Simulating Interaction Movements via Model Predictive Control and Deep Reinforcement Learning

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von

Markus Jonathan Klar
aus Augsburg

1. Gutachter:	Prof. Dr. Jörg Müller
2. Gutachter:	Prof. Dr. Gilles Bailly
Tag der Einreichung:	16.01.2024
Tag des Kolloquiums:	21.02.2024

Acknowledgements

This dissertation is the product of extensive collaboration with colleagues from my department and external collaborators. I would like to express my gratitude to my supervisor, Professor Dr Jörg Müller, for his invaluable expertise and guidance as he spent countless hours discussing, reading and revising my thesis. I would also like to thank Professor Dr Gilles Bailly, Professor Dr Michel Beaudouin-Lafon, and Professor Dr Daniel Buschek for being part of the examination committee. My heartfelt thanks go to Dr Florian Fischer, who has always been a reliable partner in the office and in discussions, as well as a good friend. Further appreciation goes to Dr Arthur Fleig, Professor Dr Miroslav Bachinski, and my other former colleagues at the University of Bayreuth whose support has been of immense value to my work. The members of the recently established User-in-the-Box projects also deserve thanks for creating exciting new approaches to user simulation and for engaging in rich discussions. I would also like to express my gratitude to Nadine Rexfort for her tireless help in all organizational matters. I am grateful to Dr Tiffany Knearem, Anne Stellberger, and Muhadj Adnan for proofreading my dissertation and providing helpful feedback. Last but not least, I would like to address my friends and family, without whom I would not have been able to get this far. Thank you.

Abstract

To evaluate the interaction between users and computers, user testing is the gold standard. However, conducting user studies can be arduous and expensive, particularly for movement-based applications in virtual or mixed reality. Additionally, their fast-paced development requires quick and cheap evaluations of interaction techniques. This thesis examines how user movements during interaction with computers can be simulated using optimal control methods. The simulation of human movements enables the automatic evaluation of interaction techniques for variables that are difficult to measure such as muscle activation, muscle fatigue or ergonomics. To enable the simulation of the whole human-computer interaction loop, this thesis presents a general framework for modeling interactions as an optimal control problem. The approach accounts for the dynamics of the application, input devices, and the human body. The latter can be exemplified through a biomechanical model of a user, which is implemented in a physics simulation. The optimal control problem is formulated as a closed control loop that translates the user's actions into application updates that are perceived by the user. The user's control strategy is then adjusted based on this perception, with the aim of achieving a pre-defined goal in the form of a cost or reward function. This thesis explores diverse control methods for the simulated user via muscle signals across various interaction scenarios. For instance, I show that the simulation of mouse-pointing with a Linear-Quadratic Regulator outperforms previous approaches in explaining user data. I demonstrate that Model Predictive Control can model 3D movement with biomechanical models and predict joint movements of users during a mid-air pointing task. In addition to classical control-theoretic approaches, I show how reinforcement learning methods can be used to simulate user movements in different VR tasks and present a system that allows developers to run simulations while working in a development environment. I conclude that the presented movement simulation is a promising approach for the automated evaluation of interactive systems and can help to improve the development of novel interaction techniques.

Zusammenfassung

Um die Interaktion zwischen Benutzern und Computern effizient zu bewerten, sind Benutzertests wichtig, aber oft aufwendig und teuer. Dies gilt besonders für bewegungsbasierte Anwendungen in virtueller oder gemischter Realität. Darüber hinaus erfordert die rasante Entwicklung dieser Anwendungen eine schnelle und kostengünstige Bewertung von Interaktionstechniken. Daher wird in dieser Abhandlung die Simulation von Benutzerbewegungen mittels Optimalsteuerungsmethoden untersucht. Die Simulation menschlicher Bewegungen ermöglicht die automatische Bewertung von Interaktionstechniken für schwer messbare Variablen wie beispielsweise Muskelaktivierung, Muskelermüdung oder Ergonomie. Um die Simulation der gesamten Mensch-Computer-Interaktionsschleife zu ermöglichen, wird in dieser Arbeit eine allgemeine Rahmenstruktur für die Modellierung von Interaktionen als Optimalsteuerungsproblem vorgestellt. Der Ansatz berücksichtigt dabei die Dynamik der Anwendung, der Eingabegeräte und des menschlichen Körpers. Das Optimalsteuerungsproblem bildet einen geschlossenen Regelkreis, der die Anwendung auf Basis der Aktionen des Benutzers aktualisiert, was wiederum vom Benutzer wahrgenommen werden kann. Die Kontrollstrategie des Benutzers kann dann entsprechend dieser Wahrnehmung angepasst werden, um ein vordefiniertes Ziel in Form einer Kosten- oder Belohnungsfunktion zu erreichen. Es wird unter Anderem gezeigt, dass die Simulation des Mauszeigers mit einem linear-quadratischen Regler Benutzerdaten besser erklären kann als vorherige Ansätze. Des Weiteren wird demonstriert, dass modellprädiktive Regelung in der Lage ist, 3D-Bewegungen mit biomechanischen Modellen zu modellieren und Gelenkbewegungen von Nutzern während einer Mid-Air-Pointing-Aufgabe vorherzusagen. Neben den klassischen steuerungstheoretischen Ansätzen wird erläutert, wie verstärktes Lernen für die Simulation von Benutzerbewegungen bei diversen VR-Aufgaben genutzt werden kann. Abschließend wird ein System vorgestellt, das Entwicklern die Möglichkeit gibt, Benutzerbewegungen zu simulieren, während sie mit einer Entwicklungsumgebung arbeiten. Zusammenfassend kann festgestellt werden, dass die Bewegungssimulation mit Optimalsteuerungsmethoden einen vielversprechenden Ansatz für die automatisierte Evaluierung interaktiver Systeme bietet und dazu beitragen kann, die Entwicklung neuer Interaktionstechniken zu verbessern.

Contents

Nomenclature

Acronyms / Abbreviations

AR	Augmented Reality
CFAT	Compute Feasible Applied Torques
DMPC	Distributed MPC
DRL	Deep Reinforcement Learning
E-LQG	Extended Linear-Quadratic Gaussian
GPU	Graphics Processing Unit
HCI	Human Computer Interaction
IC	Intermittent Control
ILQR	Iterative LQR
LQG	Linear-Quadratic Gaussian
LQR	Linear-Quadratic Regulator
MOOP	Multi-Objective Optimization
MPC	Model Predictive Control
OCP	Optimal Control Problem
OFC	Optimal Feedback Control
PPO	Proximal Policy Optimization
RGB-D	Red Green Blue Depth

RL	Reinforcement Learning
RMSE	Root Mean Squared Error
SAC	Soft-Actor Critic
VR	Virtual Reality

To do things right, first you need love, then technique.

Antoni Gaudí

1 Introduction

1.1 Movement Simulation in Human-Computer Interaction

The objectives of *Human-Computer Interaction (HCI)* include many aspects such as enhancing usability, user experience, efficiency, or ergonomics. For evaluating interfaces and interaction techniques in these regards, user testing is the gold standard. However, performing user studies is arduous and expensive. As a result, insufficient user studies are often conducted, especially in the early stages of development. This can lead to flaws in the design being discovered only later, resulting in high remediation costs. Additionally, when testing new interaction techniques with experienced users is desired, training time is necessary. This can be particularly challenging when development is under time constraints.

To enable faster and cheaper evaluation of interactive systems, models of user behavior have been investigated in HCI. These include cognitive models that focus on the goals of a user [51]. For instance, building on the understanding of human cognitive processes offered by the *Model Human Processor*, *GOMS* (goals, operators, methods, and selection rules) models provide practical tools used to predict and analyze human-computer interaction [6].

Despite being groundbreaking for modeling and simulation in HCI, these models do not cover user movement, which is an integral part of many modern interactive systems such as virtual reality (VR), augmented reality (AR), or ubiquitous computing (ubicomp) interfaces. This is problematic because studying movements and evaluating ergonomics with user testing alone can be challenging. A thorough evaluation of movement-based interaction techniques requires motion capturing, which is expensive due to the need for specialized equipment and expertise in handling resulting data. Additionally, many essential variables are intrinsic to the human body, making them challenging to quantify. These variables include muscle activations or joint loads, which can aid assessing the ergonomics of a system. This is particularly important when optimizing an interaction technique, as the result may be optimal in one aspect but may have negative effects on others. For instance, optimizing solely for a shorter movement time may result in faster muscle fatigue or muscle tension with frequent use. Another challenge is posed by the physical abilities and surroundings of users. To support inclusive applications in the future, it is essential to evaluate systems for humans with different physiologies – beyond one-size-fits-all. For instance, applications should also be evaluated and adapted for people who are confined to wheelchairs. Conducting user testing can also be challenging when accounting for different environments. This is particularly true for AR applications designed for constrained spaces, such as cubicles or airplane seats.

In silico testing might help to mitigate the need for user studies. It enables the integration of movement-based testing into the early interface development process and tools, and thus allows developers to catch issues related to ergonomics early in the design process. This simulation-based evaluation of systems can assist HCI researchers in assessing their prototypes more quickly, efficiently, and cost-effectively. Further, it will enable automatic optimization of design parameters in the future [42]. All interaction techniques are subject to such parameters, e.g., classic techniques using control-display gain [54] or arm-stretch coefficients [46], or modern approaches using input amplification [69] or ray cursor gain [1]. Selecting the best parameters is challenging since there is often an infinite amount of possible realizations and, in practice, only a limited number of design choices can be thoroughly tested due to limited financial and time resources.

Existing models of movement in HCI often inform basic principles of design. For instance, information-theoretic models such as Fitts' law [18] enable the prediction of movement time based on the size of targets. Recently, a model for the speed-accuracy trade-off for pointing movements was introduced [22]. Other works describe end-effector movement based on control-theoretic models of pointing [41] or intermittent control [39]. However, making accurate predictions of user movement and evaluating ergonomics requires models at the biological level of human action [44] and until now, models have not been able to fully capture human movement while interacting with computers.

To address this matter, detailed forward biomechanical simulations are necessary, which have not yet been established in HCI. These simulations enable the estimation of variables that are hard to quantify through user studies, such as muscle activations or fatigue. Additionally, evaluating interaction techniques with specialized biomechanical user models supports people of all abilities to enjoy using modern interfaces. Furthermore, high-fidelity physics simulations facilitate the prediction of user movements in specific interaction scenarios, such as when interacting in confined spaces, e.g., while sitting on an airplane.

In this dissertation, I therefore investigate how methods from *Optimal Feedback Control (OFC)* and *Deep Reinforcement Learning (DRL)* can be utilized to simulate human movement during the interaction with computers. To this end, I first describe how movement-based interactions can be modeled as an *Optimal Control Problem (OCP)* (Section 1.2). I propose a framework of modeling movements in HCI with OFC and demonstrate, how well mid-air pointing can be simulated using a state-of-the-art biomechanical model and *Model Predictive Control (MPC)*, comparing the resulting simulation movements to user data (Chapter 3). I evaluate simpler linear-quadratic models in their ability to simulate 1D mouse movement and show, that they outperform previous models in explaining user data (Chapters 4 and 5). Subsequently, I propose DRL approaches to simulate mid-air pointing and tracking (Chapter 6), as well as different complex VR tasks using biomechanical and perception models (Chapter 7). Finally, I introduce a system that allows developers to include simulations with biomechanical models and DRL directly into their development environments (Chapter 8).

In the subsequent chapter, I provide an overview of the works consolidated in this thesis (Chapter 2). The ensuing six chapters (Chapters 3 to 8) comprise the publications, each with a contribution statement. Subsequently, I discuss the simulation of movements in HCI and outline engaging ideas that merit further investigation (Chapter 9), before concluding the thesis (Chapter 10).

1.2 Human-Computer Interaction as an Optimal Control Problem

To enable the automatic evaluation of interactive systems with simulations, I consider the interaction between a human and a computer as an OCP. Solving this OCP will result in a simulation of user movement which can then be used to analyze and improve the system. In particular, I describe a movement-based interaction as follows. The general objective of a user is to control an application, e.g., to select an element on a display. To achieve this, the user has to move their body by choosing muscle activations. The computer then senses this movement through input devices. For example, when a controller's accelerometer registers an acceleration. This information is then processed and leads to an update of the application based on the interaction technique and application dynamics, e.g., the movement of a virtual cursor based on transfer functions (Chapter 3). Ultimately, the new state of the application is observed by the user through modalities such as visual, audio, or haptic feedback. In order to create an OCP that resembles a movement-based interaction, I will first describe models for the different parts of the system.

The user's movement is subject to the constraints raised by their physical capabilities, such as muscle strength or possible joint angles, and the physical world. To account for this, I will use either simpler models such as a spring-damper system (Chapters 4 and 5) or biomechanical models implemented in the sophisticated physics simulation MuJoCo [60] (Chapters 3 and 6 to 8). I assume that different perturbations are present in this human part of the model, such as signal dependent noise caused by the nervous system [13, 63]. Furthermore, it is important to note that humans cannot apply force directly. Instead, muscle force must ramp up over a short period of time. To account for this, I employ muscle models with delay, such as the model used by van der Helm et al. [64] (Chapters 3, 5, and 6), or the complex muscle model implemented in MuJoCo (Chapters 7 and 8). The interaction technique and application dynamics can also be modeled in various ways. They can be an implicit part of the physical model such as a mouse cursor modeled as a spring-damper system (Chapters 4 and 5). If physics simulations are used, they can be modeled as part of the physics environment such as virtual targets (Chapters 3 and 6), cursor transfer functions (Chapter 3), or buttons and joysticks (Chapter 7). Lastly, the interaction technique and application dynamics can be directly given by the application itself such as game mechanics (Chapter 8). To account for the impact of the user's perception, I investigate saccade models (Chapter 5), an RGB-D array based visual perception (Chapters 7 and 8), and haptic perception (Chapter 7). However, these perception models significantly increase the system's complexity. Therefore, in some works, I assume total observation instead (Chapters 3, 4, and 6). This means that the user knows the exact state of the application and their body, such as the position of the cursor, a virtual target, or joint angles.

In mathematical terms, I therefore define *system dynamics* that consist of the biomechanical model of the user, the input and output devices, and the application. In contrast to action-based models of interaction, movement-based interaction in the real world is continuous. This is because the real world is not divided into discrete actions, but rather is a continuous flow of movement and perception. However, since humans change their muscle control only intermittently [21] and computers technically also work on a discrete level[1], I use discrete-time models to describe the interaction. The length of a time step has to be

[1]Although faster than modeled, even the most basic operations take a non-infinitesimal time.

chosen such that a stable yet realistic simulation is possible[2]. For a given time step $k \in \mathbb{N}$, I define the current *system state* $x(k) \in \mathbb{X}$ as a combination of the biomechanical state of the user, the state of the input and output devices (e.g., controller position), and the state of the interface (e.g., cursor position). This state is constrained by the user's physical capabilities and environment (e.g., possible joint angles) and the interface (e.g., display borders), which can be implemented by constraining the *state space* $\mathbb{X}$.

The user chooses control signals $u(k) \in \mathbb{U}$ at each time step $k \in \mathbb{N}$. The possible signals are given by the control space $\mathbb{U}$ which is constrained by the physiological capabilities of the human such as maximum possible muscle activation. In the case of a biomechanical user model, this signal is then transferred to muscle activation, and, finally, to the joint torque that moves the body. This force is contingent on the user's muscle strength and typically requires some time to take full effect, as mentioned above. The resulting movements thus obey the rules defined by the biomechanical properties of the user (and of course, the laws of physics). After applying the resulting force, an updated state of the biomechanical model is attained, which is then measured by input devices. The interface state is subsequently altered based on the interaction technique employed. For instance, the user can manipulate a VR controller which is sensed by the computer and results in the corresponding movement of a virtual cursor which is rendered on a head-mounted display. The system dynamics therefore result from a combination of the biomechanical model and the physical and virtual dynamics of the interactive system. Mathematically, this can be described as follows. By applying a control $u(k)$ to the current system state $x(k)$, the system is advanced by one time step, resulting in a new state $x(k+1) \in \mathbb{X}$. The system dynamics can be written in general form as a function $f : \mathbb{X} \times \mathbb{U} \mapsto \mathbb{X}$, where

$$f(x(k), u(k)) = x(k+1). \tag{1.1}$$

The updated interface state $x(k+1)$ includes information such as a new cursor position. This state is then relayed to the user through output devices. This feedback allows the user to adjust their controls based on the difference between the desired and observed outcome of their actions. For example, the user may notice that the cursor has gone beyond its intended target, indicating an overshoot caused by excessive controller movement. As a result, they could adjust their controls to move the cursor in the opposite direction as a corrective action.

Since the user chooses a control signal at each time step, I define a sequence of controls $u(\cdot)$ that contains all control values $u(k)$ for $k \in \mathbb{N}$. By applying the system dynamics (1.1) and starting from an initial state $x_0 \in \mathbb{X}$, this results in a trajectory of system states $x(\cdot)$ which

[2]It can be useful to set a shorter time step for the actual simulation (e.g., 2 ms) than for the control (e.g., change every 40ms) to allow a stable physics simulation while still being feasible for the OCP solver. For details, see Section 3.5 in Chapter 3.

is given by

$$x(k+1) = f(x(k), u(k)) \tag{1.2}$$

for each $k \in \mathbb{N} \setminus \{0\}$ and $x(0) = x_0$. The initial state includes an initial posture of the user as well as a setting of the application, e.g., an initial target position in case of a pointing task. The resulting state trajectory $x(\cdot)$ corresponds to the entire interaction movement of the user, states the input device, and the associated updates of the interactive system.

Combining the theory of optimal human movement [61, 62] with the motivation of a user to solve the given interaction task optimally, it is possible to define an *objective function* that the user tries to minimize – subject to the constraints posed by their physiology, the interaction technique, and application. Thus, the combination of the system dynamics with this objective function defines an *OCP* that describes the complete movement-based interaction. In contrast to previous models, which often only provide end-effector movement, solving this OCP provides a forward simulation that can be used to predict human movement in interaction, up to a muscular level. Since the resulting simulation is optimal in respect to the objective function, e.g., as fast and accurate as humanly possible, it replicates the movements of experienced users. This can be difficult to achieve through user testing without time-consuming training. The user's objective function, which is also known as *cost function*, depends on the task and personal preferences. It commonly consists of terms that involve effort, task completion time, or accuracy. In control theory, the cost function is often defined using *stage costs* (or *running costs*), which only depend on the state of the system and the action chosen by the user in one time step. This stage cost is therefore defined as a function $\ell : \mathbb{X} \times \mathbb{U} \mapsto \mathbb{R}$, where $l(x(k), u(k)) \in \mathbb{R}$ describes the cost of stage $k \in \mathbb{N}$, where the system is in state $x(k)$ and the control $u(k)$ is applied. For example, in a pointing task aimed at reaching a target with minimal effort, the stage cost may be the sum of the cursor's remaining distance to the target as obtained from the current system state and the norm of the control value, since it describes the applied muscle activation.

Since the completion time of a task is unknown in general and the user chooses a sequence of controls $u(\cdot)$ as mentioned above, the optimal actions are defined as a sequence of controls $u(\cdot)$ that minimizes the *sum of stage costs* over all time steps.[3] Beginning with an initial state $x_0 \in \mathbb{X}$, the objective function is thus given by

$$J_\infty(x_0, u(\cdot)) = \sum_{k=0}^{\infty} \ell(x(k), u(k)), \tag{1.3}$$

[3]In practice, the stage cost has to be discounted since this approach will likely lead to an infinite objective function value otherwise.

where $x(\cdot)$ is the state trajectory which arises when the control is applied consecutively, following the system dynamics (1.2), i.e., $x(k+1) = f(x(k), u(k))$ for all $k > 0$ and $x(0) = x_0$. Thus, the combination of the system dynamics with the cost function defines the *OCP*, that describes the complete interaction:

$$\begin{aligned} \min_{u(\cdot)} J_\infty(x_0, u(\cdot)) &= \min_{u(\cdot)} \sum_{k=0}^{\infty} \ell(x(k), u(k)) \\ \text{such that } x(k+1) &= f(x(k), u(k)), \quad x(0) = x_0, \\ x(k) &\in \mathbb{X}, u(k) \in \mathbb{U}, \quad \text{for all } k \in \mathbb{N}. \end{aligned} \tag{1.4}$$

Through solving this OCP, a control sequence as well as a state trajectory are obtained. That means that if the system dynamics and cost function match the interaction task and user, the solution of the OCP provides a prediction of the interaction movement, starting at the initial state of the user and the interface x_0.

The methods that are available to solve the OCP (1.4) highly depend on the properties of the system dynamics f and the form of the objective function J_∞. The simplest case is linear dynamics of the form

$$f(x,u) = Ax + Bu \tag{1.5}$$

with matrices A and B of appropriate size. These dynamics are easy to handle but unfortunately do not occur often in reality. Commonly, perturbations complicate the dynamics. For example, human motion is known to be subject to signal-dependent noise [27], i.e.,

$$f(x,u) = Ax + (1+\mu)Bu, \tag{1.6}$$

where μ is a univariate Gaussian random variable. I investigate simulations with these system dynamics combined with a quadratic objective function in Chapters 4 and 5.

In the context of nonlinear dynamics, the function f may have a variety of shapes, or can even be an opaque box[4]. This means that it cannot be described as an analytical formula and instead can only be sampled. To gain a comprehensive understanding of all aspects of human-computer interactions, it is thus imperative to apply different methods. Therefore, I also employ two alternative methods to solve the OCP: *Model Predictive Control (MPC)* and *Deep Reinforcement Learning (DRL)* (see Chapters 3 and 6 to 8).

To summarize, modeling movement-based interaction as an OCP involves two tasks: defining the system dynamics and the objective function. The system dynamics include models of the user, the input and output devices, and the interactive system, enabling a

[4]I am aware that the commonly used term is "black box", but I have refrained from using it in order to emphasize the property of being "non-transparent" rather than relying on a black-white dichotomy.

forward simulation by applying user controls. The objective function depends on the task and user preferences, such as moving with low effort and quickly. It is defined as the sum of stage costs, which includes the cost of a state and control in a time step. Depending on the complexity of the system dynamics and objective function, different methods need to be applied to solve the OCP. Solving the OCP leads to a full simulation of the user movements during interaction, can provide insights into user behavior, and can be used to automatically evaluate interaction techniques.

2
Thesis Overview

In the following section, I present how the works consolidated in this thesis contribute towards the evaluation of movement-based interaction techniques. Subsequently, I provide short summaries of the papers and explain how they are interrelated.

2.1 Contributions

I provide a general framework to model movement-based interactions with OFC. Further, I apply this framework to different tasks, such as 1D mouse pointing and 3D mid-air pointing in VR, and evaluate different methods to simulate movements such as MPC. Lastly, I present systems and implementations that enable the application of the framework to further interaction techniques.

Enabling Biomechanical User Simulation Through Optimal Feedback Control

I propose a framework that enables the modeling of movement in human-computer interaction from an optimal control perspective. An interaction usually consists of an interactive system with input and output devices (e.g., controller and display), virtual interface dynamics (e.g., the application), and the task itself (e.g., clicking at a button). This task is performed by a human – the user – who aims to solve the task following some intrinsic or extrinsic motivation.

Therefore, the user takes actions that change their (physical) state (e.g., muscle activations that change the posture). The resulting state is then transmitted to the interactive system (e.g., via motion sensors) and leads to a change in the virtual dynamics (e.g., cursor movement). Subsequently, the user perceives updates of the interface through output devices (e.g., a monitor) and may adjust their actions once again based on the provided feedback, creating an interaction loop.

One substantial part of the system dynamics that describe the interaction is provided by the physical and behavioral models of the users. These are implemented as simple parameter-based models of pointing, such as the spring-damper system (Chapters 4 and 5). As an alternative, state-of-the-art biomechanical models, adjusted to the properties of reference users, are employed (Chapters 3 and 6). I incorporate complex muscle dynamics (Chapter 7) and models of visual perception (Chapters 5, 7, and 8) to further increase the realism of the simulation. These biomechanical models provide direct conclusions on variables that are difficult to measure such as muscle activation and fatigue.

Following the concept of optimal human movement control [61, 62], I assume that the user chooses their actions optimal in respect to an internalized cost function. Hence, an OCP is formulated by defining the cost function that the user aims to minimize, including criteria informed by the user's motivation, such as moving with low effort or high accuracy. These can be realized using the norm of the control vector and the distance to the target, respectively. The minimization of this cost function has to take constraints into account which are posed by the human biomechanical system, the dynamics of in- and output devices, and the virtual dynamics of the interface. Since the OCP describes an interaction *loop*, it can be solved by implementing methods from OFC. Solving the OCP ultimately provides a simulation of the human-computer interaction that can be used to evaluate interaction techniques and gain knowledge about user behavior. There are a number of ways of doing this, as detailed below.

Evaluating Different Simulation Methods with Human Data

Having defined the OCP that needs to be solved in order to simulate human-computer interactions, I investigate how different OFC and DRL methods can be used to solve the OCP. I compare various controllers for linear-quadratic systems, in particular the *Linear Quadratic Regulator (LQR)* and variants of the *Linear Quadratic Gaussian (LQG)* (Chapters 4 and 5). When using nonlinear models, such as complex biomechanical models, I implement and investigate a *Model Predictive Control (MPC)* approach to simulate interaction movements. More specifically, I collect joint movement data of humans performing a mid-air pointing task and compare different cost functions in their ability to predict this data (Chapter 3). I identify cost function parameters leading to movements that match those of real users and investigate, how changing the cost weights affects the resulting simulation movements. Notably, when compared to the reference motion, the simulation shows a smooth, convex dependence on the cost weights, which simplifies the search for suitable parameters. I also evaluate DRL approaches (Chapters 6 to 8) in their ability to produce human-like movements in different tasks, partly informed by human data.

Providing Systems for HCI Researchers and Practitioners

I present multiple systems and implementations to promote the utilization of the presented framework of modeling user movement through OCPs. Each system enables different simulations, ranging from OFC-based models of mouse pointing, over the simulation of mid-air pointing with transfer functions to complex interactions in VR applications, using biomechanical user models. The implementations provided via GitHub contain different examples such as mid-air pointing using a *Virtual Pad* interaction technique (Chapter 3) or a choice reaction task (Chapter 7). To evaluate resulting simulations, I provide user data of mid-air pointing[1] and tools to compare simulation trajectories with human movement[2].

The *SimMPC* system[3] (Chapter 3) facilitates the simulation of HCI tasks with MPC. It enables the definition of VR/AR cursor interaction techniques via transfer functions, automatic parameter identification of cost weights using a *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)* [26], and the simulation of user movement for mid-air pointing tasks with state-of-the-art biomechanical models in MuJoCo [60]. The *CFAT* tool[4] (Chapter 3) can be used to calculate feasible applied torques from motion capturing data. In particular, the tool can be used to define the maximum voluntary torques, i.e., control ranges, for the biomechanical models used in Chapters 3 and 6. The *OFC4HCI* toolbox[5] (Chapter 5) enables modeling and simulation of linear-quadratic models of human-computer interaction. For instance, it can be used to identify system parameters, simulate 1D pointing movements, and compare different linear-quadratic models in their ability to describe human movement. The *User-in-the-Box (UITB)* approach[6] (Chapter 7) includes the training and simulation of interactions with a reinforcement learning agent. The toolbox is modular and already contains different modules for the biomechanical and perceptual models, and four interaction tasks that can serve as building blocks for its application to novel applications. Lastly, the

[1] https://zenodo.org/records/7300062
[2] https://github.com/fl0fischer/uitb-tools
[3] https://github.com/mkl4r/sim-mpc
[4] https://github.com/fl0fischer/cfat
[5] https://github.com/fl0fischer/OFC4HCI
[6] https://github.com/aikkala/user-in-the-box

SIM2VR package[7] can be used to integrate biomechanical user simulations directly into VR development environments (Chapter 8). It provides an implementation for the commonly used IDE Unity[8] that can be adjusted to the task, and an interface that allows the simulated user (which is based on the UITB toolbox) to perceive the same environment as a human user would. To further increase the accessibility of our tools, I provide guidelines on how to adjust our approaches to different settings e.g, to different interaction techniques, tasks, or user models. This includes tutorial-like sections on how to simulate user movement with OFC and DRL (Chapters 3, 5, and 8).

2.2 Overview of the Presented Papers

This section provides brief introductions and contextualization of the works consolidated in this thesis. The publications themselves and their corresponding contribution statements are presented in Chapters 3 to 8. To acknowledge and value the collaborative effort that went into the presented works, I will use the plural form "we" in the following.

Simulating Interaction Movements via Model Predictive Control (Chapter 3)

To be able to simulate movement-based interactions, we have to model the human-computer interaction loop in its whole. Therefore, we build models of the interface, the input and output devices, and the user. Although many previous models of interaction are linear [39, 41], most interaction tasks already incorporate nonlinear dynamics through the interface itself such as force-to-motion functions [52]. We additionally aim to use biomechanical models to simulate user movements in a physiologically plausible and realistic way. These models also involve nonlinear dynamics such as shoulder movements, which can be challenging for OCP solvers to handle. Furthermore, our goal for modeling interaction is not only to be able to predict human movements, but also to *understand* better how humans move to interact with a computer. This is only possible if the results that our methods produce are comprehensible.

Therefore, we employ a method from mathematical control theory, namely *Model Predictive Control (MPC)*. MPC has been established as a standard method for solving linear and nonlinear problems, often involving real physical environments such as the control of refining and processing of (petro-)chemicals or polymers, power electronics, automated vehicles, or aerial swarms [49, 56, 65, 70]. The general idea of MPC is the following: Instead of solving the possibly large OCP (1.4) for which an end time is unknown, we consider only a simpler

[7]Soon to be released on GitHub.
[8]https://unity.com

$\mathrm{OCP_N}$ at each time step, which follows the same system dynamics but evolves only up to a certain MPC horizon $N \in \mathbb{N}$ into the future:

$$\begin{aligned} \min_{u(\cdot)\in\mathbb{U}^N} J_N(x_0, u(\cdot)) = \min_{u(\cdot)\in\mathbb{U}^N} \sum_{k=0}^{N-1} \ell(x(k), u(k)) \\ \text{such that } x(k+1) = f(x(k), u(k)) \text{ for all } k \in \{0, ..., N-1\}, \\ x(0) = x_0, \\ x(k) \in \mathbb{X} \text{ for all } k \in \{0, ..., N\}. \end{aligned} \tag{2.1}$$

This OCP is considerably easier to solve because it contains only a fraction of the control variables, resulting in an *optimal open-loop solution* $u^\star(\cdot)$. Then only the first part $u^\star(0)$ is applied, resulting in a new system state. This state may differ from the expected state due to disturbances such as control noise. Subsequently, the horizon is shifted by one time step, i.e., a new OCP, whose initial state coincides with the new system state, is created and then solved. This process is repeated until the task is completed or a time limit is reached.

From an HCI point of view, the open-loop solution $u^\star(\cdot)$ obtained by solving OCP (2.1) creates a prediction of the state trajectory, e.g., body and cursor positions, based on the user's planned actions in the next N time steps. After a certain amount of time – which depends on how quickly muscle activation can be adjusted – they perceive the new state of the system. This state possibly varies from the user's prediction, because of perturbations such as the noise present in the nervous system [13]. Therefore, the user makes a new prediction for the succeeding N time steps, allowing them to modify their previously planned muscle activations to improve the value of the cost function.

A simulation of the movement-based interaction is then given by the *closed-loop solution* which is defined as the formally applied controls $u^\star(0)$ of each iteration. Recently, many theorems provide mathematically proven guarantees that MPC leads to reasonable results under certain, relatively weak conditions. Dissipativity is often considered the most important factor and is usually given in physical environments [14, 23, 50].

Following the idea to consider interaction as an OCP, we define a general framework of using OFC to simulate interaction movements and explain how MPC is used to solve the OCP (1.4). Afterwards, we conduct a user study to obtain reference motion capturing data of users in a Fitts' Law-type mid-air pointing task. This data is then used to implement state-of-the-art biomechanical models of the arm and shoulder in the fast physics simulation *MuJoCo (Multi-Joint dynamics with Contact)* [60]. To reduce the complexity, we use a second-order muscle model that produces torques at each joint instead of using direct muscle control [64]. We create and implement the *Compute Feasible Applied Torque (CFAT)* tool to find appropriate maximum applicable torques for each joint.[9] After fitting the necessary parameters, we assess the efficacy of three cost functions in simulating accurate pointing movements similar to those of real users. Our analysis examines the distance cost [12, 17, 48, 59], which penalizes the distance between the cursor and target, and the control or effort cost [24, 37, 61], which quadratically penalizes the control vector's norm. Moreover, we combine these costs with either *commanded torque change costs* [32, 43, 67, 68] or *joint acceleration costs* [67]. Since the latter manages to explain the user date from our study best, it is scrutinized further. We also explain how the MPC horizon effects the simulation results and provide guidelines on choosing the right horizon length. To further help HCI researchers and practitioners to apply the simulation with MPC, we provide a step-by-step guide and give advice on adapting the approach for other applications. Overall, our findings demonstrate that MPC can simulate mid-air pointing movements that fall within the between-user variance. This shows, that the simulation with MPC can provide reasonable insights in user movement during mid-air pointing tasks. We propose that this approach has the potential to be applied to various HCI tasks and input techniques, and allow an automatic evaluation of design parameters.

An Optimal Control Model of Mouse Pointing Using the LQR (Chapter 4)

Although most HCI tasks involve nonlinearities as in the models used in the work described above, it is often possible to learn from and improve upon simplifications. In particular, OCPs with linear system dynamics with quadratic costs can be easily solved (as already explained in Section 1.2). Therefore, we also explore the abilities of the *Linear-Quadratic Regulator (LQR)* for the prediction of interaction movements (see Chapter 4). LQR leads to a linear, optimal solution for the closed loop system that can be computed offline. This means that the optimal control at a time step $k \in \mathbb{N}$ is simply of the form $u(k) = Kx(k)$, where $x(k) \in \mathbb{X}$ is the state of the system at time k and K is the solution matrix of the discrete-time algebraic Ricatti equation of appropriate size [36]. Since it is not feasible to employ nonlinear biomechanical models in this case, we choose to model 1D mouse pointing movements with a second-order linear system (often referred to as a spring-damper system). The user chooses a control signal u that acts directly on the acceleration of the mouse pointer. This simplifies how the acceleration is produced in reality, but leads to a linear quadratic OCP that can be solved with the LQR. More specifically, we identify system parameters based on the Pointing Dynamics Dataset [41], using sum squared errors. To examine the LQR's abilities to simulate mouse movements, we utilize different variants of cost functions. In particular, we consider combinations of the pointer's distance to the target and the pointer's jerk (the third derivative of the position of the mouse pointer), as commonly used in human motor control [19]. While

[9]All created systems and tools are available on GitHub, as mentioned in 2.1

jerk is penalized during the whole movement, the distance is added either at each time step or only at the time of the mouse click (taken from user data). Furthermore, premature action is also penalized until a certain time has elapsed to simulate reaction time. A factor scaling the jerk cost as well as the reaction time is also obtained by the parameter identification described above. We additionally evaluate the LQR against direct second-order lag [41] and minimum jerk models [19], showing that the LQR leads to pointing trajectories that fit best to the observed human data.

Optimal Feedback Control for Modeling Human-Computer Interaction (Chapter 5)

Following the initial investigation and success of LQR for HCI tasks, we give a proper introduction of OFC to the HCI-community and postulate a novel framework for the simulation of interaction movements by solving OCPs. We further investigate how well different simple linear models can produce one-dimensional mouse movements. First, we reevaluate the LQR, adding the same second-order muscle model as in Chapter 3 to improve the realism of the created movements. As the cost function we use the sum of distance, velocity, acceleration (or force), and control costs throughout the movement. To increase the realism of the simulated movements, we also apply the stochastic version of LQR, the *Linear-Quadratic Gaussian Regulator (LQG)*. The LQG is further extended by incorporating saccades, i.e., quick eye movements between fixations, and an imperfect initial target estimation *(E-LQG)*, based on the work of Todorov [58]. Additionally, we implement *Intermittent Control (IC)* for the given interaction task, as recently proposed by Martin et al. [39]. In Chapter 5, we use parameter identification based on the sum squared errors (for deterministic models) and the Wasserstein distance (for stochastic models), and fit the model parameters using the Pointing Dynamics Dataset [41]. We then compare the simulated trajectories of LQR, LQG, E-LQG, IC, and the previously used models of second-order lag and minimum jerk. The results show that the E-LQG predicts human 1D mouse pointing movements best in regards to the sum squared error between simulation and mean user trajectories. With the proposed framework, we can also identify different characteristics of user movement. Depending on the complexity of the chosen model, the identified model parameters can also be interpreted naturally and help to understand how users vary in their way of solving the pointing task.

Reinforcement Learning Control of a Biomechanical Model of the Upper Extremity (Chapter 6)

After exploring classic OFC methods, we investigate different *Deep Reinforcement Learning (DRL)* methods in their ability to simulate interaction movements. It is important to note, that even though OFC and DRL are different, they both aim to solve an OCP (1.4) that we use to model the interaction. The idea of DRL is to find a *policy* that provides actions depending on the current state such that they lead to the maximal accumulated *reward*. This is similar to the methods used above, which aim to minimize accumulated stage costs[10]. The policy is based on a *deep neural network*, that is updated in regular intervals during training by a so-called *agent*, that learns to control the system to obtain higher rewards. When applied subsequently, this policy produces a sequence of controls similar to the OFC methods (cf. Section 1.2). One advantage of DRL over classic OFC methods is that the agent can learn even complex tasks through exploration. Like MPC, DRL methods can work with nonlinear system dynamics because the agent only needs to be able to sample the system, i.e., perform actions, observe the new state, and receive rewards. Therefore, we use the same biomechanical and muscle model as in Chapter 3. We train a *Soft-Actor Critic (SAC)* algorithm [25] on the pointing at, and tracking of targets in front of the model. As actions, the SAC agent chooses muscle activations for each time step and joint which are then transferred to joint torques through the second-order muscle model. Instead of defining a cost function that is minimized, DRL methods use reward functions that teach the agent which actions are best. To encourage fast and accurate movements, we implement a combination of negative time and distance as reward. We assume perfect observability, i.e., the agent knows the exact position of the targets as well as his own body posture. Although human movements are straighter than those generated by the simulation, the results still exhibit similar characteristics to human motion. In particular, we show that Fitts' Law [18] and the 2/3 Power Law [35] hold for the simulated pointer trajectories. To summarize, we have shown that a DRL agent can learn to control a biomechanical user model to perform mid-air interaction tasks, showing classic characteristics that can also be observed in human movement.

Breathing Life Into Biomechanical User Models (Chapter 7)

Following our previous results with DRL methods, we aim to work on a more general approach (see Chapter 7). In this study, we expand upon the previous approach by incorporating a visual perception model and utilizing a more complex muscle model that allows for the control of individual simulated muscles instead of producing joint torques. Therefore, next to the previously demonstrated tasks of pointing and tracking, we consider the more complex tasks of choice reaction and remote car control. The choice reaction task requires the agent to identify a color displayed on a screen and press the corresponding button. In the remote car control tasks, the agent must locate the parking spot and use a joystick to control the speed of the remote-controlled car. The example above illustrates a higher-order task compared to

[10]Mathematically, minimizing costs is the same as maximizing the negative costs.

the others. This is because the agent must first learn how to move the car with the joystick before being able to steer it towards the target. Furthermore, it requires the agent to observe additional haptic feedback when touching the joystick.

To also further improve the realism of the resulting movements, and in contrast to the model in Chapter 6, we use a more sophisticated muscle model implemented in MuJoCo. Instead of controlling the joints, the agent learns to choose relative muscle controls, i.e., the change in controls of 26 muscles simultaneously. This approach was not possible with MPC because the control space grows exponentially with the number of controls which greatly slows down the solving of the OCP and leads to instabilities[11]. In contrast to the previous DRL approach in Chapter 6, the observation consists of a combination of proprioceptive feedback, tactile feedback, and visual feedback in form of a low resolution RGB-D array. Furthermore, we replace the SAC with a *Proximal Policy Optimization (PPO)* algorithm [53] which has been successfully applied in other HCI works [7, 30]. The agent gains discrete rewards for fulfilling the interaction task (e.g., reaching a virtual target or parking the remote controlled car in the correct spot), continuous negative distance rewards to encourage the agent to move towards certain targets, and negative effort rewards to ensure efficient movements. All four implemented interaction tasks are solved satisfactorily by our agent, showing that DRL methods combined with sophisticated biomechanical models and realistic perception can be used to simulate human movement even for complex interaction tasks.

SIM2VR: Integrating Biomechanical Simulations in VR Development Environments (Chapter 8)

All of the above approaches demonstrate how simulations can be produced for movement-based interactions. However, these simulations were done using specialized models of the actual interaction environment that differ from the environment a human user experiences. Although they can be used to evaluate and improve the considered interaction techniques, such as the Virtual Cursor and Pad in Chapter 3, it is therefore difficult to compare them directly to user data. Furthermore, these specialized models lack a way to easily transfer to other scenarios such as different virtual environments or tasks. The primary rationale for this is that the simulation must encompass not only the user's biomechanics, but also the interactive system itself. Since the simulated interactive system deviates from the real system, a "reality-gap" between the environment that is used to train an agent (or to obtain a feedback controller) and reality is introduced. To close this gap, we introduce *SIM2VR* – a platform the enables the seamless integration of biomechanical user simulation directly into the virtual reality development environment (see Chapter 8). More specifically, SIM2VR

[11]Often referred to as the *Curse of Dimensionality* [2] which is further discussed in Section 9.2.

provides an interface between the VR environment and the biomechanical user simulation. The interface transmits the simulated user's input movements, captured by virtual sensors, to the VR environment as if a real user were involved. Additionally, it supplies the agent with task-dependent rewards and the same visual feedback that a real user would perceive. In other words, SIM2VR allows for the use of the identical environment when training DRL agents and for the user. To demonstrate our approach on a common VR scenario, we implement a whac-a-mole VR game. We train the agent to play the game and compare the resulting simulation to data obtained via a user study in regards to performance, effort, and strategy. Specifically, we show that the simulated user achieves similar performance as the users in our study, allows prediction of effort differences in different game settings, and can be used to learn about (possibly unintended) user strategies. This enables quick evaluations and optimizations of various application settings, such as target placement or interface dynamics. It can also assist designers in improving the user interface to help users find the best strategies.

3 Simulating Interaction Movements via Model Predictive Control

Authors: Markus Klar, Florian Fischer, Arthur Fleig, Miroslav Bachinski, Jörg Müller
Status: Published in ACM Transactions on Computer-Human Interaction, Volume 30, Issue 3 [33]

The concept and framework was developed by all authors. MK implemented the simulation with MPC and the parameter fitting. FF implemented the CFAT tool. MK selected and compared the cost functions. The study was designed by all authors. MK conducted the user study. User data was preprocessed by MB, MK, and FF. MK and FF generated, processed and analyzed the simulation data. Figures were created by FF and MK. The results were interpreted and discussed by all authors. MK wrote the first draft of the manuscript. Revision and rewriting of the manuscript was done by all authors. MK is the corresponding author.

Simulating Interaction Movements via Model Predictive Control

MARKUS KLAR, FLORIAN FISCHER, and ARTHUR FLEIG, University of Bayreuth
MIROSLAV BACHINSKI, University of Bergen
JÖRG MÜLLER, University of Bayreuth

We present a Model Predictive Control (MPC) framework to simulate movement in interaction with computers, focusing on mid-air pointing as an example. Starting from understanding interaction from an Optimal Feedback Control (OFC) perspective, we assume that users aim at minimizing an internalized cost function, subject to the constraints imposed by the human body and the interactive system. Unlike previous approaches used in HCI, MPC can compute optimal controls for nonlinear systems. This allows to use state-of-the-art biomechanical models and handle nonlinearities that occur in almost any interactive system. Instead of torque actuation, our model employs second-order muscles acting directly at the joints. We compare three different cost functions and evaluate the simulation against user movements in a pointing study. Our results show that the combination of distance, control, and joint acceleration cost matches individual users' movements best, and predicts movements with an accuracy that is within the between-user variance. To aid HCI researchers and designers in applying our approach for different users, interaction techniques, or tasks, we make our SimMPC framework, including CFAT, a tool to identify maximum voluntary torques in joint-actuated models, publicly available, and give step-by-step instructions.

CCS Concepts: • **Human-centered computing → HCI theory, concepts and models**;

Additional Key Words and Phrases: Simulation, model predictive control, optimal feedback control, biomechanics, interaction techniques, mid-air pointing, AR/VR environments, maximum voluntary torques

ACM Reference format:
Markus Klar, Florian Fischer, Arthur Fleig, Miroslav Bachinski, and Jörg Müller. 2023. Simulating Interaction Movements via Model Predictive Control. *ACM Trans. Comput.-Hum. Interact.* 30, 3, Article 44 (June 2023), pages 50.
https://doi.org/10.1145/3577016

1 INTRODUCTION

Movement during interaction can be understood from an **Optimal Feedback Control (OFC)** perspective [18]: During interaction, users aim at computing muscle control signals, which control the dynamical systems of the users' bodies, which in turn control interactive systems. OFC states that users aim at minimizing an internal cost function subject to the constraints imposed by the users' bodies and the interactive systems. They do so by observing the state of the interactive system and *continuously* adjusting their controls to further their goals.

Authors' addresses: M. Klar, F. Fischer, A. Fleig, and J. Müller, University of Bayreuth, Bayreuth, Universitätsstraße 30, 95440, Germany; emails: {markus.klar, florian.j.fischer, arthur.fleig, joerg.mueller}@uni-bayreuth.de; M. Bachinski, University of Bergen, Bergen, P.O. Box 7800, 5020, Norway; email: miroslav.bachinski@uib.no.

1073-0516/2023/06-ART44
https://doi.org/10.1145/3577016

It is this observed continuity and the adjustment of controls that drives the desire to model interaction beyond summary statistics, in order to predict movement along the *entire* interaction loop between human and computer, including, e.g., joint postures or cursor trajectories, *on a moment-by-moment basis*. Taking the OFC perspective allows us to accomplish these things, by modeling interaction as an optimization problem. Here, the optimization variable is a continuous muscle signal, i.e., a function of time, that drives the user's movement, which "controls" the system. Thus, this optimization problem is usually referred to as an **Optimal Control Problem (OCP)**. The *feedback* part of OFC is due to solving the OCP in a feedback manner to model the user's ability to adjust their control during interaction, e.g., to react to unforeseen circumstances such as perturbations in the cursor movement.

Previous approaches of modeling interaction from the OFC perspective have employed linear optimal control theory, particularly the **Linear Quadratic Regulator (LQR)** [17], its stochastic extension LQG [18], and intermittent control methods [44]. These approaches differ fundamentally from our approach in that they considerably simplify the problem of computing the optimal control signals by using linear approximations to the human–computer system (e.g., a second-order spring-damper model) and quadratic cost functions. However, these limitations lead to unrealistic simplifications of the human–computer system. Typically, human movements are simulated only with simple point-mass models, since modeling the kinematic chain already leads to nonlinear dynamics. Other important nonlinear features, such as those of interactive systems (e.g., transfer functions), similarly cannot be modeled by this linear approach. Furthermore, quadratic cost functions cannot accurately reflect many tasks in Human–Computer Interaction, such as accurately hitting a button with abrupt boundaries.

In this article, we extend the OFC approach to Human–Computer Interaction to nonlinear dynamics and non-quadratic cost functions by using **Model Predictive Control (MPC)** [23]. This allows us to investigate the simulation of human movement during interaction with computers using a state-of-the-art nonlinear biomechanical model of the human upper extremity in combination with nonlinear interaction dynamics such as pointer acceleration [46]. We evaluate our approach by simulating an ISO mid-air pointing task with two different interaction techniques, each in two different settings.

MPC as a method has various strengths, such as the easy inclusion of constraints and certain theoretical functionality guarantees to provide trust and reliability, but the main idea behind MPC is *complexity reduction* in time. It takes the above OCP, which can be computationally hard to solve for the whole interaction/movement duration, and breaks it down into iterative sub-problems of much smaller duration, which are thus considerably easier to solve. After solving a sub-problem, only the first part of the resulting optimal control sequence is applied to the system, resulting in a new system state. The horizon is then shifted by one step, i.e., the next sub-problem starts with this new state. This makes the MPC a closed-loop feedback controller, which is inherently robust against perturbations that may occur during the interaction.

In summary, the contribution of this work to the field of HCI is threefold:

(1) We **propose a framework** that combines nonlinear biomechanical modeling and MPC to simulate human movement during interaction on a moment-by-moment basis (SimMPC), and a method to infer the maximum voluntary torques used in an interaction task (CFAT).
(2) We **evaluate our framework** for the example use case of mid-air pointing, including comparisons between three different cost functions in their ability to generate biomechanically plausible movements, as observed in a new user study.
(3) We **make our approach accessible** for HCI researchers by providing the sim-mpc and cfat python packages as well as a step-by-step guide on how to apply our framework to different tasks and interaction techniques.

The article is structured as follows. Related work is discussed in Section 2. The core of this article, our simulation approach using MPC (and practical advice on its use), is presented in Section 3. In Section 4, we introduce *CFAT*, a method to compute maximum voluntary torques for joint actuated models such as the one used in this article. Then follows an evaluation of our approach, applied to the use case of ISO pointing in VR, in Sections 5 and 6, where we show that our simulation is able to predict biomechanically plausible user movements. A discussion of the advantages and limitations of MPC ensues in Section 7. Section 8 explains step-by-step how to apply and extend our framework to different tasks and interaction techniques, using tracking a moving target via ray casting with a handheld VR controller as an example. Section 9 concludes the article.

The article is supplemented by our open-source project (https://github.com/mkl4r/sim-mpc), which consists of several components. The sim-mpc python package includes scripts to simulate mid-air interaction movements, compare simulation and user trajectories, generate plots, and optimize model parameters. The ISO-VR-Pointing Dataset contains marker, joint angle, and joint torque data for all mid-air pointing movements from our user study, and the SIM-MPC Dataset contains the corresponding simulation data produced for this work. The cfat python package provides an implementation of the CFAT method.

2 RELATED WORK

2.1 Forward Models of Interaction Movements

Forward models of movement during interaction with computers can predict variables such as movement duration, joint angles, or muscle activations. Depending on what they predict, they can be categorized as *summary statistics* (e.g., movement duration), *end-effector models* (e.g., end-effector position), or *kinematic chain models* (e.g., body joint trajectories).

The most widely used *summary statistics* model of the end-effector is Fitts' Law [19]. It allows to predict the overall movement time MT from the distance D and width W of the target as $MT = a + b * \log_2(D/W + 1)$ (in the Shannon formulation [43]). It is important to note that Fitts' Law has been developed to describe movement of the human hand. The fact that the same law can also be used to describe the movement of a virtual end-effector such as the mouse pointer, mediated by input devices and computer programs, is one of the great insights of HCI [8]. The parameters a and b must be identified for each user and type of movement (e.g., interaction technique) separately. Recently, more advanced models have been developed to predict, e.g., the failure rate and button press timing in moving-target acquisition tasks [37, 38].

End-effector models describe the entire trajectory of the end-effector during the movement. A classical end-effector model of hand movement is the minimum jerk model [20]. In HCI, only few works investigate the motion of the end-effector, although Bootsma et al. [6] demonstrate the importance of understanding movement in HCI beyond summary statistics. Müller et al. [47] give an introduction to end-effector models in HCI. They investigate the kinematics of mouse movements and compare four models from manual control theory. Quinn and Zhai [53] demonstrate how the minimum jerk model can be used to model finger movements during gesture typing. Jokinen et al. [33] frame touchscreen typing as a visuomotor coordination task and show that optimal supervisory control allows to generate human-like eye-hand movement patterns. Fischer et al. [17, 18] compared the applicability of different optimal control methods to simulate and predict mouse pointing trajectories, and introduced a general optimal control framework for Human–Computer Interaction. The focus there lies on controllers, such as the **Linear-Quadratic Gaussian (LQG)** Regulator, which are able to describe mouse pointing while also incorporating signal-dependent noise via a linear-quadratic OCP. The limitation to linear system dynamics rules out its application to more complex models of human biomechanics. Moreover, all of the above works have only analyzed motion in 1D or 2D, although they are in general not limited to 1D or

2D motion. The only end-effector model that is evaluated with 3D mid-air movements in HCI that we are aware of is the recent work of Bachinski et al. [2], who investigate a 2nd and a 3rd order lag for modeling mid-air movements.

In this work, we are aiming to model not only end-effector movements in 3D but using a biomechanical model of the human upper body. This allows us to obtain joint angles, velocities, and even aggregated muscle activation, each observed during interaction. Therefore, we consider *kinematic chain models* that, in contrast to pure end-effector models, also make predictions about the underlying causes of the movement by modeling the entire kinematic chain. In particular, this allows to predict ergonomic variables such as joint angles and joint moments. Most of the previous work on biomechanical models of human movement outside of HCI has concentrated on the substantially simpler 2D case and simple linked-segment models [28, 41, 61, 67]. Linked-segment models use simplified bones as sticks and hinge joints, usually without movement constraints. In movement science, the minimum torque change model [67] has been proposed, transferring the idea of the minimum jerk model (i.e., maximization of “smoothness”) to a simple 2D linked-segment model. This model requires the exact movement time as well as all joint angles, velocities, and torques of the initial and final postures as input, and yields the kinematics and dynamics of the movement between initial and final state as output. Li and Todorov [41] present a control method for a 2D linked-segment model using the **iterative Linear Quadratic Regulator (iLQR)**, which minimizes the difference between current and target posture plus quadratic control costs. However, this assumes that the final body posture is known in advance, which is not necessarily the case when the only goal is to move an end-effector (i.e., the fingertip or a virtual cursor) to a target. Moreover, the model has not yet been extended to the 3D case.

2.2 Inverse Biomechanical Simulation in HCI

In contrast to forward models, inverse biomechanical simulation takes as input human movement data and performs inverse estimations of how a specific movement was created. The method stems from the fields of biomechanics and rehabilitation and allows to compute accurate physiological indices of movements [10, 54]. Given motion capture data, it allows to estimate multiple internal variables such as joint angles, joint moments, muscle forces and activation, and neural excitation signals. At the core of the biomechanical simulation is a musculoskeletal model, which represents the kinematic, inertial, dynamic, force generation, and neural control properties of the human body [56]. Biomechanical simulation has been introduced and validated for HCI tasks as a method for ergonomic and fatigue evaluation of post-desktop user interfaces [3, 4]. It has also been used as a data generation method to develop summarization models of performance and ergonomics for arm movements [5]. Simplified biomechanical models were adapted as components of simulations for fatigue assessment tools [30, 31]. Although one current weakness of biomechanical simulations is its necessity for motion capture data collection in user experiments, these simulations have a large potential in the field of HCI, in particular for the analysis and development of AR, VR, and ubicomp user interfaces.

2.3 Deep Learning and Muscle Control

The above works from Sections 2.1 and 2.2 have investigated the control of the human body or a virtual object using either outcome-specific relationships such as Fitts’ Law, or optimization methods such as iLQR. Another approach incorporates recent tools and methods from the field of Deep Learning. Most notably, Cheema et al. [9] recently presented a method to estimate cumulative fatigue during mid-air interaction, in terms of Borg CR10 ratings. They use a 3D linked-segment arm model and **reinforcement learning (RL)** to learn a control policy for a Fitts’ law type task. They propose a novel *reward function*—the analogy of a cost function in OCPs, based on effort estimated through the *Three Compartment Controller*, and show that this generates faster and more "ergonomic" movements compared to a baseline reward of summed normalized instantaneous joint torques. Their model is shown to be able to predict the Borg CR10 ratings of the movements performed in [31] with good accuracy. However, Cheema et al. [9] did not analyze the realism of the movements generated by their approach in terms of end-effector trajectories or joint angles, but rather in terms of predicted cumulative fatigue, averaged over 12 models.

Following the work of Cheema et al., in [16] a state-of-the-art RL algorithm was used to learn to move the finger to arbitrary targets within reach. The resulting end-effector trajectories follow both Fitts' Law [19] and the $2/3$ Power Law [36]. Lately, Hetzel et al. have extended the model from [16] to simulate mid-air keyboard typing, using the same RL method [29].

Beside these works, the objective of many research works making use of Deep Learning methods is not to model or understand human motion, but rather to create interesting and realistic animations for movies or computer games. We are not aware of any works from this research area that compare the synthesized movements to actual human movements on a biomechanical level. Similarly to the works from movement science, most works have controlled the torques at the joints (e.g., [26, 49]). Control on a muscle-level has traditionally been considered to be computationally infeasible. This is due to the fact that the computation time increases exponentially with the dimensionality of the control problem, called the *curse of dimensionality*.

Recently, however, two approaches to create movements of muscle-actuated characters have been presented. Lee et al. [39] propose a two-level *imitation learning* algorithm for musculoskeletal models. A high-level controller follows a reference motion and generates target joint angles. A low-level controller then controls the muscles to generate the appropriate forces. Imitation learning assumes that reference motions from humans are available. Whether and how imitation learning approaches can generate novel interaction movements that are not available as recordings will be an important question for future research.

In contrast to imitation learning, *reference-free approaches* can synthesize novel movements based only on the model description and reward function. Jiang et al. [32] present an approach to circumvent the muscle control problem by controlling the character in joint space, while determining maximum joint torques and energy costs from a neural network, learned from a realistic model in OpenSim [10]. Jiang et al. demonstrated their technique on a leg model. Whether and how this approach can work for a significantly more complex arm model, especially taking the shoulder into account, remains open.

Control of muscles is particularly necessary when the movements are big and cover very different joint angles, as moment arms change significantly during the movement in such cases. One example used by Jiang et al. [32] is a jump for maximum height, where the joint torque network prevents overbending of the knees as an optimal strategy. However, during most interactions, the movements are small and the moment arms change only minimally during the execution. In this case, actuating the joints either based on simplified muscle dynamics (as we do in this work) or direct torque control can be a good approximation and is substantially simpler to use than complex musculotendon models. For small movements, passive forces created by ligaments and musculotendon units also play a smaller role.

3 MODELING INTERACTION AS MODEL PREDICTIVE CONTROL

In this section, we describe our approach to model and simulate human movement during interaction with the computer using MPC. We lay the theoretical foundation and provide concrete but extensible models and practical advice. A concrete use case is described in detail in Section 5.

The defining aspect of movement in Human–Computer Interaction is that users move their bodies *in order to change the state of an interactive system* such that their virtual representation,

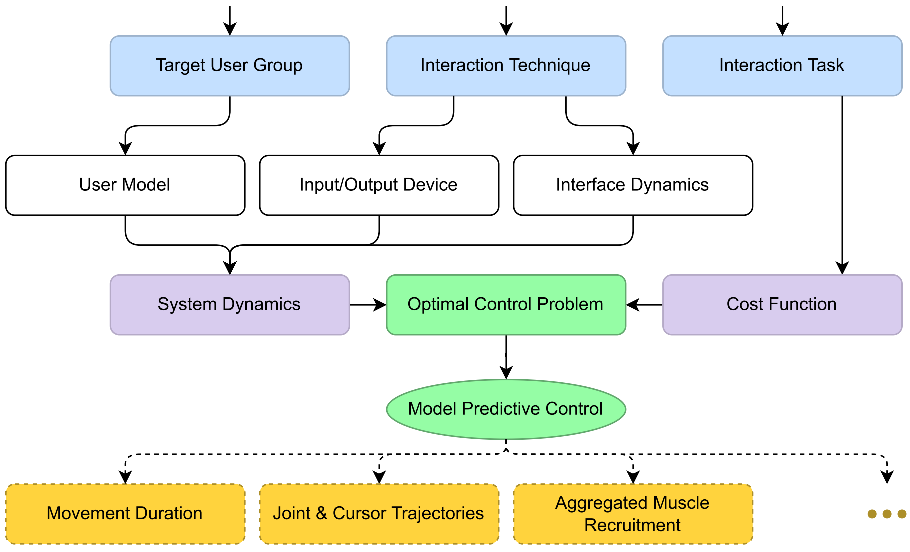


Fig. 1. Our proposed simulation framework. To run a simulation, one needs to specify the Target User Group, Interaction Task, and Interaction Technique. Based on the Target User Group, a User Model that matches one or more representative user(s) is chosen (or designed). Combined with the Input Device and Interface Dynamics—both defined by the Interaction Technique—this results in the System Dynamics. Finally, the Interaction Task imposes a specific Cost Function that the user is assumed to minimize. The resulting nonlinear OCP can then be solved with MPC, resulting in a simulation of the movement. From this, we can obtain valuable data like movement duration, joint and cursor trajectories, or aggregated muscle recruitment.

(e.g., an avatar or a cursor), reaches a desired state, e.g., selecting a button or dragging a virtual object.

When modeling this interaction, it is important to be able to deal with nonlinearities, as user movement and interaction techniques are in general nonlinear. On top of that, for most interaction techniques, there exists an infinite number of body movements that result in the same state-change in the interactive system. For example, using the mouse as input device, movements to the left result in the same cursor movement as movements to the top after a clockwise rotation of the mouse by 90 degrees. This complicates the simulation of movements, as it is unclear how the model should move in order to plausibly replicate human behavior during interaction. Therefore, in order to understand the entire interaction loop on a moment-by-moment basis, the actual state of the user's body *and* the input device needs to be taken into account, which can be achieved by utilizing a unifying and mathematically rigorous optimal control framework of interaction [18].

Our framework is depicted in Figure 1. Our model takes into account the *target user group*, the *interaction technique*, and the *interaction task*. The first influence is the *User Model*, see Section 3.1, where we match the physical properties of the target users by using state-of-the-art biomechanical models. The interaction technique consists of two parts in our framework. First, modeling of the *Input/Output device* (e.g., motion capture tracking of the index finger, or the combined use of a HTC Vive controller and **head-mounted display (HMD)**) is described in Section 3.2. Second, we define *Interface Dynamics* that determine how the user input is transferred to the virtual system (e.g., to a virtual cursor) in Section 3.3.

Readers that are already familiar with the above concepts and are mostly interested in the core method used to simulate movements may directly skip to Section 3.4. There we introduce the notation of a nonlinear *OCP*, which augments the former discussed models of the human body and the interaction technique with a *Cost Function* that formalizes the user-specific objectives for a given interaction task.

In Section 3.5, we show how the OCP can be solved via *MPC*, resulting in a simulation of the complete biomechanical chain during interaction. We thus obtain trajectories of joint angles, angular velocities, and accelerations; trajectories of cursor positions, velocities, and accelerations; and biomechanical data such as aggregated muscle recruitment.

3.1 User Model

The biomechanical properties of the user model should match those of the considered users, and the range of possible movements must be sufficient to fulfill the task. To fit in our simulation pipeline (Figure 1) and act as a part of the system dynamics, the user model should be able to be forward-simulated, i.e., to map the current body state $\mathbf{x}_{\text{user}}$ (including, e.g., joint angles and angular velocities, and the internal state of the muscles), and the aggregated muscle control signals u to the next (i.e., updated) body state $\mathbf{x}^+_{\text{user}}$ in a realistic way.[1] Formally, this mapping can be defined via a function f_{user}:

$$\mathbf{x}^+_{\text{user}} = f_{\text{user}}(\mathbf{x}_{\text{user}}, u). \tag{1}$$

In this work, a biomechanical, joint-actuated model implemented in a physics engine, coupled with second-order muscle dynamics, will take the role of f_{user}. Of course, it is possible to exchange our user model with other models of human motion.

3.1.1 Upper Extremity Model in MuJoCo.

We make use of the fast physics simulation MuJoCo [64] to handle the complex biomechanics of human motion. In [16], a MuJoCo model from the state-of-the-art OpenSim [58] musculoskeletal model from Saul et al. [56] was derived. We use this MuJoCo model for two reasons: (i) limitations in OpenSim's ability to simulate contacts—it is very difficult in OpenSim to allow a model to interact with input devices and environmental objects such as a chair or table, while preventing the model from reaching through its torso or legs −; and (ii) computation speed.

The biomechanical model has seven independent joints[2] (i.e., seven DOFs)[3] and 13 coupled joints, representing a shoulder, an elbow, and a wrist. The shoulder is the centerpiece of the model and is connected to a torso—which is made immovable during interaction for simplicity—through a set of three independent and eleven coupled joints. The three independent joints set up the angle and extent of the elevation, as well as the rotation of the upper arm. Ten of the eleven coupled joints are used to accurately describe the motion of clavicle and scapula with respect to the shoulder elevation. Since the joints build on each other, another coupled joint is used to revert the elevation angle before applying rotation. The elbow is composed of two independent joints allowing flexion-extension and pronation-supination movements. For the wrist, we use four joints, two independent and two coupled, which allow accurate flexion-extension and abduction-adduction movements of the hand. The finger joints are locked in a pointing posture, since they are less important for our example tasks and omitting them considerably simplifies the user model. The complete model is depicted in Figure 2; joint angle ranges can be found in Appendix B.1.

To avoid the *curse of dimensionality*, i.e., the exponential growth of computation time with the number of variables to be optimized, we refrain from including muscles in our MuJoCo model. Instead, we implement simplified muscles that directly act on the joints as follows. We place a

[1]Here and throughout this work, we denote states that contain a variety of different quantities in bold font, and the individual quantities in regular font.
[2]Some human joints are reflected by multiple model joints. Therefore, throughout the article, we use the term *joint* synonymously for a hinge joint in our model.
[3]degrees of freedom.

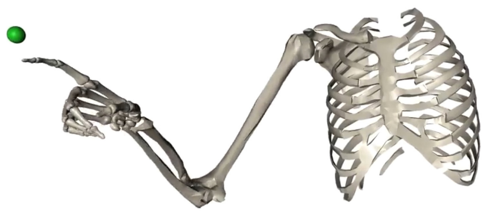


Fig. 2. Visualization of our MuJoCo user model. The green sphere models the motion tracking marker for the physical end-effector.

torque actuator that can produce positive and negative torque around the axis of each of the seven independent joints. At any given time step n, for $i \in \{1, \ldots, 7\}$ the applied torque $\tau^i(n)$ of each actuator depends on its current activation $x^i_\sigma(n)$, scaled by the maximum voluntary torque g^i for the respective joint:

$$\tau^i(n) = g^i x^i_\sigma(n). \tag{2}$$

The current activation $x^i_\sigma(n)$ of each torque actuator is obtained through a simplified second-order muscle model, which is explained in detail below.

For practical applications, one challenge with this simplified muscle model is to determine the maximum voluntary torques $g = (g^1, \ldots, g^7)^\top \in \mathbb{R}^7$ for each independent joint, as to prevent unrealistic movements. To this end, we propose *CFAT*, a tool described in Section 4, to obtain better matching torques.

3.1.2 Second-Order Muscle Dynamics. Modeling and simulating human muscles has proven to be challenging. This is not only because of the sheer amount of muscles—in the original OpenSim model [56], the shoulder and arm alone are moved by a total of 31 muscles—but also because of the complex interaction of force generation, tendon lengths, tendon positioning, and so on. Optimizing for each muscle activation simultaneously is a challenging problem, which so far has only become feasible through techniques like hierarchical optimization [42] or through aggregation. We follow the approach by van der Helm et al. [69], who aggregate muscles for each DOF using second-order dynamics. We discretize these muscle dynamics using the forward Euler method [7]. The vector $x_\sigma = (x^1_\sigma, \ldots, x^7_\sigma)^\top \in [-1, 1]^7$ contains the activation for all seven DOFs. The vector of activation derivatives is denoted by $x_{\dot\sigma} = (x^1_{\dot\sigma}, \ldots, x^7_{\dot\sigma})^\top \in \mathbb{R}^7$, and is affected by the vector of applied controls denoted by $u = (u^1, \ldots, u^7)^\top$. In formulas, the discrete-time dynamics for each DOF $i \in \{1, \ldots, 7\}$ can be described as follows, where n is the current time step and $n+1$ the next one:

$$\begin{bmatrix} x^i_\sigma(n+1) \\ x^i_{\dot\sigma}(n+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ \frac{-\Delta t}{(t_e t_a)} & 1 - \Delta t \frac{t_e + t_a}{t_e t_a} \end{bmatrix} \begin{bmatrix} x^i_\sigma(n) \\ x^i_{\dot\sigma}(n) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\Delta t}{t_e t_a} \end{bmatrix} u^i(n), \tag{3}$$

with initial constraints

$$x_\sigma(0) = \sigma_0, \; x_{\dot\sigma}(0) = \dot\sigma_0. \tag{4}$$

Here, $\Delta t = 2$ ms is the update interval, $t_e = 30$ ms and $t_a = 40$ ms are the fixed excitation and activation time constants, respectively, which are taken from van der Helm et al. [69], and σ_0 and $\dot\sigma_0$ are initial values for the activation and its derivative.[4]

[4]An idea about the magnitudes of these initial values is obtained through practical experiments, where these initial values are obtained from data for each trial, see Section 5.4.

3.1.3 Necessary Adjustments for Other Use Cases. Since we focus on the simulation of (right-handed) mid-air pointing movements of adults, we only model the upper extremity of an adult, i.e., the right arm and shoulder. However, by adjusting the MuJoCo model (which comes down to editing an XML file), e.g., to size or physique, different target user groups can be considered.

We note that, although we use a state-of-the-art biomechanical model, we do not model the complete human body. If more complex movements involving additional extremities are needed, the model needs to be augmented, respectively. For example, another arm can be added by mirroring, as has been done in [29].

Note that, in general, the interaction technique and interaction task have an influence on the choice of the user model. For example, if the input device is a handheld controller, the hand must be able to hold the controller. In particular, the (rigid) hand must be re-arranged to match the position of holding the considered device. More extensive models may contain palm and finger joints to enable fine movements. Similarly, changing the interaction task may require adjustments in the user model. If, for example, the task is to grasp and move some virtual object, the MuJoCo model would require a biomechanically more accurate model of the hand. For further details on how to adjust the MuJoCo model, we refer to the documentation provided by MuJoCo.[5]

Major changes to the user model may also affect the maximum voluntary torques, which is why, in this case, we recommend reapplying the CFAT tool described in Section 4.

3.2 Input/Output Device

In addition to biomechanics, we implement several mid-air interaction techniques. Following the scheme from Figure 1, we divide interaction techniques into their (physical) input devices (e.g., a joystick, touch screen, or motion capture system) and output devices (e.g., a monitor or HMD), and the mapping from the information that the computer receives to the virtual state that it displays.

The model of the input device should be able to realistically capture the same data from the user model as the input device captures from the real user. If, for example, a joystick senses angular movement in two axes, the model of the joystick should be able to obtain the same information. Formally, we understand an input device as a function f_{dev} that maps the user's current state $\mathbf{x}_{\text{user}}$ (e.g., body posture) and the current device state $\mathbf{x}_{\text{dev}}$ (e.g., joystick angle and/or motion capture marker position) to the updated device state $\mathbf{x}^{+}_{\text{dev}}$, i.e.,

$$\mathbf{x}^{+}_{\text{dev}} = f_{\text{dev}}(\mathbf{x}_{\text{dev}}, \mathbf{x}_{\text{user}}). \tag{5}$$

In the considered use case of mid-air pointing without any handheld device, the input device corresponds to the motion capture system PhaseSpace,[6] which allows to continuously track the movement of the user. An LED marker is placed at the tip of the right index finger, whose position is used to determine the motion of a virtual cursor. To model this input device, we use a virtual marker on our MuJoCo user model's index finger to track its position, which we denote as x_{ee}. In this particular use case, we thus have

$$\mathbf{x}_{\text{dev}} = x_{\text{ee}}.$$

Since we can obtain this data directly from MuJoCo, we do not need to implement any additional dynamics here, which would be necessary when modeling, e.g., a joystick. Therefore, the device dynamics in our case are given by the very simple mapping

$$f_{\text{dev}}(\mathbf{x}_{\text{dev}}, \mathbf{x}_{\text{user}}) = x_{\text{ee}}. \tag{6}$$

[5]https://mujoco.readthedocs.io.
[6]https://www.phasespace.com/x2e-motion-capture/.

Output modalities may also differ between interaction techniques. Most commonly, users get a visual feedback via a screen that shows how the virtual environment reacts to their input. With this information, users can evaluate their actions, e.g., through the position of a virtual cursor, and possibly change their strategy to fulfill the given task, e.g., pointing towards a virtual target. In this article, we demonstrate the simulation of mid-air pointing in VR by assuming perfect observation. This particularly implies that users always see the exact cursor position.

3.2.1 Necessary Adjustments for Other Use Cases. The MuJoCo model can easily be extended to other input devices, since many different sensors like gyroscopes as well as force, torque, or touch sensors are directly available in MuJoCo. Visual input (to the computer) can be implemented with cameras that can sense RGB pictures or just depth information. If the input device ought to be directly manipulated by the user, one needs to adjust the user model such that it can actually use the device as intended. For example, to grab and use a handheld controller, the posture of the hand would have to be adjusted to fit a controller that needs to be implemented in the same physics engine. Furthermore, for some input devices (e.g., a joystick or gamepad), fine motor finger movements are necessary, which are currently not possible with our used MuJoCo model.

To implement output devices and perception, one would need to add another layer after the *System Dynamics* in Figure 1, which maps the "real" interface state to the one perceived by the user. For example, if the output device was a 2D screen that does not allow to directly infer depth information of the regarded scene, the output device model would need to take into account the underlying projection.

3.3 Interface Dynamics

Once the human input is received, the user interface needs to be updated. For example, a change in position of the input device should entail a movement of the controlled virtual object, e.g., the virtual cursor. Additionally, the virtual world itself may have virtual dynamics. For example, throwing a virtual ball at a virtual pin may lead to that pin being knocked over.

To formalize the whole process, we use three functions. First, a *transfer function* f_{tf} transfers physical movement to virtual. As such, it depends on the current state of the input device, $\mathbf{x}_{\text{dev}}$. Next, the virtual dynamics come into play via the function f_{vd} , which takes as arguments the current state of the interface $\mathbf{x}_{\text{if}}$ (e.g., cursor or button position) and the output of the transfer function. Finally, these two components are wrapped by the wrapper function f_{if} into the Interface Dynamics of the considered interaction technique, which yields the updated virtual state $\mathbf{x}_{\text{if}}^{+}$, i.e.,

$$\mathbf{x}_{\text{if}}^{+} = f_{\text{if}}(\mathbf{x}_{\text{if}}, \mathbf{x}_{\text{dev}}) = f_{\text{vd}}\left(\mathbf{x}_{\text{if}}, f_{\text{tf}}(\mathbf{x}_{\text{dev}})\right). \tag{7}$$

Since the Interaction Dynamics is part of a *nonlinear* OCP, it is possible to include arbitrary complex virtual dynamics here (although continuity and smoothness of the functions are desirable).

The flip-side, i.e., if no explicit virtual dynamics are required, still fits in this framework. In this case, f_{if} simply is the transfer function:

$$f_{\text{if}}(\mathbf{x}_{\text{if}}, \mathbf{x}_{\text{dev}}) = f_{\text{tf}}(\mathbf{x}_{\text{dev}}). \tag{8}$$

In our case of mid-air pointing, we do not need explicit virtual dynamics and as such use (8). This is due to the fact that we only simulate single aimed movements to a static target, i.e., the only change in the interface state concerns the position of the virtual cursor, which we denote by x_{p}. This position is updated based on the transfer function that maps the (physical) end-effector position x_{ee}, which is perceived by the computer through the input device and thus is part of the input device state $\mathbf{x}_{\text{dev}}$, to the position of the virtual cursor x_{p}, which is part of $\mathbf{x}_{\text{if}}$. This leads to simple transfer functions of the form:

$$f_{\mathrm{tf}}(x_{\mathrm{ee}}) = x_{\mathrm{p}}. \tag{9}$$

But even without virtual dynamics, solely using f_{tf}, we can encompass a variety of interaction techniques.

First, we consider the class of virtual cursors in VR [50]. These simple interaction techniques introduce a displacement between the physical and the virtual hand of the user. The transfer functions for these techniques can be given in an Input-Output-Space formulation. In our case, the virtual cursor is uniquely given by an input space origin $\omega_{\mathrm{I}} \in \mathbb{R}^3$ and an output space origin $\omega_{\mathrm{O}} \in \mathbb{R}^3$. The cursor position is obtained by transferring the fingertip position in input space coordinates to the output space. The complete transfer function for *Virtual Cursor* is thus given by

$$f_{\mathrm{tf}}(x_{\mathrm{ee}}) = x_{\mathrm{ee}} - \omega_{\mathrm{I}} + \omega_{O}. \tag{10}$$

In particular, placing the end-effector at the input origin, i.e., $x_{\mathrm{ee}} = \omega_{\mathrm{I}}$, results in the cursor being at the output origin. The choice of the input origin ω_{I} influences task performance. For example, a lower input origin allows to achieve the same cursor position with a lower end-effector position, i.e., with a lowered arm, eventually resulting in more comfortable movements. To match horizontal alignment, we define the output space such that it represents a virtual 3D space in front of the user by setting $\omega_{\mathrm{O}} = (-0.1\,\mathrm{m}, 0.0\,\mathrm{m}, 0.55\,\mathrm{m})$, i.e., 10 cm right and 55 cm in front of the user.

As a slightly more complex interaction technique, we select the group of *Virtual Pad* techniques [1], which project the 3D fingertip position to a 2D cursor position. The technique can be described as using a tablet placed on a table to move a cursor on a screen in front, with the differences that the tablet is indefinitely large and that there is no need to touch it. The virtual display, i.e., the output plane on which the cursor moves, is characterized by its origin $\omega_{\mathrm{O}} \in \mathbb{R}^3$ and normal vector $n_{\mathrm{O}} \in \mathbb{R}^3$. We set this output plane to be in front of and facing the user, i.e., $\omega_{\mathrm{O}} = (-0.1\,\mathrm{m}, 0.0\,\mathrm{m}, 0.55\,\mathrm{m})$ and $n_{\mathrm{O}} = (0, 0, -1)$. An input plane is analogously defined by its origin $\omega_{\mathrm{I}} \in \mathbb{R}^3$ and normal vector $n_{\mathrm{I}} \in \mathbb{R}^3$. The cursor position is obtained in two steps: First, the fingertip is projected onto the input plane by a function $\mathrm{Proj}_{\mathrm{I}}$. Then, this point is rotated from input to output plane orientation by a function $\mathrm{Rot}_{\mathrm{IO}}$. Finally, the cursor position is translated such that it lies on the output plane. In total, the *Virtual Pad* transfer function is therefore given by

$$f_{\mathrm{tf}}(x_{\mathrm{ee}}) = \mathrm{Rot}_{\mathrm{IO}}\left(\mathrm{Proj}_{\mathrm{I}}(x_{\mathrm{ee}})\right) + \omega_{\mathrm{O}}. \tag{11}$$

Exact formulas for $\mathrm{Proj}_{\mathrm{I}}$ and $\mathrm{Rot}_{\mathrm{IO}}$ are given in Appendix A.

3.3.1 Necessary Adjustments for Other Use Cases. In the case of pointing, the presented transfer functions can easily be modified to match different interaction techniques. Modeling different pointing techniques such as ray casting [40] is also possible by adjusting the transfer function accordingly. In this case, one must add additional information of the virtual environment to the transfer function, e.g., the position and size of selectable objects. When creating new transfer functions, it is a good practice to implement them based on parameters, such as the input origin for the transfer functions used in this work. A parameter-based implementation allows for a quick adaptation and optimization of the interaction technique. Since the interface dynamics can incorporate both transfer functions and virtual dynamics, one could also model techniques where the interface has its own internal state, such as driving a virtual vehicle. In general, the virtual dynamics need to formalize the internal and mutual state dependencies of all interface objects that are relevant for the interaction, e.g., moving targets that need to be tracked by the virtual cursor, or interactable objects that may change their color or size depending on the context.

3.4 Modeling Interaction as Nonlinear Optimal Control Problem

Modeling human–computer interaction requires the use of dynamics that do not only change the state of the interface based on human input, but also capture biomechanics. Due to the redundancy of the human biomechanical system, there are infinitely many body movements that can be used to execute a given interaction task.

Building on the idea of optimal human movement control [65, 66], we assume that humans aim at behaving optimally with respect to an internalized cost function, subject to the dynamics of the human–computer-interaction system. This allows us to make use of the *optimal control framework* and rephrase the considered human–computer-interaction as nonlinear *OCP*, using an appropriate cost function as well as (discrete-time) system dynamics that describe the complete interaction loop. Formally, this can be written as

$$\begin{aligned} \min_{u(\cdot)} J_\infty(\mathbf{x}_0, u(\cdot)) &= \min_{u(\cdot)} \sum_{k=0}^{\infty} \ell(\mathbf{x}(k), u(k)) \\ \text{such that } \mathbf{x}(k+1) &= f(\mathbf{x}(k), u(k)), \quad \mathbf{x}(0) = \mathbf{x}_0, \\ \mathbf{x}(k) &\in \mathbb{X}, u(k) \in \mathbb{U}, \quad \text{for all } k \in \mathbb{N}. \end{aligned} \tag{12}$$

Here, J_∞ is the cost to minimize, which is defined by the *stage cost* or *running cost* $\ell : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ that we need to design, $f : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ is the nonlinear, continuous state transition map that takes the current state and control and yields the subsequent state according to the system dynamics, and $\mathbf{x}(\cdot)$ denotes the overall state trajectory that results from the forward simulation of the system with initial state $\mathbf{x}_0$ and control sequence $u(\cdot)$. State and control constraints are incorporated in the spaces $\mathbb{X}$ (e.g., biomechanically feasible joint angles) and $\mathbb{U}$ (e.g., maximum permissible aggregated control signal strength for each joint), respectively.

The equation $\mathbf{x}(k+1) = f(\mathbf{x}(k), u(k))$ can be written in a shorter form, analogous to the previous sections, as $\mathbf{x}^+ = f(\mathbf{x}, u)$, but we kept the current time k explicitly because it occurs in ℓ. Subsequently, we show how to apply the abstract formulation of the OCP Equation (12) to our case.

3.4.1 System Dynamics. The complete discrete-time system dynamics are obtained by combining the user model, input device, and interface dynamics that we have described in Sections 3.1, 3.2, and 3.3, respectively. These dynamics map the control signal u and the current state of the overall system $\mathbf{x} = (\mathbf{x}_{\text{user}}, \mathbf{x}_{\text{dev}}, \mathbf{x}_{\text{if}})$, consisting of user, device and interface states, to the next system state $\mathbf{x}^+$. Therefore, we can formalize the system dynamics as

$$\mathbf{x}^+ = f(\mathbf{x}, u) = \left(\mathbf{x}^+_{\text{user}}, \mathbf{x}^+_{\text{dev}}, \mathbf{x}^+_{\text{if}}\right), \tag{13}$$

where the formulas for $\mathbf{x}^+_{\text{user}}$, $\mathbf{x}^+_{\text{dev}}$, and $\mathbf{x}^+_{\text{if}}$ are given by Equations (1), (5), and (7), respectively.

In particular, in our mid-air pointing use case, the state of the complete system consists of

$$\begin{aligned} \mathbf{x}_{\text{user}} &= \begin{cases} x_{\text{qpos}}, x_{\text{qvel}}, x_{\text{qacc}} & : \text{joint angles, angular velocities, and accelerations} \\ x_\sigma, x_{\dot\sigma} & : \text{aggregated muscle activation and their derivatives,} \end{cases} \\ \mathbf{x}_{\text{dev}} &= \left\{ x_{\text{ee}} : \text{physical end-effector position, and} \right. \\ \mathbf{x}_{\text{if}} &= \left\{ x_{\text{p}} : \text{cursor position.} \right. \end{aligned} \tag{14}$$

3.4.2 Cost Function. The cost function that is assumed to be minimized by a user during interaction needs to reflect the task requirements, goals, and intrinsically motivated objectives that can represent specific user strategies. Using the notation of the OCP Equation (12), the cost function is given by a stage cost function $\ell(\mathbf{x}, u)$, which maps the state of the system $\mathbf{x}$ and the control signal u to the respective cost.

Considering mid-air pointing as an example, the task is to reach a given target with a virtual cursor. Reaching the target is often modeled as a terminal constraint. For this, however, the duration of the movement must be fixed beforehand. Since movement times vary from trial to trial and are not known in advance, they must either be estimated or calculated from experiments. However, we want to enable the simulation of human-like movements based on a model of the interaction dynamics and the user only, without relying on experimentally observed or estimated movement duration. Instead of using a terminal constraint, we thus follow a well-known approach and mitigate this problem by penalizing the distance between the cursor position $x_p \in \mathbb{R}^3$ and the target position $p^\star \in \mathbb{R}^3$ at each time step [11, 18, 51, 63]. This does not only incentivize moving the cursor towards the target, but implicitly penalizes the movement duration as well, since slow movements result in higher accumulated distance costs (note the sum in Equation (12)).

Furthermore, humans are known to prefer moving with low effort [25, 41, 65]. We implement this concept by penalizing the aggregated muscle control signal, i.e., the control vector u. As it is usually done in numerical optimization to improve the performance, we take the squared Euclidean norm, denoted by $\|\cdot\|$, in the following. If required, each element of the control vector can be scaled individually before taking the norm to replicate different effort at different joints. Since this approach would introduce additional parameters, we assume that effort is solely dependent on the normalized muscle activations. In this way, we also reward the use of stronger muscles, since, for example, the same activation in the shoulder instead of the wrist produces a higher torque with the same cost.

In addition, we introduce two different cost terms that have previously been used to model optimal human behavior. The first one corresponds to the well-established *commanded torque change* [35, 48, 71, 73], which penalizes the derivative of the commanded torques, that is, the torques that directly result from the applied motor commands. In our case, this corresponds to the derivative[7] of the applied torque τ, which we denote by $\dot{\tau}$ in the following.[8] The second, less frequently used cost term corresponds to the joint acceleration, which leads to smooth movements towards the target [71]. We denote the vector of (angular) joint accelerations by x_{qacc}, which is part of $\mathbf{x}_{\text{user}}$, see Equation (14).

With these components, we propose three different stage costs:

- **DC:** Distance and Control Costs.
  The distance between cursor and target as well as the aggregated muscle control are penalized at each time step:
$$\ell(\mathbf{x}(k), u(k)) = \|x_p(k) - p^\star\| + r_1\|u(k)\|^2 \tag{15}$$
- **CTC:** Commanded Torque Change Cost.
  This cost function adds to Equation (15) a third cost term, penalizing the commanded torque change:
$$\ell(\mathbf{x}(k), u(k)) = \|x_p(k) - p^\star\| + r_1\|u(k)\|^2 + r_2\|\dot{\tau}(k)\|^2 \tag{16}$$
- **JAC:** Joint Acceleration Costs.
  This cost function adds to Equation (15) a third cost term, penalizing the squared joint accelerations:
$$\ell(\mathbf{x}(k), u(k)) = \|x_p(k) - p^\star\| + r_1\|u(k)\|^2 + r_2\|x_{\text{qacc}}(k)\|^2 \tag{17}$$

  The *cost weights* $r_1, r_2 > 0$ define the tradeoff between the different cost terms.

[7]Due to the discrete-time setting, we take central/one-sided differences using `numpy.gradient`.
[8]We obtain τ and $\dot{\tau}$ from the activations via Equation (2).

3.4.3 Fitting Cost Weights. The choice of those cost weights has a significant impact on the resulting simulation trajectories (an evaluation of the effect of cost weights can be found in Section 6.3). To find the most appropriate weights for our cost functions, we need to evaluate different weight pairs (r_1, r_2). Since we aim at generating joint movements that are as close to human movements as possible, we evaluate a cost weight pair by comparing the resulting simulation sequence of joint angles x_{qpos} to that of a sequence of joint angles $\hat{x}_{\text{qpos}}$ obtained through a user study, considering independent joints only. More precisely, we compute the *root mean squared error (RMSE)* between simulation and experimental data, i.e.,

$$\text{RMSE}\left(x_{\text{qpos}}, \hat{x}_{\text{qpos}}\right) = \sqrt{\frac{1}{M}\sum_{k=0}^{M-1}\left\|x_{\text{qpos}}(k) - \hat{x}_{\text{qpos}}(k)\right\|^2}, \tag{18}$$

where M is the number of steps of the experimental data trajectory. To “rate” a weight pair, we sum the RMSE values for a given number S of simulated trajectories, resulting in the following loss function used for parameter optimization:

$$\mathcal{L}_{\text{param}}(r_1, r_2) = \sum_{s=0}^{S-1} \text{RMSE}\left(x_{\text{qpos}}^{(s)}, \hat{x}_{\text{qpos}}^{(s)}\right), \tag{19}$$

where $x_{\text{qpos}}^{(s)}$ denotes the simulation trajectory obtained from the cost weights r_1 and r_2, and $\hat{x}_{\text{qpos}}^{(s)}$ denotes the corresponding experimental trajectory, given a trial $s \in \{0, \ldots, S-1\}$. Since each evaluation of the RMSE Equation (18) requires solving a single OCP Equation (12) and thus results in large computation times, we decided to use a state-of-the-art derivative-free optimization algorithm that works with a low number of function evaluations and non-convex problems, and is easy to parallelize: the **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)** [27].

For evaluation purposes, we additionally compute the RMSE on state components other than joint angles, e.g., joint velocities or accelerations, as well as cursor positions, velocities, or accelerations. The respective metric is defined analogously to Equation (18), with $x_{\text{qpos}}^{(s)}$ and $\hat{x}_{\text{qpos}}^{(s)}$ being replaced by the respective quantity.

3.4.4 Necessary Adjustments for Other Use Cases. The generalized formulation as an OCP is valid for a wide range of interactions between humans and virtual objects. If the target user group or the interaction technique is varied, one has to modify the relevant parts of the system dynamics as described in Sections 3.1, 3.2, and 3.3. If an interaction task different from pointing is considered, the cost function needs to be adjusted. For example, in the case of throwing in VR, a cost penalizing the distance of a virtual ball to a target area could replace the distance cost term described above. The presented method to obtain user specific cost weights can be used for a variety of cost functions, but joint trajectories from user trials are necessary. If such data is not available, one can instead use, for example, cursor trajectories instead of joint trajectories in the loss function Equation (19).

3.5 Simulating Movements with Model Predictive Control

Since the biomechanical simulation alone has highly nonlinear dynamics, we need to solve a *nonlinear* OCP. This renders it impossible to use solvers for linear OCPs recently introduced to the HCI audience such as LQR [18, Ch. 7] or LQG [18, Ch. 8]. Solving nonlinear OCPs is generally quite challenging, and on longer time horizons, they are often computationally intractable [24]. This problem can be tackled with a receding horizon approach, also known as *MPC*. Due to its notable properties—easy to implement, handles nonlinear constraints in contrast to the LQR [12], theorems guaranteeing that MPC produces sensible results [14, 23, 55]—MPC has matured into a

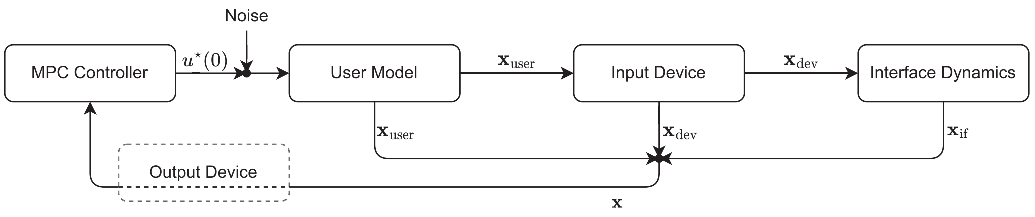


Fig. 3. High-level view on the *closed feedback loop*. The MPC Controller generates an optimal control signal $u^\star(0)$ which is perturbed by both constant and signal-dependent noise and then send to the User Model. The User Model updates the biomechanical simulation and yields a new user state $\mathbf{x}_{\text{user}}$. Based on this new user state, the Input Device then updates the device state $\mathbf{x}_{\text{dev}}$ and sends information to the Interface Dynamics, yielding the new interface state $\mathbf{x}_{\text{if}}$. All state components are then combined in the overall system state $\mathbf{x}$. The MPC Controller receives this updated state (or parts thereof) from the Output Device—closing the feedback loop.

standard control method for linear and nonlinear dynamical systems, both from the academic and application [52, 70] point of view.

The main idea of MPC is *complexity reduction* in time. The solution of the OCP Equation(12) is approximated by iteratively solving sub-problems of Equation (12) on a much shorter time horizon. The first control of the resulting optimal control sequence is then applied to the system. Iterating this process results in a closed-loop system, which is able to react to perturbations that may occur during execution (e.g., due to signal-dependent noise in the motor system [13]), without the need to handle them explicitly within each optimization step [22]. The resulting *closed feedback loop* in our framework is depicted in Figure 3.

More formally, MPC computes a *feedback law* $\mu : \mathbb{X} \to \mathbb{U}$, which maps arbitrary states $\mathbf{x} \in \mathbb{X}$ to optimal controls $u \in \mathbb{U}$, via the following MPC algorithm:

(0) Given the initial state $\mathbf{x}(0) \in \mathbb{X}$, choose the horizon length parameter $N \geq 2$ and set $n := 0$.

(1) Initialize the state $\mathbf{x}_0 = \mathbf{x}(n)$ and solve the following *open-loop*[9] OCP (we use the `scipy.optimize.minimize` from the Python `scipy` module,[10] which implements the Broyden-Fletcher-Goldfarb-Shanno (L-BFGS-B) algorithm [34, 75]):

$$\begin{aligned} \min_{u(\cdot)\in\mathbb{U}^N} J_N(\mathbf{x}_0, u(\cdot)) &= \min_{u(\cdot)\in\mathbb{U}^N} \sum_{k=0}^{N-1} \ell(\mathbf{x}(k), u(k)) \\ \text{such that } \mathbf{x}(k+1) &= f(\mathbf{x}(k), u(k)) \text{ for all } k \in \{0, \dots, N-1\}, \\ \mathbf{x}(0) &= \mathbf{x}_0, \\ \mathbf{x}(k) &\in \mathbb{X} \text{ for all } k \in \{0, \dots, N\}. \end{aligned} \tag{20}$$

Use the first value of the resulting optimal control sequence denoted by $u^\star(\cdot) \in \mathbb{U}^N$ for the feedback law, i.e., set $\mu(\mathbf{x}(n)) := u^\star(0)$.

(2) Update the state via

$$\mathbf{x}(n+1) = f(\mathbf{x}(n), \mu(x(n))), \tag{21}$$

set $n := n + 1$ and go to step 1.

We specifically differentiate between k and n to distinguish open-loop dynamics (k) from closed-loop ones (n). Design parameters include, among others, the sampling times of the state and of

[9]We call the OCP Equation (20) open-loop to emphasize that the solution of Equation (20) is not of feedback nature, i.e., cannot react to disturbances. This ability comes from the full MPC algorithm.
[10]https://docs.scipy.org/doc/.

the control. These parameters are hidden in the definition of f, which corresponds to the system dynamics (including the MuJoCo simulation in our case), and determine the resolution with which the physics are simulated, and how frequently users are assumed to be able to change their control, respectively. In order to achieve high physical accuracy, we set the sampling time of the state to 2 ms. To reflect the fact that humans are not able to adjust their behavior continuously, but only intermittently [21], we set the sampling time of the control at 40 ms (i.e., the piecewise constant control signal can be adjusted every 40 ms).

An additional design parameter introduced by the MPC algorithm is the horizon length N. Deciding on the horizon means facing a tradeoff. A longer horizon increases computation time, whereas a shorter horizon may lead to poor results. For example, if N is chosen too small, the cursor cannot be moved towards the target far enough to effectively reduce the total costs in the truncated horizon, i.e., the optimal control sequence $u^\star$ that minimizes the finite-horizon cost functional J_N does not result in the expected behavior. A more detailed analysis of the effect of N on the resulting closed-loop trajectories is presented in Section 6.4. Unless stated otherwise, we set $N = 8$ (i.e., 320 ms), as this value showed a good balance between performance and quality of simulation.

The L-BFGS-B algorithm was chosen as a solver for Equation (20) due to its computation and memory efficiency and ability to include control constraints easily. The parameters of the L-BFGS-B algorithm, which is used to solve the finite-horizon OCPs at each MPC step, are chosen as follows: objective function tolerance `ftol` $= 10^{-6}$, gradient tolerance `gtol` $= 10^{-5}$, step size for the numerical approximation of the Jacobian `eps` $= 10^{-8}$, the maximum number of objective function evaluations `maxfun` $= 10,000$, and maximum number of iterations `maxiter` $= 1{,}000$.

Previous findings suggest that human motor control signals are affected by different noise sources, e.g., sensory and motor noise [13, 28, 57, 60, 63, 68]. In order to create realistic human movements that also exhibit intraindividual variance similar to real users, perturbations can be included in the state-transition-map f. Note that, as the MPC is a closed-loop controller, we do not necessarily need to include the noise during optimization, i.e., the optimizer assumes that the system is deterministic. Instead, we include noise to the applied control $\mu(x(n))$ in step 1 of the MPC algorithm, i.e., before applying the second-order muscle model and proceeding with the next step. Applying the noise in the closed loop only considerably simplifies the OCPs and allows them to be solved efficiently. As suggested by van Beers et al. [68], we add signal-dependent and constant motor noise, i.e., two Gaussians with zero mean and a standard deviation of $0.103 \cdot \mu(x(n))$ and 0.185, respectively, to the control $\mu(x(n))$.

4 CFAT: A METHOD TO COMPUTE MAXIMUM VOLUNTARY TORQUES FOR JOINT-ACTUATED MODELS

Omitting real muscles in biomechanical models and replacing them with simplified muscles acting directly at the joints greatly simplifies computations, but it also creates another challenge. It is unclear how strong these simplified muscles need to be. Since the relative strength of each actuator has a large impact on how it needs to be actuated [32, 74], an appropriate choice of the *maximum voluntary torques* is crucial to generate biomechanically plausible movements. We, therefore, need to define the torque ranges of all actuators, i.e., the maximal positive and negative torques that can be applied at each DOF.

The natural approach to identify the torques humans apply during interaction would be to use existing Inverse Dynamics tools, as implemented in OpenSim. However, such tools obtain the complete *inter-segmental* torques acting on both independent and dependent joints, including passive forces, e.g., due to spring-dampers. In addition, the dependent joints cannot be actively actuated, but their torques emerge implicitly from the torques applied to the independent joints, i.e., the results from Inverse Dynamics cannot be used to determine the maximum voluntary torques at the independent joints.[11] Instead of relying on Inverse Dynamics, we thus apply a method similar to **Computed Muscle Control (CMC)** [62], which yields the sequence of muscle excitations that accounts for experimentally observed movements, given a fully muscle-actuated biomechanical model.

Starting with an initial posture from experimental data, the goal of our method to **compute feasible applied torques (CFAT)** is to find the sequence of applied torques that best explains the sequence of joint postures observed during an experiment. Due to the curse of dimensionality, we solve a sequence of optimization problems, one for each time step, as opposed to an optimization problem covering the entire motion, minimizing the following loss function:

$$\mathcal{L}_{\text{CFAT}}(\tau) = \alpha e_{\text{qpos}}(\tau) + \beta e_{\text{qvel}}(\tau) + \gamma e_{\text{qacc}}(\tau). \tag{22}$$

Here, the error terms $e_{\text{qpos}}(\tau)$, $e_{\text{qvel}}(\tau)$, and $e_{\text{qacc}}(\tau)$ denote the Euclidean distance between the one-step MuJoCo forward simulation with applied torques τ and the corresponding user data at this time step, in terms of joint angles, velocities, and accelerations, respectively (only incorporating the independent joints). According to our experience, penalizing an appropriate combination of joint angles, velocities, and accelerations turned out to be necessary to guarantee stability—choosing the weights $\alpha = 1{,}000$, $\beta = 50$, and $\gamma = 0.01$ showed good results in our case. After each optimization, one forward step is taken in the MuJoCo environment using the computed optimal torque. The resulting joint angles and velocities in the next time step are then used as initial values for the subsequent optimization, which returns the next optimal torques, and so on. Using this CFAT tool, we thus obtain a sequence of *applied torques* that result in the original user trajectory when sequentially applied at the DOFs of the biomechanical model. Additionally, for each trial, CFAT yields the initial activations σ_0 and their derivatives $\dot{\sigma}_0$ used in our muscle model described in Section 3.1.2.

We clean the obtained torques from outliers by removing those that deviate more than three standard deviations from the respective mean. The vectors of maximum positive and negative torques, τ^+ and τ^-, are then determined as the component-wise maximum and minimum of the computed torques τ of all considered movements.

For technical reasons, the maximum and minimum torques are normalized such that the larger of both equals one for each DOF, and the resulting values are used as boundaries for the control u.[12] The positive *scaling ratio* vector $g = \max(|\tau^-|, |\tau^+|)$, with maximum taken component-wise, is then used as a gain vector, mapping the normalized activations x_σ to the applied torques τ as in Equation (2).

It should be noted that the CFAT tool requires reference user data to measure how "human-like" a simulated joint trajectory is. In this work, we used the data from our user study, which was explicitly recorded for the considered interaction task. However, the obtained torque ranges should be appropriate for related interaction techniques and tasks as well, as was recently shown for the case of mid-air keyboard typing [29]. If major changes to the user model are made, such as modifying the physiology, running CFAT on new reference data is recommended.

5 USE CASE: ISO POINTING IN VR

As a use case, we demonstrate the applicability of our simulation framework to mid-air pointing in VR. In Section 5.1, we describe the considered task and techniques. In Sections 5.2 and 5.3,

[11] A comparison of applied torques obtained from Inverse Dynamics and CFAT can be found in the Appendix B.2.
[12] Note that, because we use the muscle model described in Section 3.1.2, the applied controls u might differ from the activation x_σ. However, given that $\Delta t < \sqrt{t_e t_a}$ holds for the second-order muscle dynamics Equation (3), the activation x_σ cannot exceed the applied controls u in absolute terms. It is thus reasonable to impose the normalized torque boundaries on u instead of x_σ.

Table 1. Interaction Techniques Used in the User Study and Simulations

Technique	Input Origin (relative to shoulder)	Input Normal Vector
Virtual Cursor Identity	$(-0.1\,\text{m}, 0.0\,\text{m}, 0.55\,\text{m})$	–
Virtual Cursor Ergonomic	$(-0.1\,\text{m}, -0.4\,\text{m}, 0.45\,\text{m})$	–
Virtual Pad Identity	$(-0.1\,\text{m}, 0.0\,\text{m}, 0.55\,\text{m})$	$(0, 0, -1)$
Virtual Pad Ergonomic	$(-0.1\,\text{m}, -0.3\,\text{m}, 0.55\,\text{m})$	$(0, 0, -1)$

All parameters are given in coordinates with respect to the right shoulder. The output origin is fixed at $(-0.1\,\text{m}, 0.0\,\text{m}, 0.55\,\text{m})$, and the output normal vector is given as $n_{\text{O}} = (0, 0, -1)$.

we proceed with a description of the user study that we conducted to collect data for the user model generation and the evaluation of our simulation. Finally, in Section 5.4, we explain how our approach can be used to replicate individual trials of the user study.

5.1 Target User Group, Interaction Techniques, and Interaction Task

Our *target user group* includes healthy adults of average size and body shape. Therefore, we do not need to make special adjustments to the biomechanical user model. Nonetheless, since we did not have user models beforehand and aim at comparing our simulation trajectories to those obtained from our user study, we derived user models that matched the biomechanical properties of the participants in the user study, as described in Section 5.3. Technically, our target user group thus corresponds to those six participants (see Section 5.2.1).

We are interested in how well our model can synthesize human movement given different *interaction techniques.* As input device, we use a motion capturing system, which tracks the position of an LED marker that is placed on the tip of the right index finger, modeled in MuJoCo as described in Section 3.2. Using the notation introduced in Section 3.3, we investigate transfer functions without any additional virtual dynamics.

For each of the two interaction technique classes *Virtual Cursor* (10) and *Virtual Pad* (11), we define a basic variant in which the input space is at the same position as the output space, i.e., $\omega_I = \omega_O$. For the virtual cursor, this means that the cursor always matches the position of the fingertip (i.e., the transfer function is the identity function), and for the virtual pad, the cursor is the orthogonal projection of the fingertip onto the input/output plane. Therefore, we refer to these techniques as *Virtual Cursor Identity/ID* and *Virtual Pad Identity/ID*, respectively. For both classes, we also consider an "ergonomic" condition, where the input space is at a lower, more comfortable height,[13] denoted as *Virtual Cursor Ergonomic* and *Virtual Pad Ergonomic* in the following. The input and output normal vectors n_{I} and n_{O}, respectively, are selected in such a way that the planes face the user and coincide for both interaction techniques. Hence, for the *Virtual Pad ID* technique, the fingertip is orthogonally projected onto a "virtual display" that is 55 cm in front of the user and on which the targets are also displayed. The *Virtual Pad Ergonomic* technique adds an additional 30 cm shift upward after the projection. Details on the input and output spaces of all considered techniques are given in Table 1.

Our *interaction task* is mid-air pointing in VR. Following the ISO 9241-9 standard, 13 targets with a diameter of 5 cm were placed on a circle of 30 cm diameter, resulting in an index of difficulty of 2.8 bits (cf. Figure 4). The center of the circle is placed 55 cm in front and 10 cm to the right of the right shoulder. We chose this placement, since most interactions with the right hand take place on the right side of the body. In each trial, the task is to move the virtual cursor as quickly

[13]In a small preliminary study, we tried different input options for both techniques, and the variants we consider here proved suitable to reach all targets comfortably.

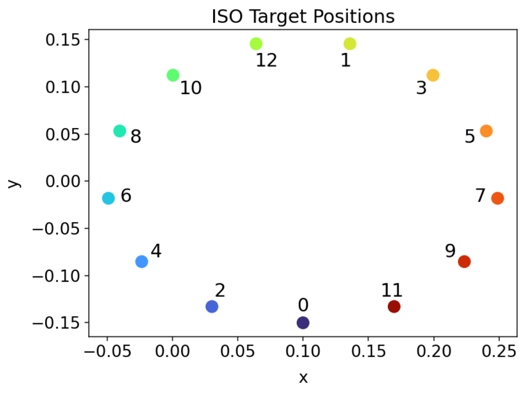


Fig. 4. The 13 targets of the ISO 9241-9 standard pointing task, which are displayed separately in ascending order. The task is to move the cursor towards the active target and keep it inside until the next target is shown.

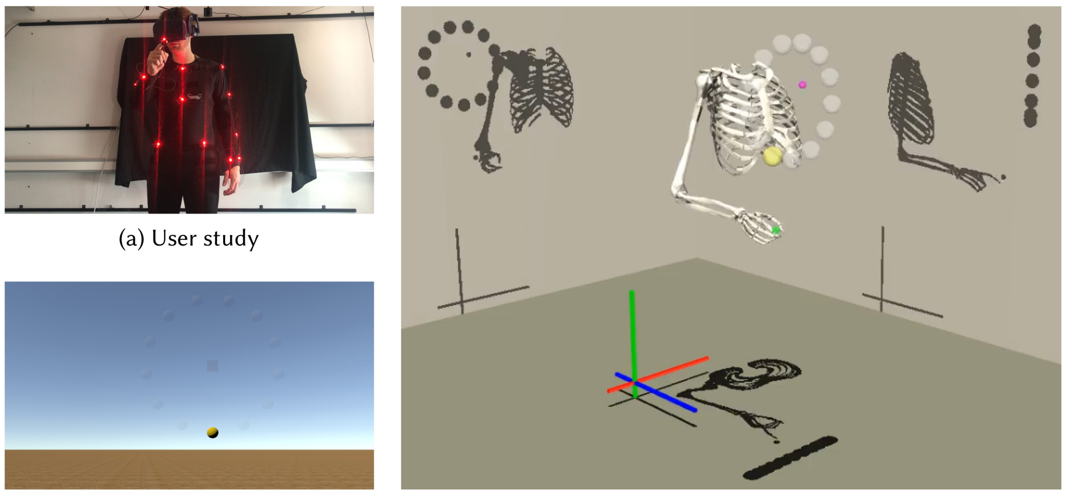


Fig. 5. **(a)** Image showing the user study. Motion is captured by tracking the movements of the LEDs. **(b)** The VR scene, which is perceived via a HMD, shows the target spheres (active: yellow; inactive: gray). **(c)** The MuJoCo simulation of the Virtual Cursor Ergonomic technique. The xyz-axes are colored in red, green, and blue, respectively. Important objects are displayed as follows. Green sphere: The virtual marker placed at the (physical) end-effector, i.e., the tip of the right index finger. Purple sphere: The virtual cursor (after applying the considered transfer function). Yellow sphere: active target. Gray spheres: inactive targets. The shadows show the orthogonal projections of the model for each dimension.

and accurately as possible toward the active target, which is represented as a yellow sphere with a diameter of 5 cm (cf. Figure 5(b)), and then hold within the target. As soon as the cursor reaches the target (with a velocity lower than 0.5 m/s to avoid early termination in case of overshoot), the next target according to the ISO 9241-9 standard is displayed after 500 ms.

5.2 User Study

We ran a user study for several reasons. First, the obtained experimental data can be used to create user-specific variants of the default biomechanical model introduced in Section 3.1. For example, in Section 4, we introduce CFAT as a tool to identify the maximum voluntary torques at each DOF, given experimentally observed user trajectories. Second, having user data allows to evaluate the quality and realism of simulated movements against observed human motion. In particular, it can be used as reference data to compare simulations for different cost functions and weights, which allows to identify the cost function parameters that best replicate observed behavior (see Section 3.4.3).

We, therefore, asked participants to perform the task described above, using the presented interaction techniques.

5.2.1 Participants. We recruited six participants (Mean Age = 28.8, SD = 6.6, 4 Male, all right-handed) from our local university campus for the study. Half of the participants had previous experience of interaction in VR, and no participants suffered from perceptual or neuromotor impairments. In the following, we refer to the different users as U1, ..., U6. All four interaction techniques are varied within subjects.

5.2.2 Apparatus and Procedure. We used a Phasespace X2E[14] motion capture system with a full-body suit to track the participants' movements at 240 Hz. The movements of the upper extremity and torso were continuously tracked by 14 optical markers placed at anatomical landmarks. Participants were immersed in Virtual Reality using a HTC Vive Pro VR headset.[15] The setup is shown in Figure 5(a). The VR scene and experimental setup were implemented in Unity3D[16] using the SteamVR plugin[17] (cf. Figure 5(b)). We aligned the coordinate systems of Phasespace and Unity as follows. We placed a Phasespace marker at the origin of a HTC Vive Pro VR controller. We then performed wanding of the interaction space using this controller, creating a set of 3D point pairs in both coordinate systems. We calculated a rigid transform between both coordinate systems using translation between the centroids to compute the translation component of the transformation, and the singular value decomposition to compute the rotation between the Phasespace and SteamVR coordinates [59].

Participants interacted with the VR scene using an end-effector marker placed at the tip of their right index finger. The movements are tracked in the Phasespace coordinate system. The cursor and target positions were only converted to the VR coordinate system right before the visualization. During the experiment, we logged the motion capture data and the experimental meta-data, as well as the timestamps at which the targets were hit.

Participants were informed about the ISO pointing task described in Section 5.1. Since we were interested in arm-only movements, participants were also instructed to only move their arm, while keeping the rest of the body as still as possible. This is important because the torso in our biomechanical model cannot move. Substantial torso movements would therefore distort the comparison between user and simulation. Nonetheless, we observed some slight torso movement in our motion capture data. For future user studies, it might therefore be helpful to use some kind of brace to keep the participant's torso stable during the task.

Since the main objective of the user study was to collect movement data for different interaction techniques, we only tested a single index of difficulty to limit the impact of fatigue. After recording a T-pose for model scaling, participants put on the HMD and performed several movements for each interaction technique. During a warm-up phase, each interaction technique was trained for at least 30 movements. Afterward, all participants performed the complete ISO task consisting of

[14]https://www.phasespace.com/x2e-motion-capture/.
[15]https://www.vive.com/de/product/vive-pro/.
[16]https://unity.com.
[17]https://valvesoftware.github.io/steamvr_unity_plugin/.

13 subsequently shown targets 5 times per interaction technique, resulting in 65 movements per interaction technique and user, or 1,560 movements in total. In order to reduce fatigue, participants were asked to take a break of one minute between interaction techniques.

5.2.3 Processing Data and Inverse Kinematics. The raw motion capture data is preprocessed according to the common conventions for biomechanical analyses [3]. The marker data is first cleaned from artifacts caused by marker occlusions and reflections based on the condition values delivered by the motion capture system, and then by filtering out outliers (i.e., segments with a difference of more than four standard deviations from the mean). The resulting gaps in the data are linearly interpolated, while keeping track of the gaps. Afterward, the data is smoothed using a Kalman filter [72], and divided into individual aimed movements using the target switch times from the experiment.

We then run the OpenSim **Inverse Kinematics (IK)** tool for each movement of any considered participant and interaction technique individually. This tool computes the joint angles for each frame of motion capture data through solving an optimization problem. To this end, it applies the kinematic constraints and freely modifies the independent joint coordinates of the model to minimize the IK loss function, which is the weighted sum of squared distances between all virtual and the corresponding experimental markers. We use a larger weight for the end-effector marker than for the other markers, as it is critical to the considered pointing task to track the end-effector as accurately as possible.

In the experimental data, the time spans between target switch and movement onset differ substantially between trials. Since we are not interested in modeling reaction times, we decided to remove these frames from user data. To this end, we determine movement onset as the time at which the acceleration of the cursor reaches 1 m/s^2 for the first time. We also removed trials that started too early (i.e., the cursor left the previous target before the new target appeared), and movements of exceptional length (i.e., the movement duration deviated more than three standard deviations from the average duration for the considered participant and interaction technique) from the dataset. In total, 158 out of 1560 recorded trials were removed, which is equivalent to 10.1%.[18]

Note that we have different time scales in simulation (2 ms) and data ($1/240$ s ≈ 4.17 ms). To be able to compare user and simulation trajectories on a moment-by-moment basis, we therefore align the two time series by applying linear interpolation on the user data.

5.3 Customized Models

In the following, we explain how the generic user model described in Section 3.1 is adjusted, both in terms of its biomechanical properties and in terms of the cost weights, which determine the tradeoff between the constituents of the cost functions introduced in Section 3.4.2.

First, we scale the models to match the kinematic and inertial properties of each participant of our user study using the OpenSim scaling tool. This tool computes ratios between pairs of markers recorded for a static posture in the experiment and the corresponding virtual markers attached to the model, only using the markers attached at the anatomical landmarks. These ratios are then used to scale the respective body segments. We ensure good quality of model scaling and marker adjustment by visually inspecting the resulting models with respect to experimental data. The scaling is then transferred from the OpenSim model to the MuJoCo model.

In addition, we adjust the joint limits to include all joint angles corresponding to the movement data of the respective participant. This is necessary because joint ranges are enforced in

[18] 114 of these trials are due to participants 2 and 5 occasionally starting their movements before the target switch.

Table 2. Joint Torque Ranges Obtained through CFAT

Joint	Torque Ranges (Nm)											
	U1		U2		U3		U4		U5		U6	
	τ^-	τ^+	τ^-	τ^+	τ^-	τ^+	τ^-	τ^+	τ^-	τ^+	τ^-	τ^+
EA	−12.74	**16.12**	−22.08	**26.12**	−14.92	**19.20**	−14.33	**18.38**	−10.99	**15.16**	−21.64	**26.73**
SE	−8.61	**20.43**	−6.91	**18.36**	−9.19	**20.92**	−7.07	**15.49**	−4.66	**17.08**	−10.05	**17.82**
SR	**−3.35**	0.70	**−4.28**	1.00	**−3.88**	0.71	**−4.03**	0.98	**−3.54**	1.37	**−5.11**	2.41
EF	0.25	**5.08**	−0.17	**5.36**	0.21	**5.88**	0.48	**5.54**	0.42	**4.81**	−0.92	**6.42**
PS	**−1.82**	1.71	**−1.36**	1.15	**−3.06**	2.73	**−0.81**	0.58	**−4.01**	3.68	**−1.42**	1.14
WD	**−2.11**	2.00	**−1.60**	1.35	**−1.98**	1.72	**−0.95**	0.57	**−1.87**	1.64	**−1.36**	1.07
WF	**−1.86**	0.78	**−1.52**	0.72	**−1.76**	0.71	**−1.24**	0.41	**−1.77**	1.02	**−1.36**	0.43

The unsigned bold values are used as respective scaling ratios g. (EA: Shoulder elevation angle; SE: Shoulder elevation; SR: Shoulder rotation; EF: Elbow flexion; PS: Pronation/Supination; WD: Wrist deviation; WF: Wrist flexion).

MuJoCo only via "soft" constraints, that is, high opponent forces are applied to postures outside the permissible region, which would reduce the reliability of the CFAT tool described in Section 4. However, it is important to note that the joint angles obtained from IK (see Section 5.2.3) are inherently dependent on the joint boundaries from the original OpenSim model (which can be found in Appendix B.1). This makes large deviations very unlikely.[19]

After scaling, we obtain the maximum voluntary torques for each user by running CFAT for all available movements. An overview of the computed maximum and minimum torques are given in Table 2. We use the initial activations σ_0 and their derivatives $\dot{\sigma}_0$ also obtained by CFAT as valid initial values for the muscle dynamics used in our simulations as described in Section 3.1.2.

To obtain reasonable cost weights, we perform parameter fitting as described in Section 3.4.3 for each user and interaction technique. That is, we identify cost weights, i.e., user strategies, that best explain observed user behavior, both in terms of general behavior and intraindividual variance. This is in contrast to previous approaches, where parameters were fitted to replicate a single user trajectory [18, 47]. To ensure computational efficiency, we create simulation trajectories for five different movement directions from the ISO task, and compute the RMSE in terms of joint angles (cf. Equation (18)) between each simulation trajectory and the respective reference user trajectory. The loss function used for the cost weight fitting is thus given by Equation (19) with $S = 5$. As described in Section 3.4.3, we use CMA-ES as a derivative-free solver. We omit motor noise during the parameter fitting, since the resulting stochastic outcome for a given set of parameters would considerably complicate the parameter search.

In cases where the optimization did not converge, we ran CMA-ES for 24 hours for each setup and took the parameter set with the lowest RMSE. The resulting cost weights for each user and interaction technique are listed in Table B.2 in the Appendix.

5.4 Simulation

Our method cannot only be used to replicate existing movements, but also to predict movements in arbitrary conditions (i.e., for different interaction techniques, tasks, and user models). To evaluate the performance of our approach, however, we need to simulate movements with the same "prerequisites" as the users in the study we are comparing to. This includes the kinematic

[19]Indeed, all user-specific joint limits were within a range of ±5 degrees around the default model values.

and inertial properties of the body as well as its initial joint configuration, which should coincide between simulation and user study.

Each aimed movement that was carried out in the user study is simulated separately. That is, for a given *reference user trajectory* (also referred to as *Baseline U1/.../U6*), we generate a corresponding simulation trajectory using the corresponding user model as described in Section 5.3 and the same interaction technique that was used in the study, i.e., we used the user-specifically scaled MuJoCo model and relevant cost weights. The simulation is shown in Figure 5(c), where our model performs a task with the Virtual Cursor Ergonomic interaction technique.

To ensure a fair comparison, we then set the torso position and orientation to that of the participant at movement onset. As mentioned above, the torso is fixed during simulation. Next, we set the initial state (including joint angles and velocities, aggregated muscle activations and their derivatives, and the virtual state of the interface, i.e., the cursor position), to the initial values of the reference user trajectory. We then synthesize the aimed movement using our MPC method. We want to emphasize that the optimization problem does *not* explicitly depend on the duration of the corresponding user movement. Instead, the receding time horizon approach allows to simulate arbitrarily long movements. Since comparing the resulting trajectories to that of the user study requires them to have equal length, we need to adjust the simulation trajectory to match the movement time of the participant in the particular trial. Therefore, the simulation stops when the movement time of the respective trial is reached.

This simulation is performed for all trials that passed the preprocessing, resulting in a total of 1,402 simulation trajectories.

6 RESULTS

In the following, we compare the ISO task trajectories resulting from our simulation to those observed during the user study described in Section 5.2. We recall from Section 5.3 that five trials from the dataset were used to fit the cost weights of the considered user and condition. In all evaluations, we exclude these five trials.

In Section 6.1, we first compare the three proposed cost functions regarding their ability to replicate and predict human movement trajectories. Using the **Joint Acceleration Costs (JAC)**, which turn out to be most suitable for simulating human pointing movements, we show in Section 6.2 that our simulation predicts user trajectories with an accuracy that is comparable to or even better than between-user comparisons, while making use of biomechanically plausible joint postures. In Section 6.3, we show that the predicted trajectories continuously depend on the choice of the cost weights r_1 and r_2, aiding the parameter optimization and paving the road to simulating new user strategies, “tailored” to some desired movement characteristics such as speed. Finally, in Section 6.4, we discuss the effect of the MPC horizon N and provide some general thumb rule on how to choose this hyperparameter.

For qualitative evaluation, we mainly focus on the following six quantities: cursor position and velocity time series, which are orthogonally projected onto the direct path between the initial and target position, as well as joint angles and velocities for both shoulder rotation and elbow flexion, as these are two of the most impactful joints for the considered mid-air movements. The angle and velocity plots of the five remaining joints are shown in Appendix B.

6.1 Comparison of Cost Functions: Joint Acceleration Costs Best Predict Human Motion

As described in Section 3.4.2, we use the following stage costs to simulate human movement in the ISO pointing task:

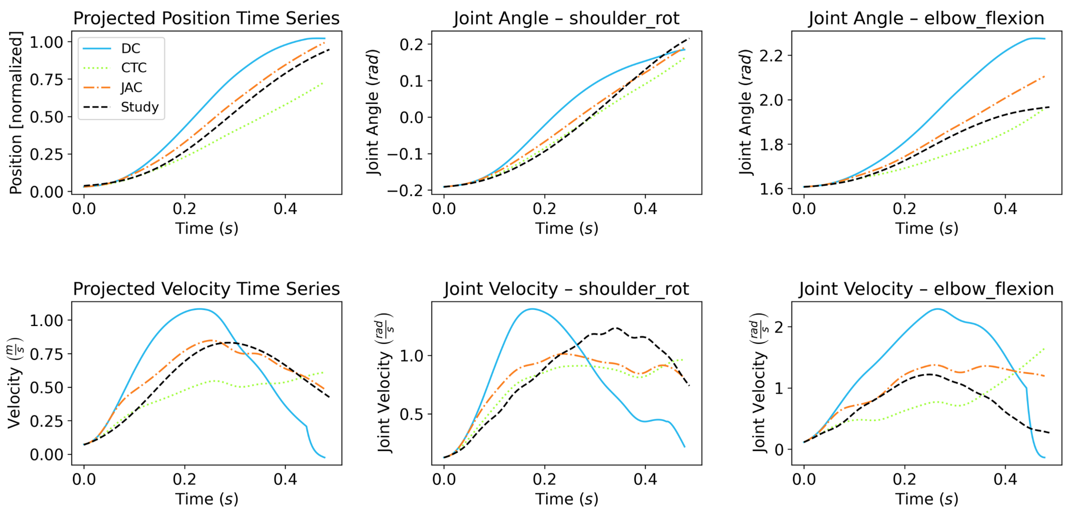


Fig. 6. Projected cursor and joint trajectories for one trial of U4 for the Virtual Pad Identity technique. The Joint Acceleration Costs (JAC; orange dashdotted lines) qualitatively explain observed user behavior best.

—**DC:** Distance and Control Costs (15),
—**CTC:** Distance, Control, and Commanded Torque Change Cost (16),
—**JAC:** Distance, Control, and Joint Acceleration Costs (17).

For each cost function, participant, and interaction technique, the respective cost weights r_1 (weight for control costs) and r_2 (weight for commanded torque change or JAC) are optimized to match joint angles between simulation and user data, as described in Section 5.3. The resulting parameter values are shown in Table B.2 in the Appendix. We evaluate the accuracy of our simulations in terms of predicted cursor and joint trajectories, both qualitatively and quantitatively.

There are clear qualitative differences between the three cost functions, as shown in Figure 6 for an example user study trial (black dashed lines; U4, Virtual Pad Identity, first movement from targets 7 to 8).

DC (blue solid lines) exhibits the highest velocities both in joint and cursor space, resulting in movements that are slightly faster than humans. The peak velocity tends to be too large, and for some trials, corrective submovements are required toward the end of the movement.

With CTC (green dashed lines), there is a considerable undershoot of the aimed target, with the cursor often not reaching the target at all within simulation time. As can be seen in the bottom left and right plots of Figure 6, penalization of commanded torque change seems to impose too restrictive constraints on the underlying joint dynamics, resulting in velocity time series of both elbow flexion and cursor that differ considerably from the typical bell-shaped velocity profiles observed in the user data.

In contrast, the simulation trajectories obtained from JAC (orange dash-dotted lines) match the human trajectories better: there are only slight differences between simulation and study in the projected cursor position and velocity profiles; it outperforms the other variants in terms of elbow flexion, and outperforms DC in shoulder rotation. Similar results can be obtained for the elevation angle, while pronation/supination as well as wrist deviation and flexion are predicted well by any of the considered cost functions (see Figure B.2 in the Appendix).

For quantitative comparison, boxplots containing the RMSEs of all ISO pointing movements for each considered cost function are shown in Figure 7, considering both cursor (top row) and joint

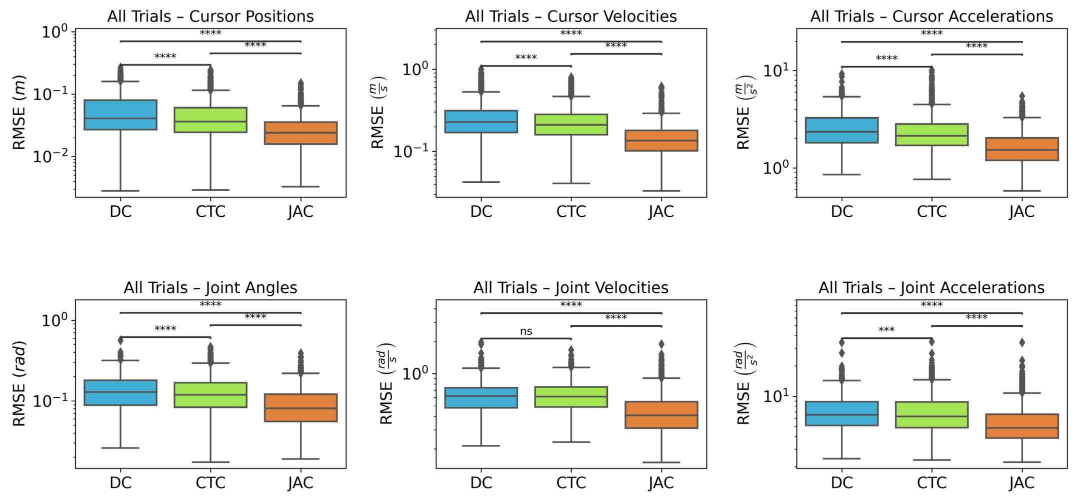


Fig. 7. Comparison of the different cost functions. The boxplots show the RMSE of all trials.

Table 3. Z-scores and p-values of the Comparisons between the Three Considered Cost Functions, Using Wilcoxon Signed Rank Tests with Bonferroni Corrections

	Cursor Z-scores			Joint Z-scores		
	position	velocity	acceleration	angle	velocity	acceleration
JAC vs. DC ($p < 0.0001$)	-22.6	-24.5	-24.9	-23.8	-26.1	-27.7
JAC vs. CTC ($p < 0.0001$)	-19.8	-21.7	-20.6	-21.4	-25.0	-25.4
CTC vs. DC	-10.8 $p < 0.0001$	-8.5 $p < 0.0001$	-8.4 $p < 0.0001$	-6.1 $p < 0.0001$	-1.4 $p = 0.17$	-3.7 $0.0001 < p < 0.001$

space (bottom row). A breakdown of the cursor position and joint angle boxplots by individual users can be found in Figure B.3 in the Appendix. Kolmogorov–Smirnov tests showed that for each of the three cost functions, none of the considered RMSE distributions fits the assumption of normality (all values $p < 0.0001$). Thus, we carried out the non-parametric Wilcoxon Signed Rank tests with Bonferroni corrections.

For the following statements, details on the results of the statistical tests are provided in Table 3. The simulation trajectories generated with JAC Equation (17) replicate the respective user study trajectories significantly better than those generated with DC. The JAC trajectories also significantly outperform the CTC trajectories in terms of RMSE. Comparing CTC to DC, some RMSE quantities yield significant differences in favor of CTC, while for others, the cost function has no or only a small significant effect.

We thus conclude that, although both CTC Equation (16) and JAC Equation (17) open the door to a better fit through an additional weight parameter r_2, by far the best results in terms of replicating observed human trajectories is obtained by JAC Equation (17), which we focus on in the following.

6.2 Simulation vs. Users: MPC is Able to Simulate User Movement in Mid-Air Pointing

We compare the movements generated by our simulation with JAC to those from the user study in terms of both projected cursor trajectories and joint postures. In particular, we show that

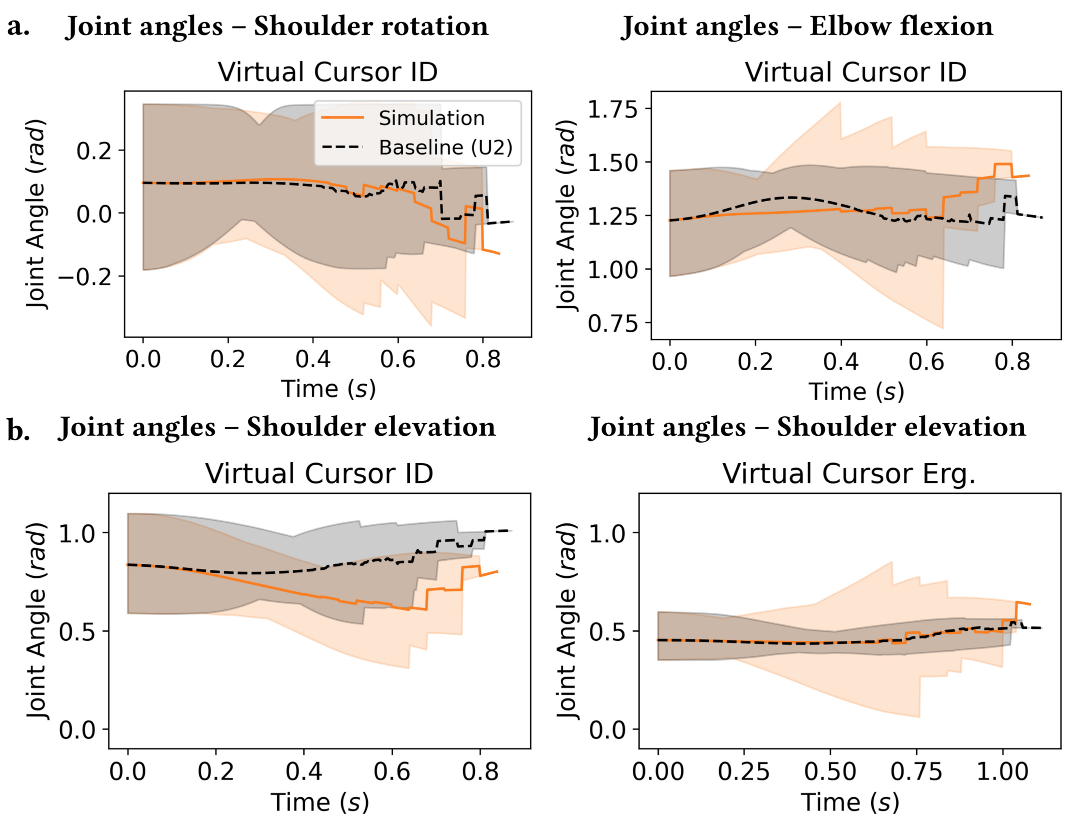


Fig. 8. **a.** The joint angle ranges predicted by our simulation for different movements in the ISO task (orange solid lines) match those observed in our user study (black dashed lines) fairly well. The mean of all movements of a single participant/user model (U2) is shown together with the entire value ranges. **b.** In the ISO task, the Virtual Cursor Identity technique (black dashed lines in left plot) requires considerably higher shoulder elevation angles than the Virtual Cursor Ergonomic technique (black dashed lines in right plot). This characteristic difference is captured well by our simulation (orange solid lines).

(1) our simulated movements exhibit biomechanically plausible joint movements,
(2) the produced cursor and joint trajectories predict human movements within between-user variability, and
(3) the method can predict motion of individual users.

(1) *Our simulated movements exhibit biomechanically plausible joint movements.* Figure 8 shows the shoulder rotation, shoulder elevation, and elbow flexion angles for one example user along with the corresponding simulation data. The user's mean angles over time (black dashed lines) are captured well by our simulation (orange solid lines) for each joint. In addition, the range of joint angles applied during any of the considered movements (black area) exhibits the same structure as in our simulation (orange area). It should be noted that these ranges only make up a relatively small portion of the admissible model joint ranges (see Table B.1). The plots for the remaining four joints are shown in Figure B.4 in the Appendix. In addition, Figure B.5 in the Appendix depicts an example simulation and user trajectories for three different movement directions showing that direction-dependent differences in the joint kinematics are also captured by our model.

There are also characteristic differences in the joint ranges when simulating different interaction techniques. Due to its shifted input space, the Virtual Cursor Ergonomic technique allows

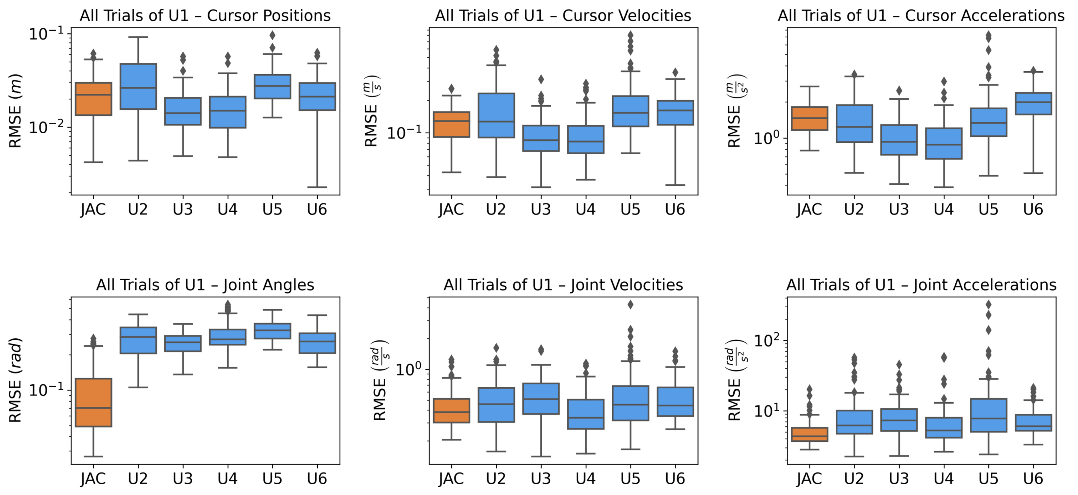


Fig. 9. RMSE between our simulation and U1 (orange whiskers), as well as between the remaining participants U2–U6 and U1 (blue whiskers), for cursor positions, velocities, and accelerations, as well as (aggregated) joint angles, velocities, and accelerations. Each whisker includes RMSE values for all interaction techniques and trials.

the participant to perform movements to arbitrary directions using considerably lower shoulder elevation angles than needed for the Virtual Cursor Identity technique, as can be inferred from the bottom plots in Figure 8 (black dashed lines, black areas). These technique-dependent movement characteristics are captured by our simulation, which predicts comparable joint ranges for both techniques (orange solid lines, orange areas).

In summary, our proposed MPC simulation is capable of generating motions that are plausible from a biomechanical perspective.

(2) *The produced cursor and joint trajectories predict human movements within between-user variability.* We argue that the movements JAC generates are within between-user variability. To this end, we first predict the movements of U1 with JAC, for all trials and interaction techniques, and compare the similarity in terms of RMSE with how well the trajectories from the remaining participants U2–U6 match those of U1. The results are displayed in Figure 9. Comparing the orange "JAC" whisker to the blue U2–U6 whiskers, Figure 9 shows that the movement trajectories generated by JAC are well within the RMSE ranges of other users, i.e., within the between-user variability. What stands out are the low RMSE values of JAC in the joint angles. It should be noted that this comparison is slightly biased because our simulation is necessarily initialized with the same joint angles as U1, while the other users might have started in slightly different postures. We extend this procedure, i.e., to infer the movements of one participant from JAC and from the respective remaining participants, to all participants, and combine the participants' RMSE values in the blue "User vs. User" whiskers in Figure 10. The plots in Figure 10 show that the results for the special case of inferring U1's movement can be extended to all participants.

We, therefore, conclude that our simulation predicts the movements of a given user not worse than other users on average, i.e., our simulation trajectories are within the between-user variability our dataset exhibits.

(3) *We can predict movement of individual users.* Besides these quantitative comparisons, we analyze how well we can predict the movement characteristics of individual users. In Figure 11, both the simulation (orange solid lines) and the corresponding reference user trajectories (black

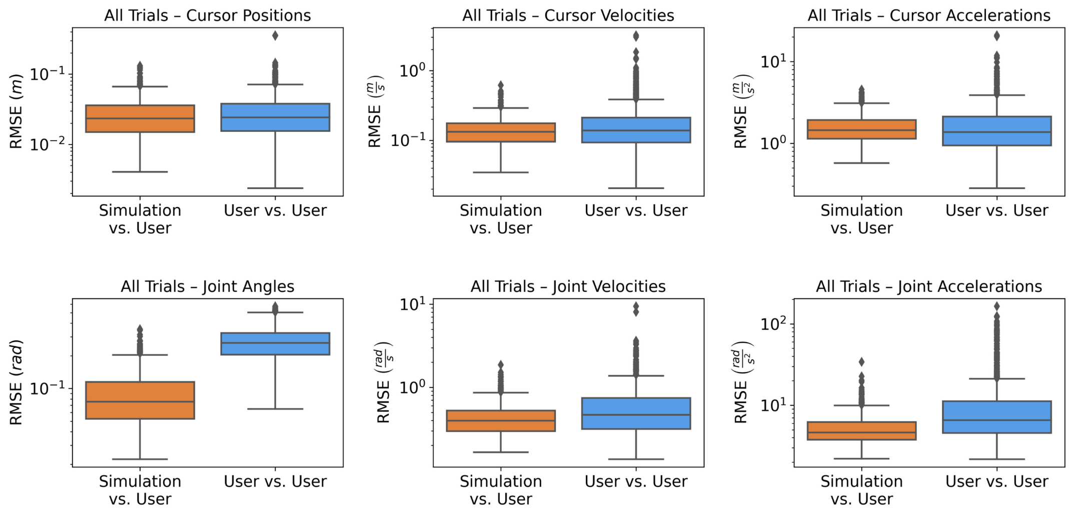


Fig. 10. RMSE comparisons, where all user study trials of a single participant (e.g., U1) are compared to either movements predicted by our simulation (*Simulation vs. User*, orange whiskers) or by the remaining five participants (e.g., U2–U6) (*User vs. User*; blue whiskers). In contrast to Figure 9, the trials of all six participants are once used as baseline.

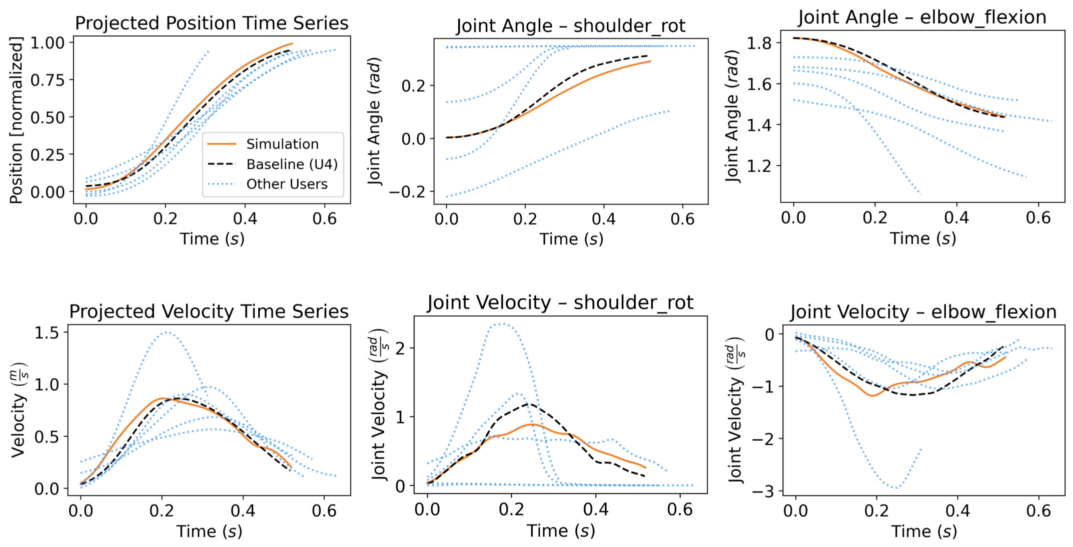


Fig. 11. Given an interaction technique (here: Virtual Pad Ergonomic) and a movement direction (here: movements from targets 1 to 2), the characteristic cursor and joint trajectories of an individual user (here: U4, black dashed lines; trajectories of the remaining users are shown as blue dotted lines for comparison) can be predicted by our simulation (orange solid lines).

dashed lines; for details, see Section 5.4) are shown for an example trial from the user study (U4, Virtual Pad Ergonomic, third movement from targets 1 to 2). We also show the respective trajectories of the remaining users for this trial (blue dotted lines). Figure 11 (left column) shows that we can match the characteristic projected position and velocity time series of a specific user. Moreover, the target is reached within a single ballistic movement, and the velocity time series exhibit

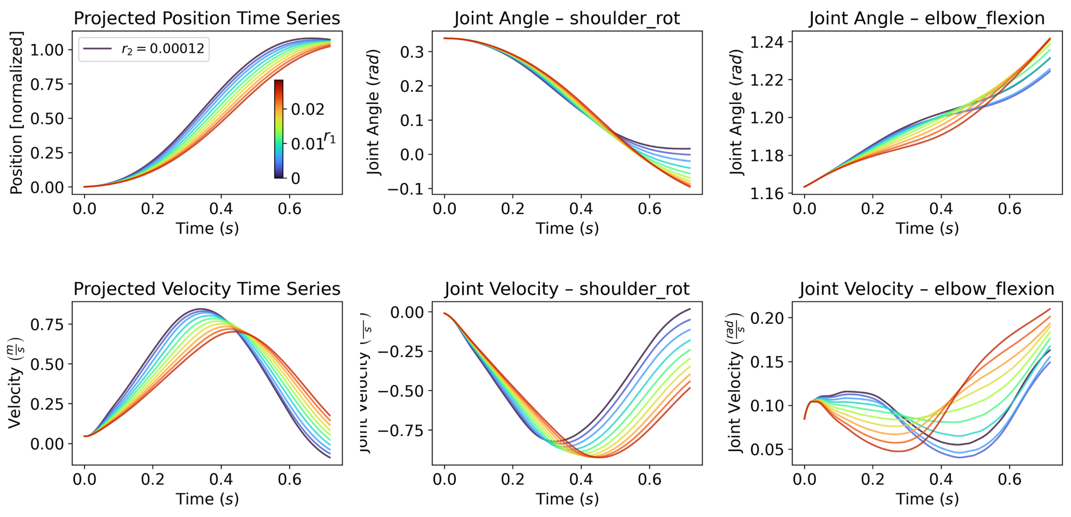


Fig. 12. Trajectories of the simulation for different cost weights r_1. It is clearly visible that an increase of r_1 results in slower, delayed movements toward the targets. Joint patterns are also (slightly) affected by the choice of r_1.

the bell-shaped velocity profile typically observed in aimed movements [45]. Figure 11 (middle and right columns) illustrates how the simulation is able to distinguish the baseline user from the remaining participants in terms of shoulder rotation and elbow flexion angles and velocities. Similar results can be observed for the remaining joints and interaction techniques, as shown in Appendix B.4.

In summary, these findings show that our simulation generates biomechanically plausible joint postures, it is capable of predicting human trajectories that are within the between-user variability of the respective interaction technique, and it allows to replicate characteristic movement patterns of individual users.

6.3 Effects of the Cost Weights

To be able to replicate characteristic movement patterns of individual users, we need to determine the cost weights r_1 and r_2 in JAC (cf. Equation (17)). To understand how much the simulation results depend on these weights, we show in this section how the trajectories change when we alter the weights, and argue that these insights can be used to generate new user *strategies*.

The most important insight is that the movement trajectories exhibit a *continuous dependence* on r_1 and r_2, i.e., if r_1 and/or r_2 change only slightly, the movement trajectories (of cursor and joints) also change only slightly. With this, we provide indicators how the cost weights should be changed in order to generate, e.g., slower movements. To this end, we start by analyzing the effects of r_1 and r_2 on the movement trajectories.

To avoid distorting the effects, we omit the motor noise from all simulations in this section.

In Figure 12, projected cursor and joint trajectories are shown for 10 different values of the control cost weight r_1, with constant joint acceleration cost weight $r_2 = 0.00012$ (U4, Virtual Cursor Ergonomic, first movement between targets 4 and 5). There is a clear decrease in peak velocity, as r_1 increases, resulting in considerably slower movements with target reached later. Moreover, the projected cursor velocity profile becomes more right-skewed, which can be explained by the

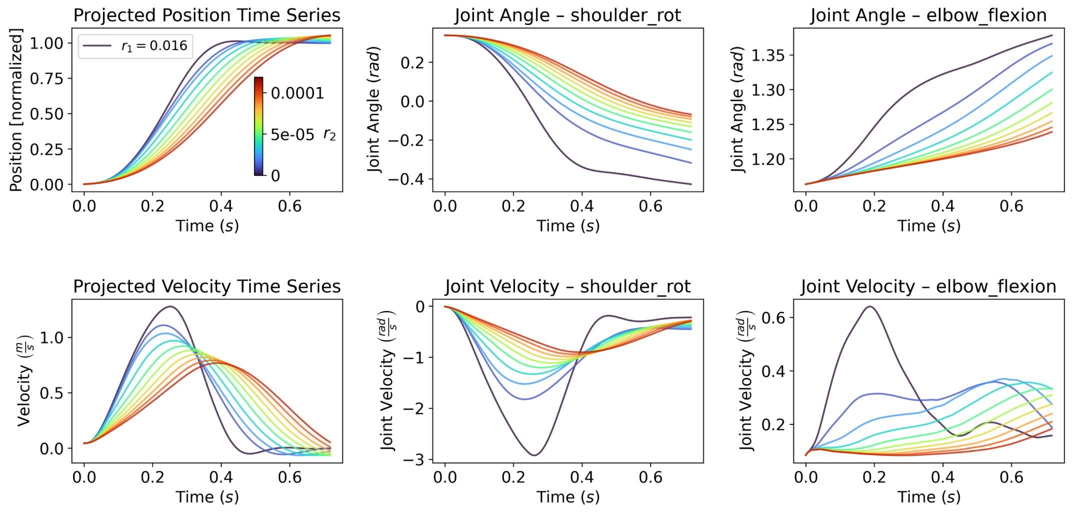


Fig. 13. Trajectories of the simulation for different cost weights r_2. The clear cursor and joint profiles that can be observed for $r_2 = 0$ (black lines) become more and more smooth as r_2 increases, resulting in a similar effect as observed for r_1.

increased incentive to apply lower torques per time step (note that the penalization of squared control signals incentives the use of multiple small control signals instead of one large control signal). This suppressive effect of increased control costs can also be observed in the shoulder rotation plots, where larger r_1 values result in considerably less negative joint velocities at the beginning of the movement, which need to be compensated later in order to reach the target. The elbow flexion is also affected by changes in the control costs weight, however, with comparably small impact (note the considerably smaller joint angle range). Note that for very large r_1 values, the only relevant objective is to reduce the control cost, i.e., it is optimal to apply no controls during the entire movement, which causes the arm to fall.

In Figure 13, the effects of the JAC weight r_2 are shown for the same participant, interaction technique, and trial, using a fixed intermediate control costs weight $r_1 = 0.016$. The effect on projected cursor trajectories is qualitatively comparable to that of r_1, albeit considerably more pronounced. Without JAC, i.e., $r_2 = 0$ (black lines), the projected velocity time series exhibits a very high peak velocity that is compensated by a corrective submovement starting after $\sim$ 450 milliseconds, resulting in a very fast movement towards the target. For shoulder rotation and elbow flexion, the changes in both joint angles and velocities strongly decrease as r_2 increases. In particular, the characteristic joint patterns that can be observed for $r_2 = 0$ (black lines) become more and more flattened as joint accelerations are more penalized.

The effects of the two cost weights on the remaining joints are depicted in Figures B.11 and B.12 in the Appendix, respectively.

To examine how well a certain cost weight pair fits one specific user, the surface plot in Figure 14 shows a simulation vs. user comparison (U4, Virtual Cursor Ergonomic) for different combinations of cost weights. Here, the z-axis shows the mean RMSE used to measure the similarity between simulation and user data trajectories (see Section 3.4.3) with respect to cursor position across all trials of U4 using the Virtual Cursor Ergonomic technique. As can be seen, the surface is clearly convex. While convexity of the RMSE with respect to the cost weights can neither be guaranteed nor expected in general, it ensures that there exists a unique global optimum, towards which most

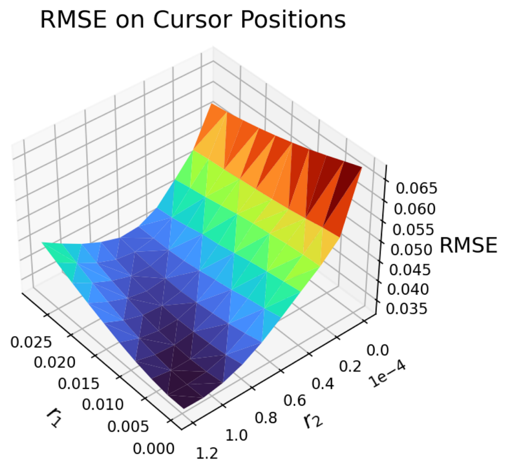


Fig. 14. Surface plot showing the performance of the simulation with different cost weights r_1 and r_2. The z-axis corresponds to the mean RMSE on cursor positions across all trials of U4 using the Virtual Cursor Ergonomic technique.

OCP solvers (including the one we use) are guided. In the considered case, the global optimum is located at $r_1 = 0$ and $r_2 \approx 1.1e-4$ (black colored valley).[20]

In summary, our analysis of cost weight effects shows that both the cursor and joint trajectories predicted by the closed MPC loop continuously depend on the cost weight parameters. This has two major advantages:

First, the parameter fitting process in finding cost weights that best reflect specific user behavior exhibits a certain robustness. This facilitates computing optimal weights and ensures that cost weights close to the optimum will already provide good results. Second, it allows to generate new user strategies by tweaking the cost weight parameters.

6.4 Effects of the MPC Horizon

To better understand the impact of the MPC horizon N, which may need manual adjustments depending on the task under consideration, in the following, we analyze how the simulated movements change with increasing horizon.

We also omit the motor noise from all simulations done in this section to avoid distorting the effect of N.

As depicted in the top left plot of Figure 15 for the same trial as used in Section 6.3, the MPC horizon N, which determines how many future steps are taken into account to select the control at a certain time step, has a considerable effect on the resulting closed-loop trajectories. Choosing N too small results in movements that are either too slow to reach the target at all ($N = 2$, black line), cross the target but do not return within a reasonable time ($N = 3, 4$, blue lines), or exhibit some considerable overshoot ($N = 5$, grass green line). Starting from $N = 8$, the differences in cursor and joint trajectories are quite small and hardly visible anymore. This suggests that a prediction horizon of $8 \cdot 40$ ms $= 320$ ms is sufficient to adequately solve our OCP via MPC.

This is also confirmed by the quantitative comparison of MPC horizons shown in Figure 16, where we again computed the mean RMSE on cursor positions, considering all trials of U4 using

[20]Note, that this does not exactly coincide with the cost weight pair found by our fitting process. This is due to the facts, that during the fitting, we use RMSE based on joint angles, and that out of computational reasons, we only considered five movements to evaluate a cost weight pair.

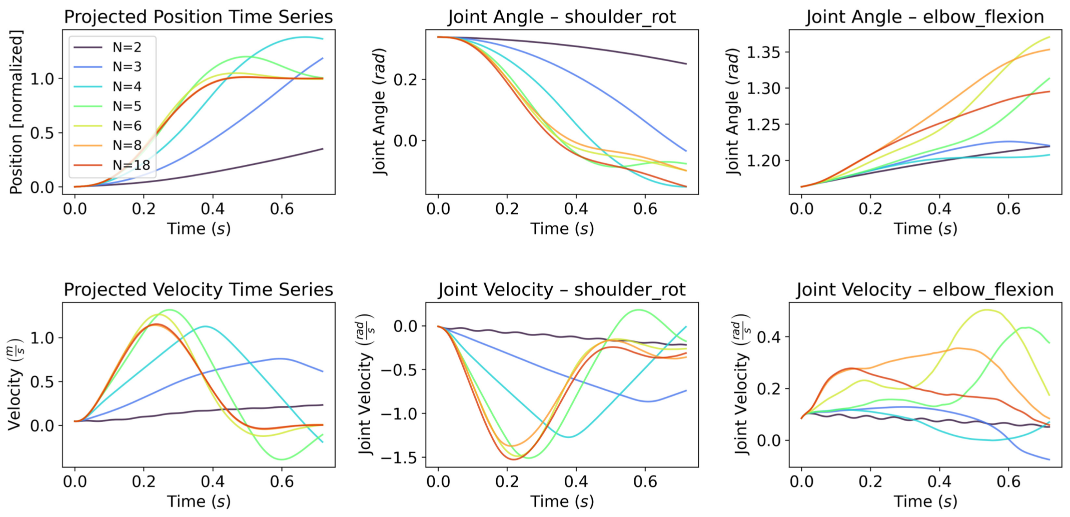


Fig. 15. Projected cursor and joint trajectories of one trial for varying MPC horizon N, using the JAC, without control noise.

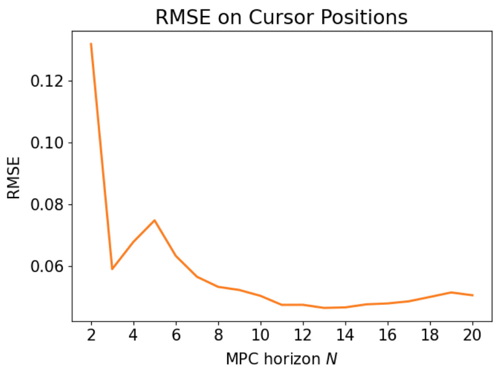


Fig. 16. Effect of the MPC horizon N on the mean RMSE between JAC and user data in terms of cursor positions, considering all trials of U4 with the Virtual Cursor Ergonomic technique. The performance clearly deteriorates when the MPC horizon N is set too low (i.e., below $N = 8$).

the Virtual Cursor Ergonomic technique, for different MPC horizons. The performance of our simulation clearly deteriorates when the MPC horizon N is set too low, i.e., $N \leq 6$. Interestingly, user trajectories are best explained by an intermediate MPC horizon of $N = 13$, while a too large horizon ($N \geq 16$) results in slightly worse mean RMSE for this interaction technique. Since the computational time exponentially increases with N, we decided to use the lowest MPC horizon replicating human behavior sufficiently well for our simulations, that is, $N = 8$.

7 DISCUSSION AND LIMITATIONS

Our framework allows to simulate interaction movements with optimal control (cf. Figure 1). It is applicable to a wide range of interaction techniques and can be adjusted to simulate different individual users. In order to help the HCI community to explore the simulation of interaction with MPC, we provide our Python code as open source at https://github.com/mkl4r/sim-mpc. In this section, we discuss how our framework can be applied, what issues should be paid particular attention to, and interesting ideas for future work.

In order to help using the framework, Section 3 discusses possible adjustments of the individual parts of the framework, and gives some ideas on how to tailor these to specific interaction techniques, users, and conditions. Moreover, we provide an example-based step-by-step guide on how to apply our framework to novel tasks in Section 8.

One point to consider is that in our simulations, the torso is fixed in space. This is obviously only a reasonable assumption when simulating interaction techniques that do not require substantial torso movement. Enabling natural torso movement is less trivial than one might think. One solution is to provide multiple "virtual" joints for the added six degrees of freedom of the torso. However, according to our experience, such virtual joints can easily lead to unnatural movements. Alternatively, the lower body can be integrated in the biomechanical simulation, leading to the added task for the controller of balancing the body.

Our presented framework assumes complete observation, i.e., complete knowledge of the system state (e.g., joint angle and cursor position). In reality, humans observe their environment, e.g., through proprioception or visual perception, and have to deal with limited information, e.g., they have to *estimate* the target position from an image on a screen. To control such a system with imperfect observation, it would be an interesting future task to add an *observer*.

Another aspect is that we only investigated movements to a single, relatively big, target size, which do not require extensive corrective submovements. If one is interested in submovements and movement times for different target sizes, MPC is certainly an interesting way to model these. Our model exhibits some submovements due to the interplay of a limited MPC horizon and movement noise. A promising direction to increase the realism of submovements, and model the change of movement times in response to different target sizes would be the introduction of an observation delay, an observer to account for both motor and sensory noise, and including perception of the target size. All of these are compatible with and can be integrated in the MPC framework.

While introduced in general terms, we apply our SimMPC framework to the use case of mid-air pointing throughout the article. In the future, it would be interesting to evaluate movements that our framework predicts for different interaction techniques and tasks. Similarly, our analysis of between-user variability was based on data from a relatively homogeneous user group (all right-handed adults without impairments and with similar backgrounds). However, since our approach explicitly provides the possibility to capture user-specific characteristics (e.g., using scaled MuJoCo bodies or maximum voluntary torques obtained by CFAT), comparable fits between *given* user data and our simulation can be expected for rather diverse target groups.

7.1 Why MPC?

We have seen in Section 6.4 that, once a cost function has been chosen (in our case, JAC), the choice of the MPC horizon N is *the* key governing factor. In practice, some trial and error is needed, but also possible here, to control the tradeoff between computation time and quality of the movement trajectory. To this end, we have provided practical guidance in Section 6.4. Once we have such an N, then MPC "works" for this application.

In comparison to some other models of interaction, MPC is particularly interesting, because it allows for an analytical understanding of the dynamics of the interaction loop. MPC has been used successfully for other applications; it is no coincidence that it is performing well in this particular setting. In this article, we have relied on user data and performed an extensive analysis on the performance of MPC. However, if one wants to very quickly check whether the effort of using MPC will be worthwhile in their application, a good indicator is the so-called *turnpike property*. It states that the computed optimal trajectories remain close to the so-called *optimal steady-state*, most of the time. As the name suggests, it is the optimal state to remain in with respect to stage costs ℓ. In our JAC case, it is determined by the state costs (e.g., keeping the cursor near the target)

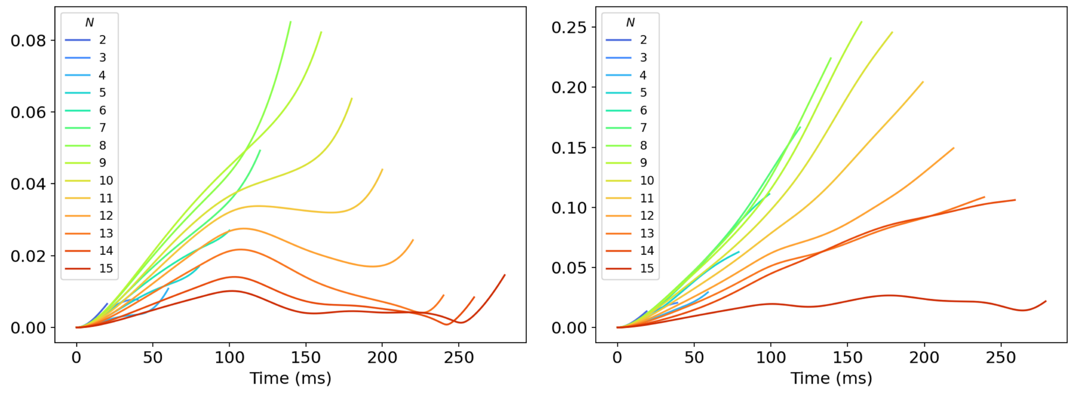


Fig. 17. Euclidean norm between the cursor position (left) and the aggregated joint angles (right) of open-loop trajectories with different MPC horizons N and a reference open-loop trajectory for $N = 16$. The *Turnpike* property is reflected in the fact that the trajectories stay longer in the vicinity of the reference as the horizon N increases. All simulations use the same trial of U4 and the Virtual Cursor Identity technique (defined in Section 5.1).

and control or effort costs (e.g., using a joint posture in the process that is not strenuous or tiring). Usually, this optimal steady-state is not known analytically, so the first step is to approximate it numerically. This can be done by solving the OCP Equation (20) over the time frame of interest. Then, instead of only applying the first value of the resulting optimal control sequence as in the MPC algorithm (Section 3.5), we compute the so-called *open-loop optimal trajectory* by simply applying the full optimal control sequence. For this purpose, the parameter N should be chosen as large as possible—in our following example, we have chosen $N = 16$. Then we compare this result to open-loop optimal trajectories for smaller N, by plotting their difference to the reference trajectory (large N). This is visualized in Figure 17.

Figure 17 shows the typical turnpike behavior: First, for all N the corresponding open-loop trajectories at some point start deviating more and more from the reference trajectory. Second, this happens later and less pronounced the higher N is. The turnpike property is an essential part in theorems that guarantee that MPC produces "sensible" results, i.e., MPC "works" [15]. This assurance provides trust and reliability, is one of the great strengths of MPC, and is the reason why it is used, e.g., in industry applications [52].

Similar guarantees exist for other controllers, such as LQR or LQG used, for example, in Reference [18]. However, unlike MPC, these cannot handle nonlinear system dynamics, which regularly occur in HCI due to nonlinear. Possibly even more important for HCI, they cannot directly handle state and control constraints, but have to resort to soft constraints via penalty terms in the cost function. These constraints include joint angle and/or torque limits imposed by human biomechanics, body position constraints when interfered by or interacting with physical objects and devices (e.g., desk, gamepad, or monitor), and boundaries on the location of the virtual objects (e.g., due to fixed bounding boxes of application windows or limited screen size).

If, on the other hand, guarantees take a back seat, RL most certainly enters the picture. RL is very powerful, can handle nonlinearities and even state and control constraints, and thus is used in many applications. In particular, RL can be used to control a biomechanical model as well [16]. However, computationally it can be very demanding to compute a single optimal movement. This is because RL learns a *policy* to then sample movements.[21] This policy may have to be completely

[21]We note that getting the policy can be advantageous depending on the use case.

retrained if the interaction technique is changed, e.g., a different input origin for the Virtual Pad is used. In contrast, MPC acts as a complexity reducer in time: instead of directly solving an optimization problem for the full movement time, in MPC, we consider subproblems on smaller time horizons, which are much easier to solve due to exponentially increasing complexity the longer the movement lasts.

From our point of view, it is therefore promising to combine the respective strengths of MPC and RL (or, more generally, Machine Learning techniques) in future applications. Following the current trend, MPC is more and more augmented by data-driven techniques (especially from the control theory community). To put it in the words of Benjamin Recht (UC Berkeley, "Reflections on the Learning-to-Control Renaissance" Keynote at the 2020 IFAC World Congress): *I still remain baffled by how Model Predictive Control is consistently underappreciated. [...] there are really very few people, especially in the Machine Learning community, who are trying to analyze why MPC is so successful, especially when it's coupled with Machine Learning.*

8 APPLYING THE SIMMPC FRAMEWORK STEP-BY-STEP

This section gives a step-by-step guide on how to use our framework for other interaction tasks or techniques. In the following, we show how to simulate the task of tracking a moving target using ray casting with a handheld VR controller. In our example, the task is to track a sphere that is constantly moving in a circle in front of the user. The approach consists of the following steps:

(1) Define user (group) and model,
(2) implement the input/output device in MuJoCo,
(3) define and implement the interface dynamics,
(4) choose the objective function (and optimize cost weights),
(5) run simulations, and,
(6) obtain trajectory data and summary statistics.

(1) We start by defining the user group and model, see Section 3.1. Formally, the model corresponds to f_{user} in Equation (1). If the goal is to generally evaluate an interaction task/technique, using one (or several) of the six provided user models of healthy humans is sufficient. If one aims to build an interaction technique for a more specific user (group), we recommend to first collect some movement data in a preliminary study, and use this data to adjust the model scaling and maximum voluntary torques as described in Section 5.3. This data does not have to stem from the task under consideration, but should contain similar movements.

(2) Next, we need to implement the input/output devices in MuJoCo, see Section 3.2. Formally, this step corresponds to defining f_{dev} and $\mathbf{x}_{\text{dev}}$ in Equation (5). In our case, we add a body to the MuJoCo model (more specifically, to the hand of the model) that corresponds to a handheld controller, with appropriate mass and inertial properties. Next, we attach a marker to the controller, which is used to infer its position and orientation (i.e., we define the physical end-effector x_{ee}). The marker should be placed such that this information match those provided by the real controller. Adding a 3D mesh and adjusting the rigid finger positions such that the hand is actually holding the device is optional but can help to place this marker correctly, improves visualization, and is recommended when collisions are expected, e.g., if the interaction space is physically limited. We define the state of the input device $\mathbf{x}_{\text{dev}}$ as the position of the (physical) end-effector x_{ee} and its orientation using a direction vector n_{ee}. The function f_{dev} defines how we obtain those values from the MuJoCo simulation, i.e., using the position of the marker as x_{ee} and calculate n_{ee} using the marker's orientation. Note that any physical action that is to have an effect on the virtual environment must be modeled in the state of the input device (e.g., button press events could be modeled using a boolean entry in the state).

(3) In the third step, we implement the interface dynamics, see Section 3.3. One part of the interface state $\mathbf{x}_{\text{if}}$ is the current position of the sphere x_{s}. The movement of the sphere is defined by the interface dynamics f_{if}. Using the position and orientation of the controller given by $\mathbf{x}_{\text{dev}}$ from the previous step, we can calculate the distance x_{d} between the ray that is cast from the controller and the center of the sphere. Note that we do not explicitly need a transfer function f_{tf} here, since we can use the position and orientation of the controller directly. The complete interface state is then given by $\mathbf{x}_{\text{if}} = (x_{\text{s}}, x_{\text{d}})$. This concludes the system dynamics of the OCP in Equation (13) that we want to solve.

(4) Next, we define the stage costs for each time step that depends on the state of our system and the control. For pointing, we suggest using a combination of distance, control, and JAC (JAC, see Equation (17)). While the distance costs reflect the actual task objective, adding control costs is generally recommended for two reasons. First, these costs create an incentive to execute the movement with as little effort as possible, and second, they regularize the optimization procedure from a mathematical point of view. When designing stage costs for tasks other than pointing, we recommend using continuous and convex functions as they lead to a better optimization performance.

In the considered case of tracking with ray casting, we decide for a combination of the squared norm of both the control and the joint acceleration vectors plus the norm of x_{d} since the goal is to keep the ray inside the sphere.[22] The stage cost is therefore given by

$$\ell(\mathbf{x}(k), u(k)) = \|x_{\text{d}}(k)\| + r_1\|u(k)\|^2 + r_2\|x_{\text{qacc}}(k)\|^2,$$

with cost weights $r_1, r_2 > 0$.

Since we use a weighted combination of different cost terms, we need to find suitable cost weights. In the considered case of tracking, we expect scaling both r_1 and r_2 relative to the modified distance costs to be sufficient to achieve reasonable results. For more severe changes to the task, or if reference user data is already available (e.g., from a preliminary study), running a parameter optimization as described in Section 3.4.3 is recommended to improve the simulation results.

(5) Before running the simulation, the MPC horizon should be chosen long enough for the considered task (see discussion in Section 6.4). Additionally, a *termination criterion* should be defined, which determines when to stop the simulation. Here, this can be directly inferred from the instructions, i.e., the task is to stay inside the target sphere for a certain amount of time. Alternatively, the simulation can also be run for a certain time, independent of the completion of the task. Finally, to be able to run a simulation, we need to define the initial state of the system. This consists of an initial posture of the biomechanical model (i.e., joint angles and velocities), initial activations of the muscles and their derivatives and the initial state of the input device and interface. This initial state can either be obtained from a user study or by running preliminary simulations with arbitrary initial states.

(6) The output of the simulation includes trajectories for all state variables such as joint angles, angular velocities, or position of virtual objects. In particular, all quantities obtained by a typical motion-capture based user study are easily available. To obtain summary statistics, we can include noise and run multiple simulations to collect an entire set of trajectories reflecting the predicted trial-to-trial variability for the task under consideration.

9 CONCLUSION

We have presented SimMPC, a framework to simulate movements during interaction with computers, combining biomechanical modeling with MPC. It allows to predict kinematic and dynamic quantities of both physical and virtual objects, including cursor positions, joint angles and velocities, and aggregated muscle recruitments, on a moment-by-moment basis, providing richer

[22]While we do not explicitly include the size of the sphere in the cost function, it can be used in the later evaluation.

information than summary statistics. Our framework allows easy combination of biomechanical models of the human body with arbitrary interaction techniques and tasks, which was not possible with existing approaches that relied on linear optimal control methods. As a use case, we have applied our approach to a joint-actuated state-of-the-art model of the upper extremity, and considered four mid-air interaction techniques for the task of ISO pointing in VR. We have shown that MPC is able to simulate user movement in mid-air pointing both in terms of cursor and joint trajectories, with an accuracy within observed between-user variability. Comparing three different cost functions, the combination of distance, control, and JAC was shown to best explain the movements observed in our user study.

In a practical sense, our approach allows to predict movement of a given individual user, and can be used to generate new user strategies. Moreover, we provide advice on how to apply our framework to simulate interactions for different target user groups, interaction techniques, or interaction tasks. We have also introduced *CFAT*, a novel tool to compute the applied torques underlying a given joint angle sequence, which we have used to infer the maximum voluntary torques that were applied in the considered ISO pointing task. We have made our code and data publicly available (https://github.com/mkl4r/sim-mpc).

The combination of biomechanical simulations with MPC could open the door for online optimization and customization of user interfaces and interaction techniques, based on the predictions of a "digital twin" simulation running in the background. The ability to evaluate the entire interaction loop between humans and computers in terms of efficiency or ergonomics with the help of realistic simulations could allow for partial replacement of costly and time-consuming user studies in the future. We, therefore, believe that the optimal control perspective on interaction and its simulation via MPC provide an interesting interpretation of and useful tools for Human–Computer Interaction.

APPENDICES

A VIRTUAL PAD TRANSFER FUNCTION

To obtain the cursor position of the Virtual Pad, we have to project the end-effector position x_{ee} onto the input plane and transfer it onto the output plane. The projection onto the input plane is given by

$$\text{Proj}_{\text{I}}(x_{\text{ee}}) = x_{\text{ee}} - ((x_{\text{ee}} - \omega_I) \cdot n_I)\, n_I, \tag{A.1}$$

where the operator $\cdot$ denotes the inner product, ω_I is the input space origin, and n_I is the input space normal. Note that, while the projected position could be denoted in 2D coordinates of the input plane, the projection maps onto the position in global 3D coordinates to be able to transfer this point correctly onto the output plane. To rotate this new point correctly, we compute the rotation matrix R that rotates the input normal vector n_I such that it equals the output normal vector n_O:

$$R = (1 - C)aa^\top + \begin{pmatrix} C & -a_3\overline{C} & a_2\overline{C} \\ a_3\overline{C} & C & -a_1\overline{C} \\ -a_2\overline{C} & a_1\overline{C} & C \end{pmatrix}, \tag{A.2}$$

where a is the rotation axis, i.e.,

$$a = (a_1, a_2, a_3)^\top = \frac{n_{\text{I}} \times n_O}{\|n_{\text{I}} \times n_O\|}, \tag{A.3}$$

$aa^\top$ is the *outer product* of a with itself, and C and $\overline{C}$ are the cosine and sine of the angle between the normals, i.e.,

$$C = \frac{n_{\text{I}} \cdot n_O}{\|n_I\|\,\|n_O\|}, \quad \overline{C} = \sqrt{1 - C^2}. \tag{A.4}$$

Since rotation is defined around the global origin, we subtract ω_I before rotating, i.e., for some point $y \in \mathbb{R}^3$:

$$\mathrm{Rot}_{\mathrm{IO}}(y) = R(y - \omega_I). \tag{A.5}$$

As the last step, we need to translate back onto the output plane by adding the output origin ω_O. The overall transfer function is thus given by

$$f_{\mathrm{tf}}(x_{\mathrm{ee}}) = \mathrm{Rot}_{\mathrm{IO}}(\mathrm{Proj}_{\mathrm{I}}(x_{\mathrm{ee}})) + \omega_{\mathrm{O}} = R(x_{\mathrm{ee}} - ((x_{\mathrm{ee}} - \omega_I) \cdot n_I)\, n_{\mathrm{I}} - \omega_I) + \omega_O. \tag{A.6}$$

B SUPPLEMENTARY FIGURES AND TABLES

B.1 MuJoCo Model

Table B.1. Default Joint Angle Ranges of the Upper Extremity Model [16]

Joint	Angle Ranges (deg/rad)			
	Min.		Max.	
Elevation **A**ngle	−90	$-\frac{1}{2}\pi$	130	$\frac{13}{18}\pi$
Shoulder **E**levation	0	0	180	π
Shoulder **R**otation	−90	$-\frac{1}{2}\pi$	20	$\frac{1}{9}\pi$
Elbow **F**lexion	0	0	130	$\frac{13}{18}\pi$
Pronation/**S**upination	−90	$-\frac{1}{2}\pi$	90	$\frac{1}{2}\pi$
Wrist **D**eviation	−10	$-\frac{1}{18}\pi$	25	$\frac{5}{36}\pi$
Wrist **F**lexion	−70	$-\frac{7}{18}\pi$	70	$\frac{7}{18}\pi$

B.2 CFAT

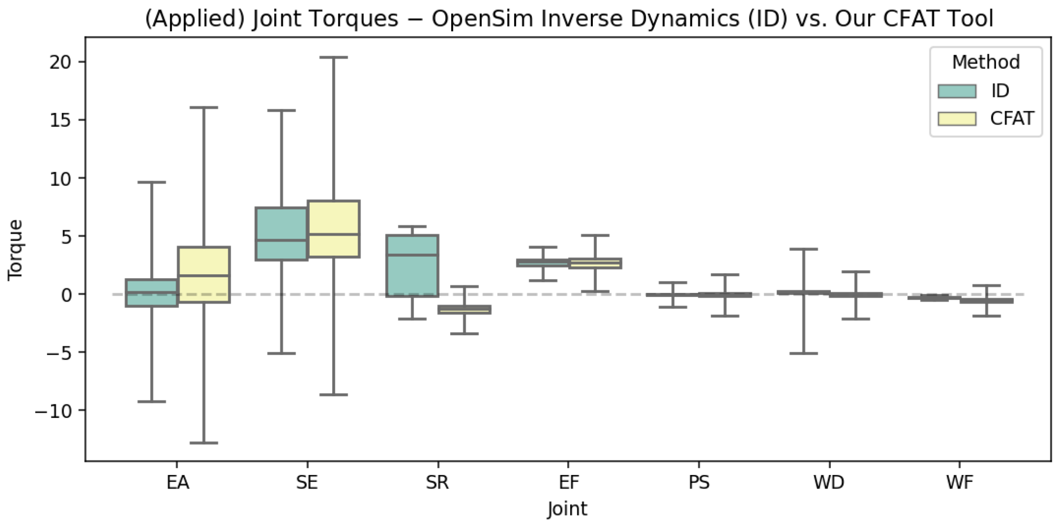


Figure B.1. Joint torques of all participants and interaction techniques, both computed using OpenSim Inverse Dynamics (ID, green) and our proposed CFAT tool (yellow). For each DOF, the colored boxes show the respective interquartile ranges (25% to 75% quantiles) and the whiskers correspond to the minimum and maximum torques after removing some outliers (see Section 4).

B.3 Comparison of Cost Functions

Table B.2. The Cost Weights Obtained by the CMA-ES Parameter Optimization for Each Participant, Condition, and Cost Function

Participant	Technique	r_1^{DC}	r_1^{CTC}	r_2^{CTC}	r_1^{JAC}	r_2^{JAC}
U1	Virtual Cursor ID	0.51	$1.49 \cdot 10^{-2}$	$1.15 \cdot 10^{-4}$	$1.18 \cdot 10^{-3}$	$4.25 \cdot 10^{-5}$
U1	Virtual Cursor Ergonomic	0.22	$1.94 \cdot 10^{-5}$	$1.18 \cdot 10^{-4}$	$9.18 \cdot 10^{-5}$	$2.23 \cdot 10^{-4}$
U1	Virtual Pad ID	$1.71 \cdot 10^{-2}$	$3.01 \cdot 10^{-2}$	$4.79 \cdot 10^{-5}$	$5.61 \cdot 10^{-2}$	$3.13 \cdot 10^{-5}$
U1	Virtual Pad Ergonomic	$8.13 \cdot 10^{-6}$	$1.03 \cdot 10^{-5}$	$4.2 \cdot 10^{-6}$	$3.03 \cdot 10^{-9}$	$1.56 \cdot 10^{-5}$
U2	Virtual Cursor ID	0.11	$5.3 \cdot 10^{-2}$	$7.13 \cdot 10^{-8}$	$7.85 \cdot 10^{-2}$	$4.66 \cdot 10^{-5}$
U2	Virtual Cursor Ergonomic	0.33	0.19	$2.85 \cdot 10^{-5}$	$2.59 \cdot 10^{-2}$	$1.68 \cdot 10^{-4}$
U2	Virtual Pad ID	0.34	$1.11 \cdot 10^{-2}$	$8.66 \cdot 10^{-5}$	$4.57 \cdot 10^{-2}$	$1.16 \cdot 10^{-4}$
U2	Virtual Pad Ergonomic	$1.51 \cdot 10^{-7}$	$1.14 \cdot 10^{-7}$	$8.95 \cdot 10^{-7}$	$6.18 \cdot 10^{-9}$	$2.71 \cdot 10^{-5}$
U3	Virtual Cursor ID	$3.86 \cdot 10^{-2}$	$2.04 \cdot 10^{-2}$	$3.54 \cdot 10^{-7}$	$2.13 \cdot 10^{-2}$	$1.1 \cdot 10^{-6}$
U3	Virtual Cursor Ergonomic	$1.72 \cdot 10^{-2}$	$6.77 \cdot 10^{-5}$	$5.04 \cdot 10^{-6}$	$2.08 \cdot 10^{-4}$	$3.8 \cdot 10^{-7}$
U3	Virtual Pad ID	0.25	$2.79 \cdot 10^{-4}$	$2.46 \cdot 10^{-5}$	$1.67 \cdot 10^{-2}$	$1.02 \cdot 10^{-5}$
U3	Virtual Pad Ergonomic	$2.72 \cdot 10^{-2}$	$1.81 \cdot 10^{-8}$	$1.11 \cdot 10^{-6}$	$2.95 \cdot 10^{-2}$	$1.83 \cdot 10^{-8}$
U4	Virtual Cursor ID	$7.84 \cdot 10^{-2}$	$1.89 \cdot 10^{-3}$	$3.84 \cdot 10^{-7}$	$1.17 \cdot 10^{-2}$	$1.05 \cdot 10^{-4}$
U4	Virtual Cursor Ergonomic	0.3	$1.26 \cdot 10^{-3}$	$5.18 \cdot 10^{-5}$	$1.57 \cdot 10^{-4}$	$1.32 \cdot 10^{-5}$
U4	Virtual Pad ID	$6.7 \cdot 10^{-2}$	$7.18 \cdot 10^{-4}$	$9.19 \cdot 10^{-5}$	$9.28 \cdot 10^{-3}$	$5.07 \cdot 10^{-5}$
U4	Virtual Pad Ergonomic	$5.87 \cdot 10^{-2}$	$2.05 \cdot 10^{-3}$	$2.48 \cdot 10^{-5}$	$3.08 \cdot 10^{-6}$	$5.73 \cdot 10^{-5}$
U5	Virtual Cursor ID	0.12	$9.5 \cdot 10^{-4}$	$2.09 \cdot 10^{-4}$	$1.31 \cdot 10^{-2}$	$3.03 \cdot 10^{-4}$
U5	Virtual Cursor Ergonomic	0.15	0.15	$6.79 \cdot 10^{-14}$	0.13	$1.03 \cdot 10^{-5}$
U5	Virtual Pad ID	$6.36 \cdot 10^{-2}$	$6.15 \cdot 10^{-2}$	$1.55 \cdot 10^{-12}$	$4.3 \cdot 10^{-2}$	$1.34 \cdot 10^{-4}$
U5	Virtual Pad Ergonomic	$3.73 \cdot 10^{-2}$	$8.03 \cdot 10^{-2}$	$5.18 \cdot 10^{-5}$	$9.05 \cdot 10^{-2}$	$1.52 \cdot 10^{-4}$
U6	Virtual Cursor ID	0.1	$1.1 \cdot 10^{-2}$	$1.92 \cdot 10^{-6}$	$1.76 \cdot 10^{-2}$	$1.3 \cdot 10^{-5}$
U6	Virtual Cursor Ergonomic	0.35	$2.45 \cdot 10^{-4}$	$1.94 \cdot 10^{-5}$	$1.05 \cdot 10^{-9}$	$5.2 \cdot 10^{-6}$
U6	Virtual Pad ID	0.39	$8.09 \cdot 10^{-3}$	$1.07 \cdot 10^{-5}$	$9.94 \cdot 10^{-3}$	$1.36 \cdot 10^{-5}$
U6	Virtual Pad Ergonomic	0.11	$3.64 \cdot 10^{-3}$	$1.79 \cdot 10^{-6}$	0.3	$9.56 \cdot 10^{-10}$

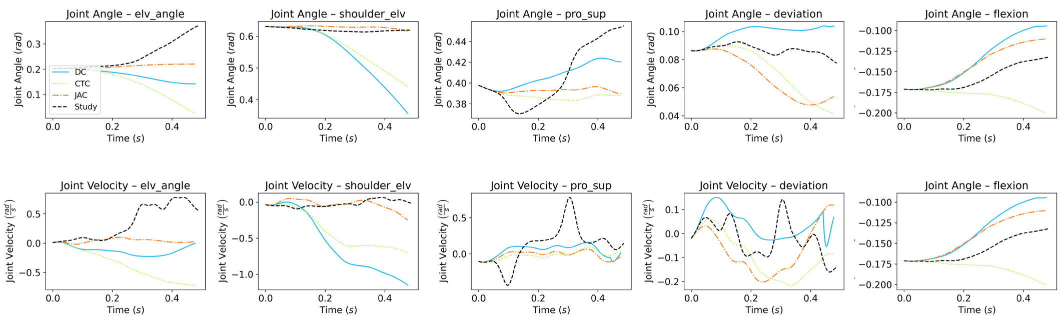


Figure B.2. Projected joint trajectories for one trial of U4 for the Virtual Pad Identity technique. The Joint Acceleration Costs (JAC; orange dashdotted lines) qualitatively explain observed user behavior best. For pronation/supination, deviation, and flexion, all considered cost function show a good fit to observed user behavior (note the small angle ranges for these joints). Remaining joints are shown in Figure 6.

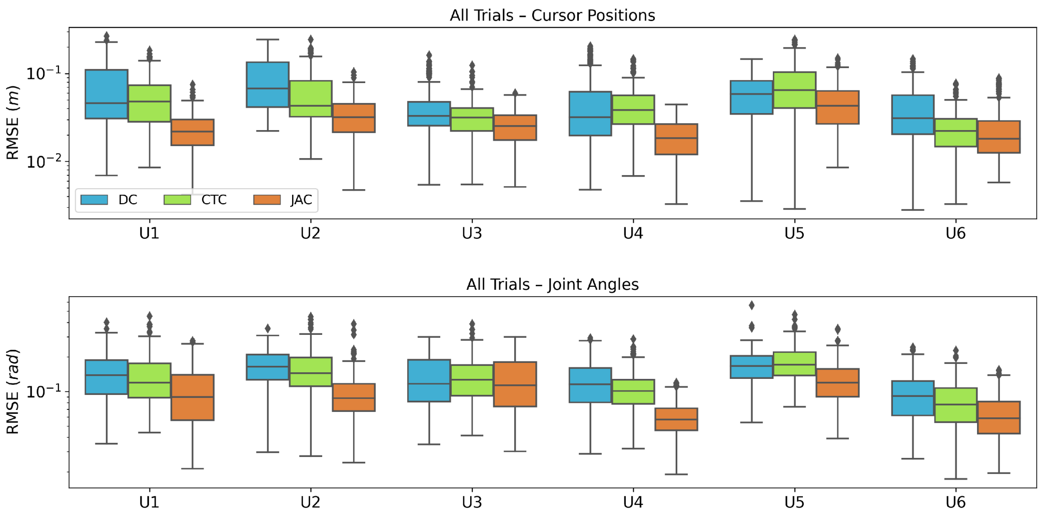


Figure B.3. Comparison of the different cost functions in terms of cursor positions and joint angles, separated by user. The boxplots show the RMSE of all trials.

B.4 MPC can Simulate User Movements

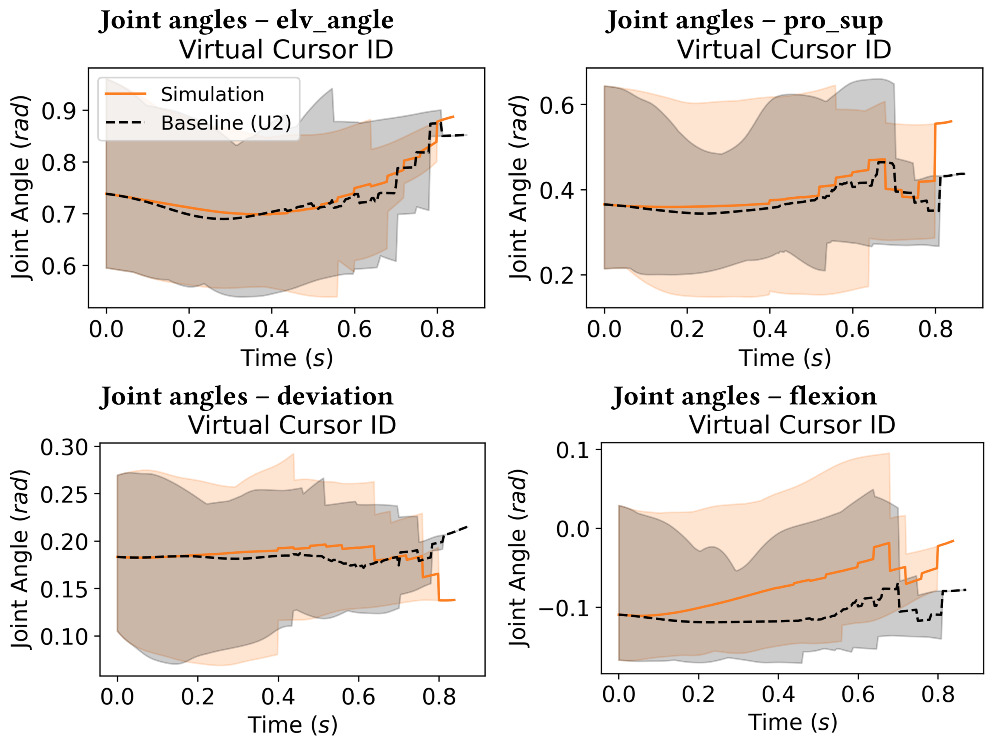


Figure B.4. The joint angle ranges predicted by our simulation for different movements in the ISO task (orange solid lines) match those observed in our user study (black dashed lines) fairly well. The mean of all movements of a single participant/user model (U2) is shown together with the entire value ranges. Remaining joints are shown in Figure 8.

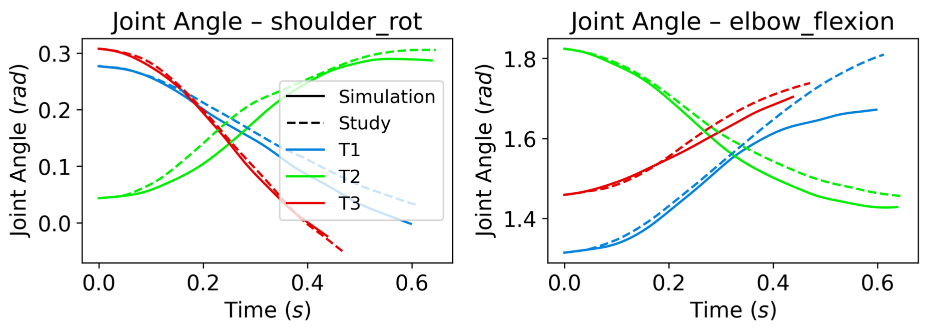


Figure B.5. U4 and simulation trajectories for different directions in the Virtual Pad Ergonomic condition. In the user study (dashed lines), the joint angle trajectories differ significantly between movement directions (the target configuration is depicted in Figure 4). These characteristic differences are captured well by our simulation (solid lines).

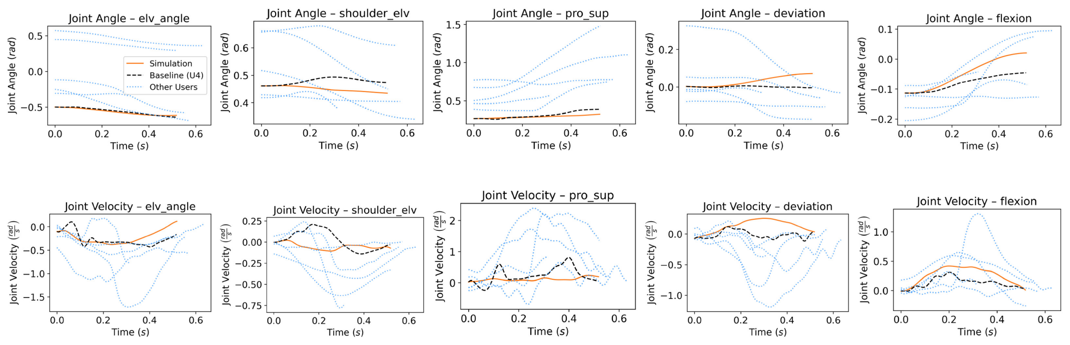


Figure B.6. Given an interaction technique (here: **Virtual Pad Ergonomic**) and a movement direction (here: movements from targets 1 to 2), the characteristic cursor and joint trajectories of an individual user (here: U4, black dashed lines; trajectories of the remaining users are shown as blue dotted lines for comparison) can be predicted by our simulation (orange solid lines). Remaining joints are shown in Figure 11.

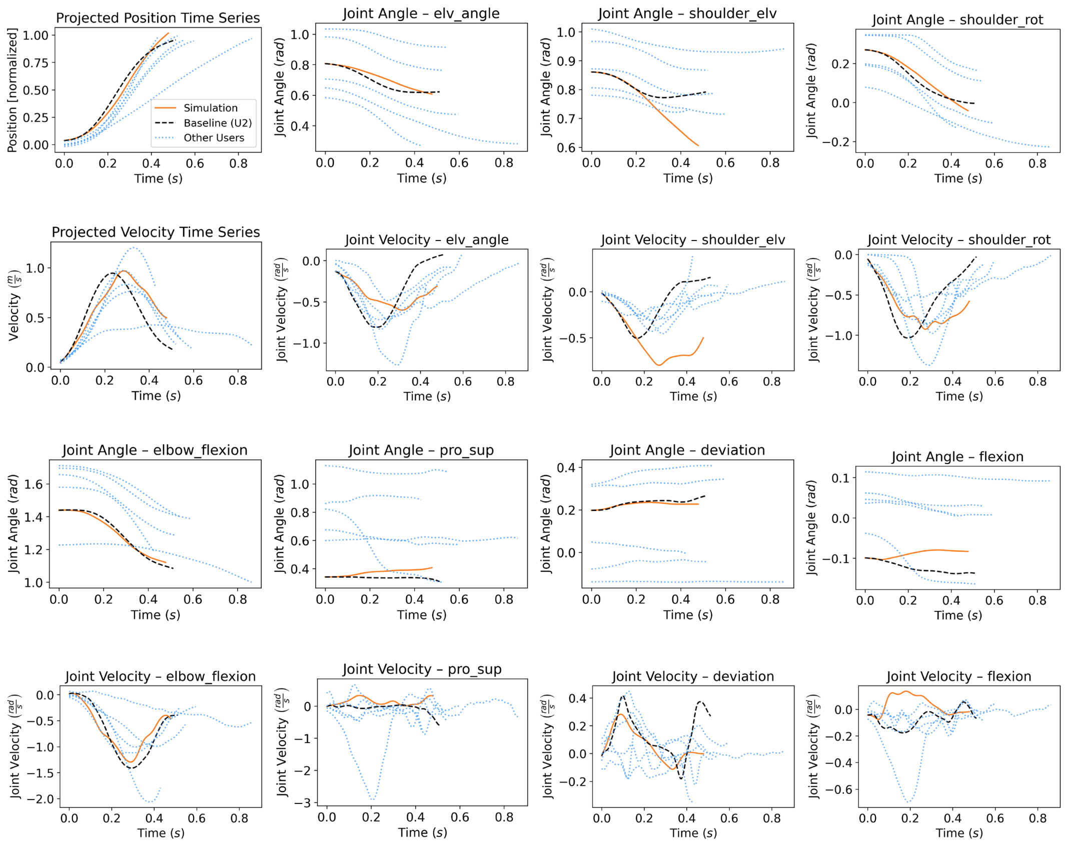


Figure B.7. Given an interaction technique (here: **Virtual Cursor ID**) and a movement direction (here: movements from targets 8 to 9), the characteristic cursor and joint trajectories of an individual user (here: U2, black dashed lines; trajectories of the remaining users are shown as blue dotted lines for comparison) can be predicted by our simulation (orange solid lines).

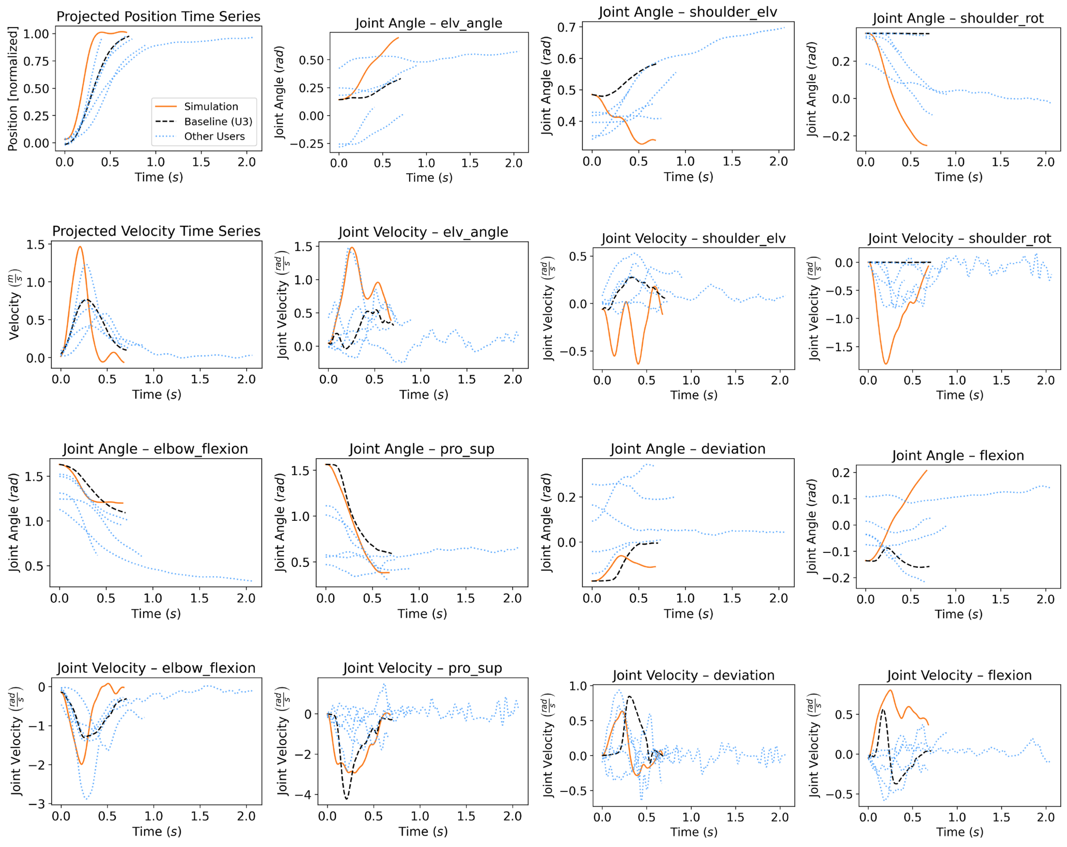


Figure B.8. Given an interaction technique (here: **Virtual Cursor Ergonomic**) and a movement direction (here: movements from targets 8 to 9), the characteristic cursor and joint trajectories of an individual user (here: U3, black dashed lines; trajectories of the remaining users are shown as blue dotted lines for comparison) can be predicted by our simulation (orange solid lines).

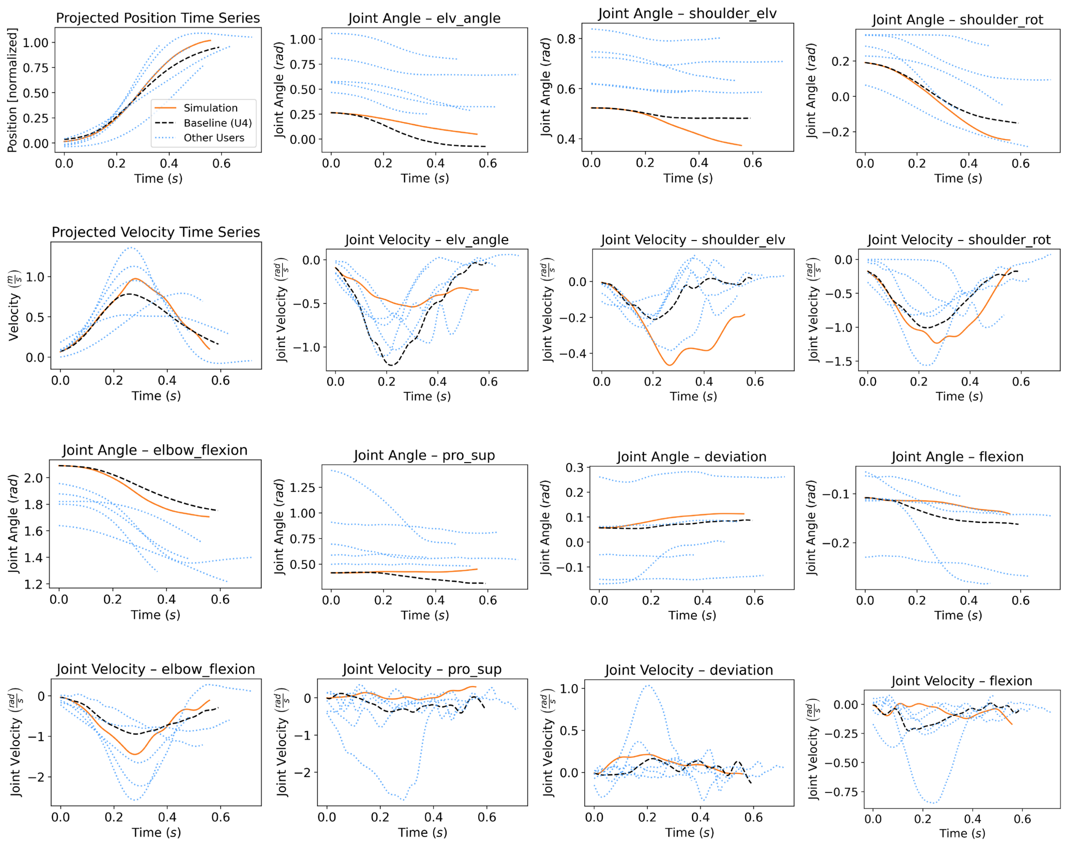


Figure B.9. Given an interaction technique (here: **Virtual Pad ID**) and a movement direction (here: movements from targets 8 to 9), the characteristic cursor and joint trajectories of an individual user (here: U4, black dashed lines; trajectories of the remaining users are shown as blue dotted lines for comparison) can be predicted by our simulation (orange solid lines).

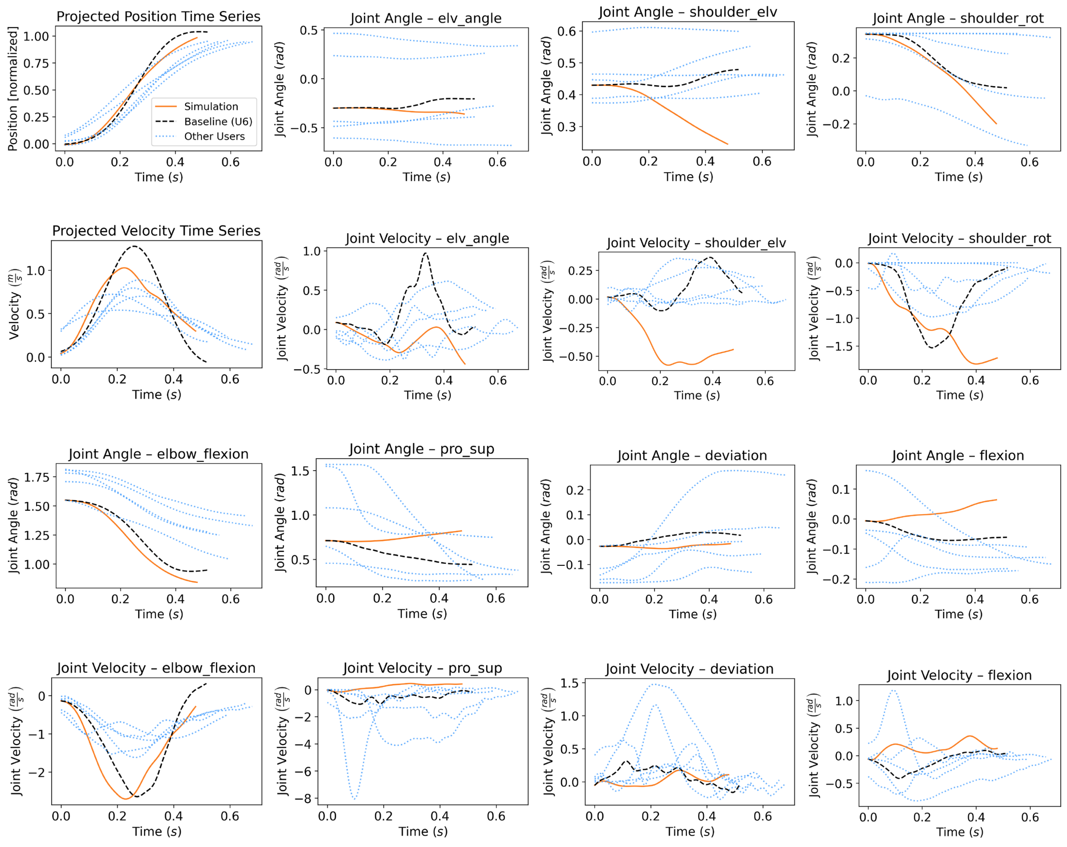


Figure B.10. Given an interaction technique (here: **Virtual Pad Ergonomic**) and a movement direction (here: movements from targets 8 to 9), the characteristic cursor and joint trajectories of an individual user (here: U6, black dashed lines; trajectories of the remaining users are shown as blue dotted lines for comparison) can be predicted by our simulation (orange solid lines).

B.5 Effects of the Cost Weights

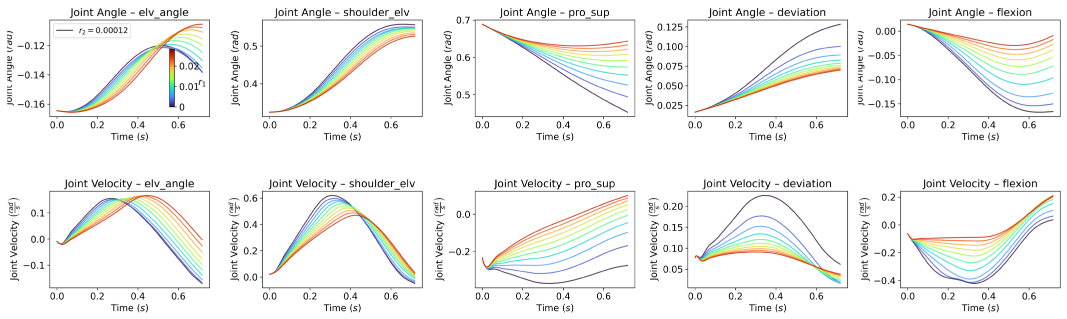


Figure B.11. Trajectories of the JAC simulation for different cost weights r_1. Remaining joints and cursor trajectories are shown in Figure 12. Shown are simulated movements for U4 in the Virtual Cursor Ergonomic condition.

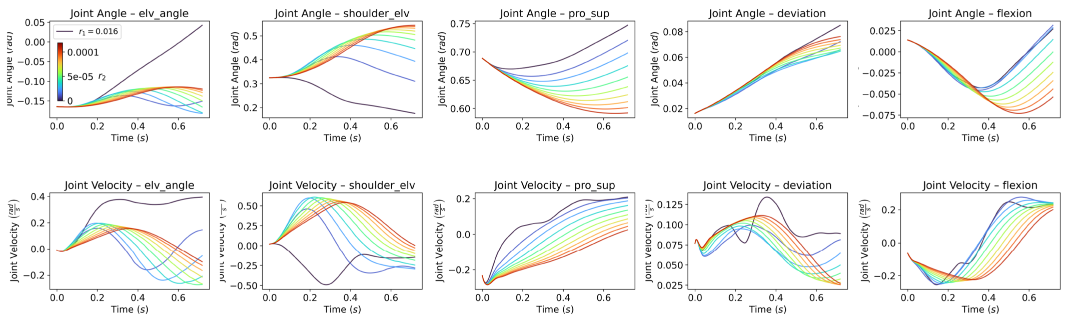


Figure B.12. Trajectories of the JAC simulation for different cost weights r_2. Remaining joints and cursor trajectories are shown in Figure 13. Shown are simulated movements for U4 in the Virtual Cursor Ergonomic condition.

B.6 Effects of the MPC Horizon

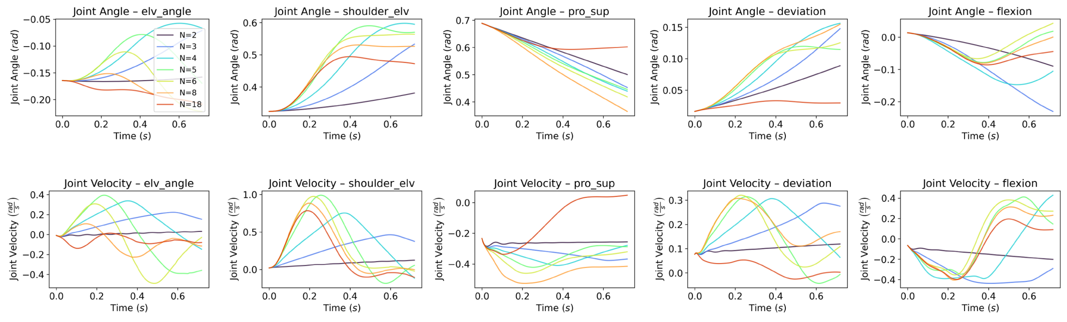


Figure B.13. Joint trajectories of one trial for varying MPC horizon N, using the JAC, without control noise. Remaining joints and cursor trajectories are shown in Figure 15. Shown are simulated movements for U4 in the Virtual Cursor Ergonomic condition.

REFERENCES

[1] Carlos Andujar and Ferran Argelaguet. 2007. Virtual pads: Decoupling motor space and visual space for flexible manipulation of 2d windows within ves. In *Proceedings of the 2007 IEEE Symposium on 3D User Interfaces*. IEEE.

[2] Myroslav Bachynskyi and Jörg Müller. 2020. Dynamics of Aimed Mid-Air movements. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI'20)*. Association for Computing Machinery, New York, NY, 1–12. DOI: https://doi.org/10.1145/3313831.3376194

[3] Myroslav Bachynskyi, Antti Oulasvirta, Gregorio Palmas, and Tino Weinkauf. 2014. Is motion capture-based biomechanical simulation valid for HIC studies? Study and implications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3215–3224.

[4] Myroslav Bachynskyi, Gregorio Palmas, Antti Oulasvirta, Jürgen Steimle, and Tino Weinkauf. 2015. Performance and ergonomics of touch surfaces: A comparative study using biomechanical simulation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*. ACM, New York, NY, 1817–1826. DOI: http://doi.acm.org/10.1145/2702123.2702607

[5] Myroslav Bachynskyi, Gregorio Palmas, Antti Oulasvirta, and Tino Weinkauf. 2015. Informing the design of novel input methods with muscle coactivation clustering. *ACM Transactions on Computer–Human Interaction (TOCHI)* 21, 6, Article 30 (Jan. 2015), 25 pages.

[6] Reinoud J. Bootsma, Laure Fernandez, and Denis Mottet. 2004. Behind Fitts' law: Kinematic patterns in goal-directed movements. *International Journal of Human–Computer Studies* 61, 6 (2004), 811–821.

[7] John Charles Butcher. 2016. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons.

[8] Stuart K. Card, William K. English, and Betty J. Burr. 1978. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics* 21, 8 (1978), 601–613.

[9] Noshaba Cheema, Laura A. Frey-Law, Kourosh Naderi, Jaakko Lehtinen, Philipp Slusallek, and Perttu Hämäläinen. 2020. Predicting Mid-Air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI'20)*. Association for Computing Machinery, New York, NY, 1–13. DOI: https://doi.org/10.1145/3313831.3376701

[10] Scott L. Delp, Frank C. Anderson, Allison S. Arnold, Peter Loan, Ayman Habib, Chand T. John, Eran Guendelman, and Darryl G. Thelen. 2007. OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering* 54, 11 (2007), 1940–1950.

[11] Jörn Diedrichsen, Reza Shadmehr, and Richard B Ivry. 2010. The coordination of movement: Optimal feedback control and beyond. *Trends in Cognitive Sciences* 14, 1 (2010), 31–39.

[12] Peter Dorato and Alexander Levis. 1971. Optimal linear regulators: The discrete-time case. *IEEE Transactions on Automatic Control* 16, 6 (1971), 613–620.

[13] A. Aldo Faisal, Luc P. J. Selen, and Daniel M. Wolpert. 2008. Noise in the nervous system. *Nature Reviews Neuroscience* 9, 4 (2008), 292–303.

[14] Faulwasser Timm, Grüne Lars, and Müller Matthias A. 2018. Economic Nonlinear Model Predictive Control. https://ieeexplore.ieee.org/document/8277242.

[15] Timm Faulwasser, Milan Korda, Colin N. Jones, and Dominique Bonvin. 2014. Turnpike and dissipativity properties in dynamic real-time optimization and economic MPC. In *Proceedings of the 53rd IEEE Conference on Decision and Control*. IEEE, 2734–2739.

[16] Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, and Jörg Müller. 2021. Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports* 11, 1 (2021), 1–15.

[17] Florian Fischer, Arthur Fleig, Markus Klar, Lars Grüne, and Jörg Müller. 2020. An optimal control model of mouse pointing using the LQR. https://arxiv.org/abs/2002.11596.

[18] Florian Fischer, Arthur Fleig, Markus Klar, and Jörg Müller. 2022. Optimal feedback control for modeling human–computer interaction. *ACM Transactions on Computer–Human Interaction* 29, 6, Article 51 (November 2022), 70 pages. DOI: https://doi.org/10.1145/3524122

[19] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement.*Journal of Experimental Psychology* 47, 6 (1954), 381.

[20] Tamar Flash and Neville Hogan. 1985. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience* 5, 7 (1985), 1688–1703.

[21] Peter Gawthrop, Ian Loram, Martin Lakie, and Henrik Gollee. 2011. Intermittent control: A computational theory of human control. *Biological Cybernetics* 104, 1-2 (2011), 31–51.

[22] Lars Grüne and Vryan Gil Palma. 2014. On the benefit of re-optimization in optimal control under perturbations. In *Proceedings of the 21st International Symposium on Mathematical Theory of Networks and Systems* MTNS 2014. University of Groningen, Groningen, 439–446. Retrieved from https://eref.uni-bayreuth.de/11269/.

[23] Lars Grüne and Jürgen Pannek. 2017. *Nonlinear Model Predictive Control*. Springer International Publishing, Cham, 45–69. DOI: https://doi.org/10.1007/978-3-319-46024-6_3

[24] Lars Grüne and Anders Rantzer. 2008. On the infinite horizon performance of receding horizon controllers. *IEEE Transactions on Automatic Control* 53, 9 (2008), 2100–2111.

[25] Emmanuel Guigon, Pierre Baraduc, and Michel Desmurget. 2007. Computational motor control: Redundancy and invariance. *Journal of Neurophysiology* 97, 1 (2007), 331–347. DOI: https://doi.org/10.1152/jn.00290.2006

[26] Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online motion synthesis using sequential Monte Carlo. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

[27] Nikolaus Hansen. 2016. The CMA Evolution Strategy: A Tutorial. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*, Association for Computing Machinery, New York, NY, 1–12. https://doi.org/10.1145/3173574.3173804 arXiv:1604.00772. Retrieved from https://arxiv.org/abs/1604.00772.

[28] Christopher M. Harris and Daniel M. Wolpert. 1998. Signal-dependent noise determines motor planning. *Nature* 394, 6695 (1998), 780–784.

[29] Lorenz Hetzel, John Dudley, Anna Maria Feit, and Per Ola Kristensson. 2021. Complex interaction as emergent behaviour: Simulating Mid-Air virtual keyboard typing using reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics* 27, 11 (2021), 4140–4149. DOI: https://doi.org/10.1109/TVCG.2021.3106494

[30] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. 2014. Consumed endurance: A metric to quantify arm fatigue of Mid-Air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'14)*. Association for Computing Machinery, New York, NY, 1063–1072. DOI: https://doi.org/10.1145/2556288.2557130

[31] Sujin Jang, Wolfgang Stuerzlinger, Satyajit Ambike, and Karthik Ramani. 2017. Modeling cumulative arm fatigue in Mid-Air interaction based on perceived exertion and kinetics of arm motion. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. Association for Computing Machinery, New York, NY, 3328–3339. DOI: https://doi.org/10.1145/3025453.3025523

[32] Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. 2019. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)* 38, 4 (2019), 1–12.

[33] Jussi Jokinen, Aditya Acharya, Mohammad Uzair, Xinhui Jiang, and Antti Oulasvirta. 2021. *Touchscreen Typing As Optimal Supervisory Control*. Association for Computing Machinery, New York, NY. DOI: https://doi.org/10.1145/3411764.3445483

[34] Nocedal Jorge and J. Wright Stephen. 2006. *Numerical Optimization*. Spinger.

[35] Mitsuo Kawato. 1993. Optimization and learning in neural networks for formation and control of coordinated movement. In *Proceedings of the Attention and Performance XIV (Silver Jubilee Volume) Synergies in Experimental Psychology, Artificial Intelligence, and Cognitive Neuroscience*. 821–849.

[36] Francesco Lacquaniti, Carlo Terzuolo, and Paolo Viviani. 1983. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica* 54, 1 (1983), 115–130.

[37] Byungjoo Lee, Sunjun Kim, Antti Oulasvirta, Jong-In Lee, and Eunji Park. 2018. Moving target selection: A cue integration model. 1–12. DOI: https://doi.org/10.1145/3173574.3173804

[38] Injung Lee, Hyunchul Kim, and Byungjoo Lee. 2021. Automated playtesting with a cognitive model of sensorimotor coordination. In *Proceedings of the 29th ACM International Conference on Multimedia (MM'21)*. Association for Computing Machinery, New York, NY, 4920–4929. DOI: https://doi.org/10.1145/3474085.3475429

[39] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)* 38, 4, Article 73 (July 2019), 13 pages. DOI: https://doi.org/10.1145/3306346.3322972

[40] Sangyoon Lee, Jinseok Seo, Gerard Jounghyun Kim, and Chan-Mo Park. 2003. Evaluation of pointing techniques for ray casting selection in virtual environments. In *Proceedings of the Third International Conference on Virtual Reality and Its Application in Industry*, Vol. 4756. SPIE, 38–44.

[41] Weiwei Li and Emanuel Todorov. 2004. Iterative linear quadratic regulator design for nonlinear biological movement systems.. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics, (ICINCO'04)*, Vol. 1. 222–229.

[42] Dan Liu and Emanuel Todorov. 2009. Hierarchical optimal control of a 7-DOF arm model. In *Proceedings of the 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. IEEE, 50–57.

[43] I. Scott MacKenzie. 1992. Fitts' law as a research and design tool in human–computer interaction. *Human–Computer Interaction* 7, 1 (1992), 91–139.

[44] J. Alberto Álvarez Martín, Henrik Gollee, Jörg Müller, and Roderick Murray-Smith. 2021. Intermittent control as a model of mouse movements. *ACM Transactions on Computer–Human Interaction (TOCHI)* 28, 5, Article 35 (Aug 2021), 46 pages. DOI: https://doi.org/10.1145/3461836

[45] Pietro Morasso. 1981. Spatial control of arm movements. *Experimental Brain Research* 42, 2 (1981), 223–227.

[46] Jörg Müller. 2017. Dynamics of pointing with pointer acceleration. In *Human–Computer Interaction—INTERACT 2017*. Regina Bernhaupt, Girish Dalvi, Anirudha Joshi, Devanuj K. Balkrishan, Jacki O'Neill, and Marco Winckler (Eds.), Springer International Publishing, Cham, 475–495.

[47] Jörg Müller, Antti Oulasvirta, and Roderick Murray-Smith. 2017. Control theoretic models of pointing. *ACM Transactions on Computer–Human Interaction (TOCHI)* 24, 4 (2017), 1–36.

[48] Eri Nakano, Hiroshi Imamizu, Rieko Osu, Yoji Uno, Hiroaki Gomi, Toshinori Yoshioka, and Mitsuo Kawato. 1999. Quantitative examinations of internal representations for arm trajectory planning: Minimum commanded torque change model. *Journal of Neurophysiology* 81, 5 (1999), 2140–2155.

[49] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)* 37, 4, Article 143 (July 2018), 14 pages. DOI:https://doi.org/10.1145/3197517.3201311

[50] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique: Non-linear mapping for direct manipulation in VR. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology.* 79–80.

[51] Ning Qian, Yu Jiang, Zhong-Ping Jiang, and Pietro Mazzoni. 2013. Movement duration, Fitts's law, and an infinite-horizon optimal feedback control model for biological motor systems. *Neural Computation* 25, 3 (2013), 697–724.

[52] S. Joe Qin and Thomas A. Badgwell. 2003. A survey of industrial model predictive control technology. *Control Engineering Practice* 11, 7 (2003), 733–764.

[53] Philip Quinn and Shumin Zhai. 2018. Modeling gesture-typing movements. *Human–Computer Interaction* 33, 3 (2018), 234–280. DOI:https://doi.org/10.1080/07370024.2016.1215922

[54] John Rasmussen, Michael Damsgaard, Egidijus Surma, Søren T. Christensen, Mark de Zee, and Vit Vondrak. 2003. Anybody-a software system for ergonomic optimization. In *Proceedings of the Fifth World Congress on Structural and Multidisciplinary Optimization*, Vol. 4. Citeseer, 6.

[55] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. 2017. *Model Predictive Control: Theory, Computation, and Design (2nd ed.).* Nob Hill Publishing.

[56] Katherine R Saul, Xiao Hu, Craig M. Goehler, Meghan E. Vidt, Melissa Daly, Anca Velisar, and Wendy M. Murray. 2014. Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model.*Computer Methods in Biomechanics and Biomedical Engineering* 5842, May 2016 (2014), 1–14. DOI:https://doi.org/10.1080/10255842.2014.916698

[57] Richard A. Schmidt, Howard Zelaznik, Brian Hawkins, James S. Frank, and John T. Quinn Jr. 1979. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review* 86, 5 (1979), 415.

[58] Ajay Seth, Jennifer L. Hicks, Thomas K. Uchida, Ayman Habib, Christopher L. Dembia, James J. Dunne, Carmichael F. Ong, Matthew S. DeMers, Apoorva Rajagopal, Matthew Millard, et al. 2018. OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS Computational Biology* 14, 7 (2018), e1006223.

[59] Olga Sorkine-Hornung and Michael Rabinovich. 2017. Least-squares rigid motion using svd. *Computing* 1, 1 (2017), 1–5.

[60] G. G. Sutton and K. Sykes. 1967. The variation of hand tremor with force in healthy subjects. *The Journal of Physiology* 191, 3 (1967), 699–711.

[61] Misaki Takeda, Takanori Sato, Hisashi Saito, Hiroshi Iwasaki, Isao Nambu, and Yasuhiro Wada. 2019. Explanation of fitts' law in reaching movement based on human arm dynamics. *Scientific Reports* 9, 1 (2019), 1–12. DOI:https://doi.org/10.1038/s41598-019-56016-7

[62] Darryl G. Thelen and Frank C. Anderson. 2006. Using computed muscle control to generate forward dynamic simulations of human walking from experimental data. *Journal of Biomechanics* 39, 6 (2006), 1107–1115. DOI:https://doi.org/10.1016/j.jbiomech.2005.02.010

[63] Emanuel Todorov. 2005. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Computation* 17, 5 (2005), 1084–1108.

[64] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 5026–5033.

[65] Emanuel Todorov and Michael I. Jordan. 2002. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience* 5, 11 (2002), 1226–1235. DOI:https://doi.org/10.1038/nn963

[66] Brian R. Umberger and Ross H. Miller. 2017. Optimal control modeling of human movement. *Handbook of Human Motion*, Müller Bertram, Wolf Sebastian I., Brueggemann Gert-Peter, Deng Zhigang, McIntosh Andrew, Miller Freeman, and Selbie William Scott (Eds.). Springer International Publishing, Cham, 1–22. https://doi.org/10.1007/978-3-319-30808-1_177-1

[67] Yoji Uno, Mitsuo Kawato, and Rika Suzuki. 1989. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics* 61, 2 (1989), 89–101.

[68] Robert J. van Beers, Patrick Haggard, and Daniel M. Wolpert. 2004. The role of execution noise in movement variability. *Journal of Neurophysiology* 91, 2 (2004), 1050–1063. DOI:https://doi.org/10.1152/jn.00652.2003

[69] Frans C. T. Van der Helm and Leonard A. Rozendaal. 2000. Musculoskeletal systems with intrinsic and proprioceptive feedback. In *Proceedings of the Biomechanics and Neural Control of Posture and Movement*. Springer, 164–174.
[70] Sergio Vazquez, Jose Rodriguez, Marco Rivera, Leopoldo G. Franquelo, and Margarita Norambuena. 2017. Model predictive control for power converters and drives: Advances and trends. *IEEE Transactions on Industrial Electronics* 64, 2 (2017), 935–947. DOI: https://doi.org/10.1109/TIE.2016.2625238
[71] Yasuhiro Wada, Yuichi Kaneko, Eri Nakano, Rieko Osu, and Mitsuo Kawato. 2001. Quantitative examinations for multi joint arm trajectory planning–using a robust calculation algorithm of the minimum commanded torque change trajectory. *Neural Networks* 14, 4-5 (2001), 381–393.
[72] Eric A. Wan and Rudolph Van Der Merwe. 2000. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. 153–158.
[73] Jiaole Wang and Masazumi Katayama. 2011. Optimal model for selecting human arm posture during reaching movement. In *Proceedings of the Advances in Cognitive Neurodynamics (II)*. Springer, 453–458.
[74] Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.
[75] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)* 23, 4 (December 1997), 550–560. DOI: https://doi.org/10.1145/279232.279236

Received 19 April 2022; revised 11 November 2022; accepted 21 November 2022

4 An Optimal Control Model of Mouse Pointing Using the LQR

Authors: Florian Fischer, Arthur Fleig, Markus Klar, Lars Grüne, Jörg Müller
Status: Pre-Print (arXiv) [16]

The models were selected by FF and JM. FF implemented the models and the parameter fitting in Matlab and visualized the model predictions. All authors analyzed and evaluated the simulation results. Figures were created by FF and MK. The results were discussed by FF, AF, MK, and JM. FF wrote the first draft of the manuscript. Revision and rewriting of the manuscript was done by all authors. FF is the corresponding author.

An Optimal Control Model of Mouse Pointing Using the LQR

Florian Fischer, Arthur Fleig, Markus Klar, Lars Grüne, Jörg Müller
University of Bayreuth, Germany

ABSTRACT
In this paper we explore the Linear-Quadratic Regulator (LQR) to model movement of the mouse pointer. We propose a model in which users are assumed to behave optimally with respect to a certain cost function. Users try to minimize the distance of the mouse pointer to the target smoothly and with minimal effort, by simultaneously minimizing the jerk of the movement. We identify parameters of our model from a dataset of reciprocal pointing with the mouse. We compare our model to the classical minimum-jerk and second-order lag models on data from 12 users with a total of 7702 movements. Our results show that our approach explains the data significantly better than either of these previous models.

Author Keywords
Pointing; Aimed Movements; Fitts' Law; Control Theory; LQR; Modeling; Second-order Lag; Minimum Jerk

CCS Concepts
•Human-centered computing → HCI theory, concepts and models;

INTRODUCTION

Interaction with computers is almost always achieved through movement of the user, measured via input devices. In the field of human motor control, there has been tremendous progress in the understanding of human movement since the 1950's and 60's, when Fitts' law [11, 12] was published. Arguably the most important modern theory of human motor control is optimal feedback control (OFC) [34, 8]. Its main strengths are *versatility* (applicable to many movement tasks) and the ability to *predict the entire movement* (including position, velocity, and acceleration of the end-effector over time, not just movement time) without relying on Machine Learning techniques, thus retaining *comprehensibility*. Despite its advantages, OFC models are not very well known in the field of Human-Computer Interaction (HCI), yet. The objective of this paper is to introduce optimal feedback control to HCI.

OFC is a family of computational models of (human) movement. These models assume that people behave rationally, i.e., optimally with respect to some cost function. In addition, people observe the state of the environment and adjust their movement in order to accomplish a given task, in a feedback manner. The interplay of the three main constituents of OFC, i.e., *optimality*, *feedback*, and *control*, is displayed in Figure 1.

As the figure suggests, the OFC framework is very versatile: Various movements such as hand or eye movements or balancing, can be explained by adjusting the *System* block (and

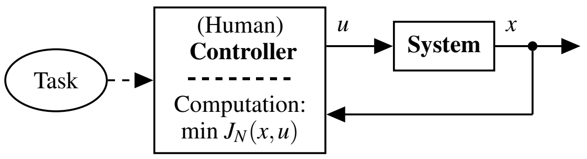


Figure 1. In our model, the user is assumed to *control* the state x of the interactive system (e.g., the mouse pointer position and velocity). We assume that the user computes the control u through *optimization*, i.e., by minimizing a cost function J_N. In this calculation the current state is taken into account through *feedback*.

the *Controller* block, if necessary). Various instructions, such as emphasizing speed vs. comfort, can be incorporated by adapting the cost function. Due to their feedback structure (also called *closed-loop*), OFC models provide intuitive insight in how humans react to disturbances during the movement, changing targets, etc.

Through OFC, we aim at connecting the field of HCI better with recent advances in neighboring scientific disciplines, such as the study of human movement in motor control [29, 13] and neuroscience [31].

From a scientific perspective, this would strengthen the field of HCI through a deeper insight into the basic constituents of interaction. We start from one of the simplest and most ubiquitous ways we interact with Personal Computers: pointing with a mouse. However, as stated above, OFC could provide a unifying framework for understanding movement in many different interactive tasks, including pointing, steering, tracking of moving targets, scrolling and zooming, with PCs, mobile devices, in AR/VR, etc.

From an engineering perspective, OFC would enable a deeper understanding of the impact of interface design parameters on the process of interaction. In the long term, these models could be used for automated optimization of the parameters of interaction techniques. Models of the dynamics of interaction would help in the design of input devices, from mice to VR controllers. Models that work in real-time could be used in predictive interfaces, which anticipate what the user wants to do and respond accordingly, such as pointing target prediction [1].

To achieve our goals, we start from a well-known model from OFC theory, presented by Todorov [32]. We believe that the best way to introduce modern motor control theory to HCI is to provide a simple model that is adapted to the above mentioned HCI purposes. Thus, we make several model simplifications, which we discuss below. These allow us to use the so-called Linear-Quadratic Regulator (LQR) as the *Con-*

troller in Figure 1, to calculate the optimal feedback control law. We explore cost functions that combine the objectives of minimizing jerk, which is the derivative of acceleration, and minimizing the distance to the target. We identify parameters of these cost functions and the underlying pointer dynamics from a dataset of reciprocal pointing [25]. We compare the ability of our model to replicate pointer movement to two other models based on the *second-order lag* [7, 21] and *jerk minimization* [13]. Both are suitable comparison candidates: the former model has been evaluated with the same dataset [25]; the latter is an established model in motor control, which has been applied in HCI context [28]. We compare the models on data from 12 users, with 7702 movements overall.

Our results show that our model is able to fit the data significantly better than the other two models. Compared to the former, our approach can generate more symmetric and plausible velocity and acceleration profiles. Compared to the latter, our approach allows to simultaneously model the movement well and reach the target. Our model can predict the entire movement with only three, intuitively interpretable parameters.

RELATED WORK

In HCI, movement, e.g., of the mouse pointer, is often reduced to summary statistics such as movement time. The dependency of movement time MT from distance D and width W of targets is usually described by Fitts' law [11, 12] as $MT = a + b\mathrm{ID}$ with Index of Difficulty (ID) defined as $\mathrm{ID} = \log_2(D/W + 1)$ [23], although alternatives such as Meyer's law exist [24]. In HCI, Fitts' law is usually interpreted from an information theoretic perspective. A very good explanation of this interpretation of Fitts' law has been provided by Gori et al. [15].

The kinematics and dynamics of movement are studied more rarely in HCI. However, in the studies of human motor control, various models describing kinematics and dynamics of human movement have been developed.

Feedback control models (also called *closed-loop models*) of movement assume that people monitor and adjust their motion on a moment-to-moment basis. These models are able to explain how users repeatedly correct errors and handle disturbances. An early closed-loop model (without optimization) has been provided by Crossman and Goodeve [7]. They assume that users observe hand and target and adjust their velocity as a linear function of the distance, as a first-order lag.

A simple, physically more plausible extension of the first-order lag is the second-order lag [7, 21]. These dynamics can be interpreted as a spring-mass-damper system similar to that implied by the equilibrium-point theory of motor control [29]. A constant force is applied to the mass, such that the system moves to and remains at the target equilibrium. This is one of the comparison models; hence, we call this approach *2OL-Eq*. Other models of human movement include VITE [4] and the models of Plamondon [26].

A fundamentally different approach to using such fixed-control models is to assume that humans try to behave optimally, according to a certain internalized cost function. Flash and Hogan [13] propose that humans aim to generate smooth movements by minimizing the jerk of the end effector. We call this model *MinJerk* in the following. Although the hypothesis that people aim to minimize jerk has been questioned, see, e.g., Harris and Wolpert [17], it is an established model and has been successfully used by Quinn and Zhai [28] to model the shape of gestures on a word-gesture keyboard. The minimum-jerk model predicts a scale-invariant trajectory (as a 5th-degree polynomial), if the exact position and time of beginning and end of the movement are known. It can be interpreted as a trajectory planning step [34] and is thus particularly appropriate for modeling movements that do not involve so-called *corrective submovements*. These have first been proposed by Woodsworth [36, 10] and typically occur after the first large movement, also called the "surge", towards the target [24]. Hence, while applicable for gestures, it remains to be seen whether this model can replicate mouse pointer data accurately. Moreover, it does not explain how people execute that trajectory, or if and how they react to disturbances, such as muscle fatigue, external perturbations, changes of the target, etc.

The theory of OFC allows to resolve the separation between trajectory planning and execution. Excellent overviews of recent progress in OFC theory are provided by Crevecoeur et al. [6] and Diedrichsen [8]. An early approach that models perturbed reach and grasp movements by using the minimum-jerk trajectory on a moment-to-moment basis was presented by Hoff and Arbib [19]. A more general, more recent and better known OFC model is proposed by Todorov and Jordan [34]. This non-deterministic model is based on an extension of the Linear-Quadratic-Gaussian Regulator (E-LQG) [32]. It assumes that users try to reach a target at a certain time while minimizing jerk. The biomechanical apparatus is modeled by second-order lag dynamics. In via-point tasks, this model qualitatively replicates movement segmentation, eye-hand coordination, visual perturbations, and other characteristics of human movement. A discussion about how this model, including state- and control-dependent noise, can be extended to more general reaching movements can be found in [33].

A fundamental limitation of the E-LQG model (and many other optimal control models, e.g., [13, 35, 17]) is that the exact movement time needs to be known in advance. One way to circumvent this issue is to use infinite-horizon OFC [20, 27, 22], i.e., to formulate the optimal control problem on an infinite time horizon. In these references, this approach, in conjunction with a cost function that includes (quadratic) distance and effort costs, was used to model end-effector movement towards a target. The movement time then emerges from the optimal control problem.

Another strand of literature that specifically deals with the duration of movement has produced the *Cost of Time* theory [18, 30, 2]. This theory assumes that humans value time with a certain (e.g., hyperbolic or sigmoidal) cost function. Thus, movement time is explicitly included in the cost function.

In summary, the fundamental question of human movement coordination has produced a substantial literature and deep understanding regarding the nature of human movement. Given that almost all interaction of humans with computers involves movement, it is surprising that this knowledge is little known in HCI. It is important to bear in mind, however, that the purposes of these models are very different from HCI. They intend to model movement of the human body per se. In contrast, in HCI we are less interested in how the body moves, and more interested in how virtual objects in the computer, such as mouse pointers, move. Movement in HCI is mediated by input devices, operating systems, and programs, requires high precision, and is often learnt very well. Therefore, these models need to be adapted and validated regarding their ability to model movement of virtual objects such as mouse pointers in interaction.

In the field of HCI, there are few publications with control models of mouse pointer movement. Müller et al. [25] compare three feedback control models (without optimization) regarding their ability to model mouse pointer movements. Ziebart et al. [37] explore the use of optimal control models for pointing target prediction. They do not make particular a priori assumptions about the structure of the cost function. Instead, they use a machine learning approach to fit a generic function with a large number of parameters (36) to a dataset of mouse pointer movements. While suitable for their purposes, we are interested in gaining more insight into the structure of the cost function. Furthermore, we believe that reducing the number of parameters (to three in our main model) reduces the risk of overfitting.

MODEL SIMPLIFICATIONS

Our approach to introducing OFC theory to HCI is by providing a model that is applicable to HCI, easy enough to understand, while still showing the benefits and strengths of OFC theory. To this end, we start with a simple model for mouse pointer movements that we validate on an HCI dataset. Based on this initial introduction of OFC to HCI, in the future we plan to incorporate extensions proposed in the motor control literature, such as sensorimotor noise and Cost of Time theory.

Our model is inspired by Todorov's E-LQG model [32]. To apply it to our HCI purposes, the following three main difficulties need to be dealt with: First, Todorov's model replicates many phenomena observed in human movement only qualitatively; there is no known method for adjusting the model to replicate specific experimental data. Second, the exact movement time needs to be known in advance, which is rarely the case in HCI. Third, motor control models usually model movement of the human body per se, e.g., movement of the hand as measured through motion capture or a stylus tablet, while the mouse has been avoided. Mouse pointer movements, however, are modified by sensor characteristics such as mouse sensor rotation and calculations on the microcontroller and in the operating system. It is unclear whether models that have been developed for understanding natural human (hand) movements are also good models for mouse pointer movements.

In this paper we present an OFC model that addresses all these points. Based on OFC theory (see Figure 1), our two key assumptions are first that control of the system is calculated via *optimization*, i.e., by minimizing a certain cost function. Second, the control is obtained in a *feedback* manner, i.e., it depends on the system state. To provide a simple model to introduce OFC to HCI and the modeling of mouse pointer movements, we make four key simplifications.

First, following existing literature, we require the cost function that users are assumed to minimize to be *quadratic*. In pointing tasks, people aim at bringing the end-effector to the target. For various settings, this has been modeled in OFC literature through *quadratic distance costs* that penalize the distance of the end-effector to the target center [32, 8, 27], see also [14]. At the same time, people aim at minimizing their effort and moving smoothly. The common model for the latter is that users aim to minimize the jerk of the movement [13]. Thus, similar to Todorov [32], we assume the cost function to include terms for penalizing the distance between pointer and target as well as terms to penalize the jerk.

Second, we assume *linear dynamics* of the mouse pointer (the *System* block in Figure 1). More precisely, as in Todorov [32], our system dynamics are described by a second-order lag.

With the third and fourth simplification, we deviate from Todorov [32]: We assume that there are no internal *delays* in the model. Moreover, we do not model noise and thus have a *deterministic model*. As a result, our approach quantitatively predicts position and velocity of the mouse pointer over time. In this deterministic setting, fitting the model parameters to the behavior of particular users in a specific task becomes easier.

To summarize, we assume *optimal closed-loop behavior* with respect to a *quadratic cost function* (that penalizes the jerk as well as the distance to the target) and subject to *linear system dynamics* (second-order lag) with *no delay* and *no noise*. These simplifications allow us to solve the optimal control problem using a simple optimal feedback controller, LQR, as explained in the next section.

THE MODEL

Since mouse sensor data are available in discrete time, we use discrete-time dynamics. The state of the system is given by a vector x_n that includes the position and velocity of the virtual mouse pointer. The user controls the mouse pointer by a force u_n, which influences the state x_n. Both are given at the discrete time steps $n \in \{1,\dots,N\}$ up to some final $N \in \mathbb{N}$. The next state x_{n+1} depends on the current state x_n and control u_n, as described by

$$x_{n+1} = Ax_n + Bu_n, \tag{1}$$

where the initial state x_1 is given. In this, the matrix A describes how the system, e.g., the mouse pointer dynamics described by a second-order lag, evolves when no control is exerted. The matrix B describes how the control influences the system. In this paper we look at 1D pointing tasks, in which the mouse can only be moved horizontally. Thus, in our case, the state x_n encodes the horizontal position and velocity of the pointer, denoted by $p_n \in \mathbb{R}$ and $v_n \in \mathbb{R}$, respectively, as well as a target position $T \in \mathbb{R}$ for technical reasons (in order to later be able to compute the distance to the target), i.e.,

$$x_n := (p_n, v_n, T)^\top . \tag{2}$$

This model can easily be extended to 2D or 3D pointing tasks by augmenting x_n and u_n with the respective components for the additional dimensions.

As a model for the mouse pointer dynamics we use the second-order lag, as depicted in Figure 2(a). The parameters of the model are the stiffness of the spring $k > 0$ and the damping factor $d > 0$. The mass is a redundant parameter and does not change the qualitative behavior of the model. We therefore set it to 1. In continuous time, we denote the position of the mouse pointer as $y(t)$, and its first and second derivatives with respect to time (i.e., velocity and acceleration) as $\dot{y}(t)$ and $\ddot{y}(t)$, respectively. The behavior is then described by the second-order lag equation

$$\ddot{y}(t) = u(t) - ky(t) - d\dot{y}(t), \tag{2OL}$$

cf. Figure 2(b). We derive a discrete-time version of (2OL) via the forward Euler method, with a step size of $h = 2ms$, where the two milliseconds correspond to the mouse sensor sampling rate. From this, we obtain the matrices A and B for (1) as

$$A := \begin{pmatrix} 1 & h & 0 \\ -hk & 1-hd & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} 0 \\ h \\ 0 \end{pmatrix}. \tag{3}$$

This process is similar to the one used by Todorov [32].

Next, we design the cost function J_N that we assume the user to minimize, based on our modeling assumptions. We want to penalize the jerk and the distance to the target. Ideally, no distance costs should occur within the target, which is a box with target width W. Unfortunately, this is infeasible in our LQR setting, where we need cost terms to be quadratic. To circumvent this limitation, we construct the distance costs such that we have lower costs inside the target and higher costs outside. At time step n, the *remaining* distance to the target is given by $D_n := |p_n - T|$, and we define the resulting distance costs as the square of that:

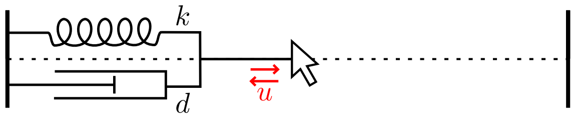
$$D_n^2 = (p_n - T)^2. \tag{4}$$

As in Todorov [32], the jerk in our case corresponds to the derivative of the control u. We call j_n the approximation of the jerk at time step n obtained by backward differences, i.e., $j_n := (u_n - u_{n-1})/h \approx \dot{u}_n$. We square this term to get positive values only. A weight factor $r > 0$ describes how important the jerk is compared to the positional error (4). Thus, our jerk costs are

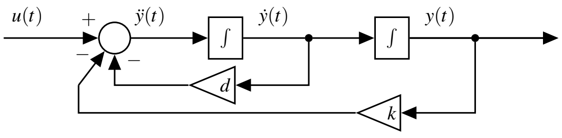
$$rj_n^2 = r\left(\frac{u_n - u_{n-1}}{h}\right)^2. \tag{5}$$

Formally, this approach requires a value u_0 to be chosen, which we will explain later.

Our overall cost function J_N will depend on different summations of the distance costs (4) and the jerk costs (5) over multiple time steps. In order to design a cost function J_N that explains user behavior best, we explore three different cost functions of this type later in the paper.



(a) Mouse pointer model with spring and damper



(b) Control-flow diagram

Figure 2. Illustrations of the second-order lag (2OL)**.**

In conclusion, we model the process of pointing through the following optimal control problem:

$$\min_{x,u} J_N(x,u) \quad \text{subject to} \quad x_{n+1} = Ax_n + Bu_n, \tag{OCP}$$

for a given initial control u_0 and initial state x_1, and where the matrices A and B are given by (3) and the function J_N is some summation of (4) and (5) over multiple time steps.

We assume that the user computes the *optimal* control u_n, which we denote by u_n^*, in a feedback manner. It has been proven that for these kinds of problems the optimal control u_n^* depends linearly on the state [9]. In our case, the optimal control u_n^* can be calculated simply by multiplying a matrix $-K_n$ with the state x_n, extended[1] by the previous control u_{n-1}^*:

$$u_n^* = -K_n \begin{pmatrix} x_n \\ u_{n-1}^* \end{pmatrix}. \tag{6}$$

The matrix K_n is called the *feedback gain* at time step n. It can be computed directly, given the matrices A, describing the mouse pointer dynamics, and B, describing how control influences the mouse pointer, and the cost function J_N. This is done by solving the appropriate *Discrete Riccati Equation*, see [32, Theorem 7].

The main question now is whether this optimal feedback corresponds to users' behavior, i.e., if our approach is suitable to describe pointing tasks. For this purpose, we note that there are several free parameters that we can choose: the spring stiffness k, the damping d, and the jerk weight r. The goal is to choose these parameters such that users' behavior is approximated best.

PARAMETER FITTING

In contrast to the non-deterministic E-LQG model of Todorov [32], one main strength of our deterministic model is that we can imitate user data without information about the end time of the movement. In addition, the calculation of optimal parameters is simplified by eliminating uncertainties. In

[1]This extension is required in order to penalize the jerk as in (5).

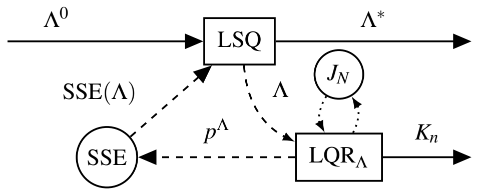


Figure 3. Starting with an initial parameter set $\Lambda = \Lambda^0$, the least squares (LSQ) algorithm obtains the sum squared error value (SSE) for the currently considered parameter set Λ. To do this, it calls LQR$_\Lambda$, which sets up the respective optimal control problem (OCP) and obtains the corresponding optimal feedback gain K_n. The resulting position time series p^Λ is used to compute SSE(Λ), which is transmitted back to LSQ. As an LSQ algorithm, we use MATLAB's nonlinear least squares algorithm `lsqnonlin`, which uses a gradient-based search method to obtain the next set of parameters Λ until it convergences to an optimal parameter set Λ^* with minimal SSE. Finally, Λ^* is returned along with the respective optimal feedback gain matrices K_n.

this way, our model can replicate the behavior of a particular user in a particular task. To this end, we need to fit the free parameters k, d, and r, to the data. We denote the set of these parameters by $\Lambda = \{k,d,r\}$. The goal is to find the optimal set, Λ^*, in the sense that our model, with parameters Λ^*, yields a pointer trajectory that is as similar as possible to that of the user. To achieve this, we measure the difference between the model trajectory p^Λ and the user trajectory p^{USER} using the sum squared error (SSE):

$$\text{SSE}(\Lambda) = \sum_{n=1}^{N} \left(p_n^\Lambda - p_n^{\text{USER}}\right)^2. \tag{7}$$

We then apply the least squares (LSQ) algorithm depicted in Figure 3 to find the optimal parameter set Λ^* minimizing (7).

Least-squares-based algorithms may converge to local minima and not find a global minimum. Therefore, we execute the whole fitting process several times for randomly chosen starting parameter sets Λ^0. According to our simulations, 100 of such sets sufficed to provide results that would not improve further by iterating on more starting parameter sets.

POINTING TASK AND DATASET

To evaluate our model, we use the *Pointing Dynamics Dataset*. Task, apparatus, and experiment are described in detail in [25]. The dataset contains the mouse trajectory for a reciprocal pointing task in 1D for ID 2, 4, 6, and 8.

Pointing movements almost always start with a reaction time, in which velocity and acceleration of the pointer are close to zero. In real computer usage, the user usually takes some time to decide whether to move the mouse and to locate the target before initiating the movement. Therefore, one could speak of the movement beginning once the acceleration of the pointer reaches a certain threshold.

In the Pointing Dynamics Dataset we use, the trial started immediately when the previous trial was finished, i.e., after the mouse click, not when the user initiated the next movement. This results in a considerable variation in reaction times. Since some variants of our approach as well as the methods from the literature we use for comparison cannot properly handle reaction times, in each trial we ignore the data before the user starts moving. To be exact, we drop all frames before the acceleration reaches 0.5% of its maximum/minimum value (depending on the movement direction) for the first time in each trial.

Moreover, we ignore user mistakes by dropping the failed and the following trial. From all other trials of all participants and all tasks – 7732 trajectories in total – we have removed another 30 for which the optimally fitted damping parameter d was an outlier (more than three standard deviations from the mean). This was necessary due to numerical instabilities that occurred for these parameters, leading to erroneous calculations of the optimal control. All remaining 7702 trajectories are used in the later evaluation.

We use the raw, unfiltered position data in our parameter fitting process to avoid artifacts. The dataset also contains derivatives of user trajectories, which were computed by differentiating the polynomials of a Savitzky-Golay filter of degree 4 and frame size 101 [25]. We use this (filtered) data only for the computation of the reference control u_0 (see the next chapter) and for illustration purposes.

For the following plots, unless stated otherwise, we display one certain representative user trajectory, namely the 21st movement to the right of participant 1 for the ID 8 task with 765px distance and 3px target width. For comparison and validation, the plots of all 7702 trajectories are provided in the supplementary material.

ITERATIVE DESIGN OF THE COST FUNCTION

In this section we describe the iterative design of our cost function J_N that is utilized in the algorithm depicted in Figure 3. The three resulting approaches are denoted by 2OL-LQR with the corresponding numbering.

First Iteration: Distance Costs at Endpoint (2OL-LQR$_1$)

In our first iteration we use a cost function similar to the one used by Todorov [32] for the E-LQG model. In this function, jerk costs occur at every step. Distance costs, however, only occur in the time step in which the mouse is clicked (time step N). In particular, no distance costs occur at other time steps. Thus, the cost function is given by

$$J_N(x,u) = D_N^2 + r \sum_{n=1}^{N-1} j_n^2, \tag{8}$$

where $D_N = |p_N - T|$ is the *remaining* distance to the target center at the end of the movement, r is the weight of the jerk, and $j_n = (u_n - u_{n-1})/h$ is the jerk at time step n.

The initial pointer position and velocity are set from the data, i.e., $x_1 = (p_1^{\text{USER}}, v_1^{\text{USER}}, T)^\top$. Although the choice of u_0 does not have a direct impact on the system dynamics, the trajectory heavily depends on its value. This is due to j_1 penalizing the deviation of u_1 from u_0, which carries over to j_2, and so

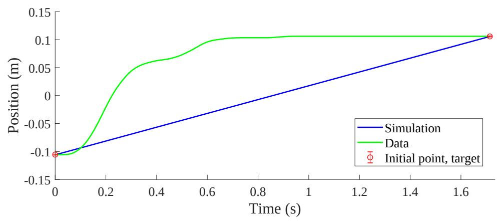


Figure 4. First iteration (2OL-LQR$_1$): Using a cost function similar to the one proposed by Todorov results in the model (blue) not replicating the data (green) well.

on.[2] We define u_0 such that if the first control u_1 coincides with u_0, the model will replicate the initial acceleration from the data a_1^{USER}, i.e., $u_0 = kp_1^{\text{USER}} + dv_1^{\text{USER}} + a_1^{\text{USER}}$.

The approach of using cost function (8) suffers from two major problems. First, as illustrated in Figure 4, the generated trajectories do not fit our data. In particular, the target is reached only at exactly the time of the mouse click. In contrast, our data shows that for high IDs, the users reach the vicinity of the target much earlier and then spend considerable time with small corrective submovements close to the target. The reason for this different behavior is that the cost function (8) sets the incentive to settle at the target only at the final time step N, while the jerk is penalized in every time step.

The second problem is that the cost function must include the exact time of the mouse click a priori. This makes the cost function very difficult to use for the simulation of human behavior in pointing tasks, if we cannot or do not want to prescribe a specific clicking time.

Hence, we propose a slightly modified cost structure in the LQR algorithm to take these considerations into account.

Second Iteration: Summed Distance Costs (2OL-LQR$_2$)

Both issues of the first iteration can be attributed to the fact that the remaining distance to target is only penalized at the time of the mouse click. Hence, we now penalize both the jerk and the distance between pointer position and target during the whole movement. Having summed costs over the entire movement is a standard approach in optimal control for such tracking tasks [5]. Our new cost function is

$$J_N(x,u) = D_N^2 + \sum_{n=1}^{N-1} \left(D_n^2 + rj_n^2\right), \tag{9}$$

where $D_n = |p_n - T|$ is the *remaining* distance to the target center after time step n. This changes the meaning of N: Instead of being the exact clicking time, it can now be interpreted as the maximum time allowed for the task. Thus, it is now much less important to set N accurately.

Optimal solutions of this approach with respect to the new cost function (9) approximate most of the considered user trajectories well, and much better than 2OL-LQR$_1$, cf. Figure 7.

[2]For example, setting $u_0 = 0$ might result in an implausibly high acceleration at the start of the movement, similar to 2OL-Eq.

Third Iteration: Reaction Time (2OL-LQR$_3$)

As explained in the dataset section, we prefer to model only the movement itself, excluding the reaction time. Thus, our second iteration does not model reaction time. In some cases, however, it is desirable to model it explicitly. In this section we present an objective function that achieves this.

To this end, we add a parameter $\delta > 0$ that should describe the reaction time. Due to our discrete time setting, we introduce $n_\delta \in \{1,\ldots,N\}$ as the discrete time step closest to δ. The idea is to adjust the cost function such that it incentivizes standing still until n_δ, to take reaction time into account.

We achieve this by splitting the cost function in two parts, before and after n_δ. In the first part, we assume that users are not aware of the target position or have at least not processed all required information for initiating the motion. In both cases, users should have no interest in changing their control. Therefore, we do not penalize the distance to the desired position in that time frame and employ a much higher jerk penalization compared to the main movement phase. More precisely, r is replaced by $f(n)\cdot r$, where $f(n)$ is, for the most part, an approximation of a very large constant c, e.g., $c = 100000$.[3] In the second part, i.e., starting from time step n_δ, we use the cost function (9) from 2OL-LQR$_2$.

In total, the cost function of 2OL-LQR$_3$ is

$$J_N(x,u) = D_N^2 + \sum_{n=1}^{n_\delta - 1} f(n) r j_n^2 + \sum_{n=n_\delta}^{N-1} \left(D_n^2 + rj_n^2\right). \tag{10}$$

There are several ways to obtain the reaction time δ and thus n_δ. One way is to determine it directly from the data, e.g., as the time when the acceleration passes a certain threshold. Another approach is to include it as an additional parameter to be optimized by the LSQ algorithm. We have chosen the latter approach and it works well according to our results.

RESULTS

In this section we evaluate our main model, 2OL-LQR$_2$, by comparing it to the minimum-jerk model from [13] (MinJerk) and the second-order lag with equilibrium control from [25] (2OL-Eq). We also investigate how the parameters of our model change for different tasks (IDs) and different users. Finally, we demonstrate the ability of 2OL-LQR$_3$ to model movements including a reaction time.

Minimum-Jerk Model by Flash and Hogan (MinJerk)

Flash and Hogan [13] show that the minimum-jerk trajectory between two points is a fifth-degree polynomial. They assume that velocity and acceleration are zero at the start and at the end of the movement, and explain how the parameters of this polynomial can be computed under these conditions. However, in our dataset, velocity and acceleration are not necessarily zero, neither at the beginning nor at the end of the movement. Therefore, before we delve into the results, we present the following technique to derive the parameters

[3]To aid the LSQ optimization process, we use a smoothed version of the piecewise constant sequence of jerk weights $c\cdot r$ and r, i.e., $f(n) := (c-1)\exp\left(\frac{1}{n_\delta - 1} - \frac{1}{n_\delta - n}\right) + 1$ for $n \in \{1,\ldots,n_\delta - 1\}$.

of the minimum-jerk polynomial under these different conditions.

Deriving the MinJerk Polynomial
In [13], the minimum-jerk polynomial is given by

$$p^{\mathrm{MinJerk}}(t) = \sum_{i=0}^{5} c_i \left(\frac{t}{t_f}\right)^i, \tag{11}$$

with coefficients $c_0, \ldots, c_5$ and where t_f is the final time of the movement. In our discrete-time setting, we evaluate the polynomial only at times $t_n = (n-1)h$, $n \geq 1$. In this case, the final time is given by $t_f = (\tilde{N}-1)h$, where $\tilde{N}$ is the last time step[4] and h is the same step size as before. Thus, the position at time step n is given by

$$p_n^{\mathrm{MinJerk}} = \sum_{i=0}^{5} c_i \left(\frac{n-1}{\tilde{N}-1}\right)^i. \tag{12}$$

The coefficients $c_0, \ldots, c_5$ are computed from the data: c_0 is the initial position, i.e., $c_0 = p_1^{\mathrm{USER}}$. The coefficients c_1 and c_2 are computed from initial velocity v_1^{USER} and acceleration a_1^{USER}. Since we have to take into account factors arising from differentiation, we arrive at $c_1 = v_1^{\mathrm{USER}} t_f$ and $c_2 = a_1^{\mathrm{USER}} t_f^2/2$. The remaining coefficients c_3, c_4, c_5 can be computed by solving the system of linear equations

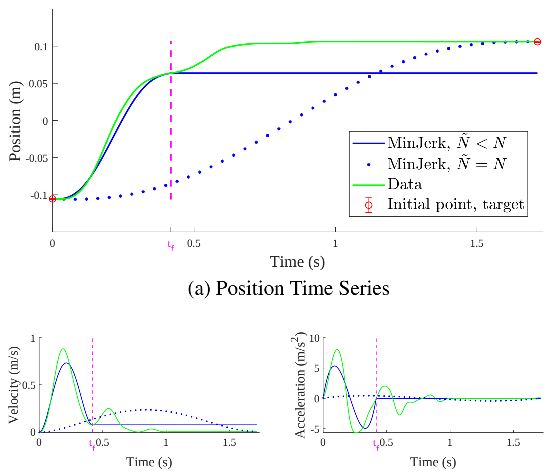
$$\begin{pmatrix} 1 & 1 & 1 \\ 3 & 4 & 5 \\ 6 & 12 & 20 \end{pmatrix} \begin{pmatrix} c_3 \\ c_4 \\ c_5 \end{pmatrix} = \begin{pmatrix} p_{t_f}^{\mathrm{USER}} - c_0 - c_1 - c_2 \\ v_{t_f}^{\mathrm{USER}} t_f - c_1 - 2c_2 \\ a_{t_f}^{\mathrm{USER}} t_f^2 - 2c_2 \end{pmatrix}, \tag{13}$$

where $p_{t_f}^{\mathrm{USER}}$, $v_{t_f}^{\mathrm{USER}}$, and $a_{t_f}^{\mathrm{USER}}$ are, respectively, the pointer position, velocity, and acceleration at the final time.

Results for MinJerk
The MinJerk model has been derived from data of an experiment that did not involve any corrective submovements [13]. This leaves two possibilities to fit the model to our data, which does show extensive corrective submovements. If MinJerk is used for modeling the entire movement, i.e., until time step N, the fit is very poor (see Figure 5; dotted line). Instead of a quick movement towards the target with extensive corrective submovements, as in our data, the model predicts a slow, smooth movement, reaching the target only at the time of the mouse click.

Therefore, we use MinJerk for only the first, rapid movement towards the target (the "surge"). Similar to [25], we determine the end of the surge (t_f in Figure 5) from the data as the first zero-crossing in the acceleration time series after the deceleration (for movements to the left: acceleration) phase. After that, we assume that the pointer does not move. As illustrated in Figure 5 (blue solid line), this results in a good fit of the surge phase, at least for movements that exhibit a clear surge phase. However, the target is not reached, causing a poor overall fit.

[4]We specifically do not use N for reasons elaborated below.



(a) Position Time Series

(b) Velocity Time Series (c) Acceleration Time Series

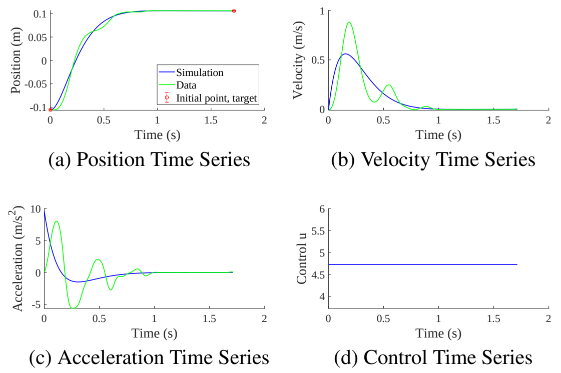
Figure 5. For the MinJerk model, we have to decide whether we want to model the surge well, but not reach the target (blue solid line with constant continuation after t_f), or reach the target, but not model the entire movement well (blue dotted line). In this paper we have chosen the former option. In this case t_f is the final time of the surge.

In conclusion, MinJerk is a good model for the surge phase but not suitable for describing motions that contain extensive corrective submovements.

Second-order Lag Equilibrium Control (2OL-Eq)
The 2OL-Eq model is a discrete version of (2OL) with $u \equiv kT$. It is given by the system dynamics $x_{n+1} = Ax_n + Bu_n$ with matrices A and B from (3) and initial condition $x_1 = (p_1^{\mathrm{USER}}, v_1^{\mathrm{USER}}, T)^\top$. With this particular choice of control, the pointer moves towards the target T and stays there. The target position T, together with zero velocity and acceleration, constitutes an equilibrium in this case; hence the name "equilibrium control". This constant control is the main difference to our approach, in which the control values u_n are optimized with respect to some cost function J_N.

For the 2OL-Eq model, we optimize the spring stiffness k and the damping d with the same parameter fitting process and the same SSE objective function (7) that we use for our 2OL-LQR approach.

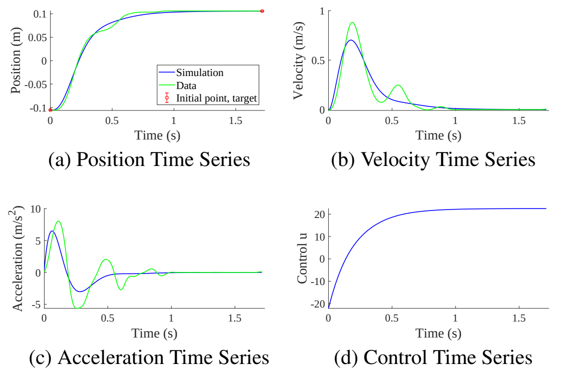
The behavior of the 2OL-Eq is shown in Figure 6. Visually, the model captures user behavior well in terms of pointer position, cf. Figure 6(a). The velocity time series depicted in Figure 6(b), however, is asymmetric in the 2OL-Eq case, while the user shows a more symmetric, bell-shaped velocity profile. The biggest difference appears in the acceleration time series. The user performs a symmetric and smooth N-shaped acceleration. In contrast, the acceleration of the 2OL-Eq jumps instantaneously at the start of the movement, and then rapidly declines. This can be explained with the physical interpretation of the 2OL-Eq as a spring-mass-damper system: Since u is constant in this model, as the system is released, the spring instantaneously accelerates the system with a force that is proportional to the extension of the spring. Because human mus-



(a) Position Time Series (b) Velocity Time Series

(c) Acceleration Time Series (d) Control Time Series

Figure 6. Due to the constant control, 2OL-Eq yields a much less symmetric velocity and acceleration profile during the surge than the user data.



(a) Position Time Series (b) Velocity Time Series

(c) Acceleration Time Series (d) Control Time Series

Figure 7. Our second iteration model 2OL-LQR$_2$ models the entire movement well. However, the acceleration in the surge phase is slightly less symmetric than the one of the user.

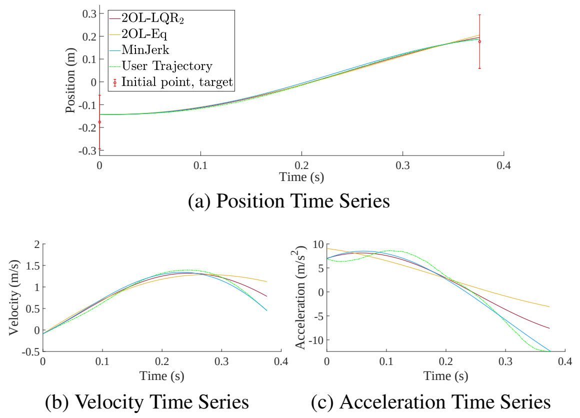
cles cannot build up force instantaneously [29], this behavior is not physically plausible.

Our Model 2OL-LQR$_2$ vs. MinJerk and 2OL-Eq

Qualitative Comparison

For the qualitative comparison, we performed a visual analysis of model behavior on the entire dataset. Although in the figures we illustrate a particular movement of a specific participant, we recall that the behavior is representative and the plots of all 12 participants and all 4 IDs are provided as supplementary material.

The behavior of our model 2OL-LQR$_2$ is shown in Figure 7. Overall, the model approximates the position rather well over the entire movement, cf. Figure 7(a). Corrective submovements, which start at around $t = 0.4s$, are not replicated well by any of the three models (see Figures 5, 6, and 7). Our model slightly underestimates the maximum velocity and the velocity profile is less symmetric than the data. Similar effects can be observed in the acceleration, see Figure 7(c).



(a) Position Time Series

(b) Velocity Time Series (c) Acceleration Time Series

Figure 8. ID 2 tasks without a correction phase are well approximated by each of the three considered models (here: Participant 1, 1275px distance, 425px target width, 35th movement to the right).

Compared to MinJerk, our model 2OL-LQR$_2$ explains the surge phase similarly well, while not quite capturing the symmetry observed in many acceleration time series as the one depicted in Figures 5, 6, and 7.[5] However, as a major improvement compared to MinJerk, 2OL-LQR$_2$ captures the entire movement, not just the surge phase. We emphasize that MinJerk is given the end point of the surge, as well as position, velocity and acceleration at that point, while our model is not given that information.

Compared to 2OL-Eq, our model captures position, velocity, and acceleration much better. The reason for this is that, in contrast to 2OL-Eq, the control time series shown in Figure 7(d) is not constant but changes over time. This often leads to a more N-shaped acceleration time series and a more bell-shaped velocity time series, as predicted by Flash and Hogan [13] and in many cases confirmed by our data.

ID 2 tasks play a special role, as they (usually) do not involve corrective submovements, see Figure 8. In this case, all three models match the position data. Visible differences in the fit appear in the velocity and acceleration data.

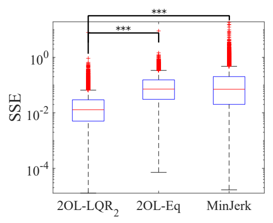
Quantitative Comparison

In the following, we provide a quantitative comparison across all 7702 trajectories. The resulting SSE values of all three models are shown in Figure 9(a), on a logarithmic scale. In addition, we measure the *Maximum Error* between model and user trajectories, i.e.,

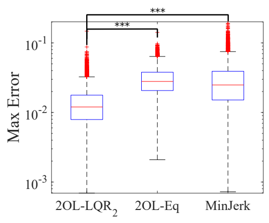
$$\max_{n=1,\ldots,N} |p_n^{\Lambda} - p_n^{\text{USER}}|, \qquad (14)$$

which is depicted in Figure 9(b). As can be seen from both Figures, our model 2OL-LQR$_2$ is able to capture human behavior substantially better in terms of SSE and in terms of Maximum Error than both the 2OL-Eq and MinJerk models.

[5]There are some cases in which asymmetric acceleration time series do occur. Our model 2OL-LQR$_2$ is able to approximate these profiles reasonably well and is not limited to, e.g., an N-shaped acceleration profile, as is the case with MinJerk.



(a) SSE



(b) Maximum Error

Figure 9. SSE and Maximum Error values of our model 2OL-LQR$_2$ compared to 2OL-Eq and MinJerk for the user trajectories of all participants and all tasks (logarithmic scale).

Model	SSE			Maximum Error		
	Mean	SE	SD	Mean	SE	SD
2OL-LQR$_2$	0.03	0.001	0.10	0.014	0.0001	0.009
2OL-Eq	0.11	0.002	0.16	0.03	0.0001	0.013
MinJerk	0.21	0.006	0.56	0.035	0.0025	0.022

Table 1. Mean value, standard error (SE), and standard deviation (SD) of the SSE and Maximum Error values of each model applied to the 7702 user trajectories.

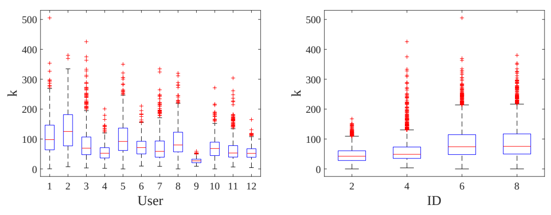
Kolmogorov-Smirnov tests showed that the distributions of SSE for the three models do not fit the assumption of normality (all values $p < 0.0001$). Thus, we carried out a Friedman Test (i.e., a non-parametric test equivalent to a repeated measures one-way ANOVA). The main factor included in the analysis was which model was used: 2OL-LQR$_2$, 2OL-Eq, or MinJerk. The significance level was set to 0.05. The test indicated that the SSE between the three models was significantly different ($\chi^2(2) = 8492.78$, $p < 0.001$, $n = 7702$).

Additional Wilcoxon Signed Rank tests with Bonferroni corrections showed that the SSE was significantly lower in the 2OL-LQR$_2$ model when compared to the 2OL-Eq model ($Z = -74.87$, $p < 0.001$), or to the MinJerk model ($Z = -68.49$, $p < 0.001$). The findings are analogous for the maximum deviations of the simulated trajectories from the data (Friedman Test, $\chi^2(2) = 9106.12$, $p < 0.001$, $n = 7702$), with Wilcoxon Signed Rank tests ($p < 0.001$) showing that 2OL-LQR$_2$ approximates user trajectories significantly better than both 2OL-Eq and MinJerk. Summary statistics of both measures for all three models can be found in Table 1.

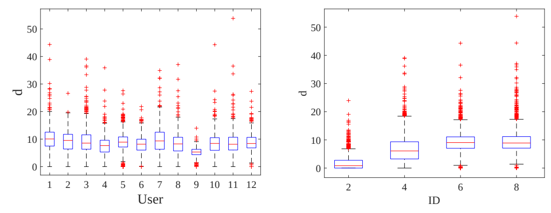
Parameter Distribution of 2OL-LQR$_2$

Figures 10(a)-(c) (left) show the ranges of the three 2OL-LQR$_2$ parameters k, d, and r, optimized for the user trajectories of all tasks with ID > 2, grouped by participants.[6] As can be seen, different participants are characterized by differing parameter sets. For example, participant 2 is characterized by a high spring stiffness k, an above-average damping d, and a very low jerk weight r. In contrast, participant 9 is characterized by a very low spring stiffness k, a very low damping d, and a very high jerk weight r. Since in our case higher jerk penalization enforces less rapid changes in control, from the jerk weight r it can be inferred how much *effort* the user is willing to put into the task: a higher r can be interpreted as less effort.

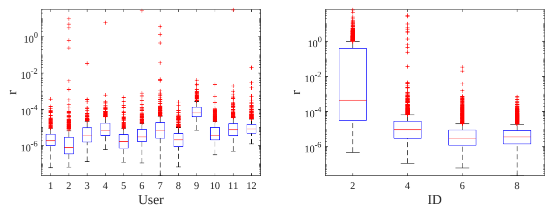
[6] The parameters for ID 2 tasks differ from those of ID > 2 tasks. Due to limited space, we focus on the latter in these plots. For the sake of completeness, the figures including ID 2 tasks can be found in the supplementary material.



(a) Parameter k



(b) Parameter d

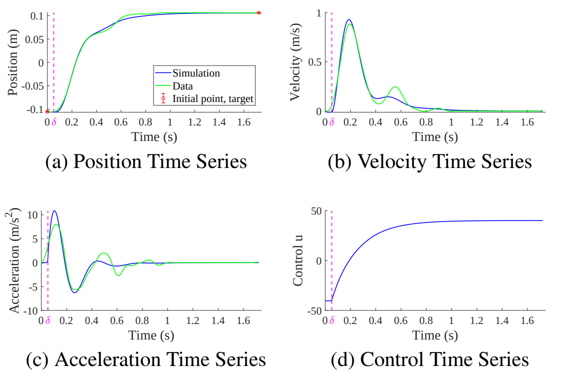


(c) Parameter r (logarithmic scale)

Figure 10. Parameters of our model 2OL-LQR$_2$, optimized for all considered trajectories of all participants and all tasks, grouped by participants (left, only ID 4, 6, 8 tasks) and by ID (right). For reasons of clarity, both plots for parameter d do not include the five biggest outliers ranging between 58 and 181.

Figures 10(a)-(c) (right) illustrate the ranges of the parameters k, d, and r, optimized for the user trajectories of all participants, grouped by ID of the task. All three parameters show characteristic variations by ID. The spring stiffness k increases noticeably from ID 4 to ID 6. The damping parameter d is considerably lower for ID 2 tasks. This confirms the observation that participants show oscillatory behavior in tasks with low IDs, as reported before in [16, 3, 25]. These oscillations also play a role in the large variance of r for ID 2. For the other IDs, r declines only slightly with ID, i.e., the effort is almost independent of the task difficulty.

The impact of the parameters on model behavior is however not straightforward, because a change in one of the parame-



(a) Position Time Series (b) Velocity Time Series (c) Acceleration Time Series (d) Control Time Series

Figure 11. Our third iteration model 2OL-LQR$_3$ allows to model individual movements by including reaction time.

ters does not only influence the movement directly, but also results in a different optimal control sequence, which likewise affects the solution trajectory.

Modeling Individual Movements Including Reaction Time

Our model 2OL-LQR$_2$ does not take reaction time into account. However, this is possible with our third iteration, 2OL-LQR$_3$. Only in this section, we thus explicitly do *not* drop any frames at the beginning of the trials. Results for the same representative trial as before are shown in Figure 11. Clearly, there is no change in control and thus in acceleration before time δ, which can loosely be interpreted as a reaction time. Looking closely at the initiation of the acceleration, we observe that our model initiates the movement later than the user but with a higher acceleration. The reason is that the optimizer treats δ as a free parameter to minimize the SSE of the entire position time series. Thus, while movements including reaction time can be approximated by 2OL-LQR$_3$ quite well, the parameter δ itself does not necessarily resemble the true reaction time.

DISCUSSION AND FUTURE WORK

In this paper we have explored a simple OFC model for mouse pointer movements. We assumed *optimal closed-loop behavior* with respect to a *quadratic cost function* (penalizing jerk and distance) and subject to *linear system dynamics* with *no delay* and *no noise*. These simplifications lead to a number of limitations of our model.

First, all models that we compared do not model corrective submovements well. Although our models can recreate corrective submovements (e.g., in Figure 11), they are smaller in amplitude than those of the users. Future research should put more emphasis on replicating these submovements in more detail by extending the model.

Second, due to its deterministic nature, our model cannot replicate the variability of human movements. It produces a typical movement of a specific user, but it produces the same movement every time. In future work we plan to explore stochastic models to better capture human variability.

Third, we note that although our cost function (9) of our main model, 2OL-LQR$_2$, incentivizes a short(er) movement time due to summed distance costs, it does not explicitly model minimizing the total movement time. If the latter is desired (e.g., as part of the experimental design), then in future work the model can be extended by modifying the cost function using the Cost of Time theory.

Despite these limitations, our 2OL-LQR$_2$ model matches our data well, and significantly better than 2OL-Eq or MinJerk. We achieve this with only three parameters, which have an easily understandable interpretation as spring stiffness k, damping d, and effort, related to r. We only need these parameters, the target position, and initial conditions. In contrast to MinJerk, our model does not need to know the point in time and space where the surge movement ends. Most importantly, our model does not require knowledge about the exact time when the target is reached. Compared to 2OL-Eq, our model yields a more bell-shaped velocity time series and a more N-shaped acceleration time series, without implausibly high acceleration at the start of the movement. In addition, our model explains how users differ from each other in properties (stiffness, damping) and effort.

The biggest strength is that the OFC perspective makes our model very flexible and easily extensible. In particular, it can readily be extended to other instructions, such as emphasizing speed vs. comfort. It can also be extended to different tasks, such as 2D or 3D pointing, 6 DoF docking tasks, etc.

It is important to highlight that our model is a pure end-effector model of the movement of the mouse pointer. We do not explicitly model biomechanics, sensor characteristics, or transfer functions in the operating system. Incorporating these is possible, albeit yielding nonlinear system dynamics, and therefore making the model more complex. Our simple model already works quite well for modeling mouse pointer movements. This reinforces our argument that OFC is a promising theory to better understand movement, such as movement of the mouse pointer, during interaction and is thus a valuable addition to the HCI community.

CONCLUSION

In this paper, we have modeled mouse pointer movements from an optimal control perspective. More precisely, we have investigated the Linear-Quadratic Regulator with various objective functions. We found that our model 2OL-LQR$_2$ fits our data significantly better than either 2OL-Eq [25] or MinJerk [13]. We require a number of simplifying assumptions (linear dynamics, quadratic costs). Despite these, mouse pointer movements of real users can be explained well. Moreover, this is achieved with only three, intuitively interpretable, parameters, which allow to characterize users by properties (stiffness, damping) and effort. In conclusion, we believe that the optimal feedback control perspective is a strong, flexible, and very promising direction for HCI, which should be further explored in the future.

REFERENCES

[1] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. 2005.

Predictive interaction using the delphian desktop. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. ACM, 133–141.

[2] Bastien Berret and Frédéric Jean. 2016. Why Don't We Move Slower? The Value of Time in the Neural Control of Action. *Journal of Neuroscience* 36, 4 (2016), 1056–1070. DOI: http://dx.doi.org/10.1523/JNEUROSCI.1921-15.2016

[3] Reinoud J. Bootsma, Laure Fernandez, and Denis Mottet. 2004. Behind Fitts' law: kinematic patterns in goal-directed movements. *International Journal of Human-Computer Studies* 61, 6 (2004), 811–821.

[4] Daniel Bullock and Stephen Grossberg. 1988. Neural Networks and Natural Intelligence. Massachusetts Institute of Technology, Cambridge, MA, USA, Chapter Neural Dynamics of Planned Arm Movements: Emergent Invariants and Speed-accuracy Properties During Trajectory Formation, 553–622. http://dl.acm.org/citation.cfm?id=61339.61351

[5] Y. Chan and J.-P. Maille. 1975. Extension of a linear quadratic tracking algorithm include control constraints. *IEEE Trans. Automat. Control* 20, 6 (December 1975), 801–803. DOI: http://dx.doi.org/10.1109/TAC.1975.1101101

[6] Frederic Crevecoeur, Tyler Cluff, and Stephen H. Scott. 2014. The Cognitive Neurosciences, 5th ed. MIT Press, Cambridge, MA, USA, Chapter Computational Approaches for Goal-Directed Movement Planning and Execution, 461–477.

[7] E. R. F. W. Crossman and P. J. Goodeve. 1983. Feedback control of hand-movement and Fitts' law. *The Quarterly Journal of Experimental Psychology* 35, 2 (1983), 251–278.

[8] Jörn Diedrichsen, Reza Shadmehr, and Richard B. Ivry. 2010. The coordination of movement: optimal feedback control and beyond. *Trends in Cognitive Sciences* 14, 1 (2010), 31 – 39. DOI: http://dx.doi.org/10.1016/j.tics.2009.11.004

[9] P. Dorato and A. Levis. 1971. Optimal linear regulators: The discrete-time case. *IEEE Trans. Automat. Control* 16, 6 (December 1971), 613–620. DOI: http://dx.doi.org/10.1109/TAC.1971.1099832

[10] Digby Elliott, Werner Helsen, and Romeo Chua. 2001. A century later: Woodworth's (1899) two-component model of goal-directed aiming. *Psychological bulletin* 127 (06 2001), 342–57. DOI: http://dx.doi.org/10.1037//0033-2909.127.3.342

[11] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6 (1954), 381–391.

[12] Paul M. Fitts and James R. Peterson. 1964. Information capacity of discrete motor responses. *Journal of experimental psychology* 67, 2 (1964), 103.

[13] Tamar Flash and Neville Hogan. 1985. The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. *Journal of neuroscience* 5 (1985), 1688–1703.

[14] J. Gori and O. Rioul. 2018. Information-Theoretic Analysis of the Speed-Accuracy Tradeoff with Feedback. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 3452–3457. DOI: http://dx.doi.org/10.1109/SMC.2018.00585

[15] Julien Gori, Olivier Rioul, and Yves Guiard. 2018. Speed-Accuracy Tradeoff: A Formal Information-Theoretic Transmission Scheme (FITTS). *ACM Trans. Comput.-Hum. Interact.* 25, 5, Article 27 (Sept. 2018), 33 pages. DOI: http://dx.doi.org/10.1145/3231595

[16] Yves Guiard. 1993. On Fitts's and Hooke's laws: Simple harmonic movement in upper-limb cyclical aiming. *Acta psychologica* 82, 1 (1993), 139–159.

[17] Christopher M. Harris and Daniel M. Wolpert. 1998. Signal-dependent noise determines motor planning. *Nature* 394, 6695 (1998), 780–784. DOI: http://dx.doi.org/10.1038/29528

[18] Bruce Hoff. 1994. A model of duration in normal and perturbed reaching movement. *Biological Cybernetics* 71, 6 (01 Oct 1994), 481–488. DOI: http://dx.doi.org/10.1007/BF00198466

[19] Bruce Hoff and Michael A. Arbib. 1993. Models of Trajectory Formation and Temporal Interaction of Reach and Grasp. *Journal of Motor Behavior* 25, 3 (1993), 175–192. DOI: http://dx.doi.org/10.1080/00222895.1993.9942048 PMID: 12581988.

[20] Y. Jiang, Z. Jiang, and N. Qian. 2011. Optimal control mechanisms in human arm reaching movements. In *Proceedings of the 30th Chinese Control Conference*. 1377–1382.

[21] Gary D. Langolf, Don B. Chaffin, and James A. Foulke. 1976. An Investigation of Fitts' Law Using a Wide Range of Movement Amplitudes. *Journal of Motor Behavior* 8, 2 (1976), 113–128. DOI: http://dx.doi.org/10.1080/00222895.1976.10735061 PMID: 23965141.

[22] Zhe Li, Pietro Mazzoni, Sen Song, and Ning Qian. 2018. A Single, Continuously Applied Control Policy for Modeling Reaching Movements with and without Perturbation. *Neural Computation* 30, 2 (2018), 397–427. DOI: http://dx.doi.org/10.1162/neco_a_01040 PMID: 29162001.

[23] I. Scott MacKenzie. 1992. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Human–Computer Interaction* 7, 1 (1992), 91–139. DOI: http://dx.doi.org/10.1207/s15327051hci0701_3

[24] David E. Meyer, Richard A. Abrams, Sylvan Kornblum, Charles E. Wright, and J. E. Keith Smith. 1988. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological review* 95, 3 (1988), 340.

[25] Jörg Müller, Antti Oulasvirta, and Roderick Murray-Smith. 2017. Control Theoretic Models of Pointing. *ACM Trans. Comput.-Hum. Interact.* 24, 4, Article 27 (Aug. 2017), 36 pages. DOI: http://dx.doi.org/10.1145/3121431

[26] Réjean Plamondon and Adel M. Alimi. 1997. Speed/accuracy trade-offs in target-directed movements. *Behavioral and brain sciences* 20, 02 (1997), 279–303.

[27] Ning Qian, Yu Jiang, Zhong-Ping Jiang, and Pietro Mazzoni. 2013. Movement Duration, Fitts's Law, and an Infinite-Horizon Optimal Feedback Control Model for Biological Motor Systems. *Neural Computation* 25, 3 (2013), 697–724. DOI: http://dx.doi.org/10.1162/NECO_a_00410 PMID: 23272916.

[28] Philip Quinn and Shumin Zhai. 2016. Modeling Gesture-Typing Movements. *Human–Computer Interaction* (2016), 1–47. DOI: http://dx.doi.org/10.1080/07370024.2016.1215922

[29] Richard A. Schmidt and Timothy D. Lee. 2005. *Motor Control and Learning*. Human Kinetics.

[30] Reza Shadmehr. 2010. Control of movements and temporal discounting of reward. *Current Opinion in Neurobiology* 20, 6 (2010), 726 – 730. DOI: http://dx.doi.org/10.1016/j.conb.2010.08.017 Motor systems, Neurobiology of behaviour.

[31] Reza Shadmehr and Steven P. Wise. 2005. *The Computational Neurobiology of Reaching and Pointing*. MIT Press.

[32] Emanuel Todorov. 1998. Studies of goal-directed movements. Massachusetts Institute of Technology. (1998).

[33] Emanuel Todorov. 2005. Stochastic Optimal Control and Estimation Methods Adapted to the Noise Characteristics of the Sensorimotor System. *Neural Computation* 17 (2005), 1084–1108.

[34] Emanuel Todorov and Michael I. Jordan. 2002. Optimal feedback control as a theory of motor coordination. *Nature neuroscience* 5, 11 (2002), 1226–1235.

[35] Y. Uno, M. Kawato, and R. Suzuki. 1989. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics* 61, 2 (01 Jun 1989), 89–101. DOI: http://dx.doi.org/10.1007/BF00204593

[36] Robert Sessions Woodworth. 1899. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements* 3, 3 (1899), i.

[37] Brian Ziebart, Anind Dey, and J. Andrew Bagnell. 2012. Probabilistic Pointing Target Prediction via Inverse Optimal Control. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces (IUI '12)*. ACM, New York, NY, USA, 1–10. DOI: http://dx.doi.org/10.1145/2166966.2166968

APPENDIX

2OL-LQR EQUATIONS

The 2OL-LQR model can be described as the time-discrete linear-quadratic optimal control problem with finite horizon $N \in \mathbb{N}$

$$\begin{aligned} &\textit{Minimize} \quad J_N(x,u) = \sum_{n=1}^{N} x_n^\top Q_n x_n + \sum_{n=1}^{N-1} (u_n - u_{n-1})^\top R_n (u_n - u_{n-1}) \\ &\textit{with respect to } u = (u_n)_{n\in\{1,\dots,N-1\}} \subset \mathbb{R} \textit{ given } \bar{x}_1 \in \mathbb{R}^3,\, \bar{u}_0 \in \mathbb{R} \end{aligned} \tag{15a}$$

where $x = (x_n)_{n\in\{1,\dots,N\}} \subset \mathbb{R}^3$ with $x_n = (p_n, v_n, T)^\top$ satisfies

$$\begin{aligned} x_{n+1} &= Ax_n + Bu_n, \quad n \in \{1,\dots,N-1\}, \\ x_1 &= \bar{x}_1, \end{aligned} \tag{15b}$$

with sampling time $h > 0$ and system dynamics matrices

$$A = \begin{pmatrix} 1 & h & 0 \\ -hk & 1-hd & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ h \\ 0 \end{pmatrix} \tag{15c}$$

based on the (approximated) second-order lag.
The state cost matrices are defined by

$$Q_n = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3}, \quad n \in \{1,\dots,N\}, \tag{16}$$

which implies

$$x_n^\top Q_n x_n = (T - p_n)^2 = D_n^2, \tag{17}$$

i.e., the distance $D_n = |T - p_n|$ between mouse and target position is quadratically penalized at every time step $n \in \{1,\dots,N\}$. In our case of one-dimensional pointing tasks, the control cost matrices are scalar and given by

$$R_n = \frac{r}{h^2} \in \mathbb{R}, \quad r > 0, \quad n \in \{1,\dots,N-1\}, \tag{18}$$

which yields

$$(u_n - u_{n-1})^\top R_n (u_n - u_{n-1}) = r_n \left(\frac{u_n - u_{n-1}}{h}\right)^2, \tag{19}$$

i.e., the squares of the "jerk" terms $j_n = \frac{u_n - u_{n-1}}{h}$ are penalized with some jerk weight r at every time step $n \in \{1,\dots,N-1\}$. Because of the penalization of the *differences* in control, each control value u_n^* of the optimal control sequence u^* minimizing $J_N(x,u)$ given some initial state $\bar{x}_1$ and some initial control $\bar{u}_0$ explicitly depends on the preceding control value u_{n-1}^*. For this reason, we need to introduce **information vectors**

$$\mathcal{I}_n = \begin{pmatrix} x_n \\ u_{n-1} \end{pmatrix} \in \mathbb{R}^4, \quad n \in \{1,\dots,N\}. \tag{20}$$

Furthermore, we expand the system matrices A and Q_n by an additional zero row and column and add an additional one to the control matrix B in order to propagate the previous control u_{n-1}:

$$\mathcal{A} = \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & h & 0 & 0 \\ -hk & 1-hd & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4},$$

$$\mathcal{B} = \begin{pmatrix} B \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ h \\ 0 \\ 1 \end{pmatrix} \in \mathbb{R}^{4\times 1},$$

$$\mathcal{Q}_n = \begin{pmatrix} Q_n & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4},$$
$$n \in \{1,\dots,N\}. \qquad (21)$$

Using this notion, (15) is equivalent to the following optimal control problem:

$$\text{Minimize} \quad \mathcal{J}_N(\mathcal{I},u) = \sum_{n=1}^{N} \mathcal{I}_n^\top \mathcal{Q}_n \mathcal{I}_n + \sum_{n=1}^{N-1} (u_n - u_{n-1})^\top R_n (u_n - u_{n-1})$$
$$\text{with respect to } u = (u_n)_{n\in\{1,\dots,N-1\}} \subset \mathbb{R} \text{ given } \bar{x}_1 \in \mathbb{R}^3,\ \bar{u}_0 \in \mathbb{R} \qquad (22a)$$

where $\mathcal{I} = (\mathcal{I}_n)_{n\in\{1,\dots,N\}} \subset \mathbb{R}^4$ with $\mathcal{I}_n = (x_n, u_{n-1})^\top$ satisfies

$$\mathcal{I}_{n+1} = \mathcal{A}\mathcal{I}_n + \mathcal{B}u_n, \quad n \in \{1,\dots,N-1\},$$
$$\mathcal{I}_1 = \bar{\mathcal{I}}_1 = \begin{pmatrix} \bar{x}_1 \\ \bar{u}_0 \end{pmatrix}, \qquad (22b)$$

with sampling time $h > 0$ and where $u_0 = \bar{u}_0$ applies. Moreover, we define

$$\mathrm{I}_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \in \mathbb{R}^{3\times 4}, \quad \mathrm{I}_u = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{1\times 4}, \qquad (23)$$

which implies

$$\mathrm{I}_x\mathcal{I}_n = x_n \in \mathbb{R}^3, \quad \mathrm{I}_u\mathcal{I}_n = u_{n-1} \in \mathbb{R}, \quad n \in \{1,\dots,N\}, \quad (24)$$

i.e., I_x respective I_u are the matrices that extract the state x_n respective the control u_{n-1} from the information vector $\mathcal{I}_n$ for any $n \in \{1,\dots,N\}$.

It can be shown that the unique solution $u^* = (u_n^*)_{n\in\{1,\dots,N\}}$ to the optimization problem (22) (and thus to the original optimization problem (15) as well) is given by

$$u_n^* = -K_n\mathcal{I}_n^*, \quad n \in \{1,\dots,N-1\},$$
$$K_n = (R_n + \mathcal{B}^\top \mathcal{S}_{n+1}\mathcal{B})^{-1}(\mathcal{B}^\top \mathcal{S}_{n+1}\mathcal{A} - R_n\mathrm{I}_u),$$
$$n \in \{1,\dots,N-1\}, \qquad (25)$$

where the symmetric matrices $\mathcal{S}_n \in \mathbb{R}^{4\times 4}$ can be determined by solving the **Modified Discrete Riccati Equations**

$$\mathcal{S}_n = \mathcal{Q}_n + \mathrm{I}_u^\top R_n \mathrm{I}_u + \mathcal{A}^\top \mathcal{S}_{n+1}\mathcal{A} -$$
$$-(\mathcal{A}^\top \mathcal{S}_{n+1}\mathcal{B} - \mathrm{I}_u^\top R_n)(R_n + \mathcal{B}^\top \mathcal{S}_{n+1}\mathcal{B})^{-1}(\mathcal{B}^\top \mathcal{S}_{n+1}\mathcal{A} - R_n\mathrm{I}_u) \qquad (26a)$$

for $n \in \{1,\dots,N-1\}$ backwards in time with initial value

$$\mathcal{S}_N = \mathcal{Q}_N. \qquad (26b)$$

5
Optimal Feedback Control for Modeling Human-Computer Interaction

Authors: Florian Fischer, Arthur Fleig, Markus Klar, Jörg Müller
Status: Published in ACM Transactions on Computer-Human Interaction, Volume 29, Issue 6 [17]

The Optimal Control Framework of HCI was developed by all authors. The models were selected and adapted by FF and JM. FF implemented the models, the parameter fitting, and the evaluation tools. User data was preprocessed by FF, with help of MK. All authors analyzed and evaluated the simulation results. Figures were mainly created by FF, with help of MK. The results were interpreted and discussed by all authors. FF wrote the first draft of the manuscript. Revision and rewriting of the manuscript was done by all authors. FF is the corresponding author.

Optimal Feedback Control for Modeling Human–Computer Interaction

FLORIAN FISCHER, ARTHUR FLEIG, MARKUS KLAR, and JÖRG MÜLLER,
University of Bayreuth

Optimal feedback control (OFC) is a theory from the motor control literature that explains how humans move their body to achieve a certain goal, e.g., pointing with the finger. OFC is based on the assumption that humans aim at controlling their body optimally, within the constraints imposed by body, environment, and task. In this article, we explain how this theory can be applied to understanding Human-Computer Interaction (HCI) in the case of pointing. We propose that the human body and computer dynamics can be interpreted as a single dynamical system. The system state is controlled by the user via muscle control signals, and estimated from observations. Between-trial variability arises from signal-dependent control noise and observation noise. We compare four different models from optimal control theory and evaluate to what degree these models can replicate movements in the case of mouse pointing. We introduce a procedure to identify parameters that best explain observed user behavior. To support HCI researchers in simulating, analyzing, and optimizing interaction movements, we provide the Python toolbox *OFC4HCI*. We conclude that OFC presents a powerful framework for HCI to understand and simulate motion of the human body and of the interface on a moment-by-moment basis.

CCS Concepts: • **Human-centered computing → HCI theory, concepts and models;**

Additional Key Words and Phrases: Optimal control, OFC, Human-Computer Interaction, modeling, parameter fitting, aimed movements, mouse pointing, LQR, LQG, second-order lag, minimum jerk, Intermittent Control

ACM Reference format:
Florian Fischer, Arthur Fleig, Markus Klar, and Jörg Müller. 2022. Optimal Feedback Control for Modeling Human–Computer Interaction. *ACM Trans. Comput.-Hum. Interact.* 29, 6, Article 51 (November 2022), 70 pages.
https://doi.org/10.1145/3524122

1 INTRODUCTION

We address the problem of understanding, and modeling, how users control a virtual end-effector when interacting with computers. Traditionally, the field of **Human–Computer Interaction** (**HCI**) has concentrated on models such as Fitts' Law [39, 40], predicting summary statistics of the movement such as movement time. Recently, more attention has been paid to modeling the underlying process by which the end-effector is controlled, predicting not only movement time, but end-effector position, velocity, and acceleration sequences, as well as applied forces (e.g., [34, 38, 98, 160]).

Authors' address: F. Fischer, A. Fleig, M. Klar, and J. Müller, University of Bayreuth, Bayreuth 95440, Germany; emails: {florian.j.fischer, arthur.fleig, markus.klar, joerg.mueller}@uni-bayreuth.de.

1073-0516/2022/11-ART51
https://doi.org/10.1145/3524122

We argue that, in order to understand how users control user representations (e.g., mouse pointer) [120], or virtual objects, the field of HCI needs to learn more from human motor control. While human motor control mainly addresses the question of how humans control the movement of their body, the theories developed there also apply to and can be adapted to the question of how humans control the state of a computer, e.g., movement of the mouse pointer.

In the field of human motor control, modern understanding of human movement is based on the theory of **optimal feedback control** (**OFC**) [33, 142]. This theory understands the human body, and possibly the environment the body is interacting with, as a dynamical system that can be controlled, e.g., via muscle control signals. Body and environment put constraints on this control, e.g., via the system dynamics and constant and signal-dependent motor noise. The theory assumes that humans continuously observe the state of their own body and the environment they are interacting with, e.g., by processing visual and proprioceptive signals. Humans are assumed to control their body optimally with respect to an internalized cost function, while respecting the constraints given by the system dynamics and motor noise.

We believe that the OFC framework enables a better connection between the field of HCI and recent advances in neighboring scientific disciplines, such as the study of human movement in motor control [42, 117] and neuroscience [123]. However, OFC models are not very well known in the field of HCI, yet. In particular, it has not yet been shown whether these models, developed to model how humans control their body, can be used to model how users behave during interaction.

The objective of this work is to examine the applicability of OFC to HCI, using the example of mouse pointing. The contribution of this article is fourfold:

First, we propose a unifying optimal control framework for understanding movement in interaction with computers. This framework allows to predict the kinematics and dynamics of the entire movement trajectory, including, e.g., end-effector position, or muscle excitation.

Second, we present the first qualitative and quantitative evaluation to what degree different optimal control models (either open- or closed-loop, deterministic or stochastic) can replicate movements of the mouse pointer. To the best of our knowledge, these models have not yet been evaluated quantitatively regarding their ability to predict movement trajectories during interaction. We also discuss the possibilities and limitations of the presented models regarding their suitability for other HCI tasks such as target tracking, path-following, or handwriting.

Third, we propose a generic parameter fitting process, which can be used to identify the components of both the system dynamics and the cost function that best explains observed user behavior, using any desired optimal control model. For each of the presented models, we systematically analyze the individual effects of the parameters and show how the proposed parameter fitting can be used to explain typical differences between users and/or task conditions, which would remain hidden when using summary statistics only.

Fourth, we provide *OFC4HCI*, an open-source toolbox accessible from our GitHub repository[1] that contains the underlying Python code of this article. This toolbox includes easy-to-use scripts for three main use cases: running simulations of human pointing movements using any of the presented control methods, comparing the resulting trajectories to data from the Pointing Dynamics Dataset, and optimizing the parameters of a given control model. While the focus of this toolkit currently is on (one- or multidimensional) pointing tasks, using the toolkit, extensions to other HCI tasks such as target tracking, keyboard typing, or gesture-based input methods are possible.

Our results suggest that stochastic OFC models are able to explain average user behavior significantly better than models that only account for simplified movement dynamics (second-order lag) or pure kinematic models (jerk minimization). In addition, stochastic models such as

[1]https://github.com/fl0fischer/OFC4HCI.

Linear-Quadratic Gaussian Regulator (**LQG**) are able to fit the *distribution* of entire trajectories, given a specific user and task condition. Moreover, the fitting is significantly better than using the recently proposed **Intermittent Control** (**IC**) model [160] with respect to both KL divergence [76] and the 2−Wasserstein distance [104] serving as evaluation metrics. The considered deterministic OFC model, which does not take into account any noise terms, is able to predict average user behavior, given a slightly modified cost function.

We strongly believe that a proper modeling of the underlying control process can provide intuition to interface designers as to why users move the way they do during interaction, and enables a deeper understanding of the impact of parameters of the interface and input device on the process of interaction. In the long term, such models could be used for automated optimization of the parameters of interaction techniques and input devices. Models that work in real-time could further be used in predictive interfaces, which anticipate what the user wants to do and respond accordingly, such as pointing target prediction [4]. While we will focus on the example of mouse pointing throughout this article, the framework we present is generic and suitable for a wide range of pointing devices, using, e.g., joysticks, keyboards, pens, touch-based input, mid-air gestures, and so on.

The article is structured as follows:

In Section 2, we start with a short overview of existing models and methods from the fields of HCI, Human Motor Control, and Optimal Control Theory. The proposed *optimal control framework for HCI* is then introduced in Section 3. The models presented in this article are evaluated against an existing dataset of one-dimensional pointing movements, which is described in Section 4.1. The generic parameter fitting process we use to identify the model parameters that best explain observed user behavior is described in Section 4.2.

In Sections 5–10, different optimal control models are presented, analyzed, and adapted to the case of mouse pointing. Moreover, the predicted movements are compared against user data. Since this article is also supposed to serve as a tutorial to OFC for HCI researchers and interaction designers, we start with an analysis of the individual components of the OFC framework before combining them into a final model. In Section 5, we start with a basic model of movement dynamics, the second-order lag, which has been used to describe the overall human–computer system dynamics [98] and serves as a baseline for the presented optimal control models. The idea of (open-loop) optimal control is introduced in Section 6, using the minimum jerk model [42]. In Section 7, both movement dynamics and the assumption of optimality are integrated into one closed-loop OFC model, the Linear-Quadratic Regulator, which is based on the assumptions of linear dynamics and quadratic costs [141]. From a didactic point of view, it is important to develop a thorough understanding of deterministic OFC before progressing to **stochastic OFC** (**SOFC**) models. For this reason, we first start with the (substantially simpler) deterministic case, which can be used to predict *average* human movement. In Sections 8 and 9, we extend this framework to the general stochastic case by adding different sensory-input models along with Gaussian motor and sensory noise (thus denoted as Linear-Quadratic Gaussian Regulator). We compare the SOFC models to a recently proposed IC model [160], which is briefly described in Section 10.

Finally, both qualitative and quantitative comparisons between all considered models are given in Section 11. Difficulties and limitations of the proposed framework with regard to its applicability to other HCI tasks are discussed in Section 12, together with some practical advice for HCI researchers, and conclusions are drawn in Section 13.

2 RELATED WORK

In the field of HCI, interaction is most commonly understood as a sequence of discrete actions, which is reflected in the classification of tasks, such as *command selection* or *target acquisition* [19].

In particular, movement, e.g., of the mouse pointer, is often reduced to summary statistics. The most prominent example is the dependency of movement time MT from distance D and width W of targets, which is described by Fitts' Law [39, 40] as $\mathrm{MT} = a + b\,\mathrm{ID}$, with **Index of Difficulty** (**ID**) usually defined as $\mathrm{ID} = \log_2(D/W + 1)$ [89]. This affine relationship has shown to apply for a variety of tasks, including reciprocal tapping [39], mouse pointing and dragging [21, 48], eye-gazing [64, 151], reaching with a joystick [21, 64], and ellipse drawing [96]. A very good explanation of the information theoretic interpretation of Fitts' Law has been provided by Gori et al. [51].

While aggregated metrics of movement trajectories, e.g., *movement variability* or *movement offset* [88], have been used to evaluate task accuracy since the early days of HCI [63], predictive models of movement kinematics and dynamics are less common. Exceptions include the works of Williamson [152, 153], which introduce an information-theoretic model of interaction with a focus on the amount of uncertainty that is apparent in different sensor and control channels, and Müller et al. [98], in which three feedback control models (without optimization) are compared regarding their ability to model mouse pointer movements. However, the former model is originally designed for the specific needs of brain-computer interfaces, particularly inference of the user's intention based on noisy signal channels, whereas the latter models only describe the biomechanical apparatus, while high-level factors affecting the movement trajectory such as concrete task requirements or intrinsic motivations are neglected. Ziebart et al. [159] explore the use of inverse optimal control models for pointing target prediction. They do not make particular a priori assumptions about the structure of the cost function. Instead, they use an inverse optimal control approach to fit a generic function with a large number of parameters (36) to a dataset of mouse pointer movements. While Ziebart et al. [159] focus on the application of inverse optimal control to pointing target prediction; in this article, we investigate the ability of OFC models to model movement of the mouse pointer more quantitatively. From an engineering perspective, several interaction techniques that take into account the underlying end-effector kinematics have been proposed, including cursor jumping [4, 99], target expansion [90], and increased cursor activation areas [24, 95]. These approaches are either based on target likelihood estimates [99, 159] or extrapolate sensor data measured during runtime [4, 90, 95]. Other methods compare observed trajectories to a set of pre-defined templates in order to predict the desired end-point ("kinematic template matching") [106]. In general, these methods are restricted to the kinematic end-effector level, i.e., they ignore the movement dynamics of the human body (which play a crucial role for non-standard interaction techniques such as gesture-based input) and cannot be used to model interaction with dynamic objects.

In addition to their functional use in HCI, movement dynamics have been a research focus within the field of **Motor Control** for a long time. Various models have been developed, all of which predict complete trajectories, e.g., end-effector position, velocity, and acceleration profiles over the entire movement (e.g., [17, 18, 42, 43, 53, 65, 77, 97, 109]). Biomechanical and neural models, in addition, explain how these trajectories are dynamically generated. This can be either done on the joint-, muscle-, or neuronal level, incorporating quantities internal to the human body such as joint angles, joint moments, muscle forces and activations, or neural excitation signals [8, 69, 70, 101, 114, 144].

Many models of motor control are also capable of modeling the characteristic between-trial variability that is typically observed in human movements. This variability is mainly attributed to multiple sources of noise within the human biomechanical and neural system, most of which can be modeled as additive or multiplicative Gaussian random variables [68, 118, 125, 129, 138, 141]. Signal-dependent noise terms, e.g., Gaussians with zero mean and with a standard deviation that linearly depends on the magnitude of the muscle control signal, are also considered responsible for the well-known *speed-accuracy tradeoff* in goal-directed human movements [58, 118, 129]. These noise terms have the effect that larger control signals, which might increase the speed of the end-effector, also result in larger deviations from the desired end-effector position. Users thus face a tradeoff between accurate achievement of the desired goal and fast, but noisy movements.

Another well-established finding from human motor control refers to the amount of information that is used when selecting a specific control signal. Several experiments suggest that information that becomes available to the controller during the movement, e.g., proprioceptive and/or visual signals regarding the end-effector, are utilized to adjust control signals online and to account for unexpected perturbations [93, 136, 142, 149, 154]. This is reflected by feedback control models of movement, which are able to explain how users correct errors and handle disturbances during the movement. An early closed-loop model has been provided by Crossman and Goodeve [29]. They assume that users observe hand and target and adjust their velocity as a linear function of the distance, as a first-order lag. A physically more plausible extension of the first-order lag is the second-order lag [29, 79]. These dynamics can be interpreted as a spring-mass-damper system, where a constant force is applied to the mass, such that the system moves to and remains at the target equilibrium. Because of its simplicity and widespread use, we use this model as a baseline, called *2OL-Eq*. Other models of human movement include VITE [18] and the models of Plamondon [110].

The *desired trajectory hypothesis* [71] assumes that whenever disturbances occur (e.g., due to internal control noise or external perturbations), feedback is used to push the end-effector toward a predetermined, deterministic trajectory that results from a separated planning phase. In contrast, Todorov and Jordan [142] have demonstrated that deviations are corrected only if they interfere with the task performance, i.e., deviations that are irrelevant for achieving the desired goal remain ignored. This *minimum intervention principle* particularly implies that all task-specific requirements (end-point position, movement time, accuracy, etc.) need to be reflected by an internal formulation that the controller has access to.

Optimal control models provide exactly this internal representation by assuming that humans try to behave optimally with respect to a certain internalized cost function. Flash and Hogan [42] proposed that humans aim at generating smooth movements by minimizing the jerk of the end-effector. We call this model *MinJerk* in the following. Although the hypothesis that people aim to minimize jerk has been questioned, see, e.g., Harris and Wolpert [58], the minimum jerk model is one of the most established models. For example, it has been successfully used by Quinn and Zhai [112] to model the shape of gestures on a word-gesture keyboard.

Most modern theories of motor control are based on OFC, i.e., they combine the assumptions of optimality and continuously perceived feedback for closed-loop control. Excellent overviews of recent progress in OFC theory are provided by Crevecoeur et al. [28] and Diedrichsen [33]. An early approach that models perturbed reach and grasp movements by using the minimum-jerk trajectory on a moment by moment basis was presented by Hoff and Arbib [61]. A more general, more recent, and better known OFC model is the LQG [62, 87], which was mainly used by Todorov to model human movement from a sensorimotor perspective [138, 141, 142]. In this work, we will present and discuss the assumptions and limitations of this model, and analyze its applicability to standard HCI tasks such as mouse pointing.

An important limitation of the LQG model (and many other optimal control models, e.g., [42, 58, 144]) is that the exact movement time needs to be known in advance. One way to circumvent this issue is to use infinite-horizon OFC [66, 85, 111], i.e., to formulate the **optimal control problem (OCP)** on an infinite-time horizon. With such models, (quadratic) distance and effort costs are usually applied continuously, resulting in an optimal trajectory that consists of both a transient phase (where the end-effector is moved toward the target) and a steady-state equilibrium (where the end-effector is kept at the target). The movement time thus emerges implicitly from the OCP.

Another strand of literature that specifically deals with the duration of movement has produced the *Cost of Time* theory [10, 60, 122]. To account for the fact that humans value earlier achievement more than later achievement, this theory assumes that time is explicitly penalized with a certain cost function (usually hyperbolic or sigmoidal).

Recently, methods from the field of **Reinforcement Learning (RL)** have gained increased attention. These methods are also based on the principles of optimal control theory; however, they do not require the system dynamics to be known in terms of equations and formulas, but solely rely on sampling from an environment that is usually assumed a black-box to the controller. For this reason, they are generally applicable to arbitrarily complex systems including highly non-linear dynamics and discontinuous cost functions [130].

Cheema et al. [25] have applied recent RL methods to predict fatigue during mid-air movements, using a torque-actuated linked-segment model of the upper limb. Building on this work, it has recently been shown that RL applied to a more realistic upper-limb model allows to synthesize human arm movements that follow both Fitts' Law and the 2/3 Power Law and can predict human behavior in mid-air pointing and path following tasks [38]. Moreover, an extension to mid-air keyboard typing has been proposed [59].

In theory, policy-gradient RL methods can also be applied to model interaction on a muscular level, using state-of-the-art biomechanical models of the human body [72, 81, 100, 137]. However, the high complexity of the neuromuscular system has so far imposed considerable restrictions to each of these approaches, including the reduction of degrees of freedom [72, 100, 137] and the omission of muscle activation dynamics [81, 100]. Most importantly, for most RL algorithms no theoretical convergence guarantees exist, which complicates a profound interpretation or replication of the resulting simulation results [130]. For this reason, in this article we focus on the well-understood theory of optimal control, as this allows us to use convergence guarantees more often, which makes us less reliant on intuition and experience. For example, the **Linear-Quadratic Regulator** (**LQR**) introduced in Section 7 is guaranteed to converge to the optimal movement trajectory, given a fixed set of parameters. This is a decisive advantage compared to pure RL-based methods, as it allows to compare optimal trajectories for different task conditions, cost functions, and user models.

In summary, the fundamental question of human movement coordination has produced a vast literature and a deep understanding of the nature of human movement. Given that almost every interaction of humans with computers involves movement of the body, it is surprising that this field is little known, and applied, in HCI. It is important to bear in mind; however, that most of these theories intend to model movement of the human body per se. In HCI, it is also relevant how users control the movement of user representations (e.g., mouse pointers) [120] and virtual objects in the computer. Since the control of user representations and virtual objects is mediated by input devices, operating systems, and programs, requires high precision, and is often learnt very well, it is unclear how the theory of human motor control can be applied to the HCI context. To our knowledge, it has not yet been investigated whether the above optimal motor control models can be applied to HCI tasks such as mouse movements. Adapting and validating such models regarding their ability to model HCI tasks such as pointing thus remains an open research question for HCI.

In order to leverage the strengths of recent motor control theory in the field of HCI, we believe that a general optimal control framework for HCI is necessary, which can explain both *how* and *why* humans behave in interaction with arbitrary interfaces on a continuous level. Such a framework constitutes a natural extension of the principle of "designing interaction, not interfaces" [7] by conceptualizing interaction based on neuroscientific, psychological, and biomechanical insights within one coherent and mathematically profound framework.

In the following section, we will introduce the optimal control framework for HCI and explain its main constituents using the example of mouse pointing.

3 INTRODUCING THE FRAMEWORK

Modern motor control theory assumes that humans aim at controlling their movements optimally, given the constraints imposed by the body and environment. Important constraints are imposed by physics, e.g., via Newton's second law, i.e., force equals mass times acceleration, and by the muscles, which cannot create forces instantaneously, but need to build up muscle activation (and thus force) over time. In the case of HCI, constraints are imposed not only by the human body, but also by the input device, sensor, and processing within the computer. Furthermore, the human perceptual system does not have direct access to the state of the world, but can only observe certain variables that depend on the state and needs to build up an internal estimate of the true world state over time.

Since these properties are characteristic for almost any HCI task, we propose a generic *optimal control framework for HCI*, which consists of four submodels that continuously interact with each other:

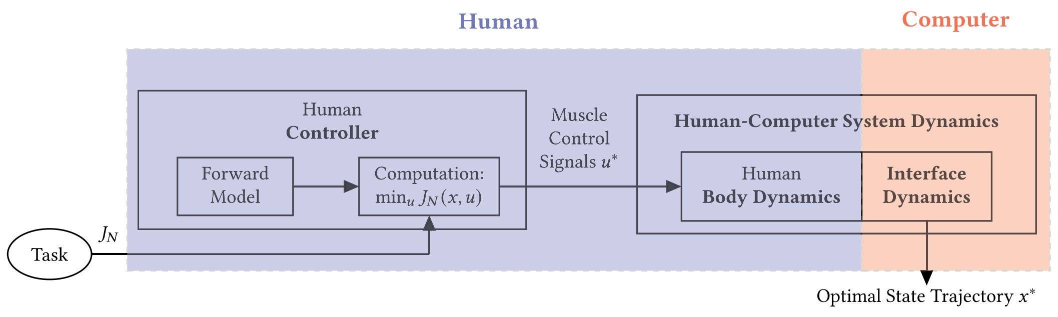
— The **Human–Computer System Dynamics**, which describe the biomechanics of the considered body parts as well as the dynamics of the resulting interaction with the application interface via an input device;
— The **Human Controller**, i.e., the decisive part of the brain, which selects the new muscle control signals;
— The **Feedback**, which models how environmental information is sensed by the human; and
— The **Human Observer**, a cognitive model for how perceived sensory signals from the Feedback are processed and evaluated.

Since the computer operates in discrete time, we use discrete-time dynamics, i.e., we consider timesteps $n \in \{0, \dots, N\}$ up to a final step $N \in \mathbb{N}$, with each time step corresponding to h seconds. However, the proposed framework is designed to be as general as possible and the following explanation also applies to the continuous-time case.

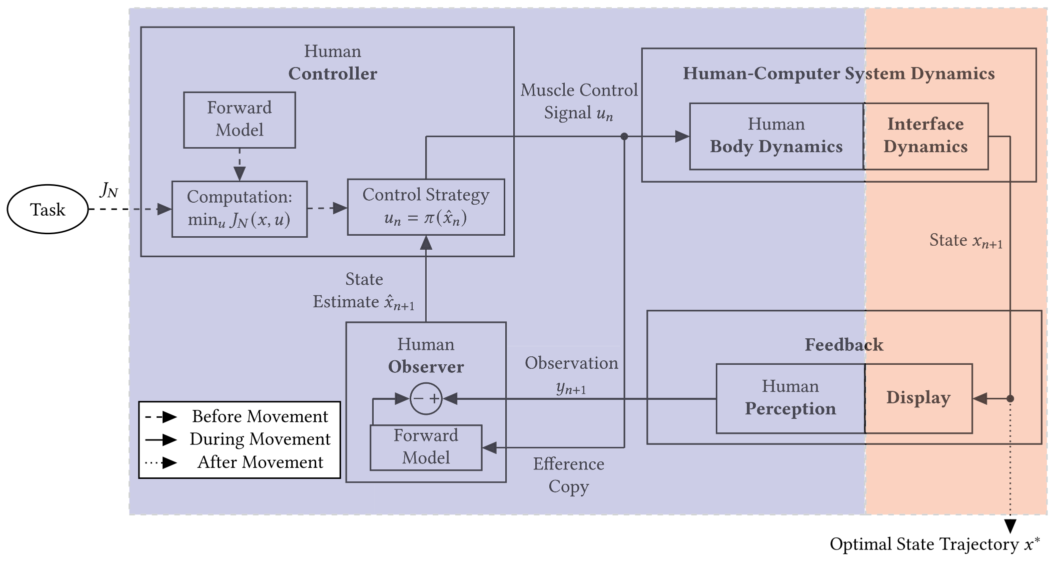
Figure 1 illustrates the relationship between the four components of the HCI loop, specifically distinguishing between *open-loop* and *closed-loop* optimal control models. In the following, we will give a brief description of the proposed framework, with focus on the differences between both variants. Subsequently, the four submodels are explained in detail in separate subsections, introducing a more technical and mathematically rigorous notation. Readers who are already familiar with the differences between open- and closed-loop models can proceed directly to Section 3.2.

3.1 Open-Loop vs. Closed-Loop Models

Open-loop models, as depicted in Figure 1(a), cannot infer any information from the *System Dynamics* after applying muscle control signals u. For this reason, the *Human Controller* block in Figure 1(a) does not depend on the output of the *System Dynamics* block, but depends only on an internal *Forward Model*. In particular, the *Feedback* and *Human Observer* blocks are not part of the generic open-loop framework at all. For this reason, open-loop models allow for a strict separation between planning and execution phase of a movement, i.e., trajectories x^* that are optimal with respect to the objective function J_N can be obtained in a two-step process. First, an (open-loop) OCP is solved (*Computation* block), i.e., a sequence of controls $u^* = (u_n^*)_{n \in \{0,\dots,N-1\}}$ is found such that $J_N(x, u)$ becomes minimal among all permissible control sequences u, given an initial state (see Section 3.3 for more details). Second, the resulting optimal control sequence is applied in a


Human
Computer
Human Controller
Forward Model
Computation: $\min_u J_N(x,u)$
Muscle Control Signals u^*
Human-Computer System Dynamics
Human Body Dynamics
Interface Dynamics
J_N
Task
Optimal State Trajectory x^*

(a) Open-Loop Optimal Control



(b) Closed-Loop Optimal Control (Discrete-Time Formulation)

Fig. 1. In our **optimal control framework for HCI**, the user is assumed to *control* the state x of the interactive system, which incorporates both the body state (e.g., arm and finger position) and the interface state (e.g., mouse pointer position and velocity), and which evolves according to the Body and Interface Dynamics. We assume that the user computes the controls u through *optimization*, i.e., by minimizing a cost function J_N (e.g., incorporating time or effort costs) that depends on the task. (a) In an *open-loop model*, this calculation is only based on an internal Forward Model of the Human-Computer System Dynamics. The optimal state trajectory x^* is obtained by applying the resulting muscle control signals u^* in one forward pass. The Forward Model does not have to coincide with the System Dynamics. (b) A *closed-loop model* takes into account effects that appear only after execution. The key difference in the Computation block is that, instead of optimal control signals u^*, it yields an optimal Control Strategy π that is computed before movement onset. At each time step n, this Control Strategy is used to map an arbitrary (estimated) state $\hat{x}_n$ to the corresponding optimal control u_n. Based on the resulting state x_{n+1}, an observation y_{n+1} is obtained via Feedback, which incorporates descriptions of both the Display and the Human Perception. The Human Observer then compares this observed state to an expected state it computes using an efference copy of the current control signal u_n and the Forward Model. Based on the resulting difference between expected and observed signals, an internal state estimate $\hat{x}_{n+1}$ is computed and used to select the next control u_{n+1}, and so on.

forward pass, i.e., the system dynamics are evaluated at each timestep $n \in \{0, \ldots, N-1\}$ to obtain the subsequent optimal state x^*_{n+1}.

Note that if the system dynamics are deterministic and the forward model internally used to compute u^* matches these dynamics, there is no advantage in computing the optimal controls *online*, that is, only at the time they are executed during the forward pass. However, there are few scenarios where the internal forward model used for optimization cannot fully predict the actual behavior of the human body and interface dynamics, i.e., the outcome of the Human-Computer System Dynamics block in Figure 1. This is particularly the case

— if the OCP is stochastic, e.g., in the case of motor noise,
— if there is a mismatch between internal model and actual system dynamics, or
— if unexpected disturbances occur that the internal model did not account for.

If one of these assumptions holds (which usually is the case in practice), the controller will benefit from any information it receives *during execution*, as this allows to condition the choice of an individual control u_n on the true state in a **closed-loop** manner, instead of using a prior state estimate. Note that the feedback loop immediately eliminates the strict separation between planning and execution phase, which is prevalent for open-loop models. Instead, the optimal control sequence u^* needs to be computed online in an iterative manner. In particular, all information available to the controller at a timestep $n \in \{0, \ldots, N-1\}$ is used to compute the muscle control signal u^*_n at this timestep, which is applied to the *Human Body Dynamics*. In combination with the *Interface Dynamics*, this results in a new system state x_{n+1}. The state (or partial information thereof, see Section 3.4) is then fed back into the controller, which again selects the next control u^*_{n+1} based on this information, i.e., a *sensorimotor* control loop between Human and Computer is established (see Figure 1(b)). In particular, the optimal state trajectory x^* does not only depend on the control sequence u, but also vice versa. Since in such models, feedback is given to the controller during execution, these models are often denoted as **OFC models** [28, 33].

It is important to note that with many OCP solution methods that take into account feedback during execution (including the ones presented in this article), the actual optimization can be performed offline, i.e., before the controls are applied to the actual system dynamics. Instead of a single optimal control sequence u^*, such methods usually yield an optimal *Control Strategy*, that is, a function $\pi : \mathcal{X} \to \mathcal{U}$ that maps an arbitrary state $x \in \mathcal{X}$ to a corresponding control $u \in \mathcal{U}$ that is optimal starting from this state.

In the later sections, we will present both an open-loop model (Section 6) as well as different variants of one of the most widely used OFC models (Sections 7–9).

3.2 Human–Computer System Dynamics

The Human–Computer System Dynamics form the basis of each optimal control model of HCI and consist of two parts: the Human Body Dynamics and the Interface Dynamics.

Human Body Dynamics. Given a vector of neural muscle control signals u, the Human Body Dynamics describe how these signals are transformed into joint torques and accelerations. The corresponding joint postures are obtained via integration. If required, kinematic models that map joint postures to world-centered positions of arbitrary body parts can be included as well, e.g., to get more realistic movement of the wrist or the index finger based on the computed joint angles.

Interface Dynamics. In interaction with computers, the forces and accelerations resulting from body dynamics are applied to the physical input device, e.g., the mouse device. Inertial properties of this *physical end-effector* determine the input device motion, which is sensed, filtered, and mapped to a motion of the corresponding *virtual end-effector*. For example, in the case of mouse pointing, the mouse pointer position might result from the application of a pointing transfer function to the mouse device velocity measured via optical sensors [22, 23]. The virtual end-effector is then used for interaction with graphical user interfaces. These dynamics of the computer part, including physical properties of the input device as well as visualizations and animations that appear from interaction with buttons, sliders, and so on, are summarized within the Interface Dynamics block in Figure 1.

Combined, the Human–Computer System Dynamics yield the new state vector x, which incorporates all relevant information about the current state of the human body, of the physical and/or the virtual end-effector, and of the applications. In the discrete-time formulation used in this article, both the state and control vectors x_n and u_n are given at timesteps $n \in \{0, \ldots, N\}$ up to a final step $N \in \mathbb{N}$, with each timestep corresponding to h seconds. The next state x_{n+1} depends on the current state x_n and control u_n, which in most cases can be formalized as

$$x_{n+1} = f(x_n, u_n), \tag{1}$$

where x_0 is a given initial state.[2]

This system dynamics function f can be either *deterministic* or *stochastic*. In the deterministic case, starting from the current state $x_n \in \mathbb{R}^k$ ($k \in \mathbb{N}$) and applying a control $u_n \in \mathbb{R}^m$ ($m \in \mathbb{N}$), the subsequent state $x_{n+1} \in \mathbb{R}^k$ is uniquely determined by the function $f : \mathbb{R}^k \times \mathbb{R}^m \rightarrow \mathbb{R}^k$. In the stochastic case, x_{n+1} randomly emerges from a set of possible successor states, according to a conditional probability distribution p, i.e., $x_{n+1} = f(x_n, u_n) \sim p(\cdot \mid x_n, u_n)$. Examples of stochastic systems include, e.g., *Body Dynamics* with internal motor noise or *Interface Dynamics* with noisy input signals.

It is important to note that the controller, which is described in Section 3.3, is agnostic to the partitioning of the system dynamics into effects attributed to the *Body Dynamics* and effects attributed to the *Interface Dynamics*. All the controller need to know is the *overall* system dynamics f (or an internal approximation of it), which maps an arbitrary state-control pair (x_n, u_n) to the subsequent state x_{n+1} reached after h seconds. Thus, an optimal control model of HCI can be instantiated in two ways. The first one is to include accurate submodels of arbitrary granularity (e.g., a separate model for each muscle activation, arm and hand dynamics, input device dynamics, and application dynamics), and combine them along the interaction loop into one aggregated system dynamics function f. However, the framework also allows to test whether some generic dynamics, such as a spring-mass-damper system or simplified muscle activation dynamics, are suitable to model the *overall Human–Computer System Dynamics* for a given task setting. The focus of this article will be on the latter approach, since we aim at starting with an easily understandable and well-established model from the field of Human Motor Control, and test whether these dynamics are applicable to the context of mouse pointing. We believe that this approach is well suited for introducing optimal control methods to HCI without going too much into (biomechanical) detail.

The system dynamics of all models considered in this article are linear in both the state and the control, i.e.,

$$x_{n+1} = f(x_n, u_n) = Ax_n + Bu_n. \tag{2}$$

Here, the matrix A describes how the dynamics evolve when no control is exerted. The matrix B describes how the control influences the system.

At first glance, this assumption seems to be very limiting, especially with regards to the complexity of the human neuro- and biomechanical system, as well as of most interaction methods and application GUIs, However, the tools and methods proposed in this article for the case of linear dynamics are also beneficial for more complex models of interaction, which, for example, include

[2]This is closely related to a continuous-time formulation based on differential equations, where the control is assumed to be piecewise constant (i.e., it only changes once every h seconds).

muscle-driven models of the human body or non-linear pointer acceleration functions. Using a reference trajectory, it is always possible to linearize a non-linear system around this particular trajectory in order to obtain a linear system of the form Equation (2), which locally approximates the non-linear one. While linearization-based extensions of the considered optimal control models to the non-linear case have been proposed [83, 132, 143], an application of these methods to typical HCI tasks would be an interesting area for future work.

The main advantage of using linear dynamics is that, when combined with quadratic costs and Gaussian noise, the resulting OCP (see Section 3.3) can be solved analytically and thus quickly and exactly. Finally, the linear case is easier to understand and formalize and thus well suited for the explanatory purposes of this article.

In the case of mouse pointing, which usually requires only small movements of the arm, the hand, and the input device, linearization around a single trajectory, i.e., using constant system matrices A and B as in (2), is a reasonable initial approach to model (moderate) mouse movements. Indeed, we will show that linear system dynamics can account for many phenomena that are characteristic in mouse pointing.

3.3 Human Controller

In general, various control sequences can produce the same movement trajectory. For example, the arm can rest on the table or stay in the air, as long as the mouse device is controlled appropriately. This is referred to as the *joint-redundancy problem* [18]. For a large number of degrees of freedom, e.g., motor signals that are applied to individual muscles, the same goal can be achieved with different controls, raising the question of which control is actually chosen by the **central nervous system** (**CNS**) and why.[3] This fundamental question, however, cannot be answered using movement dynamics alone. Instead, the optimal control framework has been proposed to address this question [44, 73, 142].

Optimal Control Problems (OCPs). Optimal control methods make use of a specific cost function, which is to be minimized. Previous approaches include minimization of either jerk [42, 61], peak acceleration [103], end-point variance [58], duration [3, 131], or torque-change [144], among others. Different objectives can be combined in one cost function to model tradeoffs, e.g., between accuracy and effort [83, 138], accuracy and stability [86], or jerk and movement time [60]. Recently, it has been argued that several abilities associated with intelligence such as knowledge, perception, or imitation naturally emerge from behaving optimally with respect to an ultimate goal [124]. For goals that can be expressed by an adequate cost function, this particularly suggests that the optimal control framework is able to explain intelligent behavior.

In general, the (finite-horizon) *discrete-time OCP* is given by

$$\begin{aligned} &\textit{Minimize} \quad J_N(x,u) = g_N(x_N) + \sum_{n=0}^{N-1} g(x_n,u_n), \\ &\textit{with respect to } u = (u_n)_{n\in\{0,\dots,N-1\}} \subset \mathcal{U} \subset \mathbb{R}^m, \end{aligned} \tag{3a}$$

where $x = (x_n)_{n\in\{0,\dots,N\}} \subset \mathcal{X} \subset \mathbb{R}^k$ satisfies

$$\begin{aligned} x_{n+1} &= f_{\text{int}}(x_n,u_n), \quad n \in \{0,\dots,N-1\}, \\ x_0 &= \bar{x}_0 \end{aligned} \tag{3b}$$

for some given initial state $\bar{x}_0 \in \mathcal{X} \subset \mathbb{R}^k$.

[3]Moreover, in the case of muscle-driven simulations, the set of feasible controls is relatively small compared to the total decision space, which makes it even less clear how appropriate controls are internally found [145].

Here, f_{int} denotes the dynamics of the internal model used for optimization. In Figure 1, this corresponds to the *Forward Model* block within the Human Controller. In most optimal control models, including those considered in this work, the internal model is assumed to be exact, i.e., it matches the actual system dynamics (corresponding to the *Human–Computer System Dynamics* block in Figure 1), which are analogously described by some function f. The *objective function* $J_N(x, u)$ that we want to minimize consists of some *terminal cost* function $g_N : \mathbb{R}^k \to [0, \infty[$ and the sum of *running costs* $g : \mathbb{R}^k \times \mathbb{R}^m \to [0, \infty[$ accumulated over N timesteps. These might be chosen dependent on the task under consideration. For example, in a tracking task, the distance between end-effector and target could be penalized in each step, whereas in a steering task, large costs might be applied whenever one of the bounds is reached. The sets $\mathcal{X}$ and $\mathcal{U}$ can be used to restrict the states and controls that are permissible at each timestep. In the following, however, we will set $\mathcal{X} = \mathbb{R}^k$ and $\mathcal{U} = \mathbb{R}^m$, i.e., we do not impose any restrictions. The initial state $x_0 = \bar{x}_0$ is assumed to be given, and the (unique) optimal solution (x, u) to an OCP (assuming that it exists) is denoted by (x^*, u^*) in the following.

For *deterministic OCPs*, both the internal model and the actual system dynamics are given by deterministic functions $f_{\text{int}} : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}^k$ and $f : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}^k$, respectively. For *stochastic OCPs*, these dynamics are replaced by conditional probability distributions (see Section 3.2). It is important to note that albeit in stochastic OCPs, the concrete successor state x_{n+1} resulting from the application of a hypothetical control u_n in the state x_n is not available to the controller during optimization, the underlying transition probability density function $p(\cdot \mid x_n, u_n)$ usually is.[4] Stochastic OCPs are thus capable of modeling the between-trial variability that typically occurs in human movement.

The methods to find the optimal solution of an OCP depend on its problem structure, i.e., the properties of the cost function (e.g., differentiability, convexity), the system dynamics (e.g., linearity, stochasticity), and the permissible state space $\mathcal{X}$ and control space $\mathcal{U}$. Often, the optimal solution can only be determined approximately, using numerical methods such as Multiple Shooting [15], Direct Collocation [11], or RL [130]. However, in some cases an explicit solution formula exists that yields the exact (and unique) optimal control sequence u^*. In this article, we will focus on the most widely known class of OCPs that allows for such an analytical solution scheme: those with linear system dynamics and convex, quadratic costs.

3.4 Feedback & Human Observer

In the closed-loop case of the optimal control framework for HCI (Figure 1(b)), the *Feedback* block accounts for the fact that usually not all information included in the state x are (immediately) available to the user. First, the visual output, which is shown on the *Display*, is created based on the respective state components. This information is then sensed and processed by the *Human Perception*, which describes how visual, proprioceptive, and/or auditive signals are perceived and integrated into the stream of observations $y = (y_n)_{n \in \{0,\dots,N-1\}}$ the controller can condition on. The same holds for information on the own body state, which is directly obtained from the *Human Body Dynamics*, e.g., via proprioceptive input signals.

In general, these observations might be delayed, noisy, or incomplete. To decide for an appropriate control u_{n+1} at timestep $n + 1$, thus an estimate $\hat{x}_{n+1}$ of the true current state x_{n+1} is required. This estimate is computed by the *Human Observer*, which compares the observed state y_{n+1} to an expected state it computes using an efference copy of the most recent control signal u_n and the

[4]In the case of unknown transition dynamics p (or f, in the deterministic case), the controller would need to rely on sampling transitions from the environment in order to be able to estimate the expected future costs of different controls. This problem is addressed in *Model-Free RL*.

Forward Model.[5] Based on the resulting difference between expected and observed signals, an internal state estimate $\hat{x}_{n+1}$ is computed, which is then used by the Human Controller to select the next muscle control signal u_{n+1}, resulting in the above discussed closed interaction loop.

3.5 Applications of the Proposed Framework

In this article, we analyze and compare several optimal control models of interaction. It is important to note that not all of the considered models exploit the complexity of the generic models depicted in Figure 1. For example, the closed-loop LQR model includes a trivial perceptual model in that it assumes that the controller has complete access to the true system state x immediately. Another example is the open-loop MinJerk model, which does not include task-specific system dynamics.

Each optimal control model, however, yields a continuous representation of all relevant quantities of the interactive system. In contrast to summary statistics such as Fitts' Law, this allows to simulate and predict complete movement trajectories on both the kinematic and the dynamic level. It also allows to analyze the effects of the control u and of different cost terms incorporated in the objective function J_N, on the human body and the interface (e.g., user representations, buttons, or sliders).

Most importantly, the modularity of the proposed framework enables high flexibility and generalizability. For example, it is possible to analyze the effects of different input devices and/or GUIs on movement trajectories and control sequences, using the same description of the human biomechanical and perceptual system. Additionally, a given interface can be evaluated for different tasks such as pointing, dragging, steering, and so on, by modulating the internal objective function accordingly. The resulting continuous representations can then be evaluated with respect to different metrics, e.g., remaining distance to target [33], effort [121], fatigue [25], movement time [131], and so on.

Finally, the framework can be used to reverse-engineer the internal objective function (*inverse optimal control*) as well as properties of the human biomechanical system (*system identification*), such that the resulting trajectories best fit some experimentally observed user trajectories. Before we explain how to identify such model-specific parameters using a data-driven parameter fitting procedure (see Section 4.2), we give a brief overview of the experimental data we use to evaluate the presented models in this article.

4 USER TRAJECTORIES AND PARAMETER FITTING

4.1 The Pointing Dynamics Dataset

For the evaluations in this article, we use the *Pointing Dynamics Dataset*. Task, apparatus, and experiment are described in detail in [98]. The dataset contains the mouse trajectories for a reciprocal mouse pointing task in 1D for ID 2, 4, 6, and 8 (12 participants, 8 task conditions, and 7732 trajectories in total). We use the raw, unfiltered position data in our parameter fitting process to avoid artifacts from the filtering process. In this section, we explain how we pre-process this data for the purposes of this article.

Pointing experiments both in the reciprocal and discrete Fitts' paradigm introduce reaction times as experimental artifacts. In real mouse usage, users first decide on a pointing target themselves, and then start moving the mouse. In this sense, the pointing process can be considered initiated as soon as the pointer begins to move. In contrast, in the experimental paradigm used in [98], the next trial started as soon as the user clicked the mouse in the previous trial. The target

[5]In this article, we assume perfect system knowledge, i.e., the forward model consists of the same system dynamics and perception functions as used in the actual interaction loop.

given to the user appeared at that instant. This introduces a potential confound in the starting time of each trial. The beginning of each trial can be partly attributed to belong to the end of the previous trial, and partly to a reaction time adjusting to the new target. During this time, velocity and acceleration of the pointer are close to 0. This reaction time shows considerable variation both within and between participants.

Because the focus of this article is not on modeling reaction times, we remove them from each mouse movement. To this end, we drop all frames before the velocity reaches 1% of its maximum/minimum value (depending on the movement direction) for the first time in each trial.[6]

Since the deterministic optimal control models considered in this article can only predict average user behavior, we compute *mean trajectories* for each user, task condition, and direction from the raw data, resulting in 192 mean trajectories. This is done as follows.

First, we remove outlier trials, where at any timestep the position was more than three standard deviations from the respective mean. This was the case for 397 trajectories in total, i.e., 5.1% of all trials. We found this to be necessary as the averaging process is highly sensitive to outliers. In particular, delayed movement onsets, which, e.g., might have occurred due to a lack of attention of the participant, would inject a high bias into the computed statistics.

Second, in order to make trials of different length comparable, we assume that the pointer would not move after the mouse click. Given a set of trials to be averaged (with reaction times removed as described above), movements shorter than the longest one are extended by their last position, zero velocity, and zero acceleration to achieve the same length. To avoid unnecessarily long trajectories for conditions where a few of the recorded trials were of exceptional length, we additionally remove trials with duration longer than three standard deviations from the mean duration of the respective condition, before extending the remaining trajectories to maximum length. This was the case for 87 trajectories in total (i.e., 1.1% of all trials), with a maximum of two trials removed per user, task condition, and direction. Finally, we average the resulting trajectories on a frame by frame basis. We also compute the respective sample covariance matrices at each timestep to capture the between-trial-variability observed from user data.

4.2 Parameter Identification

In this section, we present a method to identify the parameters of a given instantiation of the optimal control framework introduced in Section 3 that best explain experimentally observed user behavior. More specifically, for a given interaction model and a given dataset of user trajectories, we aim at finding the model-specific parameter values such that the resulting trajectories approximate a subset of user trajectories (e.g., all trajectories of a specific user for some task condition) as closely as possible. In the following, we will apply this procedure to each of the presented models, using the Pointing Dynamics Dataset as reference data. Since only stochastic models can account for the between-trial variability typically observed in human movements, we need to distinguish between the deterministic and the stochastic case in the following.

In the deterministic case, we use the squared error between predicted and observed mouse pointer position, summed over time (***sum squared error (SSE)***) as a measure of distance between simulation and mean user trajectories. Given a model with parameter vector Λ, where p_n^{Λ} denotes the position time series of the resulting simulation trajectory, and given a mean user position time series p_n^{USER}, the goal is to find the parameter vector Λ^* such that the loss function of the parameter

[6]For improved temporal alignment of the individual trajectories and to remove outliers that sometimes occur at the beginning of a movement, we additionally assume that the acceleration remains positive/negative (depending on the movement direction) for at least 40 ms after the initial time. If no timestep is found that satisfies both criteria, we discard the entire movement. This was the case only for a single movement (ID 2, participant 5).

fitting process

$$\mathcal{L}(\Lambda) = \mathrm{SSE}(\Lambda) = \sum_{n=0}^{N} \left(p_n^{\Lambda} - p_n^{\mathrm{USER}}\right)^2, \tag{4}$$

takes its minimum in Λ^*. This is done for each mean trajectory, resulting in 192 optimal parameter vectors.

Minimizing Equation (4) with respect to Λ can be considered a *least-squares problem* [14], where each function evaluation of $\mathcal{L}(\Lambda)$ requires computation of the respective model simulation trajectory to obtain p_n^{Λ}. In the case of optimal control models, this particularly implies that an OCP must be solved to obtain p_n^{Λ} for a given Λ, that is, the complete parameter fitting process consists of two nested optimizations.

Stochastic models, in contrast, yield a sequence of distributions of the state, such as the distributions of pointer position and velocity, over multiple trials. These distributions can be used to sample individual trajectories. We measure the "similarity" between two distributions using the 2−*Wasserstein distance* (often denoted as *Earth mover's distance*) [104]. Given two normal distributions ρ_1 and ρ_2 with means μ_1 and μ_2 and covariance matrices Σ_1 and Σ_2, respectively, the 2−Wasserstein distance can be written as

$$W_2\left(\rho_1, \rho_2\right) = \left(\|\mu_1 - \mu_2\|_2^2 + \mathrm{tr}(\Sigma_1) + \mathrm{tr}(\Sigma_2) - 2\mathrm{tr}\left((\Sigma_1\Sigma_2)^{\frac{1}{2}}\right)\right)^{\frac{1}{2}}, \tag{5}$$

and can be interpreted as the amount of work required to transform the probability distribution ρ_1 into the probability distribution ρ_2 (and vice versa). In the following, we will use this formula to measure the distance between the simulation state distribution of a model with parameter vector Λ, ρ^{Λ}, and the empirically observed state distribution, ρ^{USER}, at some timestep $n \in \{0, \ldots, N\}$, i.e., $\rho_1 = \rho_n^{\Lambda}$ and $\rho_2 = \rho_n^{\mathrm{USER}}$.

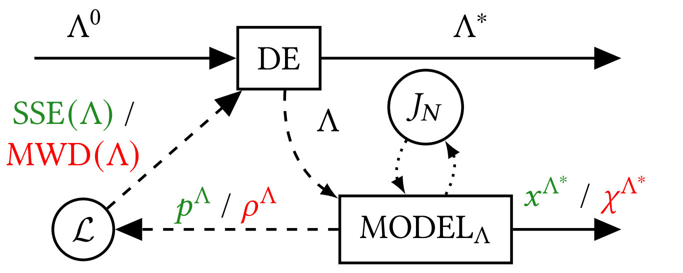
One advantage of this measure is that it is only based on the relative distance between the two means and covariance matrices of the two distributions, independent of the magnitude of these quantities. This is in contrast to the KL divergence [76], which increases as the variance of the reference distribution decreases. In the special case where both distributions have the same, diagonal covariance matrix, the 2−Wasserstein distance corresponds to the Euclidean distance between the means of both distributions.

As a measure for the similarity between complete sequences of distributions (e.g., of mouse pointer positions and velocities), we use the ***mean Wasserstein distance (MWD)*** over time:

$$\mathcal{L}(\Lambda) = \mathrm{MWD}(\Lambda) = \frac{1}{N+1}\sum_{n=0}^{N} W_2\left(\rho_n^{\Lambda}, \rho_n^{\mathrm{USER}}\right). \tag{6}$$

In both the deterministic and the stochastic case, we solve the (outer) optimization problem of minimizing $\mathcal{L}(\Lambda)$ with respect to Λ using *differential evolution* [127], which is a simple, gradient-free global optimization algorithm suitable for continuous parameter spaces. This algorithm has proven to yield robust and reliable results even for ill-conditioned problems [5]. Of course, more efficient optimization methods, e.g., gradient-based ones, are always desirable, and algorithmic differentiation is a promising step forward in that regard. The main question here is the applicability of algorithmic differentiation in the case of iteratively alternating between control and estimation problems, as is required for the considered case of LQG with signal-dependent noise (see Section 8.1). Pursuing this endeavor, however, might very well enable real-time predictions of parameter effects on the entire interaction loop.

Figure 2 gives an overview of our parameter identification process for both the deterministic and the stochastic case. The parameter boundaries for all models introduced below are given in



Deterministic Parameter Fitting | Stochastic Parameter Fitting

Fig. 2. Starting with an initial parameter vector $\Lambda = \Lambda^0$, the differential evolution (DE) algorithm obtains the loss $\mathcal{L}(\Lambda)$ (SSE value in the deterministic and MWD in the stochastic case) for the currently considered parameter vector Λ. To do this, it calls MODEL$_\Lambda$, which computes the resulting model trajectory sequence x^Λ (or a sequence of state distributions χ^Λ). In case of optimal control models, this requires an inner optimization with respect to a model-specific objective function J_N. The resulting position time series p^Λ included in x^Λ (or the sequence of position-velocity distributions ρ^Λ included in χ^Λ) is used to compute the loss $\mathcal{L}(\Lambda)$ of the current parameter vector Λ, which, in turn, is used by the DE algorithm. After obtaining the loss for a few requested parameter vectors Λ, the DE algorithm chooses the next parameter vectors Λ until Λ^* with minimum loss is found. Finally, Λ^* is returned along with the respective optimal trajectory x^{Λ^*} (or the optimal sequence of state distributions χ^{Λ^*}).

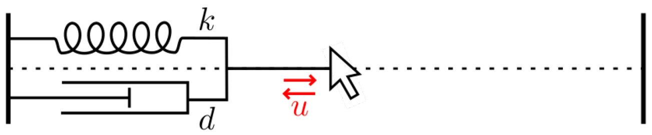


Fig. 3. Mouse pointer model with spring and damper.

Table B.1 in the Appendix. Descriptions of how discrete parameters are relaxed in order to optimize them using standard continuous optimization methods are given in the respective model sections.

5 POINTING AS A DYNAMICAL SYSTEM: THE SECOND-ORDER LAG

One of the basic models of mouse pointer dynamics is the *second-order lag*, which has been used as a baseline in several papers, including [98, 160]. We therefore also include it as a baseline. The parameters Λ of the model are the stiffness of the spring $k > 0$ and the damping factor $d > 0$. In the setting described below, the mass is a redundant parameter, and we thus set it to 1. In continuous time, we denote the position of the mouse pointer as $y(t)$, and its first and second derivatives with respect to time (i.e., velocity and acceleration) as $\dot{y}(t)$ and $\ddot{y}(t)$, respectively. The behavior is then described by the second-order lag equation:

$$\ddot{y}(t) = u(t) - ky(t) - d\dot{y}(t). \tag{2OL}$$

An intuitive illustration of these dynamics is given in Figure 3: Assuming that the mouse pointer is fixed at one edge of the screen via a spring, the parameters k and d correspond to the stiffness and the damping of this spring, respectively, and the control value u can be interpreted as the force acting on the mouse pointer. In particular, the pointer acceleration $\ddot{y}$ is assumed to be directly proportional to the control u (apart from the damping and stiffness terms), i.e., (2OL) defines a (linear) dynamical system of second order. A control flow diagram of the model is shown in Figure B.1 in the Appendix.

Given a target position $T \in \mathbb{R}$, it can be shown that for the particular choice $u \equiv kT$, with $k, d > 0$, the position $y(t)$ approaches T for large enough t, independent of the initial position $y(0) = y_0$ and initial velocity $\dot{y}(0) = \dot{y}_0$ [57]. More precisely, the state $(y, \dot{y}) = (T, 0)$, i.e., the desired target T is reached and the velocity is 0, is an equilibrium, meaning that once reached, that state will forever be maintained. The resulting trajectory, which is uniquely determined given y_0, k, d, and T, is often referred to as *second-order lag trajectory*, and can be used to model mouse pointing movements toward a given target T. Since the control $u(t)$ is constant in time and converges toward the equilibrium state, we denote this variant of (2OL) as *2OL-Eq* in the following.

Following the general notation of (2), we derive a discrete-time version of (2OL), with a step size of two milliseconds, i.e., $h = 0.002$, which corresponds to the mouse sensor sampling rate. Considering our example case of 1D pointing tasks, in which the mouse can only be moved horizontally, the state x_n encodes the horizontal position and velocity of the pointer, denoted by $p_n \in \mathbb{R}$ and $v_n \in \mathbb{R}$, respectively, i.e.,

$$x_n = (p_n, v_n)^\top \in \mathbb{R}^2. \tag{7}$$

Using the forward Euler method,[7] we obtain the matrices A and B for (2) as

$$A = \begin{pmatrix} 1 & h \\ -hk & 1 - hd \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ h \end{pmatrix}. \tag{8}$$

This model can easily be extended to 2D or 3D pointing tasks by augmenting x_n and u_n with the respective components for the additional dimensions.

5.1 Analysis of Parameters

Since the 2OL model is an important baseline for mouse pointing dynamics, in this section, we provide an analysis and intuitive understanding of influence of the model parameters on model behavior.

While the convergence of 2OL-Eq toward a target T of fixed width $W > 0$ can be easily shown under the assumptions described above, both the time until this target is reached first (i.e., the time at which the remaining distance to target is smaller than W) and the *transient behavior* (i.e., specific characteristics of the trajectory until this time) largely depend on the parameters k and d. In the following, we analyze the effects of the stiffness k and the *damping ratio* ζ, which is defined as

$$\zeta = \frac{d}{2\sqrt{k}}, \tag{9}$$

as this is easier to interpret than the actual damping parameter d. Note that given two of the three parameters k, d, and ζ, the remaining one (in this case d) is uniquely determined by the others and can be easily computed.

The position, velocity, and acceleration time series of typical 2OL-Eq trajectories are shown in Figure 4. The initial and target values stem from an ID 4 task condition from the Pointing Dynamics Dataset. The most characteristic feature of 2OL-Eq trajectories is the large positive acceleration at the beginning of the movement. This is due to the model being second-order, i.e., the control u is proportional to the acceleration $\ddot{y}$ (apart from the damping and stiffness terms). The velocity profile is typically left-skewed, since the deceleration phase is considerably longer than the acceleration phase. As a consequence, the peak velocity is reached relatively early during the movement.

[7]While we could also use the exact solution here, the (fairly good) approximation via forward Euler yields matrices that are more suitable for our explanatory purposes.

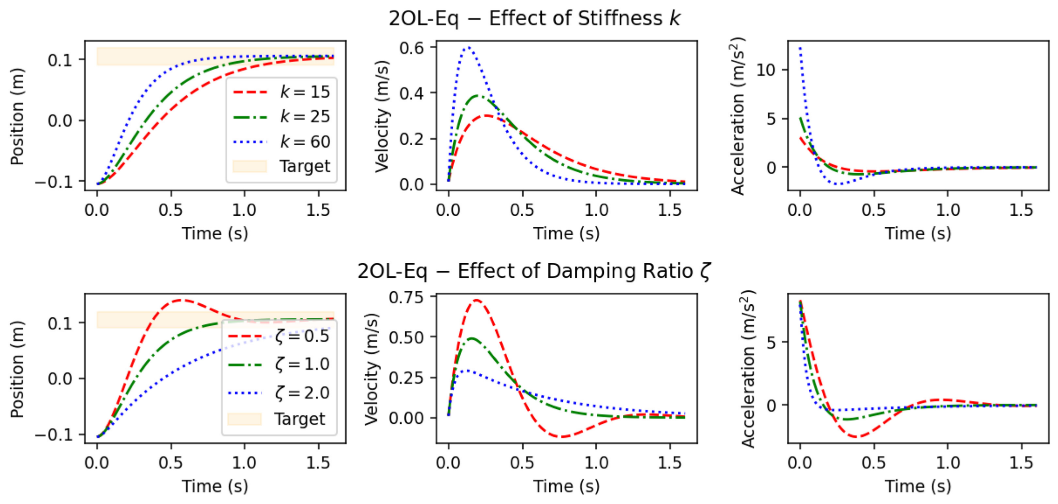


Fig. 4. Position, velocity, and acceleration time series of typical 2OL-Eq trajectories with target shown as the orange box. **Top:** Effect of stiffness parameter k with fixed damping ratio $\zeta = 1$ (red dashed: $k = 15$, green dash-dotted: $k = 25$, and blue dotted: $k = 60$). **Bottom:** Effect of damping ratio ζ with fixed stiffness parameter $k = 40$ (red dashed: $\zeta = 0.5$, green dash-dotted: $\zeta = 1$, and blue dotted: $\zeta = 2$). The stiffness parameter k mainly affects the instantaneous initial acceleration and thus the speed at which the target is approached. The parameter ζ determines the relative amount of damping, with $\zeta < 1$ (underdamped) resulting in oscillations around the target and $\zeta > 1$ (overdamped) leading to trajectories that reach the target later.

Given a constant damping ratio ζ, the stiffness parameter k mainly affects the initial acceleration, and consequently the peak velocity and the time at which the target is reached first. As can be seen in the top row of Figure 4, a large stiffness (blue dotted line) leads to a high initial acceleration and peak velocity, and thus the target is reached earlier than with lower stiffness values (red dashed line). For damping ratios $\zeta < 1$ (red dashed line in the bottom row of Figure 4), i.e., the damping d is small compared to the stiffness k, oscillations occur in the trajectories, leading to multiple peaks in velocity and acceleration time series and to *overshooting* in the position time series (the so-called *underdamped* case). For $\zeta = 0$, the pointer does not even converge toward the target, but oscillates indefinitely (not shown). If $\zeta > 1$ (the so-called *overdamped* case, blue dotted line in the bottom row of Figure 4), the pointer converges toward the target slowly, without oscillations. If $\zeta = 1$, the trajectory is *critically damped*, which means that the pointer reaches (and stays at) the equilibrium (i.e., the target T) in minimum time.

5.2 Results of Parameter Fitting

For each combination of participant, task, and direction, we identify the parameters $\Lambda = (k, \zeta)$ that best explain the corresponding mean trajectory from the Pointing Dynamics Dataset, using the deterministic parameter fitting process described in Section 3. The loss function is the SSE on position, see Equation (4). Note that the results obtained from our parameter fitting do not exactly match those presented in [98], since we apply a different pre-processing to the experimental user trajectories and optimize the parameters with respect to the positional error only.

The fitted trajectory for a representative ID 4 task condition is shown in Figure 5. As discussed in [98], the main differences between model and human behavior are the less symmetric velocity profile and the large initial accelerations produced by the 2OL-Eq model. In particular, the user

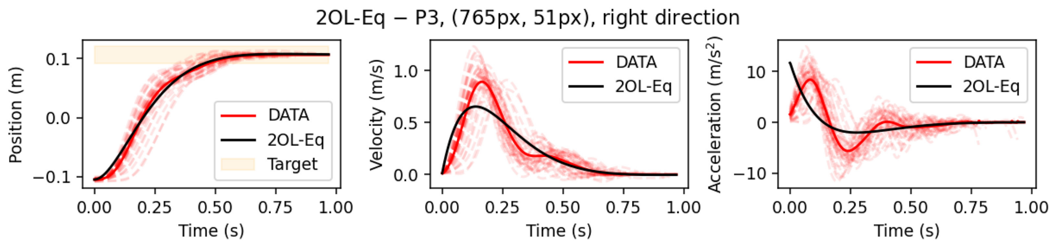


Fig. 5. While the position time series visually matches the observed user data quite well, 2OL-Eq yields a much less symmetric velocity and acceleration profile during the surge (here: up to 0.36 s). This is due to the assumption of constant equilibrium control, which results in a physically implausible instantaneous peak acceleration.

trajectories exhibit velocity profiles that are close-to-symmetric and bell-shaped, at least for the initial ballistic movement toward the target (the "surge" [98]), which is consistent with previous findings [94]. The differences can be explained with the physical interpretation of the 2OL-Eq as a spring-mass-damper system: Since u is constant in this model, as the system is released, the spring instantaneously accelerates the system with a force that is proportional to the extension of the spring. Because human muscles cannot build up force instantaneously [117], this behavior is not physically plausible.

In Figure 6, the optimal values of k and ζ are given for all participants, tasks, and movement directions, both grouped by participants (left) and by ID (right). Interestingly, different behavior between individual users is mainly captured by different stiffness values k. Participant 2, for instance, seems to be characterized best by a large stiffness (between 46.8 and 107.6, with mean 78.3), while the trajectories of participant 9 are best explained by a considerably lower stiffness (between 10.3 and 18.3, with mean 14.4) in the 2OL-Eq model. In contrast, the damping ratio ζ seems to center around 0.7–0.76, independent of the participant. Instead, it is mainly influenced by the ID of the task. Lower IDs tend to result in a lower damping ratio, with trajectories of ID 2 tasks being considerably more underdamped than others. This is consistent with previous findings [17, 53, 98] and might be explained by the reciprocal nature of the considered pointing task, where participants alternately moved between two given targets (which we denote as initial and target position for a given movement direction) without dwell time.

In summary, the stiffness k mostly seems to account for movement strategies that are characteristic of specific users, whereas the damping ratio ζ mainly differs between indices of task difficulty.

5.3 Discussion

The main shortcomings of the 2OL-Eq as a model of mouse pointer movements are the unrealistically high initial acceleration and the resulting skewed velocity profile. This is mainly due to the assumption of equilibrium control, while the literature suggests that the motor control signals are actively changed during the movement [13, 47, 138]. From a conceptual standpoint, the 2OL-Eq only describes the passive dynamics of the mouse pointer as a differential equation. It does not separately model the user's "brain" or intention in controlling these dynamics. In particular, it does not describe what the user is trying to achieve. This can be explained by optimal control models.

6 POINTING AS OPTIMAL OPEN-LOOP CONTROL: THE MINIMUM JERK MODEL

An elementary model of aimed movements that assumes that users behave optimally according to an internal cost function is the *minimum jerk model* by Flash and Hogan [42]. This model, which

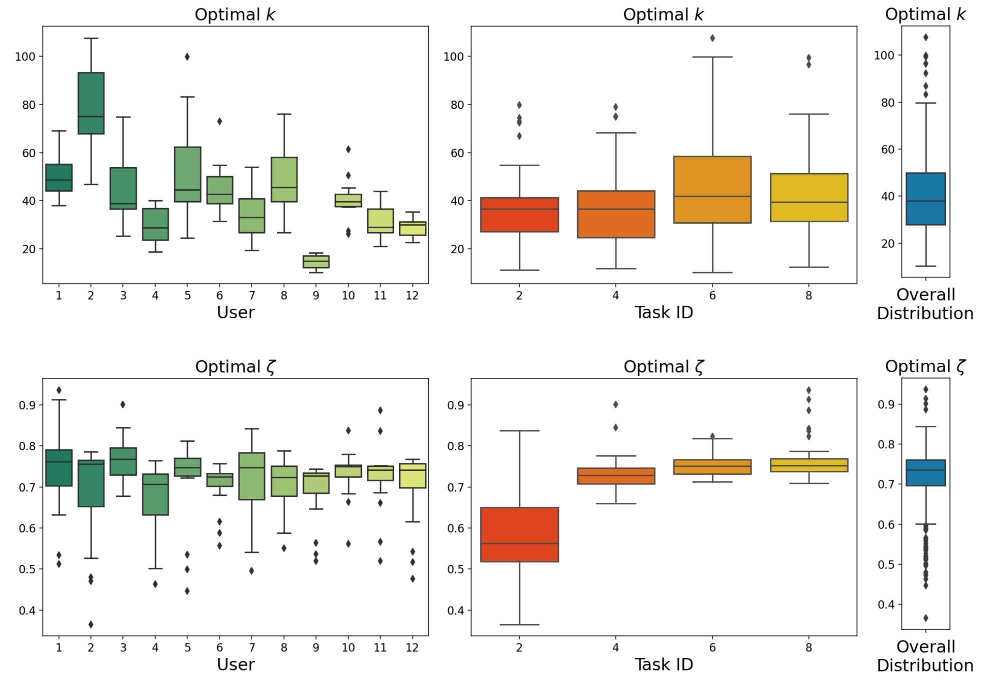


Fig. 6. Parameters of 2OL-Eq, optimized for the mean trajectories of all participants, tasks, and directions, grouped by participants (left) and by ID (middle), and as overall distribution (right). While the optimal stiffness parameter k considerably differs between participants, the damping ratio ζ is mainly affected by the task ID. Interestingly, all resulting simulation trajectories are underdamped (mean $\zeta = 0.71$), with ζ increasing as ID increases.

we will refer to as *MinJerk* in the following, assumes that the objective of users is to generate smooth movements by minimizing the jerk of the end-effector, i.e., the time derivative of the end-effector's acceleration, while reaching the target exactly at a prescribed movement time with zero final velocity and acceleration. Within HCI, this model has been successfully used by Quinn and Zhai [112] to model the shape of gestures on a word-gesture keyboard.

The model assumes that the movement is controlled open-loop, and thus cannot explain how users would correct for disturbances or inaccurate execution. Similar to 2OL-Eq, the choice of parameters and boundary values already determines the complete trajectory. However, there are a few important differences to 2OL-Eq. First, MinJerk does not only require information about the initial, but also about the final state (as a third-order model, initial and final position, velocity, and acceleration need to be specified). Second, the overall movement time, which is denoted by N_{MJ} in the following, needs to be known in advance. This is in contrast to the discrete-time formulation of 2OL-Eq.

In discrete time, minimizing jerk corresponds to minimizing the differences between subsequent accelerations. While in principle, the MinJerk OCP could be transformed into a closed-loop discrete-time system similar to Equation (2) (but with time-dependent matrices A_n and B_n, see [61]); here, we make use of the analytical solution of the original continuous-time problem, and afterwards discretize the resulting 5th-degree polynomial with respect to time. For arbitrary initial state $x_0 = (\bar{p}_0, \bar{v}_0, \bar{a}_0)$ and final state $x_{N_{MJ}} = (\bar{p}_{N_{MJ}}, \bar{v}_{N_{MJ}}, \bar{a}_{N_{MJ}})$, where the third component is the respective end-effector acceleration, the (discrete-time) MinJerk trajectory is given by

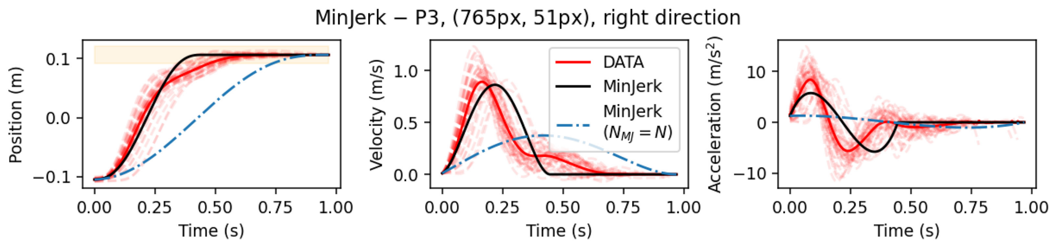


Fig. 7. Visually, MinJerk trajectories show a relatively good fit to observed user data, as long as the duration parameter N_{MJ} is chosen to cover only the surge (black solid line; here, $N_{MJ} = 223$ corresponds to 0.446s). However, the MinJerk model predicts zero velocity at the end of the surge, which results in a considerably worse overall fit of user trajectories that exhibit clear submovements. If N_{MJ} is set as total movement duration N, the resulting simulation trajectory (dash-dotted blue line) does not fit the data at all.

$$x_n = \sum_{i=0}^{5} c_i \left(\frac{n}{N_{MJ}}\right)^i, \tag{10a}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_f & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5t_f^2 & 0 & 0 & 0 \\ -10 & -6t_f & -1.5t_f^2 & 10 & -4t_f & 0.5t_f^2 \\ 15 & 8t_f & 1.5t_f^2 & -15 & 7t_f & -t_f^2 \\ -6 & -3t_f & -0.5t_f^2 & 6 & -3t_f & 0.5t_f^2 \end{pmatrix} \begin{pmatrix} \bar{p}_0 \\ \bar{v}_0 \\ \bar{a}_0 \\ \bar{p}_{N_{MJ}} \\ \bar{v}_{N_{MJ}} \\ \bar{a}_{N_{MJ}} \end{pmatrix}, \tag{10b}$$

where $t_f = N_{MJ}h$ denotes the final time, and h is the fixed time interval between two consecutive timesteps.

6.1 Extension to Complete Trajectories

The MinJerk model has been derived from data of an experiment that did not involve any corrective submovements [42]. However, in mouse pointing tasks with large ID, submovements occur regularly. If MinJerk is used for modeling of the entire movement, i.e., until the final timestep N, the fit is thus very poor (see blue dash-dotted lines in Figure 7). Instead of a quick movement toward the target with corrective submovements, as in our data, the model predicts a slow, smooth movement, and reaching the target only at the final time.

A much better fit is obtained by using MinJerk only for the first, rapid movement toward the target (the "surge"), and assuming that the pointer does not move afterwards. To this end, we define the (extended) MinJerk trajectory as follows. For $n \leq N_{MJ}$, x_n corresponds to the minimum jerk polynomial Equation (10) with $\bar{p}_0$, $\bar{v}_0$, and $\bar{a}_0$ taken from user data, $\bar{p}_{N_{MJ}} = T$, and $\bar{v}_{N_{MJ}} = \bar{a}_{N_{MJ}} = 0$. For $n > N_{MJ}$, the trajectory is constantly extended by the final state of the MinJerk polynomial, i.e., $x_n = x_{N_{MJ}} = (T, 0, 0)^\top$.

The effect of the parameter N_{MJ} in our MinJerk model is shown in Figure 8. Varying this parameter allows to model variable peak velocities and accelerations. However, with N_{MJ} being a pure scaling parameter of the trajectory, the velocity profile is always bell-shaped and the acceleration profile N-shaped. Moreover, the target center is reached at timestep $n = N_{MJ}$ by definition, i.e., the model naturally cannot account for corrections that typically occur after the surge. As illustrated in Figure 7 for the same ID 4 task as above (black solid lines), this can result in a considerably worse overall fit of the trajectory, at least for movements that consist of several submovements.

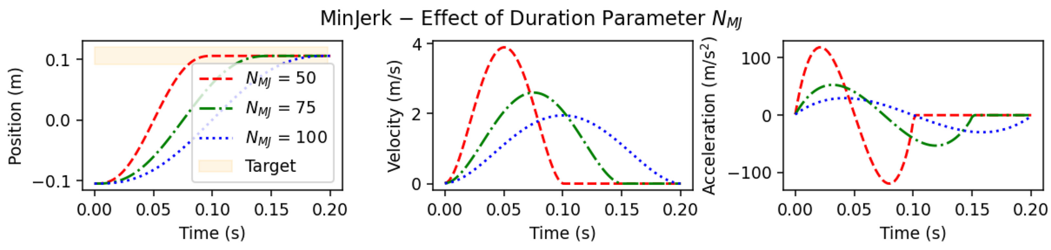


Fig. 8. Position, velocity, and acceleration time series of different MinJerk trajectories with target shown as the orange box. The parameter N_{MJ} determines the end time of the symmetric and smooth jerk-minimizing movement, after which the trajectory is constantly extended by its final position value and zero velocity and acceleration (red dashed: $N_{MJ} = 50$, green dash-dotted: $N_{MJ} = 75$, blue dotted: $N_{MJ} = N = 100$).

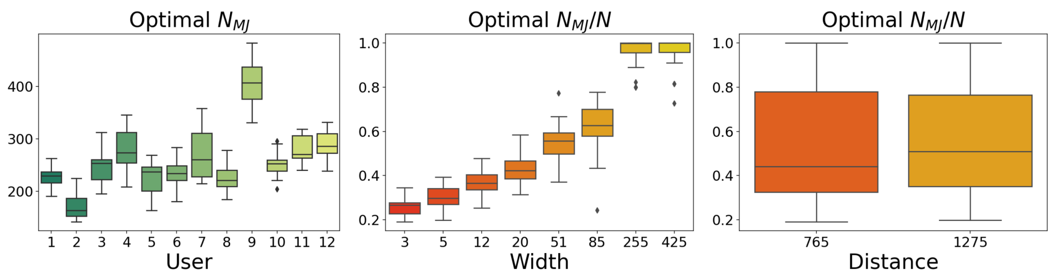


Fig. 9. Duration parameter N_{MJ} of MinJerk, optimized for the mean trajectories of all participants, tasks, and directions. **Left:** Absolute parameter values, grouped by participants. **Middle:** Parameter values relative to the total movement duration N, grouped by target width. **Right:** Parameter values relative to the total movement duration N, grouped by distance to target.

6.2 Results of Parameter Fitting

Similar to the parameter fitting process for 2OL-Eq, we identify the optimal duration parameter N_{MJ} with respect to positional SSE for each participant, task, and movement direction, using the respective mean trajectory as our reference. To improve convergence of the used optimization algorithm, we relax this parameter by allowing for continuous values of N_{MJ} in Equation (10) and compute the discrete-time MinJerk states x_n for $n \leq \lceil N_{MJ} \rceil$.

The optimal values of N_{MJ} grouped by participants are shown in the left plot of Figure 9. Different users can be characterized by different values for N_{MJ}. Interestingly, the user-specific effects match those observed for the stiffness parameter k in the 2OL-Eq model fairly well. Indeed, there is an inverse-linear relationship between k and N_{MJ}, as illustrated in Figure B.2 in the Appendix. Comparing the effects of k (Figure 4, top row) and N_{MJ} (Figure 8) on the respective model trajectories; however, this is not very surprising. Both a higher stiffness k and a lower surge duration N_{MJ} lead to a faster movement that reaches the target earlier (note that participant 9 moved considerably slower (average duration: 1.7s) than the rest of the participants (average duration: 0.95s)). Differences between the effects of these two parameters are mainly related to the initial acceleration, which scales with k in 2OL-Eq, but is always fixed in MinJerk, and to the skewness of the velocity profile, which is only affected by k.

In contrast, the average surge duration does not differ much between different task IDs (not shown). However, the *relative time spent in the surge*, i.e., N_{MJ}/N, clearly increases as the target width increases, while it is unaffected by the distance between initial and target center (see middle and right plot of Figure 9). This is consistent with previous findings, which suggest that the skewness of the velocity profile and thus the duration of corrective submovements is mainly determined by the target size, while the distance to target has a much smaller effect on the relative duration of the initial ballistic movement [16, 98, 135].

6.3 Discussion

The minimum jerk model can explain the shape of the initial ballistic movement (the surge) toward the target very well. However, both initial and terminal conditions need to be known in advance. The same holds for the overall movement time, unless it is identified through a parameter fitting process, using experimentally observed user data. It should also be noted that it is difficult to explain conceptually why users should aim at minimizing the jerk of the movement (see, e.g., Harris and Wolpert [58]).

The main limitation of the model is that it is a pure kinematic model. That is, the trajectory of the mouse pointer is uniquely defined given the initial and terminal conditions. The underlying reasons for the movement, such as the acting forces, are not explained. In particular, the model does not involve any explanation of the underlying biomechanics of the user, not even as a point-mass model such as 2OL-Eq. Due to its deterministic nature, it cannot account for the between-trial variability typically observed in user movements (see red dashed lines in Figure 7). Furthermore, as an open-loop model, the movement trajectory is completely specified at the beginning of the movement, and in its standard form, the model cannot react to perturbations or inaccuracies in the movement. In order to explain how users react to visual or proprioceptive feedback, models based on OFC are required.

7 POINTING AS OPTIMAL FEEDBACK CONTROL: THE LQR

In general, the OCP given by Equation (3) is very difficult to solve, since no solution method is known that guarantees convergence toward the global optimum without imposing (fairly strong) assumptions on the costs and system dynamics (e.g., convexity, continuous differentiability, etc.). One important subclass of problems where an analytic solution method exists is the case of **linear dynamics** and **quadratic costs**. The solution in this case is given by the LQR.

These OCPs usually are of the following form:

$$\begin{aligned} &\textit{Minimize} \quad J_N(x,u) = \sum_{n=0}^{N} x_n^\top Q_n x_n + \sum_{n=0}^{N-1} u_n^\top R_n u_n, \\ &\textit{with respect to } u = (u_n)_{n\in\{0,\ldots,N-1\}} \subset \mathbb{R}^m, \end{aligned} \tag{11a}$$

where $x = (x_n)_{n\in\{0,\ldots,N\}} \subset \mathbb{R}^k$ satisfies

$$\begin{aligned} x_{n+1} &= Ax_n + Bu_n, \quad n \in \{0,\ldots,N-1\}, \\ x_0 &= \bar{x}_0 \end{aligned} \tag{11b}$$

for some given initial state $\bar{x}_0 \in \mathbb{R}^k$.

As before, x_n is the state of the human–computer system, u_n is the control (e.g., muscle excitation), the matrix A describes the dynamics of the human–computer system if no control is applied, i.e., $u \equiv 0$, and B describes how the control influences the system (e.g., the force generated by muscles). The matrices Q_n and R_n can be interpreted as coefficients or weights for the state and control costs, respectively, where, e.g., the former formalizes that users aim to reach the target and the latter formalizes that they aim at doing so with minimal effort. Note that in our case, the controls u_n are one-dimensional ($m = 1$), i.e., the matrix R_n only consists of a single entry.

Regarding the optimal control framework for HCI, the minimization in Equation (11a) corresponds to the *Human Controller* block in Figure 1(b), and Equation (11b) corresponds to the

Human–Computer System Dynamics block. The *Feedback* and *Human Observer* blocks in Figure 1(b) are assumed trivial, i.e., observation and internal state estimate both equal the actual state of the system ($y_{n+1} = \hat{x}_{n+1} = x_{n+1}$). This is clearly different from the open-loop case depicted in Figure 1(a), where the controls are independent from the state estimates.

In the following, we use the muscle model and the cost function that have been used by Todorov [141] in the case of the (stochastic, and therefore significantly more complex) Linear-Quadratic Gaussian regulator, which we discuss in Section 8. To understand the stochastic case, it is useful to first consider the deterministic case, which we introduce in this section. The simplified second-order muscle model that we use has been proposed by van der Helm [147], and obtains control signals as input $u(t)$ and yields the forces applied to the end-effector $F(t)$ as output

$$F(t) = u(t) - \tau_1\tau_2\ddot{F}(t) - (\tau_1 + \tau_2)\dot{F}(t), \tag{12}$$

with time constants τ_1, $\tau_2 > 0$. Throughout this section, we use $\tau_1 = \tau_2 = 0.04$.

A discrete-time approximation of these dynamics is obtained by the *Forward Euler method* with time interval $h > 0$,

$$\begin{aligned} f_{n+1} &= f_n + \frac{h}{\tau_2}\left(g_n - f_n\right), \\ g_{n+1} &= g_n + \frac{h}{\tau_1}\left(u_n - g_n\right), \end{aligned} \tag{13}$$

where f_n and g_n denote the muscle activation (corresponding to force) and excitation at timestep n, respectively. Recall that $h = 0.002$ corresponds to the 2 ms sampling rate of the mouse sensor. Following Todorov [141], we assume a unit mass of 1 kg of the hand-mouse system.

Combining these muscle dynamics with a second-order integrator, this leads to the following system dynamics matrices:

$$A = \begin{pmatrix} 1 & h & 0 & 0 & 0 \\ 0 & 1 & h & 0 & 0 \\ 0 & 0 & 1 - \frac{h}{\tau_2} & \frac{h}{\tau_2} & 0 \\ 0 & 0 & 0 & 1 - \frac{h}{\tau_1} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{h}{\tau_1} \\ 0 \end{pmatrix}. \tag{14}$$

Here, the state $x_n = (p_n, v_n, f_n, g_n, T)^\top \in \mathbb{R}^5$ (i.e., $k = 5$ in Equation (11b)) consists of the pointer position p_n and velocity v_n, as well as muscle force f_n and muscle excitation g_n. Moreover, the fixed target T is included in the state for technical reasons. Note that, in contrast to the 2OL-Eq model, the controls u_n do not equal a fixed, target-dependent value (that is, we do not prescribe pure equilibrium control), but are chosen to minimize the cost J_N.

In principle, the cost matrices Q_n and R_n can be chosen freely. Based on [141], we derive them from the following assumptions:

— Users aim at minimizing the distance between pointer and target.
— Users aim at staying inside the target after reaching it.
— Users aim at minimizing the effort required to fulfill the task.

Ideally, no distance costs should occur within the target, which is assumed to be a box of width W. However, a fundamental limitation of the LQR setting is that cost terms need to be quadratic in the states and controls. Therefore, costs that are 0 everywhere within the target are unfortunately infeasible within the LQR setting. To approximate such costs, we construct the distance costs such that we have lower costs inside the target and higher costs outside. In particular, we penalize the squared remaining Euclidean distance between the end-effector position p_n and the desired target

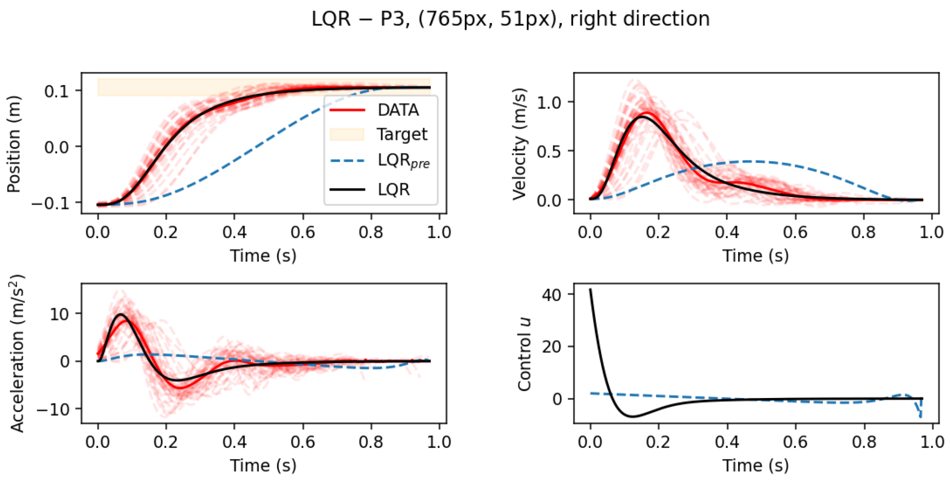


Fig. 10. If state costs are only applied at the final timestep N (blue dashed line), the LQR cannot capture the correction phase, which is highly pronounced for tasks with sufficiently small target. With continuous state costs (black solid line), LQR trajectories visually exhibit a good fit, although the peak acceleration is slightly higher. In contrast to MinJerk, the end of the surge phase does not need to be prescribed, but emerges implicitly from the model.

position T, which is given as

$$D_n^2 = |p_n - T|^2. \tag{15}$$

To create an incentive to keep the end-effector inside the target once it is reached, the squared velocity v_n^2 and the squared force f_n^2 (which can be interpreted as acceleration, since unit mass is assumed) are penalized as well, weighted with $\omega_v, \omega_f \geq 0$. All these cost terms are quadratic with respect to the state x_n, i.e., a positive semi-definite matrix Q_n can be found such that

$$x_n^\top Q_n x_n = D_n^2 + \omega_v v_n^2 + \omega_f f_n^2. \tag{16}$$

At the same time as minimizing the distance to the target, we assume that users aim at minimizing their effort. This assumption is well-established in motor control theory, motivated by both neuroscientific findings and mathematical requirements [54, 83, 142].[8] Moreover, we assume that the effort cost matrices are constant in time, i.e., $R_n = R$ holds for all $n \in \{0, \dots, N-1\}$, and normalized with respect to the duration of the movement for better comparability between conditions. In particular, we choose

$$R = \frac{\omega_r}{N-1}, \tag{17}$$

where the weight parameter $\omega_r > 0$ determines how much effort the user is willing to invest to reduce the distance to the target more quickly.

While in [141], the state costs Equation (16) were only applied at the final time $n = N$, which must be known in advance, in the deterministic LQR case without neuromotor noise, this does not work. As can be seen in Figure 10, for tasks with fairly small targets that require a considerable correction phase, the resulting LQR trajectory (dashed blue lines) does not resemble the observed user behavior at all. The main reason for this is that distance costs that only occur at a single timestep do not create an incentive to reach this state earlier than necessary. The constant

[8]In OCPs, penalizing the controls u acts as a *regularizer*, i.e., it constrains the subspace of optimal solutions, which often results in a unique optimal solution.

incurrence of control costs adds to this, resulting in an optimal policy that chooses relatively small controls during most of the movement, while shortly before the final timestep N, larger controls are applied to reach the target "just in time" with low velocity and acceleration.

This problem can be addressed in two ways. Either neuromotor noise can be included, as it is done in [141] and in the next section. Or, in the deterministic case, behavior more similar to humans can be achieved by assuming that the state costs from Equation (16) are applied *during the entire movement*. This clearly creates an incentive to move the pointer toward the target from the start, in order to reduce the sum of the distance costs.

In summary, the objective function of our LQR model is given by

$$J_N^{(\mathrm{LQR})}(x,u) = \sum_{n=0}^{N} \left(D_n^2 + \omega_v v_n^2 + \omega_f f_n^2 \right) + \frac{\omega_r}{N-1} \sum_{n=0}^{N-1} u_n^2. \tag{18}$$

We assume that the user computes the *optimal* control, denoted by u_n^*, in a feedback manner, based on the current state (i.e., the model is *closed-loop*). It has been proven that for these kinds of problems, the optimal control u_n^* depends linearly on the state [35], i.e.,

$$u_n^* = \pi(x_n) = -L_n x_n, \tag{19}$$

holds for some uniquely determined *feedback gain matrices* L_n. Recall that in our case, the control u_n is one-dimensional. Since the state x_n is a vector in $\mathbb{R}^5$, L_n is thus a 1×5 matrix.

Given the matrices A and B as well as the cost function $J_N^{(\mathrm{LQR})}$, these feedback gain matrices can be computed once before movement onset (i.e., at the planning stage) by iteratively solving the corresponding *Discrete Riccati Equation* (see Appendix; details are given in [138, Theorem 7]). This results in a very fast on-line computation of the optimal controls, yet taking into account the most recent state observations.

7.1 Analysis of Parameters

In Figure 11, the individual effects of the cost function weights ω_v, ω_f, and ω_r are shown. A higher velocity cost weight ω_v (top plots, blue dashed lines) results in a lower peak velocity, as expected, since velocity is penalized quadratically. Keeping the remaining parameters constant, this leads to a less symmetric velocity profile, as higher velocities toward the end of the movement are necessary to compensate for the lower peak. Moreover, the target is reached later. Note that this only occurs as long as the velocity cost weight is not dominant. Otherwise, for large enough ω_v, there is no incentive to reach the target at all.

Similar effects (peak values, symmetry) can be observed in the acceleration profile for the force cost weight ω_f (since the forces are applied to a unit mass and thus can be interpreted as acceleration) and in the control profile for the effort cost weight ω_r. Moreover, large force cost weights lead to a constant, positive velocity at the end of the movement (middle plots, velocity profile), i.e., the pointer moves across the target instead of staying inside (middle plots, position profile). In contrast, the magnitude of the effort cost weight ω_r mainly affects the duration of the surge (bottom plots, position profile), while the target is still reached with relatively low velocity and acceleration for moderate values of ω_r.

7.2 Results of Parameter Fitting

Analogously to the parameter fitting process for the 2OL-Eq and MinJerk models, we identify optimal values for the cost weights ω_v, ω_f, and ω_r collected in the paramter vector Λ for each mean trajectory. Since these parameters only define the objective function $J_N^{(\mathrm{LQR})}$ of the OCP, and the system matrices A and B are uniquely determined given the above fixed values of m, τ_1, and τ_2,

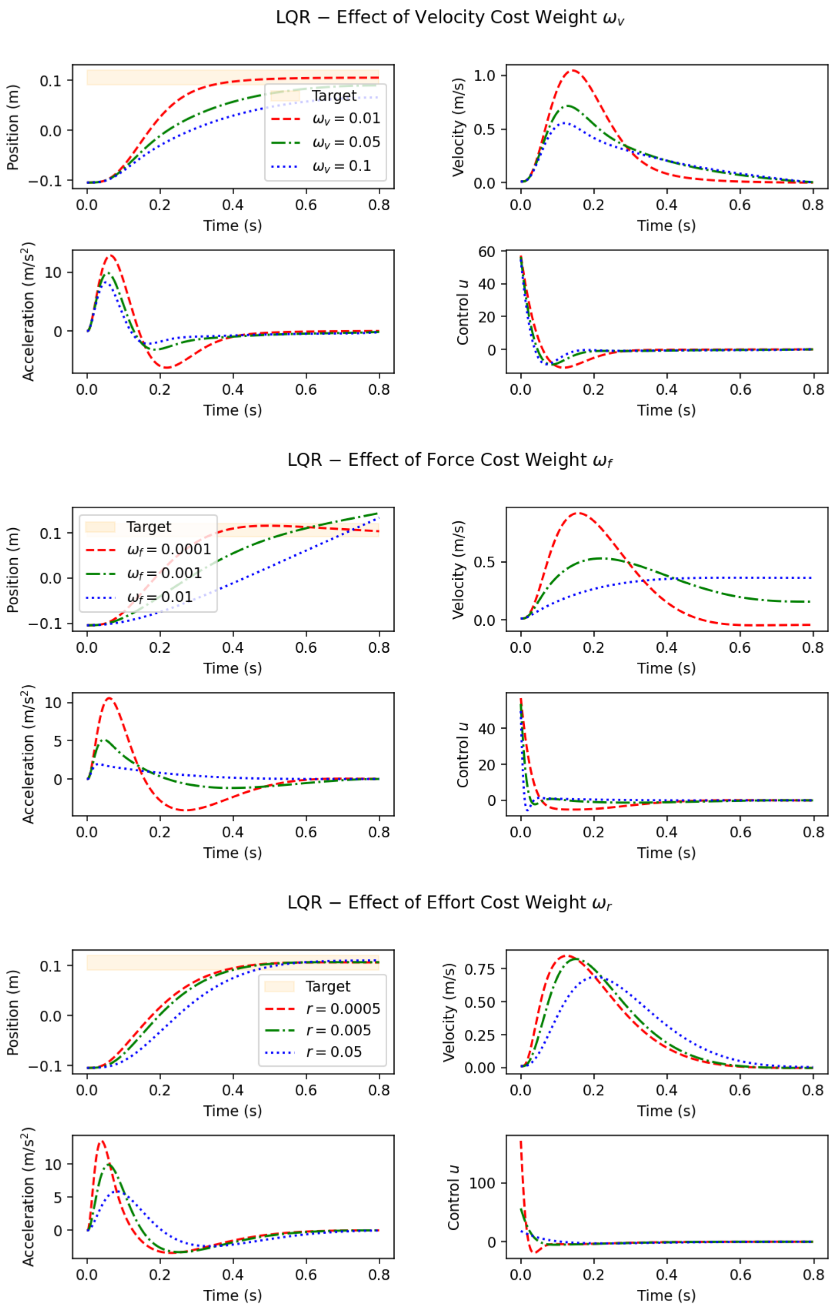


Fig. 11. Position, velocity, acceleration, and control time series of typical LQR trajectories with target shown as the orange box. **Top:** Effect of velocity cost weight ω_v with fixed cost weights $\omega_f = 0$, $\omega_r = 5e\text{-}3$ (red dashed: $\omega_v = 0.01$, green dash-dotted: $\omega_v = 0.05$, and blue dotted: $\omega_v = 0.1$). **Middle:** Effect of force cost weight ω_f with fixed cost weights $\omega_v = 0$, $\omega_r = 5e\text{-}3$ (red dashed: $\omega_f = 1e\text{-}4$, green dash-dotted: $\omega_f = 1e\text{-}3$, and blue dotted: $\omega_f = 1e\text{-}2$). **Bottom:** Effect of effort cost weight ω_r with fixed cost weights $\omega_v = 0.01$, $\omega_f = 1e\text{-}4$ (red dashed: $\omega_r = 5e\text{-}4$, green dash-dotted: $\omega_r = 5e\text{-}3$, and $\omega_r = 0.05$).

the parameter fitting for the LQR can be regarded as an inverse OCP [67].[9] As usual, we use the bi-level approach described in Section 4.2, i.e., at each iteration of the parameter fitting method, the OCP subject to the parameter vector Λ is solved as described above.

7.2.1 Qualitative Results. The resulting optimal LQR trajectory (for the same ID 4 task considered for the previous models) is shown as the black solid line in Figure 10. The position time series is approximated fairly well at first glance, even better than by the MinJerk trajectory (see Figure 20; quantitative comparisons between all models are given in Section 11.2). Peak velocity and acceleration are also very close to the values of the respective user trajectory, albeit the maximum acceleration is higher and the timing of the minimum and maximum acceleration does not match exactly. However, it is important to note that the duration of the surge phase was not explicitly built into the LQR model,[10] but emerges naturally from the interplay of the optimal parameters ω_v, ω_f, and ω_r. In contrast, in the MinJerk model, the duration of the surge phase needs to be either known in advance or determined using the parameter fitting process.

7.2.2 Quantitative Results. All optimal parameter values, grouped by both user and task ID, are shown in Figure 12 (note the logarithmic scale for ω_f and ω_r).

Each of the three parameters exhibits a large between-user variability. Interestingly, the between-user effects show similar trends and characteristics for all cost weights. For example, trajectories of participant 9 are characterized by very large weights (i.e., relatively small distance costs, since the weights affect every term in the cost function but the distance costs), while the behavior of participant 2 is explained best by considerably lower weights (i.e., relatively high distance costs). Moreover, these characteristic differences in the behavior of individual users are similar to those identified for the surge duration N_{MJ} in MinJerk (see Figure 9 (left)). In contrast, the task difficulty does not seem to have a clear effect on the optimal values of the cost weights. Hence, our findings suggest that the identified optimal parameters rather encode user-specific characteristics than the task under consideration.

7.3 Discussion

The LQR combines beneficial features of both 2OL-Eq (movement duration emerges from the model) and MinJerk (smooth movements with "close-to-bell-shaped" velocity profiles, as observed in many user studies). With a continuous penalization of remaining distance to target, velocity, and force *during the entire movement*, the LQR model is able to explain *average* user behavior in terms of position, velocity, and acceleration profiles. However, whether the modified cost terms are plausible from a biomechanical and neuroscientific perspective is debatable [8, 113]. Moreover, as the MinJerk model, due to its deterministic nature, the LQR model cannot account for the considerable between-trial variability, which is typically observed in user movements.

In the next section, we will thus introduce signal-dependent control noise, which in combination with continuous effort costs and only terminal distance, velocity, and force costs, allows to replicate user behavior similarly well as the proposed LQR variant, with the additional benefit of obtaining *a distribution of* optimal trajectories.

[9] Including the system dynamics parameters m, τ_1, and τ_2 in the parameter fitting process did not improve the fit to observed user behavior.

[10] In theory, it would be possible to include prior knowledge about submovements and thus induce time-dependent behavior by choosing the time-dependent cost matrices Q_n and R_n appropriately.

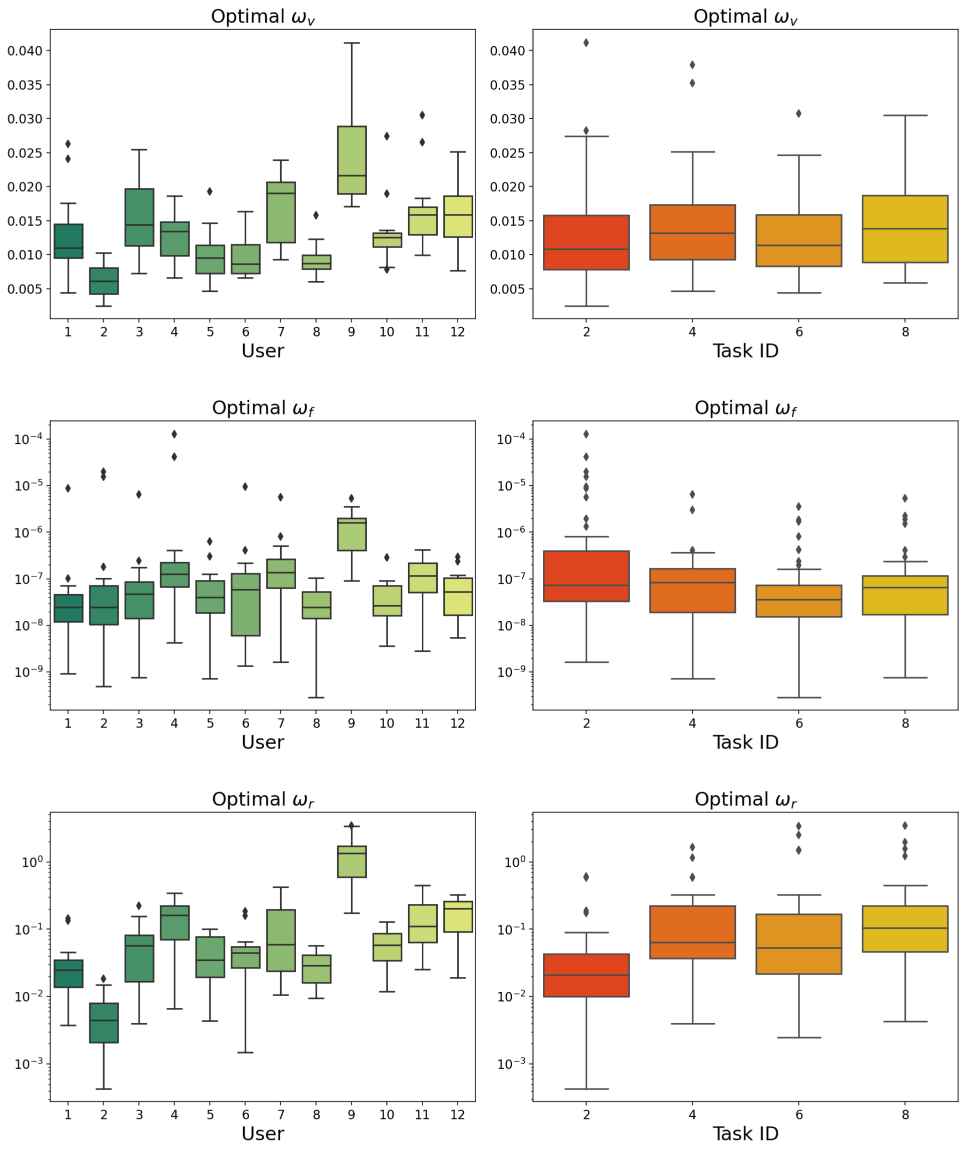


Fig. 12. Parameters of the LQR model, optimized for the mean trajectories of all participants, tasks, and directions, grouped by participants (left) and by ID (right). For better visibility, the optimal values of ω_f and ω_r are plotted on a logarithmic scale.

8 POINTING AS OPTIMAL FEEDBACK CONTROL SUBJECT TO SIGNAL-DEPENDENT MOTOR NOISE: THE LQG

While the LQR model visually captures typical *average* user behavior relatively well (see Figure 10; for a quantitative comparison between different models, see Section 11), it has one major drawback: The proposed optimal trajectory is necessarily deterministic and thus cannot account for the large variability observed among multiple trials of the same participant for the same task [32, 142].

An extension that allows to consider different sources of variability, which might occur during the sensorimotor control loop, is the *LQG* [62, 87, 141]. It belongs to the class of ***Stochastic***

Optimal Feedback Control (SOFC) models, as it takes into account the variance resulting from various noise terms.

In the following, we will present and analyze the LQG model introduced by Todorov for reaching movements [141]. In comparison to the LQR model from Section 7, this model includes signal-dependent Gaussian control noise and an observation model with additive Gaussian noise. Moreover, all state costs (i.e., distance, velocity, and force costs) are only applied at the final timestep N.

8.1 Linear-Quadratic Gaussian Regulator (LQG)

Starting from the linear-quadratic control problem (11), we extend the deterministic LQR model presented in the previous section with stochastic noise terms. In principle, there are two main types of noise terms that can be included in the system dynamics: additive noise and multiplicative noise.[11] While the former introduces the same level, or variability, to the system at each timestep, the variance of multiplicative noise depends on the system variables themselves. In the considered case of *signal-dependent (multiplicative) control noise*, a larger magnitude of the applied control u_n thus results in a higher uncertainty about the subsequent state x_{n+1}. From a neuroscientific perspective, this dependency can be justified by the empirically observed effect of neural motor commands on the variance of the resulting motor-neuronal firing [26, 68, 118, 129]. In particular, signal-dependent noise can account for well-established phenomena such as cosine tuning, muscle synergies, smooth movements, and the tradeoff between speed (or duration) and the end-point accuracy of a movement [58, 139, 140].

Introducing the *control noise level* $\sigma_u > 0$ and the sequence $(\eta_n)_{n\in\{0,\dots,N-1\}}$ of (univariate) Gaussian random variables $\eta_n \sim \mathcal{N}(0;1)$, the resulting discrete-time system dynamics can be written as

$$\begin{aligned} x_{n+1} &= Ax_n + (1+\sigma_u\eta_n)Bu_n, \quad n \in \{0,\dots,N-1\}, \\ x_0 &\sim \mathcal{N}(\bar{x}_0, \Sigma_0), \end{aligned} \tag{20}$$

where the initial state x_0 is drawn from a multivariate Gaussian distribution with given mean $\bar{x}_0$ and covariance matrix Σ_0. The specific values for $\bar{x}_0$ and Σ_0 can be extracted from data, see Section 8.2. Following [141], we do not include additive control noise, albeit this would be easily possible in the proposed framework.

As in the previous section, the state $x_n \in \mathbb{R}^5$ contains pointer position, velocity, force, and muscle excitation, as well as the target position. We assume that the controller, which needs to decide for a control $u_n \in \mathbb{R}$ at each timestep $n \in \{0,\dots,N-1\}$, does not have complete access to the current state of the system x_n. The main reason is that usually, not all information stored in x_n is observable. This means that the control u_n at timestep $n \in \{0,\dots,N-1\}$ may depend only on some observation $y_n \in \mathbb{R}^l$ ($l \in \mathbb{N}$), but not on the true state x_n. Moreover, the observation can be noisy, i.e., errors during observation may occur. More specifically, to maintain linear model dynamics, we assume that the (noisy) observation y_n is linear in x_n, i.e.,

$$y_n = H_n x_n + G\xi_n. \tag{21}$$

[11]In this article, we only consider *white noise*, i.e., the noise terms that occur at different timesteps are assumed to be independent. However, the LQG model could be extended to incorporate temporarily correlated noise by augmenting the state space accordingly (for details, see [138, Section 3.4.3]).

For simplicity, we assume that position, velocity, and force values can be observed immediately in global coordinates, while muscle excitation and target position cannot be directly observed, i.e.,

$$H_n = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \tag{22}$$

which results in

$$H_n x_n = (p_n, v_n, f_n)^\top . \tag{23}$$

The error that occurs during observation is modeled by the additive noise term $G\xi_n$. The 3×3 matrix G determines the individual noise levels. As in [141], it is given by

$$G = \sigma_s \; diag(0.02, 0.2, 1), \tag{24}$$

where $\sigma_s > 0$ is a scaling parameter and the constants were chosen in order to reflect the different magnitudes between position, velocity, and force (i.e., acceleration). The vector ξ_n is a three-dimensional Gaussian random variable, i.e., $\xi_n \sim \mathcal{N}(0; I_3)$, where I_3 denotes the 3×3 identity matrix. In particular, the observation y_n is a three-dimensional vector.

The objective function is related to the one used in the LQR model, Equation (18), i.e., a weighted combination of distance, velocity, and force costs, plus effort costs. The two differences are: (i) distance, velocity, and force costs incur only at the final timestep N, while only the effort costs incur continuously throughout the movement, and (ii) since state and control are not deterministic due to noise, we use the expected value of these terms, denoted by $\mathbb{E}[\cdot]$. The objective function is thus given by

$$J_N^{(\mathrm{LQG})}(x, u) = \mathbb{E}\left[D_N^2 + \omega_v v_N^2 + \omega_f f_n^2 + \frac{\omega_r}{N-1}\left(\sum_{n=0}^{N-1} u_n^2\right)\right]. \tag{25}$$

In total, this results in the following stochastic OCP:

$$\begin{gathered} Minimize \quad J_N^{(\mathrm{LQG})}(x, u), \\ with\ respect\ to\ u = (u_n)_{n \in \{0,\dots,N-1\}} \subset \mathbb{R}^m, \end{gathered} \tag{26a}$$

where $y = (y_n)_{n \in \{0,\dots,N-1\}} \subset \mathbb{R}^l$ satisfies

$$y_n = H_n x_n + G\xi_n, \quad n \in \{0, \dots, N-1\}, \tag{26b}$$

and $x = (x_n)_{n \in \{0,\dots,N\}} \subset \mathbb{R}^k$ satisfies

$$\begin{gathered} x_{n+1} = Ax_n + (1 + \sigma_u \eta_n)Bu_n, \quad n \in \{0, \dots, N-1\}, \\ x_0 \sim \mathcal{N}(\bar{x}_0, \Sigma_0). \end{gathered} \tag{26c}$$

We specifically use the dimensions m, l, and k throughout to underline the generality and easy expandability of the LQG model. In our case, we recall that $m = 1$, $l = 3$, and $k = 5$.

Since the true state x_n at timestep n is not available to the controller, it needs to compute internal state estimates $\hat{x}_n$ based on the information available. Under the LQG assumptions (linear dynamics, quadratic costs, Gaussian noise), the state estimates $\hat{x}_n$ can be computed using a linear estimator:

$$\hat{x}_{n+1} = A\hat{x}_n + Bu_n + K_n(y_n - H_n\hat{x}_n), \quad n \in \{0, \dots, N-1\}. \tag{27}$$

Here, y_n and $H_n\hat{x}_n$ denote the obtained and the expected sensory input at time step n, respectively (recall the *Human Observer* block in Figure 1(b)), where initially, at $n = 0$, we set $\hat{x}_0 = \bar{x}_0$. The matrix K_n, which is to be determined, specifies to which degree the observed differences between these quantities should be taken into account for the computation of the subsequent state estimate $\hat{x}_{n+1}$. As a trivial example, consider $K_n = 0$, which corresponds to the open-loop case, i.e., no sensory information is used by the controller. When it comes to choosing a "good" matrix K_n, the above LQG assumptions allow to analytically derive the matrix that is optimal in the sense that it minimizes the objective function $J_N^{(\mathrm{LQG})}$. The resulting optimal estimator is known as the *Kalman Filter*.[12]

Moreover, it can be shown that, similar to the LQR case, the optimal closed-loop solution $u^* = (u_n^*)_{n\in\{0,\dots,N-1\}}$ is linear in the state estimate $\hat{x}_n$, i.e., there exist unique *feedback gain matrices* L_n, $n \in \{0,\dots,N-1\}$ such that

$$u_n^* = \pi(\hat{x}_n) = -L_n\hat{x}_n, \tag{28}$$

holds. As in the previous section, L_n is a 1×5 matrix. For more details, we refer to [141].

It is important to note that in the considered case of signal-dependent control noise, the feedback gain matrices L_n and the Kalman gain matrices K_n non-trivially depend on each other. More precisely, each feedback gain matrix L_n depends on the subsequent Kalman filter matrices $(K_i)_{i\in\{n+1,\dots,N-1\}}$, and each Kalman filter matrix K_n depends on the previous feedback gain matrices $(L_i)_{i\in\{0,\dots,n-1\}}$. Thus, the feedback gain matrices L_n can only be computed backward in time, starting at the final timestep $n = N$, whereas the Kalman filter matrices K_n can only be computed forward in time, starting at the first timestep $n = 0$. In particular, one must be given in order to optimally determine the other. Fortunately, iterating alternately between the two optimizations results in a coordinate descent algorithm, which can be shown to converge toward a (local) minimum [141].[13] This iterative computation of the matrices K_n and L_n can be done offline, i.e., before movement onset.

As suggested by Todorov [141], we therefore initialize the Kalman gain matrices to 0, i.e., we first compute the optimal open-loop control strategy matrices L_n, and iteratively compute K_n and L_n until the resulting objective function value converges toward its optimum. In particular, we terminate the iterative optimization procedure in the ith step if the relative improvement of $J_N^{(\mathrm{LQG})}$ falls below a predefined threshold $\epsilon_J > 0$, i.e.,

$$\left| \frac{\left(J_N^{(\mathrm{LQG})}\right)_{i+1} - \left(J_N^{(\mathrm{LQG})}\right)_i}{\left(J_N^{(\mathrm{LQG})}\right)_i} \right| \leq \epsilon_J, \tag{29}$$

or after a maximum number of iterations (set to 20; we found that this value sufficed to obtain parameters that did not change considerably afterwards). In our implementation, we use ϵ_J = 1e-3.

We do not make use of the *adaptive Kalman filter*, as it was proposed for an LQG model designed for via-point movements [138]. The adaptive Kalman filter computes Kalman gain matrices K_n that explicitly depend on the observations $(y_i)_{i\in\{0,\dots,n-1\}}$ received so far, i.e., this needs to be done online. Moreover, since these observations are stochastic, the matrices K_n (and thus the expected behavior) differ between several runs. In addition, it was empirically observed that the adaptive Kalman filter might be unstable [141]. In contrast, the non-adaptive Kalman filter from [141] computes matrices K_n that only depend on information available *before movement onset.* The resulting a priori expectations over the state sequences can be directly used in the loss function of the parameter fitting process to compare the expected outcome of different parameter vectors.

In principle, using an adaptive filter only during runtime (with L_n optimized with respect to the non-adaptive filter) could further reduce the expected total costs. However, this effect has shown to be minor [141], which is why we refrain from using an adaptive filter at all.

[12]In the case of state-dependent observation noise, i.e., if G is replaced by $G_n(x_n)$, the Kalman Filter is still optimal among all linear estimators [138].

[13]Numerically, it has been shown that this algorithm even converges toward the global minimum [141].

8.2 Analysis of Parameters

The above mentioned a priori expectations over state trajectories, which are used to compute the non-adaptive Kalman filter, allow to compare the stochastic results of the LQG controller between different sets of parameters. In total, there are five parameters we aim at optimizing:

- the (terminal) velocity cost weight ω_v,
- the (terminal) force cost weight ω_f,
- the effort cost weight ω_r,
- the signal-dependent control noise level σ_u, and
- the observation noise level σ_s.

The first three parameters correspond to those from Section 7.1. In particular, setting all other parameters as well as the initial variance to 0 and using exact initial state estimates $\hat{x}_0 = \bar{x}_0 = x_0$ and $\hat{\Sigma}_0 = \Sigma_0 = 0$, the stochastic OCP Equation (26) equals the deterministic OCP (11), i.e., the LQR is a special case of the LQG. However, in contrast to the LQR model from Section 7, distance, velocity, and force costs are only applied at the final timestep N. The effect of the individual cost weight parameters is thus much smaller.

In principle, the same fixed noise level parameters σ_u and σ_s could be used for all participants and task conditions. However, choosing them from literature is difficult since their effects strongly depend on the assumptions and system dynamics of the considered model.

As shown in Figure 13 (top plots), an increase in the velocity cost weight ω_v results in a higher peak velocity, which is attained earlier. This can be explained by the corresponding decrease of the end-point velocity, which effectively reduces the total costs. This is also visible from the entries of the optimal feedback gain matrices L_n, which are depicted in Figure 13 as well. The magnitude of the position component[14] of L_n determines the importance of correcting the error between end-effector and target at timestep n, whereas the magnitude of the velocity component of L_n determines the importance of adjusting the end-effector velocity. For the LQG model, an increase in the velocity cost weight ω_v results in a later peak of the velocity component, i.e., the controller would correct for a large velocity that occurs very shortly before the end of the movement, in order to reduce the terminal velocity costs. In the position component of the feedback gain matrices, a second, earlier peak occurs for moderate values ($\omega_v = 1$, green dash-dotted line). This suggests that it is particularly important to the controller to eliminate deviations from the target both at the end of the surge phase (peak at 0.212 s for the considered case) and at the end of the movement (peak at 0.68 s). For large velocity cost weights ($\omega_v = 20$, blue dotted line), the earlier peak becomes more dominant and remains the only peak, i.e., end-point errors that occur shortly before the end of the movement are not corrected anymore, as this would result in a larger end-point velocity and thus a higher total expected cost. In addition, the (relative) length of the correction phase increases as the terminal velocity cost weight increases. It is important to note that these effects only occur under both observation noise and signal-dependent control noise, i.e., for σ_u, $\sigma_s > 0$.

Terminal force costs only, i.e., without terminal velocity costs, cannot account for the typically observed corrective movements following the surge (see Figure 13, middle plots). In particular, there is no incentive to reduce the velocity toward the end of the movement. Instead, the acceleration is reduced, which is visible from the constant velocity at the end of the movement for large ω_f

[14]We recall that to compute the resulting control u_n, each entry of the matrix L_n is multiplied by the corresponding entry of the state x_n, see Equation (28). In our case, L_n is a 1×5 matrix, which allows for easy interpretation of the matrix entries. For example, the first entry of L_n is multiplied by the first entry of x_n, which is the position, and hence referred to as the *position component* of L_n.

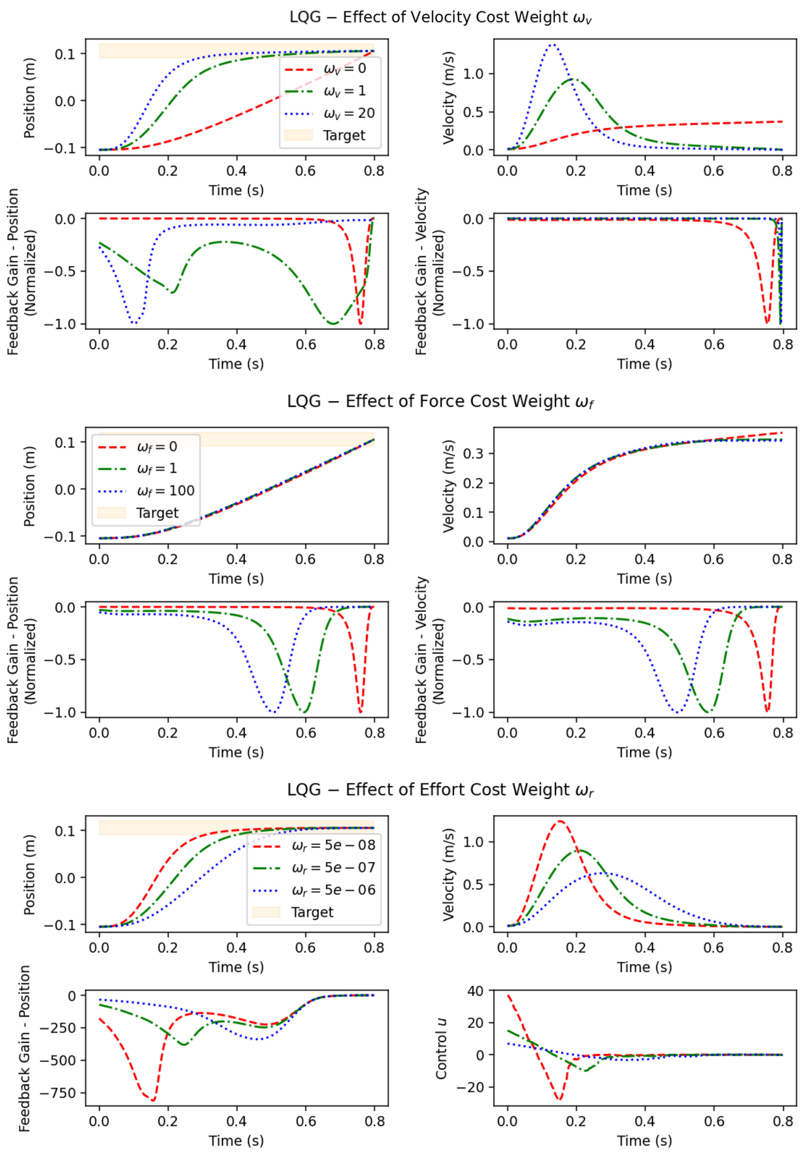


Fig. 13. Typical LQG trajectories with target shown as the orange box, as well as selected entries of the corresponding feedback gain matrices, and control time series, with noise levels $\sigma_u = 0.2$, $\sigma_s = 0.5$. **Top:** Effect of velocity cost weight ω_v with fixed cost weights $\omega_f = 0$, $\omega_r =$ 1e-7 (red dashed: $\omega_v = 0$, green dash-dotted: $\omega_v = 1$, and blue dotted: $\omega_v = 20$). **Middle:** Effect of force cost weight ω_f with fixed cost weights $\omega_v = 0$, $\omega_r =$ 1e-7 (red dashed: $\omega_f = 0$, green dash-dotted: $\omega_f = 1$, and blue dotted: $\omega_f = 100$). **Bottom:** Effect of effort cost weight ω_r with fixed cost weights $\omega_v = \omega_f = 2$ (red dashed: $\omega_v =$ 5e-8, green dash-dotted: $\omega_v =$ 5e-7, and blue dotted: $\omega_v =$ 5e-6).

(green dash-dotted and blue dotted lines). Similarly to the velocity cost weight ω_v, the force cost weight ω_f also affects the time period at which deviations from the desired terminal position and velocity are corrected. As ω_f increases, this period shifts forward, i.e., late-occurring deviations are not corrected anymore.

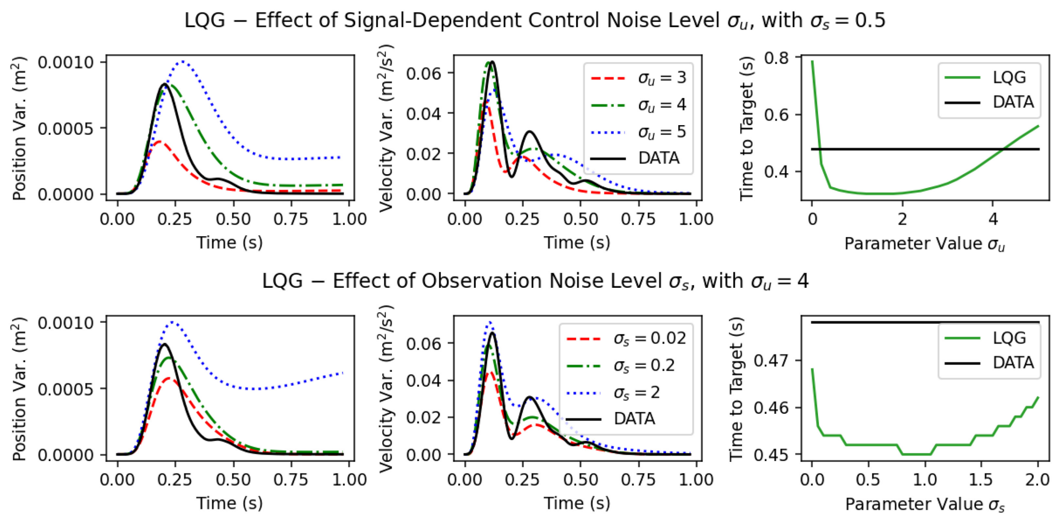


Fig. 14. **Top:** Effect of the signal-dependent noise level σ_u on the variance time series of the LQG position and velocity profiles (red dashed: $\sigma_u = 3$, green dash-dotted: $\sigma_u = 4$, and blue dotted: $\sigma_u = 5$) and on the time until the target is reached; all simulations with observation noise level $\sigma_s = 0.5$. **Bottom:** Effect of the observation noise level σ_s on the variance time series of the LQG position and velocity profiles (red dashed: $\sigma_s = 0.02$, green dash-dotted: $\sigma_s = 0.2$, and blue dotted: $\sigma_s = 2$) and on the time until the target is reached; all simulations with signal-dependent noise level $\sigma_u = 4$.

For positive velocity and force cost weights ω_v and ω_f, an increase of the effort cost weight ω_r leads to a trajectory that reaches the target later, with lower peak velocity (see Figure 13, bottom plots). This is intuitive, since higher effort costs reduce the magnitude of the optimal control signals, resulting in lower accelerations and velocities. The terminal velocity is close to 0 for any moderate ω_r.

Note that the shape of the optimal control sequences differs considerably from the deterministic LQR model (see Figure 11). In the LQG model, the control usually attains its maximum at the beginning, then linearly decreases toward its minimum, and increases again toward 0. This is mainly due to the velocity and the acceleration being penalized only at the final timestep N, which allows to reach a higher peak velocity and acceleration (achieved through larger control signals at the beginning of the movement; also note the large (non-normalized) positional feedback gain values in bottom left plot). Under the assumption of signal-dependent control and constant observation noise, the control is very close to 0 during the correction phase (which, for ID $\geq$ 4, makes up a considerable part of the movement). In Figure 13 (bottom right plot), this is shown for moderate control and observation noise levels $\sigma_u = 0.2$, $\sigma_s = 0.5$.

If the control noise σ_u is set large enough, similar effects can be observed without any observation noise, i.e., $\sigma_s = 0$ (not shown). In contrast, large observation noise levels σ_s cannot compensate a missing control noise. This is intuitively plausible—since the controller is aware that there is no control noise, the deterministic system states resulting from the closed-loop system become perfectly predictable through the internal (correct) forward model, i.e., there is no need to rely on noisy observations at all.

The effects of the control noise level σ_u (with $\sigma_s = 0.5$) and the observation noise level σ_s (with $\sigma_u = 4$) are shown in Figure 14. In the left and in the middle plots, the variance of the resulting LQG position and velocity profiles is plotted against time. The black solid lines show a representative variance profile from the Pointing Dynamics Dataset. Visually, the combination of signal-dependent control noise and constant observation noise can explain the characteristic

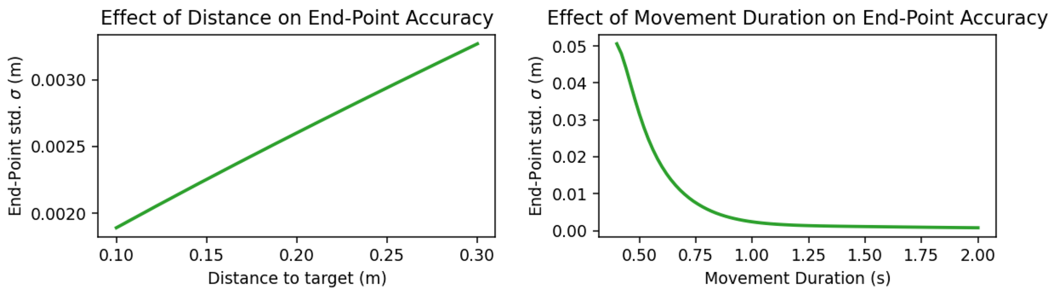


Fig. 15. Under signal-dependent noise and given a fixed movement duration (here: 0.97s), the end-point standard deviation of the LQG linearly increases with distance. Given a fixed distance (here: 0.21m), the end-point standard deviation inverse-quadratically decreases with movement duration.

variance profiles relatively well. Under signal-dependent noise, the application of smaller controls in the second half of the movement (during the correction phase, which follows the surge) results in a decreasing movement variability toward the end of the movement. This is in accordance with the two-phase positional variance profiles that are typically observed in aimed movements [50, 55, 78]. From an information-theoretic perspective, the decaying rate of these profiles can be explained by a user-specific channel capacity [50]. In the considered LQG model, the idea of user-specific control and observation noise levels affecting the (expected) end-effector variability provides a different interpretation. However, very large and physically implausible noise levels ($\sigma_u \approx 1{-}4$) are required to account for the substantial variance observed in the position and velocity profiles of mouse pointing movements. The velocity variance profiles, which are typically bimodal [30, 55], can also be replicated by the noise model of the LQG, albeit the second peak is usually less pronounced in the simulation.

The effect of the signal-dependent noise level σ_u on the (average) time until the target is reached is depicted in the top right plot of Figure 14, with total movement duration N corresponding to 0.97s. Note that the time a target is reached does not have to coincide with the total movement duration. This is because the movement does not end upon reaching the target, but instead is the experimentally observed time between two mouse clicks. To distinguish these two, we will abbreviate the average time until the target is reached by *time to target* in the following.

Without signal-dependent noise, i.e., $\sigma_u = 0$, there is little incentive to reach the target much earlier than at timestep N, which is the only timestep at which the positional error, velocity, and acceleration are penalized. As σ_u increases, the movement duration rapidly decreases. This can be again explained by the need to apply lower controls toward the end, in order to reduce the uncertainty regarding the final end-effector position. Signal-dependent noise thus induces a strategy that could be described as "doing most of the work at the beginning to be on the safe side". However, if σ_u becomes too large, its effect on the movement time reverses: The more signal-dependent noise, the more time is required to reach the target. This is intuitive, since higher control noise levels result in a larger positional variance of the end-effector, which, in turn, forces the controller to more rely on the received observations when updating the internal state estimate. In combination with a positive observation noise level σ_s, an increase in σ_u thus results in a larger uncertainty regarding the own position, which makes it difficult to reach the target at the same time. Such an increase of the time to target with σ_u cannot be observed if observation noise is omitted (not shown).

The assumption of signal-dependent control noise is also in accordance with the well-known speed-accuracy tradeoff, which suggests that faster movements result in a larger end-point variance [58, 142, 154]. As can be seen in Figure 15 (left), for a fixed movement duration N, the positional end-point accuracy (i.e., the standard deviation of the terminal end-effector position) linearly increases with the initial distance to target. This is plausible, as larger movements require higher controls u_n, and is consistent with empirical findings [49, 138, 141, 155]. Similarly, an inverse relationship between movement duration and end-point standard deviation can be observed (see Figure 15 (right)).

For the observation noise level σ_s, similar effects on the variance profiles and the time to target can be observed, given a fixed control noise level σ_u (bottom row of Figure 14). The increasing variance in position and velocity as σ_s increases can also be explained by the increasing uncertainty regarding the own end-effector position as σ_s increases, as this uncertainty in turn affects the variance of the applied muscle control. For moderate observation noise levels σ_s, this effect decreases toward the end of the movement, since smaller controls are applied then, i.e., the future system states become less dependent on the internal state estimates. However, if σ_s becomes too large, the internal state estimates are not precise enough to reliably move the end-effector toward the target. In this case, the positional variance remains constant or even increases toward the end of the movement (blue dotted line in the bottom left of Figure 14). Note, however, that *on average*, the target is still reached early during the simulation (after 0.46 s for $\sigma_s = 2$, which is even slightly faster than the representative user trajectory for the considered task shown as black line in the bottom right plot of Figure 14).

In summary, the mutual dependency between observations and applied controls, i.e., between the Kalman gains K_n and the feedback gains L_n, implies a positive effect of both signal-dependent control noise and constant observation noise on the movement variability, whereas the effect on the time to target depends on the absolute value of both noise levels. However, the effects of σ_s are considerably smaller than the effects of σ_u (note the logarithmic scale of σ_s for both variance profiles, and the smaller linear scale in the time to target). In particular, observation noise only has an effect *in combination with control noise*, because otherwise the system is deterministic, i.e., the controller does not need to rely on sensory input at all.

8.3 Results of Parameter Fitting

Using our parameter fitting process, we identify the optimal values of all five parameters for each combination of participant, task condition, and direction within the Pointing Dynamics Dataset. Since the LQG model yields a sequence of state *distributions*, we use the stochastic parameter fitting variant described in Section 4.2, with 2–Wasserstein distance applied to the position-velocity components of the respective state distributions as the loss function.

The parameters of the initial distribution, $\bar{x}_0$ and Σ_0, from Equation (26c) are specified as follows. We choose $\bar{x}_0 = (p_0^{\text{USER}}, v_0^{\text{USER}}, 0, 0, T)^\top$, where p_0^{USER} and v_0^{USER} denote the average initial position and velocity, respectively, of all trials for the considered combination of participant, task condition, and direction. The covariance Σ_0 of the initial state is defined as follows. The components for position and velocity correspond to the sample covariance matrix empirically observed from these user trajectories, as described in Section 4.1. All other components are set to 0.

8.3.1 Qualitative Results. The optimal LQG solution for the same representative ID 4 task as used for the previous models is shown in Figure 16, where both the mean trajectories and the 95% point-wise confidence bands are plotted. The mean position and velocity profiles (green solid lines) visually match those from user data (red solid lines) very well. The same holds for the acceleration time series, apart from the second submovement (starting around 0.3 s), which is slightly less pronounced in the simulation. The most notable differences are that in the LQG model, there is a slightly larger positional variance at the transition from the ballistic to the corrective phase of the

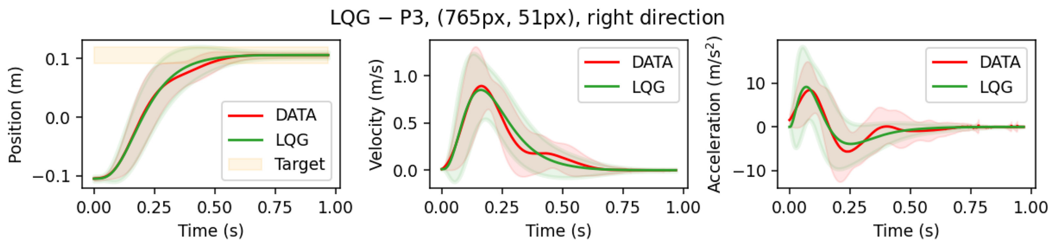


Fig. 16. Visually, LQG trajectories show a good fit to typical user trajectories. The mean trajectories (solid lines) as well as the 95% point-wise confidence bands match very well at first glance. Some simulation movements exhibit a slightly too low velocity at the beginning of the movement. The LQG acceleration profiles are obtained by applying a Savitzky–Golay filter of degree 3 and frame size 15 to the respective velocity profile and differentiating the corresponding polynomials.

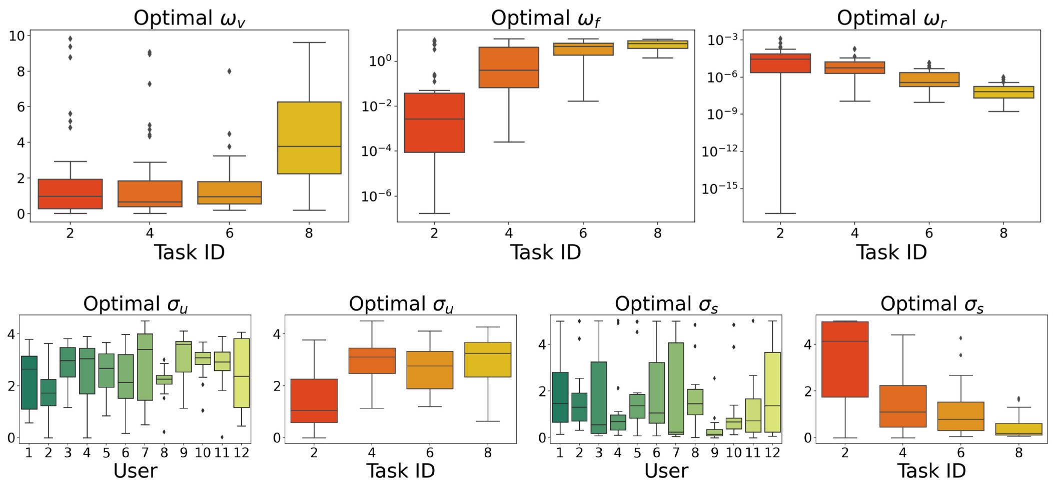


Fig. 17. Parameters of the LQG model, optimized for the trajectory sets of all participants, tasks, and directions, grouped by participants (left) and by ID (right). Note that the optimal values of ω_f and ω_r are plotted on a logarithmic scale.

movement (see also Figure 20), and a larger variance in velocity and acceleration at the beginning of the movement.

8.3.2 Quantitative Results. In Figure 17, optimal parameter values of the LQG are shown, grouped by both user and task ID. For better visibility, the values of ω_f and ω_r are plotted on a logarithmic scale.

In contrast to the LQR model, the optimal cost weights are more affected by the task ID than by the individual participants. Since in the LQG model, distance, velocity, and force costs are only applied in the final timestep N, the corresponding weights can be interpreted as importance of the endpoint accuracy constraint relative to keeping the required effort low.

The force cost weight ω_f monotonously increases with task ID, whereas the velocity cost weight ω_v takes similar values for ID 2, 4, and 6 tasks, and is considerably larger for ID 8 tasks. Since the velocity cost weight ω_v mainly affects the relative time spent in the surge phase (see Figure 13, top plots), the latter finding suggests that movements for very difficult tasks (ID 8) exhibit a considerably longer correction phase. Similarly, the force cost weight ω_f determines the time period at which positional errors and large velocities are corrected (see Figure 13, middle plots). The monotonous increase of ω_f with task ID thus implies that the more difficult the task, the less attention is paid to deviations near the end of the movement.

The effort cost weight ω_r exponentially decreases as the task ID increases. This is not surprising, since the controller does not have explicit knowledge on the target width W, but only on the distance to target D (via the initial end-effector and the target position, which are both included in $\bar{x}_0$). Instead, the desired increase of end-point accuracy as W decreases needs to be implemented via the cost weights. The observed decrease of the effort cost weight ω_r as the task becomes more difficult, which is equivalent to a relative increase of the terminal costs, can thus be interpreted as a higher importance of keeping the end-effector inside the target (higher accuracy) with small velocity and force at the final time step N (higher stability).

The task ID also has an effect on the two noise level parameters σ_u and σ_s. The signal-dependent control noise level σ_u takes considerably lower values for ID 2 tasks, while the observation noise level σ_s decreases as the task becomes more difficult.

While in contrast to the LQR model, the effect of the participants on the optimal cost weights is less pronounced (see Figure B.3 in the Appendix), different users are clearly characterized by different noise levels. For example, the trajectories of participant 8 can be explained by a small control noise level σ_u (and a rather large observation noise level σ_s), whereas participant 9 is best explained by a larger control noise level σ_u (and a small observation noise level σ_s).

8.4 Discussion

The LQG model assumes that users behave optimally with respect to a combination of terminal distance, velocity, and force costs, as well as continuous effort costs, within the constraints imposed by the human–computer system dynamics, and subject to signal-dependent motor noise and constant observation noise. Using the presented stochastic parameter fitting, this allows for an excellent replication of user trajectories.

As shown in Figure 16, trajectories resulting from the presented parameter fitting process capture both the average behavior and between-trial variability that is typically observed in mouse pointing movements.

However, the observation model from [141] has some shortcomings, as it assumes that all relevant quantities are perceived instantaneously in global coordinates and perturbed by additive Gaussian noise only. Moreover, the target position is assumed to be perfectly known during the entire movement. Thus, online comparisons between the target signals obtained from an appropriate observation model (i.e., H_n such that $H_n x_n$ includes T, see Equation (23)) and those predicted by the internal model would not yield any additional benefit. This, however, is in contradiction to many empirical observations, which suggest that visual stimuli are internally used whenever available, even in the considered case of serial movements between the same two targets [116, 138].

In the following section, we thus extend the LQG model by considering both fixation-centered and world-centered sensory input signals, inspired by [138].

9 POINTING AS OPTIMAL FEEDBACK CONTROL SUBJECT TO SIGNAL-DEPENDENT NOISE AND SACCADES: THE E-LQG

The observation model of the LQG model in Section 8, which was taken from [141], has some major drawbacks. In particular, it assumes that the position, velocity, and force of the end-effector can be directly observed in world-centered coordinates, only perturbed by additive Gaussian noise whose magnitude is also known to the controller.

In the following, we will thus present an extension of this LQG model, denoted by *E-LQG*, which includes a more complex and physiologically plausible human observation model. The main concepts are taken from [138], with an adaption to the considered case of mouse pointing. In contrast to [138], we assume that the end-effector position cannot be observed via proprioception (i.e., in world-centered coordinates), but only from visual input (i.e., relative to the current eye fixation position), since it corresponds to the mouse cursor position in our case.

Compared to the LQG model from Section 8, the E-LQG model

— models eye movement based on accurate saccades between the initial and the target position,
— distinguishes between visual input (in fixation-centered coordinates) and observations of the eye fixation position (in world-centered coordinates), and
— works with an imperfect initial target estimate, which is updated during the movement based on sensory input.

In the following, these differences are described in more detail.

Eye Saccades. For the considered goal-directed movements, sensory input can be assumed to be based on *eye saccades*, i.e., fast movements of the fovea between two fixation points [74, 75, 157]. The number and choice of fixation points usually depends on the complexity of the observed scene, the underlying goal (i.e., which information should be extracted from visually input), and the salience of individual objects, among others [46, 119, 133]. However, previous experiments on via-point tasks suggest that a single and precise movement of the fovea toward the aimed target is sufficient for the considered case of reciprocal pointing toward clearly delimited target areas [138]. Following [138], we thus can assume that at the beginning of the movement, the eye fixation corresponds to the initial position (regarding the repetitive movements from the Pointing Dynamics Dataset, this is equivalent to the target position of the previous movement). At a certain time during the arm movement, the gaze is assumed to move toward the target, which is then fixated until the end of the trial.

However, the eye saccades are decoupled from the rest of the movement in the sense that the controller can neither modify the time of the saccade nor the fixation points. Instead, we assume that both fixations are accurate (which can be argued by combining "possible corrective saccades in one "saccade" moving the eyes from one target to another" [138]), and optimize the saccade time within the outer parameter fitting process.[15]

Visual Input. We assume that visual input signals yield information regarding the position of the end-effector (i.e., the mouse pointer), the target, and the initial position, each *relative to the eye fixation position* (i.e., in fixation-centered coordinates).[16]

Based on the principles of foveal and peripheral vision [82, 128], these observations are assumed to be disturbed by noise that linearly increases in the distance between the respective object (pointer, initial/target box) and the eye fixation point. This is a major difference to the LQG model from Section 8.1, which included additive observation noise only. In addition, both the end-effector velocity and acceleration (which corresponds to force due to the assumption of unit mass) are perceived from visual input channels with additive noise, i.e., the magnitude of the observation noise is assumed independent of the distance to the eye fixation. The rationale for using additive noise here is that the minimum detectable difference between velocities is known to hardly differ between the peripheral and the foveal field. This particularly implies that useful observations of the end-effector velocity can be obtained independent of whether the end-effector moves close to the fixation point [91].

[15]In the future, it will be interesting to include the eye position in state space and make it controllable via some (simplified) eye dynamics, similar to the simplified muscle dynamics that are used to control the end-effector.

[16]Note that depending on whether the saccade has already taken place, the relative initial position (before the saccade) or the relative target position (after the saccade) equals 0 and thus can be discarded from the observation space.

Proprioceptive and Eye Fixation Input. Proprioceptive signals are signals that refer to the own body position and movement. While the visual input channel yields fixation-centered observations of the end-effector and the target position, proprioception could be used to obtain world-centered estimates of the own body position and orientation, e.g., of the arm, hand, or head. The used Human–Computer System Dynamics, however, do not explicitly distinguish between quantities corresponding to the human body and quantities corresponding to the input device or interface. Instead, it gives the overall dynamics that are directly applied to the virtual end-effector (see the discussion in Section 3.2). In the considered case of mouse pointing, the end-effector corresponds to the mouse cursor shown on the display, which cannot be perceived proprioceptively. In contrast to [138], we thus do not include world-centered observations of the end-effector position in the proposed model.

Besides feedback on the end-effector position and orientation, the human body usually also provides information about the eye position. In recent years, a large debate has evolved about whether the cortical eye position is rather obtained via proprioception or using internal efference copies of "outflow" signals [92, 150]. For details on the neurophysiological mechanisms underlying coordinated eye-hand movements, we refer the interested reader to the excellent overview given in [123]. Regarding the observation model used for the E-LQG model, we assume that the eye fixation point (perturbed by additive noise) can be perceived in world-centered coordinates. Note that the eye fixation dynamics are not part of the Human–Computer System Dynamics, but are implicitly modeled via the following workaround. Th e two attainable values are included in the state, and the time of the instantaneous switch is determined via some parameter (more details are given in Section 9.1).

In summary, the eye fixation is assumed to take only two different values during an aimed movement: the initial position (before the saccade) and the target position (after the saccade). In particular, the (perturbed) target position can be observed in world-centered coordinates, as soon as the saccade has taken place.

Internal Target Estimate. We assume that at the beginning of the movement, the controller is not aware of the exact target position. This is intuitively plausible, since even in the considered case of reciprocal tasks, where the users know that the two same targets will appear alternately, visual input signals are known to be used to improve the (rough) prior target estimates during the movement [116, 138].

In both the LQG and the E-LQG models, the internal state estimates $\hat{x}_n$ include an estimate of the desired target. However, the controller in the LQG model is given an *exact* initial estimate $\hat{x}_0$, i.e., $\hat{x}_0$ includes the correct (mean) initial position, velocity, force, muscle excitation, and target position. Since the target is known to be constant during the movement, this immediately implies a correct target estimation during the complete movement. In contrast, in the E-LQG model, we assume that the target component of the initial estimate differs from the actual target position T. For simplicity, we assume that this initial target estimate corresponds to the *initial position* T_0 (see Section 9.1), that is, the center of the target box of the previous movement in the Pointing Dynamics Dataset; note that both initial and target box were permanently displayed in the experiment [98]. This is in agreement with the reciprocal nature of the movements from the Pointing Dynamics Dataset, where the initial position should in turn equal the target position of the preceding movement. During the mouse movement, the internal target estimate, i.e., the target component of $\hat{x}_n$, is then updated based on the perceived sensory input signals y_n.

9.1 LQG with Extended Observation Model (E-LQG)

Based on the above assumptions, we modify the LQG model presented in Section 8 as follows.

In order to model eye fixation of the initial position T_0, we first need to include T_0 in the state space. We thus define $x_n = (p_n, v_n, f_n, g_n, T_0, T)^\top \in \mathbb{R}^6$, i.e., the state vector now consists of the end-effector position, velocity, force, muscle excitation, the (fixed) initial position, and the (fixed) target position. Moreover, we introduce the *saccade timestep* n_s, which defines the time at which the eye fixation switches from initial to target position. In order to be able to optimize this saccade timestep within the stochastic parameter fitting process from Section 4.2, we relax this parameter by allowing continuous values, i.e., $n_s \in [0, N]$.

The observations y_n are more complex than in the LQG model, see Equation (21). They are given by the following observation model[17]:

$$y_n = H_n x_n + G_n(x_n)\xi_n, \tag{30}$$

where the observation matrix H_n depends on whether the saccade has taken place or not. Before the saccade has taken place, the unperturbed observations $H_n x_n$ include the end-effector velocity, the end-effector acceleration (corresponding to muscle activation and force), the initial position (i.e., the center position of the initial boundary box, which corresponds to the eye fixation position), as well as the end-effector and target position, both relative to the the initial position. After the saccade has taken place, the unperturbed observations $H_n x_n$ include velocity, acceleration, the target position (which now corresponds to the eye fixation position), as well as the end-effector and initial position, both relative to the the target position. At saccade timestep $n = \lfloor n_s \rfloor$, we use a convex combination of the two observations.[18] To this end, we formally define the two matrices,

$$H^{(T_0)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}, \quad \text{and} \quad H^{(T)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}, \tag{31}$$

and introduce $\{n_s\}$, which denotes the fraction part of n_s, i.e., $\{n_s\} = n_s - \lfloor n_s \rfloor$. Then, we can define the observation matrix H_n as

$$H_n = \begin{cases} H^{(T_0)}, & \text{if } n < \lfloor n_s \rfloor, \\ H^{(T)}, & \text{if } n > \lfloor n_s \rfloor, \\ \{n_s\}H^{(T_0)} + (1 - \{n_s\})H^{(T)}, & \text{if } n = \lfloor n_s \rfloor. \end{cases} \tag{32}$$

The signal-dependent observation noise is introduced by the second term on the right-hand side of Equation (30). Here, the vector ξ_n is a five-dimensional Gaussian random variable, i.e., $\xi_n \sim \mathcal{N}(0; I_5)$, where I_5 denotes the 5×5 identity matrix. The observation noise matrix G_n is defined by

$$G_n(x_n) = \begin{cases} diag(\sigma_v, \sigma_f, \sigma_e, \gamma|p_n - T_0|, \gamma|T - T_0|), & \text{if } n < \lfloor n_s \rfloor \\ \begin{aligned} diag(\sigma_v, \sigma_f, \sigma_e, \gamma\left(\{n_s\}\left(|p_n - T_0|\right) + (1 - \{n_s\})\left(|p_n - T|\right)\right), \\ \gamma\left(\{n_s\}\left(|T - T_0|\right) + (1 - \{n_s\})\left(|T_0 - T|\right)\right)), \end{aligned} & \text{if } n = \lfloor n_s \rfloor \\ diag(\sigma_v, \sigma_f, \sigma_e, \gamma|p_n - T|, \gamma|T_0 - T|), & \text{if } n > \lfloor n_s \rfloor \end{cases}. \tag{33}$$

The end-effector velocity and force are perturbed by visual noise levels σ_v and σ_f, respectively. Similarly, the eye fixation position is perturbed by the gaze noise level σ_e. The magnitude of the

[17]Note that the generality and flexibility of the proposed framework in principle allows to incorporate multiple and imprecise saccades, as well as more sophisticated approaches to model visual input [37]. However, this is beyond the scope of this work.

[18]This relaxation ensures that the gradient of the state sequence x with respect to n_s becomes non-zero, which is necessary to apply standard continuous optimization methods within the parameter fitting process.

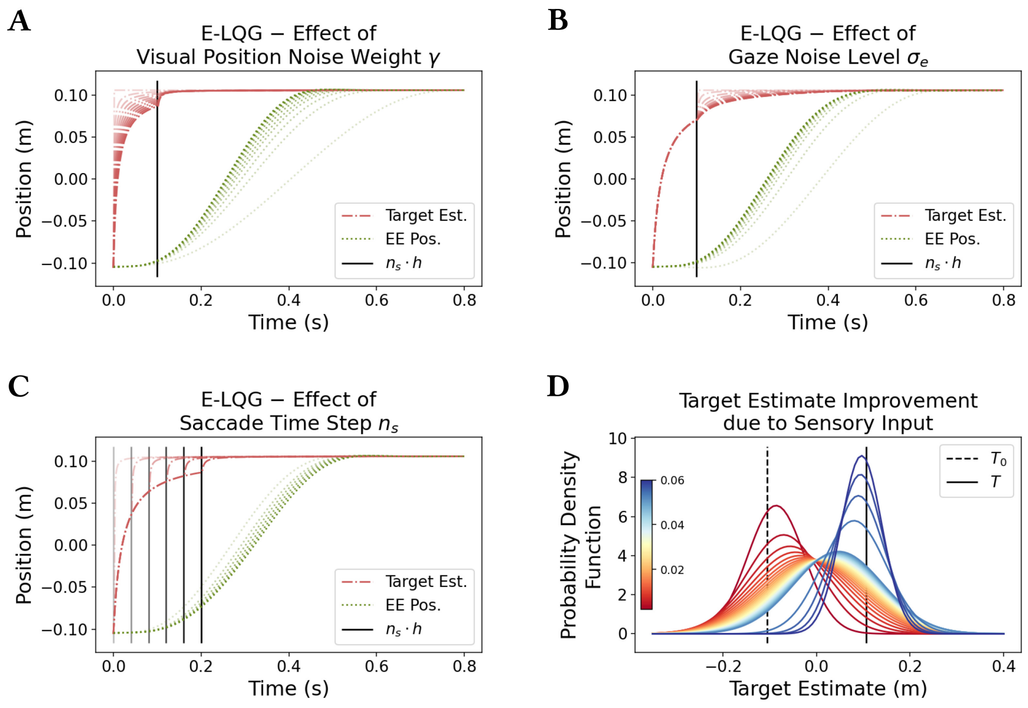


Fig. 18. Internal estimates of the target position in the E-LQG model, as well the resulting (expected) position profile of the end-effector. In (A)–(C), darker lines correspond to larger parameter values. We set $\omega_v = 2$, $\omega_f = 0.02$, $\omega_r = 1\text{e-}7$, $\sigma_v = 5$, $\sigma_f = 1$, $\sigma_u = \sigma_c = 0$, $h = 0.002$, and performed at least three iterations to compute the Kalman and feedback gain matrices K_n and L_n, see Equation (29). (A): Effect of position perception noise weight γ (between 0 and 5) with fixed gaze noise level $\sigma_e = 0.1$ and saccade timestep $n_s = 50$, i.e., at 0.1 s. (B): Effect of gaze noise level σ_e (between 1e-10 and 0.1) with fixed position perception noise weight $\gamma = 10$ and saccade timestep $n_s = 50$. (C): Effect of saccade timestep n_s (between 0 and 100) with fixed position perception noise weight $\gamma = 10$ and gaze noise level $\sigma_e = 0.1$. (D): Development of the internal target estimate probability density function over time, with position perception noise weight $\gamma = 10$, gaze noise level $\sigma_e = 0.1$, and the saccade occurring after 0.05 s.

visual position observations depends on the respective distance to the eye fixation point, scaled by the parameter γ. This is consistent with Weber's Law [27], which claims that the minimum required stimulus changes that lead to a considerable change in the visual perception (that is, changes that are larger than the perceptual noise) are linear in the absolute value of the respective signal, suggesting that the perceptual noise linearly depends on this absolute value as well [85].

9.2 Analysis of Parameters

The effect of the parameters γ, σ_e, and n_s on both the internal target estimate and the resulting (expected) end-effector position time series is shown in Figure 18(A)–(C), with darker lines corresponding to larger parameter values. The parameter γ denotes the scaling weight of the Gaussian noise term added to the visually observed positions, which is multiplied by the respective distance to the eye fixation point. The constant magnitude of the Gaussian noise added to the eye fixation position is denoted by σ_e, and n_s denotes the timestep at which the saccade occurs.

Starting with an eye fixation of the initial position (which is known at the beginning of the movement and thus correctly estimated), a higher position perception noise weight γ leads to a slower update of the internal target estimate from initial position to true target position (red dash-dotted lines in plot A). Interestingly, this does not delay the end-effector movement, but rather results in a faster movement toward the target (green dotted lines in plot A). A possible explanation for this phenomenon could be the that moving the end-effector early after the saccade toward the internal target estimate can improve the internal estimate of the end-effector position, since the eyes fixate the target center after the saccade. This means that the variance of the end-effector position observations linearly decreases as the distance between end-effector and target decreases. Since the terminal costs create an incentive to keep the end-effector at the target center with zero velocity and acceleration in the final timestep N, it is thus important to obtain reliable estimates of both the target position (which is quite accurate after the saccade has taken place, see next paragraph) and the own end-effector position early in time, to avoid expensive last millisecond corrections. A large scaling parameter γ intensifies this problem, as a smaller distance between end-effector and target is necessary to achieve the same amount of visual observation noise. Thus, a larger position perception noise weight γ results in an earlier movement toward the target. However, this effect only holds for moderate values ($\gamma \leq 5$); if the visual observation noise γ becomes too large, more time is required to obtain a reliable internal target estimate, resulting in a more tentative (i.e., slower) movement toward the estimated target position (not shown).

Similar effects can be observed for the gaze noise level σ_e, that is, the (constant) magnitude of Gaussian noise that is added to the observation of the eye fixation. As soon as the saccade toward the target has taken place (after 0.1s in the shown example), the target estimate is significantly improved, as it can be estimated from both visual input and the eye fixation observation. Thus, σ_e mainly determines the convergence rate of the internal target estimate *after* the saccade (red dash-dotted lines in plot B). Moreover, a larger σ_e incentivizes a (slightly) faster movement toward the internal target estimate, in order to further improve this estimate and thus reliably keep the end-effector inside the actual target at the end of the movement, when terminal costs incur.

The effect of the saccade timestep n_s on both the target estimate and the end-effector position profile is shown in plot C. An increase in n_s delays the movement toward the target, as world-centered information on the target position, which become available to the controller at time n_s via observation of the eye fixation point, considerably improve the internal target estimate, i.e., it is worth waiting a little longer.

In plot D, the development of the internal target estimate over time is depicted. Starting with the prior target estimate T_0 (dashed line), the mean of the normal distributions shifts toward the true target T (solid line) as more sensory input becomes available. Note that the largest improvements occur at the beginning and after the saccade, i.e., after 0.05 s. The variance first increases[19] and then slowly decreases toward 0.

9.3 Results of Parameter Fitting

The E-LQG model uses the following nine parameters:

— the (terminal) velocity cost weight ω_v,
— the (terminal) force cost weight ω_f,
— the effort cost weight ω_r,
— the signal-dependent control noise level σ_u,
— the velocity perception noise level σ_v and the force[20] perception noise level σ_f,

[19]Note that the initial variance estimate of the target component equals 0, i.e., the LQG model initially assumes that the target is at the initial position T_0 with probability 1, which is why we skip the density function at time 0 and plot the density function starting at the second timestep, i.e., after 0.002 s.
[20]Recall that the controlled system is assumed to have unit mass, i.e., the applied force is equivalent to the acceleration.

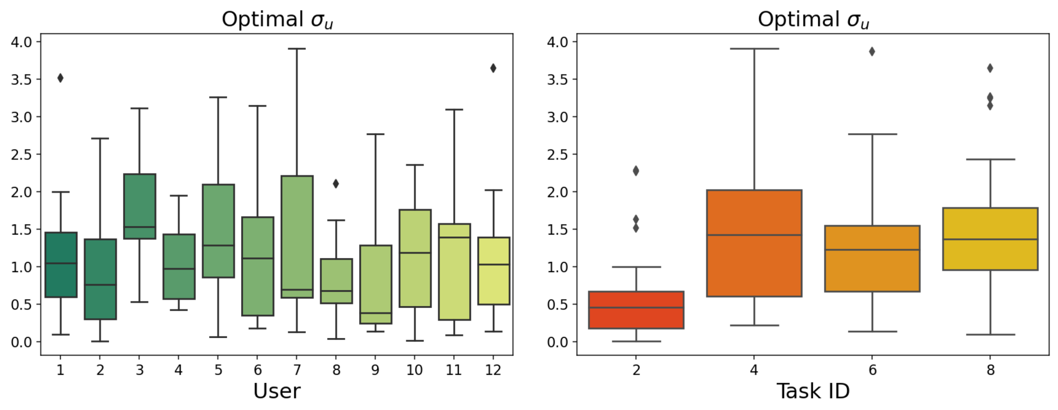


Fig. 19. Control noise parameter σ_u of the E-LQG, optimized for the mean trajectories of all participants, tasks, and directions, grouped by participants (left) and by ID (right).

— the gaze noise level σ_e,
— the position perception noise weight γ, and
— the saccade timestep n_s.

We again identify the optimal parameter vector for each combination of participant, task condition, and movement direction, using the stochastic parameter fitting procedure from Section 4.2. Qualitatively and quantitatively, the results do not change much from the LQG model. We thus focus only on parameters which exhibit noticeable differences.

As shown in Figure 19, the optimal control noise level σ_u exhibits clear differences between individual users and task IDs, similar to the LQG case. However, the optimal values are considerably lower (the mean value amounts to 1.18 (E-LQG) and 2.52 (LQG)). In Section 11, we show that the E-LQG model replicates the observed user trajectories similarly well compared to the LQG model. This suggests that the extended observation model of E-LQG allows to replicate the typical variance profiles with lower (i.e., more plausible) signal-dependent control noise levels.

The decrease of observation noise with higher ID, which was clearly noticeable for the single observation noise parameter σ_s in the LQG model, occurs only for the velocity perception noise level σ_v (see Figure B.4 in the Appendix; note the logarithmic scale in both plots). Instead, the magnitude of the noises σ_f and γ do not exhibit a clear dependency on the ID, and vary between individual participants. This suggests that the observation model of E-LQG makes it more meaningful than the LQG model. The optimal parameter values of σ_e and n_s increase with ID (at least for ID $\geq$ 4), and also exhibit characteristic differences between individual users (see Figure B.5 in the Appendix).

10 THE INTERMITTENT CONTROL MODEL (IC)

In the remaining part of this article, we will compare the presented models against each other, both qualitatively and quantitatively. In particular, we analyze and discuss the ability of the stochastic models LQG and E-LQG to predict not only individual trajectories, but entire trajectory distributions. Since 2OL-Eq, which we use as a baseline for the deterministic optimal control models, is not capable of predicting movement variability, we need another, stochastic baseline model. We decided to use a model from IC theory, which recently has been proposed by Martin et al. [160]. In the following, we will give a short overview of the similarities and differences between IC and OFC.

In both IC and classical OFC models, internal models of the interaction loop are used to find controls that are optimal with respect to a certain cost function. As a major difference, OFC continuously integrates the stream of obtained sensory input signals to account for unexpected disturbances and correct internal state estimates accordingly, whereas IC only intermittently makes use of these observations [34, 45, 105, 160]. In particular, IC allows to include a minimum open-loop interval between two successive events, in which no feedback is available to the controller [160]. From a neuroscientific perspective, this is consistent with the theory of *psychological refractory periods*, which assumes the existence of short periods of time after a visual stimulus has been processed, in which the controller cannot react to further changes in the environment [45, 134].

More precisely, in IC, an open-loop control based on an internal representation of the system dynamics (the so-called *hold*) is applied until the difference between predicted and observed state exceeds some predefined threshold, i.e., until the unaccounted disturbances become too large. In this case, an event is triggered, which updates the internal model based on a new sample from the continuously perceived stream of observations. Afterwards, the open-loop control that is optimal for the updated internal model is applied until the next event is triggered, and so on, resulting in an IC. IC models can thus be regarded as a hybrid of open-loop and closed-loop models.

We decided to use the IC model from [160], since it is based on similar assumptions (optimal control with respect to accuracy, stability, and effort costs, subject to the constraints imposed by the system dynamics), and also has been applied to mouse movements, using the same Pointing Dynamics Dataset and a parameter identification process similar to ours. Moreover, the IC model is also able to replicate movement variability in terms of phase space probability distributions [160]. It thus constitutes a suitable baseline for the considered stochastic OFC models.

However, the variability is generated completely differently in the two approaches. In LQG/E-LQG, between-trial variability arises from noise terms that are explicitly modeled in the Human–Computer System Dynamics (e.g., signal-dependent control noise, or observation noise), which allows to analytically compute the expected mean and covariance matrices (see Section 8.1). In contrast, both the control and observations dynamics of the IC model from [160] are assumed deterministic. While motor and/or observation noise could be included in principle, the IC model is based on the LQR, i.e., it is not capable of taking into account the expected variance due to such noise terms when computing the optimal control strategy.

In contrast, in the IC model, motor variability is only due to a *multiple-model approach*, i.e., multiple movements are generated by using different parameter vectors, which are randomly drawn from a bank of identified parameter vectors. This is a major difference from the presented SOFC models, where we have identified only *one parameter vector* for each user, task condition, and direction, such that the resulting trajectory distribution captures both average user behavior and between-trial variability.

Further differences between the IC model and the LQG/E-LQG include the system dynamics (in the IC model, the same fourth-order dynamics are used, but with slightly different time constants $\tau_1 = \tau_2 = 0.05$), and the observation model, which for the IC only yields (unperturbed) positional information.

10.1 Technical Details

The IC simulation trajectories, which we will use as a baseline for the LQG and the E-LQG models in the following, were generated by Martin et al. [160]. For each combination of participant and task condition, they performed 200 simulations of 20 subsequent “slices” (i.e., combinations of rightward and leftward movements), by randomly drawing from a bank of 20 identified parameter vectors. Since we analyze unidirectional movements in this article, we split each of these slices at the respective target switch time, resulting in a total of 4,000 simulated IC movements for each participant, task condition, and direction.

For the comparison between the IC simulation data and observed user data, we then clip the trajectories of *both* datasets to some $N \in \mathbb{N}$, since the lengths of the IC trajectories were chosen to match the lengths of the respective dataset slices (in contrast to LQG/E-LQG, where the optimal trajectory distribution sequence necessarily yields sample trajectories of pre-defined, equal length). We define the maximum IC trajectory length N_{IC} using the same outlier criteria (both with respect to trajectory length and position values at each timestep) as for the maximum user data trajectory length N_{USER} (see Section 4.1), and then cut both distribution sequences to length $N = \min(N_{\text{IC}}, N_{\text{USER}})$.[21] We also remove the reaction times from all IC simulation trajectories, using the same procedure as for the Pointing Dynamics Dataset (see Section 4.1).

Finally, we compute the sample mean and covariance matrices of the resulting set of IC trajectories on a frame by frame basis, resulting in one trajectory distribution sequence for each participant, task condition, and direction.

11 COMPARISON BETWEEN MODELS

In the following, we provide a detailed comparison of the six presented models, both qualitatively and quantitatively.

11.1 Qualitative Comparison

A comparison of all simulation trajectories for the regarded user and task condition can be found in Figure 20, where the mean position and velocity is shown for all considered models, and the variance in position and velocity is shown for all stochastic models.

Deterministic Models. The deterministic models can only predict average behavior (top row). The 2OL-Eq trajectory (blue lines) has a too large velocity at the beginning of the movement, which is a direct consequence of the high initial acceleration, as discussed in Section 5. The trajectory of the MinJerk model (orange lines) exhibits a perfectly bell-shaped velocity profile, with peak velocity very close to that observed in the user trajectory (black lines). However, MinJerk cannot explain the required corrective submovements toward the end of the movement. Instead, it assumes that the target is reached after the first ballistic movement, i.e., the surge. For trajectories with clearly visible submovements, this results in a considerable worse overall fit of both the position and the velocity time series. In addition, the duration of this surge does not emerge from MinJerk, but needs to be explicitly fitted to the desired user trajectory. In contrast, the LQR model (green lines) approximates the mean trajectory well in terms of position and velocity, although the corrective movement is not pronounced.

Stochastic Models. Both the LQG and the E-LQG models do not only model average behavior well (top row), but also account for the between-trial variance observed in both position and velocity profiles (bottom row). The variance profiles of the LQG model (red lines) are similar to those of the user data (black lines). For some trials (as the one shown in Figure 20), the E-LQG trajectory distribution sequence (purple lines) fits slightly worse in terms of positional and velocity variance; in particular, the peaks of both variance profiles are considerably lower for the E-LQG compared

[21]Note that, given a participant, task condition, and direction, the length N used to compute a measure of similarity between simulation and user trajectories, such as MWD, might thus be slightly lower for IC than for the other models, where N was set to N_{USER}.

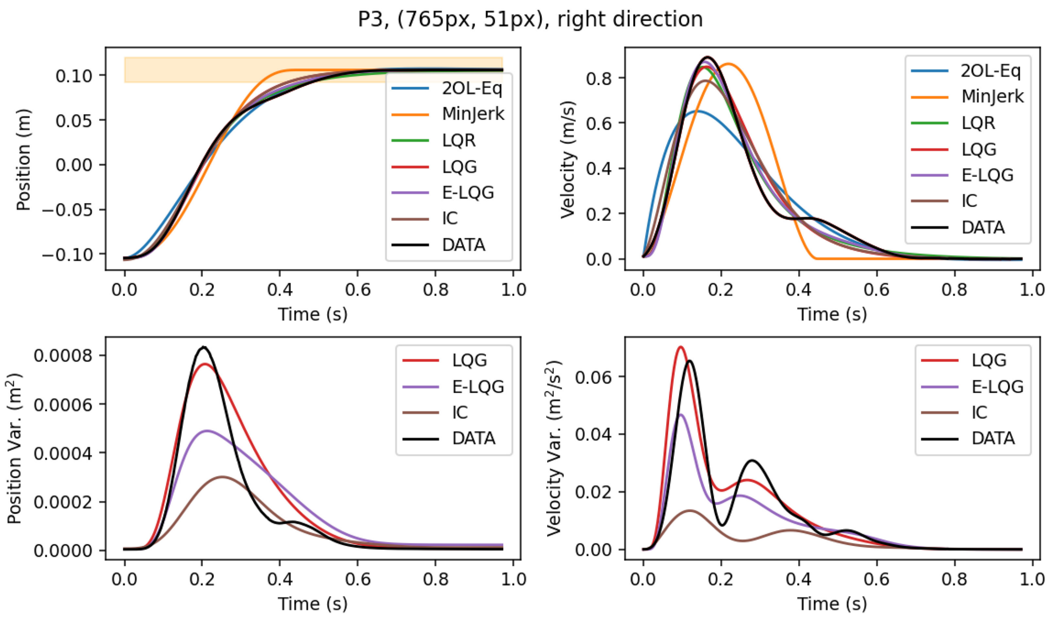


Fig. 20. Comparison between all considered models in terms of mean and variance of both position and velocity, using the same task condition (participant 3, ID 4 (765px distance, and 51px width), rightward movement) as in the previous sections.

to user data. The IC simulation trajectories (brown lines) exhibit an even lower variance in terms of both position and velocity. Moreover, the peak velocity of IC is lower compared to user data and all other considered models (except for 2OL-Eq).

Corrective Submovements. Corrective submovements are not replicated well by any of the six models. MinJerk is extended by a constant position value after the surge and thus naturally cannot account for granular corrections of the end-effector position, which are visible in the velocity time series of the user data, starting around 0.36 s. The remaining models slowly reduce the velocity toward the end of a movement. However, clear submovements, i.e., additional peaks in the velocity profile (around 0.42 s in the user data), are not visible.

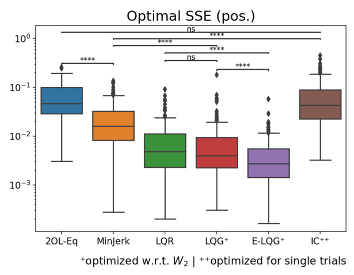
11.2 Quantitative Comparison

We start with a comparison of how well each model is able to predict average user behavior, i.e., how close their simulated trajectories resemble the mean trajectories computed from the Pointing Dynamics Dataset. Although the parameter fitting of the deterministic models (2OL-Eq, MinJerk, and LQR) was performed with respect to positional SSE only, we also evaluate the SSE with respect to mean velocity and acceleration. In addition, we measure the positional *Maximum Error* between model and user trajectories, i.e.,

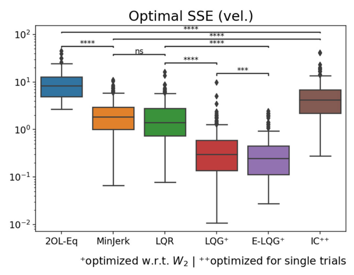
$$\max_{n=0,\ldots,N} \left| p_n^{\Lambda} - p_n^{\text{USER}} \right|, \tag{34}$$

and analogously the Maximum Error in velocity and acceleration.

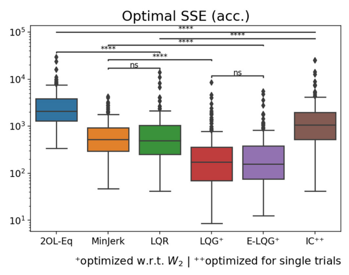
In addition to the 2−Wasserstein distance, we also consider the **mean *KL divergence (MKL)*** [76] over time to further evaluate the performance of the stochastic models (LQG, E-LQG, and IC). Moreover, we compare the mean trajectories of the stochastic models with respect to SSE and



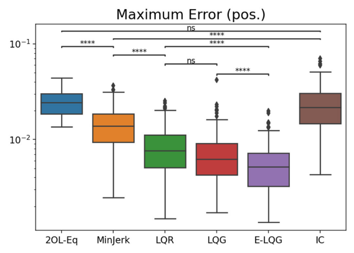
(a) SSE - Position



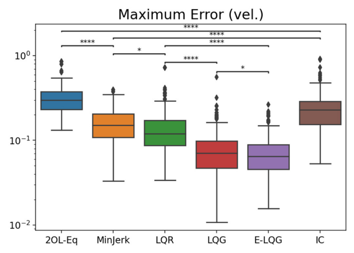
(b) SSE - Velocity



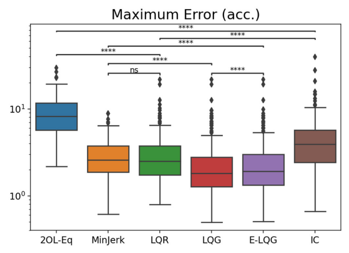
(c) SSE - Acceleration



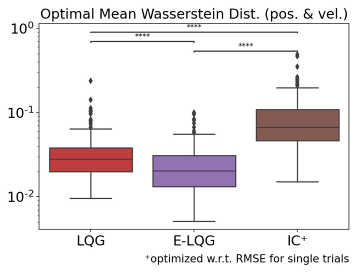
(d) Maximum Error - Position



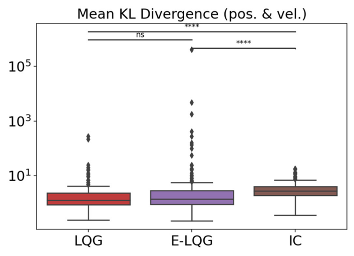
(e) Maximum Error - Velocity



(f) Maximum Error - Acceleration



(g) MWD - Position & Velocity



(h) MKL - Position & Velocity

Fig. 21. (a)–(f): SSE and Maximum Error values of all considered models, regarding the mean position, velocity, and acceleration time series of all participants and all tasks. (g)–(h): Mean Wasserstein distance and mean KL divergence of LQG, E-LQG, and IC, using sequences of Gaussian distributions over end-effector position and velocity, for all participants and all tasks. The asterisks indicate whether the difference between two bars is significant according to Wilcoxon Signed Rank tests with Bonferroni corrections (*: $p \leq 0.05$, **: $p \leq 0.01$, ***: $p \leq 0.001$, ****: $p \leq 0.0001$; n.s.: $p > 0.05$). Note the logarithmic scale in each plot.

Maximum Error, albeit these models were optimized to resemble the entire variability of observed user behavior.

Figure 21 displays the quality of the fit for all six models on a logarithmic scale.

Kolmogorov–Smirnov tests showed that the distributions of positional SSE for all six models do not fit the assumption of normality (all values $p < 0.0001$; the same holds for all other considered measures). Thus, we carried out a Friedman Test (i.e., a non-parametric test equivalent to a repeated measures one-way ANOVA), using Bonferroni corrections. The main factor included in the analysis was which model was used: 2OL-Eq, MinJerk, LQR, LQG, E-LQG, and IC. The test indicated that the SSE between the six models was significantly different ($\chi^2(2) = 638, p < 0.0001$, $n = 192$).

Additional Wilcoxon Signed Rank tests with Bonferroni corrections showed that the positional SSE of the E-LQG is significantly smaller than that of both LQG and LQR (E-LQG vs. LQG: $Z = -4.9$; E-LQG vs. LQR: $Z = -5.4$; $p < 0.0001$), while there were no significant differences between the latter two. However, the positional SSE of both LQG and LQR is significantly lower when compared to the MinJerk model (LQG vs. MinJerk: $Z = -9.9$; LQR vs. MinJerk: $Z = -8.4$; $p < 0.0001$), which, in turn, shows smaller values than 2OL-Eq ($Z = -9, p < 0.0001$). The fit of the mean IC trajectories with respect to positional SSE is comparable to 2OL-Eq (non-significant differences), with MinJerk ($Z = -9, p < 0.0001$) fitting the mean position profiles significantly better.

The findings are analogous for the maximum deviations of the simulated position time series from the data (Friedman Test, $\chi^2(2) = 685.4, p < 0.0001, n = 192$), with Wilcoxon Signed Rank tests showing that the E-LQG model approximates user trajectories significantly better than both the LQR ($Z = -6.9, p < 0.0001$) and the LQG ($Z = -5.4, p < 0.0001$) models. Moreover, they showed that the LQR model approximates user trajectories significantly better than MinJerk ($Z = -9.3$, $p < 0.0001$), and that MinJerk approximates user trajectories significantly better than both the 2OL-Eq ($Z = -10.5, p < 0.0001$) and IC ($Z = -8.9, p < 0.0001$) models.

Regarding velocity, a Friedman Test indicated that SSE ($\chi^2(2) = 794.4$, $p < 0.0001$, $n = 192$) and Maximum Error ($\chi^2(2) = 761.9$, $p < 0.0001$, $n = 192$) were significantly different between the six models, with Wilcoxon Signed Rank tests showing that E-LQG approximates velocities significantly better than LQG ($Z = -4.3, p < 0.001$ for SSE and $Z = -3.1, p < 0.05$ for Maximum Error), LQG better than LQR (SSE: $Z = -11.7$, Maximum Error: $Z = -11.3$; $p < 0.0001$), LQR comparable to MinJerk (LQR is slightly better in terms of Maximum Error, with $Z = -2.7$ and $p < 0.05$, while the differences in SSE are non-significant), MinJerk better than IC (SSE: $Z = -8.4$, Maximum Error: $Z = -7.7$; $p < 0.0001$), and IC better than 2OL-Eq (SSE: $Z = -9.7$, Maximum Error: $Z = -9.9$; $p < 0.0001$). The clear lead of LQG and E-LQG over LQR in this respect was to be expected, as LQG and E-LQG were optimized to minimize the Wasserstein distance with respect to *both* position and velocity.

Regarding acceleration, a Friedman Test indicated that SSE ($\chi^2(2) = 723.6, p < 0.0001, n = 192$) and Maximum Error ($\chi^2(2) = 539.5, p < 0.0001, n = 192$) were also significantly different between the six models, with Wilcoxon Signed Rank tests showing that acceleration is approximated significantly better by both LQG and E-LQG than by LQR or MinJerk (E-LQG vs. LQR – SSE: $Z = -12$, Maximum Error: $Z = -10$; LQG vs. LQR – SSE: $Z = -12$, Maximum Error: $Z = -11.6$; E-LQG vs. MinJerk – SSE: $Z = -8.4$, Maximum Error: $Z = -4.7$; LQG vs. MinJerk – SSE: $Z = -8$, Maximum Error: $Z = -4.7$; $p < 0.0001$), MinJerk and LQR show no significant differences in acceleration, both LQR and MinJerk approximate acceleration significantly better than IC (LQR vs. IC – SSE: $Z = -11.6$, Maximum Error: $Z = -11.2$; MinJerk vs. IC – SSE: $Z = -8$, Maximum Error: $Z = -8.9$; $p < 0.0001$), and IC approximates acceleration significantly better than 2OL-Eq (SSE: $Z = -8.1$, Maximum Error: $Z = -9.3$; $p < 0.0001$). In terms of SSE, no significant differences between LQG and E-LQG were found, whereas in terms of Maximum Error, LQG approximates user acceleration profiles significantly better than E-LQG ($Z = -7.1, p < 0.0001$).

In summary, the OFC model LQR achieves similar or better fits than both the dynamics model 2OL-Eq and the open-loop model MinJerk on all accounts, while being able to incorporate both control dynamics and objectives that are assumed to be optimized, given a specific interaction task. In addition, the duration of the initial ballistic movement (i.e., the surge), emerges naturally from the model and does not need be known in advance (in contrast to MinJerk). The mean trajectories predicted by the stochastic extensions LQG and E-LQG approximate average user behavior significantly better than the LQR model in terms of velocity and acceleration, which could also result from taking the velocity explicitly into account in the LQG/E-LQG parameter optimization. IC strategies on the other hand show a sub-optimal fit of mean user trajectories. A probable reason is that parameters were fitted to match particular user trajectories instead of trial-independent trajectory distributions [160].

Regarding the MWD and the MKL (see Figure 21(g)–(h)), both the LQG and the E-LQG trajectories show a significantly better approximation of user behavior than the IC trajectories according to Wilcoxon Signed Rank tests (LQG vs. IC – MWD: $Z = -12$, MKL: $Z = -7.9$; E-LQG vs. IC – MWD: $Z = -12$, MKL: $Z = -5.3$; $p < 0.0001$). Between the LQG model and its extension E-LQG, significant differences in favor of the E-LQG were only found in terms of MWD ($Z = -10.1$, $p < 0.0001$).

These results suggest that the LQG and its extension E-LQG capture both average user behavior and the specific variance profiles observed in position and velocity time series better than all other considered models.

12 DISCUSSION AND FUTURE WORK

In the following, we provide a discussion of the above results. We also discuss the applicability of the proposed framework to HCI tasks other than mouse pointing.

12.1 Deterministic Models

As shown in Section 5, the optimal damping ratio ζ of 2OL-Eq is always lower than 1 and exhibits a relatively small between-user variance. This implies that 2OL-Eq considers all regarded user movements as underdamped, which, to the best of our knowledge, has not yet been shown. While this could provide an indication that users rather tend to over— than undershoot for spatially-constrained 1D mouse movement, this is not consistent with our findings from all considered optimal control models. We thus conclude that the second-order dynamics of 2OL-Eq are not sufficient to capture the complex human behavior that is already apparent in 1D end-effector trajectories. In other words, interpreting the mouse cursor as a mass attached to one edge of the screen via a spring and a damper might suggest an underdamped system; however, it is the interpretation itself that seems to be inappropriate. This can also be seen from the left-skewed velocity profiles caused by (unrealistic) instantaneous peak acceleration, whereas typical user trajectories rather exhibit bell-shaped velocity profiles, as it is captured by the remaining models.

The user-specific values of the stiffness k in 2OL-Eq, which indicates how fast the end-effector is moved toward the target, are closely related to those identified for the surge duration parameter N_{MJ} from MinJerk. This indicates that different parameters in different models can play a similar role in explaining user behavior when fit through our parameter fitting process, even for dynamics-only (2OL-Eq) and kinematics-only (MinJerk) models.

The MinJerk model is able to predict typical velocity and acceleration profiles. However, in its standard form, it only covers the first ballistic movement and does not account for any corrective movements. Our proposed variant, which is constantly extended by its last position, thus faces a tradeoff between (i) fitting the perfectly bell-shaped velocity profile to that observed in user trajectories during the surge (which, however, is often truncated), and (ii) exhibiting a non-zero velocity during the subsequent correction phase. A tempting approach to resolve this issue is an iterative-submovement version of MinJerk, which is composed of the minimum jerk trajectories for a number of identified path segments [148]. Indeed, such a concatenation has been shown to account for the characteristics of typically observed trajectories, e.g., in case of handwriting [36], gesture typing [112], or arm movements [41, 80]. At its core, however, it requires the manual definition of when a submovement starts and terminates. Even more critically, the kinematic properties of the end-effector (i.e., the position, velocity, and acceleration) need to be known at the beginning and at the end of each path segment. Thus, several *via-points* need to be placed along the path, none of which can be inferred from the task description. This contradicts the minimum intervention principle [142], which suggests that only the "deviations" that interfere with task performance are being corrected. Moreover, several user studies have shown that there is no fixed via-point users aim at when being asked to repetitively navigate around a setup of obstacles [86]. In summary, such an extension might be appropriate to replicate kinematic characteristics of human movements to a certain extent. However, it cannot resolve the fundamental unsuitability of the MinJerk model for explaining how *and why* motion is generated, as the underlying movement dynamics are not modeled at all.

Instead, we propose to follow the argumentation of Liu and Todorov [86], which have empirically shown that (apart from the physical constraints induced by the environment, i.e., the Human–Computer System Dynamics in Figure 1) there are no internalized "hard" constraints users would comply *at any cost*, but only a (non-trivial) tradeoff between several objectives. or example, users asked to reach a small target *as accurate as possible* and *within a certain time limit*, which might be very difficult to achieve together, were shown to relax both requirements to some extent [86], rather than trying to keep exactly on schedule. This is consistent with the *cost combination hypothesis* [8], which argues that the flexibility in coordinating motor behavior is due to the optimization with respect to a *combined* objective function, e.g., including both accuracy and effort costs. An extension of these theories is the "reward is enough" hypothesis [124], which claims that different forms of intelligent behavior (e.g., learning, social intelligence, or generalization) can be directly deduced from maximizing an internalized reward function (which can be considered equivalent to minimizing a cost function). All of these findings support the idea that human movement arises as the result of an internal optimization, with objectives that can be directly deduced from the cognitive system and the task-specific instructions, and are thus fully compatible with the proposed optimal control framework for Human–Computer Interaction.

Combining the assumption of such a task-dependent optimal control with a simplified muscle model that yields overall fourth-order system dynamics, as we have done with LQR,[22] allows to replicate average mouse pointing trajectories both qualitatively and quantitatively. In particular, the LQR model yields a significantly better fit of average user trajectories than both 2OL-Eq and MinJerk (in terms of velocity and acceleration, LQR and MinJerk show comparable fits). However, we found it necessary to penalize both the remaining distance to target as well as its time derivatives (velocity and acceleration, which equals to applied force under the assumption of unit mass) in every time step. In contrast, the LQR model with state costs only applied at the final timestep does not replicate typical mouse pointing movements (for ID $\geq$ 4), as it predicts a smooth movement toward the target with symmetric, bell-shaped velocity profile and thus cannot account for the large correction phase typically observed toward the end of the movement (see Figure 10).

12.2 Stochastic Models

The stochastic extension of the LQR model—the LQG model—is naturally capable of modeling and explaining between-trial variability observed from experimental data. It includes both signal-dependent control noise and constant observation noise. In contrast to the LQR model, the state cost does not need to incur in every timestep, but only at the final time. We also introduced an extension of the LQG model, called E-LQG, which incorporates both visual input and observations of the eye fixation point, assuming accurate saccades between the initial and the target position, thus being more plausible from a visuomotor perspective.

[22]Note that the LQR model (as well as all other presented models) depends on the task under consideration, since the target position needs to be included in state space, and the several objectives (accuracy, stability, and effort costs) were derived from the task description.

Both mean and variance of typical mouse trajectories can be replicated fairly well by the proposed LQG and E-LQG models. In terms of SSE, E-LQG approximates mean user trajectories as good as or significantly better than all other models, including LQR. In terms of the mean Wasserstein distance, E-LQG shows significant improvements over LQG, suggesting that the extended observation model indeed captures characteristic end-effector variance profiles of mouse pointing tasks considerably better than the saccade-free observation model proposed by Todorov [141].

For the considered 1D reciprocal mouse pointing movements, we found that the assigned noise levels considerably vary between individual participants, with some of the variability better explained by large signal-dependent control noise and some better explained by higher observation noise. In contrast, the identified optimal cost weights depend on task ID rather than the user. We hypothesize that the tradeoff between end-point accuracy (determined by a large distance weight) and stability (determined by large velocity and force weights) explicitly given by the experimental instructions was interpreted differently for each task condition, leading to different optimal control strategies depending on the task difficulty.

The good fit of the LQG and the E-LQG models in terms of variance is partially attributed to relatively high signal-dependent noise levels. For the LQG model, the average optimal value of σ_u amounts to 2.52, that is, an (unbiased) deviation from the desired control value u_n with a magnitude of 252% of this value can be expected at each timestep $1n \in \{0, \ldots, N\}$. For the E-LQG model, the fitted signal-dependent control noise levels are reduced by approximately 50% compared to the LQG model, i.e., more of the variability that is observed from user data can be explained by the extended observation model instead of attributing it to large control noise. The control noise levels are still relatively high (1.18 on average, i.e., the desired control and the effectively applied control differ by 118% of the desired control value on average).

In contrast, the literature suggests signal-dependent control noise levels between 10% and 25%, based on several empirical findings [26, 58, 146]. We believe that this mismatch does not render the LQG inappropriate in explaining human movements during interaction on a continuous level. Instead, the large amount of variability observed in the Pointing Dynamics Dataset can at least partially be attributed to *temporal noise*, i.e., different cognitive reaction and/or motor activation times between individual trials. This is consistent with previous findings, which suggest that both signal-dependent and temporal noise (as well as constant noise) are required to explain the characteristic variance profiles observed in two-dimensional goal-directed arm movements [146]. In this article, we have cut off the reaction times of each trial as accurately as possible. However, it is difficult to reliably identify reaction times using properties of the (one-dimensional) end-effector trajectories only. Moreover, the reciprocal nature of the considered bi-directional movements further enhances the variance of the movement onset times, as these become susceptible to learning effects as well as fatigue and a temporary lack of attention (among others), whose overall effect is unclear. These effects are not accounted for in the assumption that the internal forward model is identical to the actual Human–Computer System Dynamics. In particular, when using more complex dynamics, it might be too restrictive. Using an inaccurate internal model might act as an additional source of variability (often referred to as *dynamic uncertainty* [12]). In the future, it will be interesting to consider different internal models, extend the LQG model by temporal delays of uncertain length (preliminary attempts to include *fixed* reaction times can, e.g., be found in [138]), and to combine it with RL-based methods that allow to model optimal learning behavior [12].

Another point that requires a more thorough discussion is the choice of parameters that are included in the parameter fitting process. For each considered model, we decided to optimize all parameters that we suspected to differ between users or task conditions, and which could not be directly inferred from literature (in contrast to, e.g., the time constants in LQR, LQG, and E-LQG). However, for some parameters (e.g., the force perception noise level σ_f) the identified values for each participant span several magnitudes, suggesting rather minor effects on the resulting movement trajectories. In this case, it might be reasonable to conduct a second parameter identification, where these parameters are set constant. Similarly, parameters corresponding to task-independent user strategies or inherent body characteristics could be identified once per user instead of differentiating between task conditions.

It is also important to note that the fit of a given model can be expected to improve with the number of optimized parameters (i.e., degrees of freedom of the model). This implies a tradeoff between model simplicity (low number of interpretable parameters) and goodness of fit (ability to "explain" observed data), which can be captured by several measures, including the *Akaike information criterion* and the *Bayesian information criterion* [126]. However, these criteria are not directly applicable to our model comparison, since the considered models not only differ in terms of parameters, but most importantly with respect to their scope, which is also reflected in the used reward function, system dynamics, and observation model. For example, LQG can account for movement variability even with fixed (non-zero) noise levels, while LQR naturally cannot. Instead of comparing the proposed models only based on a single quantitative value, their scope and the phenomena that can be predicted should also be taken into account (also see the three-stage process proposed at the end of Section 12.4).

Compared to a recently proposed IC [160], simulation trajectories of the LQG and the E-LQG models exhibit a considerably better fit in terms of all considered metrics (SSE, Maximum Error, MWD, and MKL). Qualitatively, both OFC models predict the positional variance at the beginning of the movement and during the correction phase, as well as the velocity variance profiles, more accurate than the IC. However, it is important to note that the IC simulations are based on a multiple-model approach, i.e., an individual parameter vector is identified for every single trial. In particular, the variability of the IC only results from random sampling from this set of identified parameter vectors during run-time. In contrast, the LQG/E-LQG parameters were explicitly fitted to match the user position and velocity *distributions*, incorporating all trials of a given user, direction, and task condition. In other words, the variability of LQG/E-LQG is intrinsic to the considered stochastic OCP. While the between-trial variability of IC has shown to match the trajectory variance of the Pointing Dynamics Dataset relatively well in terms of phase plane densities [160]; it seems that the time sequences of state distributions are replicated considerably better by both the LQG and the E-LQG models. This suggests that a well-defined model of the sources of variability, such as the signal-dependent control noise and observation noise in the (E-)LQG models, is necessary to replicate the characteristic development of position and velocity variance over time.

A promising next step would thus be to develop an IC model that includes both signal-dependent control noise and observation noise, and investigate its ability to account for the characteristic variance profiles observed for mouse pointing. A rigorous analysis of the individual components of the new IC model, as we have done for several OFC models in this article, would allow to examine whether a combination of OFC and IC theory (i.e., continuously perceived observations, motor and observation noise, and intermittency of control) may account for typical phenomena such as reaction times, bell-shaped velocity profiles, and characteristic variance profiles.

One major limitation of the presented OFC models that has not yet been discussed is the need to determine the total movement duration in advance. While this is true for most optimal motor control models, including the minimum torque-change [144] and the minimum end-point variance [58] models, a few attempts have been made toward a model that predicts movement time instead of requiring it. These approaches include the open-loop constrained minimum-time model [131], which can be solved analytically in case of linear dynamics, Markov decision processes using state- and action-space discretizations and being solved via dynamic programming [86], as well as models that explicitly assign an optimal "cost of time", either based on prior assumptions on the human sensorimotor system [122] or computed via an inverse optimal control approach [10].

Another promising framework is *infinite-horizon OFC* [108, 111], which is based on the same assumptions as the finite-horizon framework presented in this article, the main difference being that the total movement duration does not need to be specified in advance. Instead, the controller is assumed to apply the optimal, time-independent steady-state strategy for *both* the transient movement and the subsequent posture maintenance phase (the so-called *steady-state-control hypothesis*). While infinite-horizon OFC constitutes an interesting alternative that allows for a more in-depth analysis of the speed-accuracy tradeoff using Fitts' Law type studies, it might be inappropriate for tasks without a clear posture maintenance phase, or where it is unclear whether the controller should consider such a phase during planning, e.g., in fast, repetitive movements as those from the dataset considered in this article [111]. Its applicability to HCI thus needs to be explored in future work, possibly using a similar approach as in this work.

12.3 Application to Other HCI Tasks

In this article, we analyzed the applicability of optimal feedback control models to 1D pointing tasks. In this section, we discuss how these models can be applied to other common tasks in HCI, to highlight the generalizability and limitations of these models. The main limitations of the LQR/LQG approach are their restriction to linear dynamical systems and quadratic costs. If one of these properties does not hold for a particular task, nonlinear OFC approaches need to be applied.

While the extension of LQR/LQG to via-point tasks[23] with fixed passage times is straightforward (see [138]), tasks where only the order of targets is specified, but not the specific times they are reached, cannot be directly covered by the proposed models. This is mainly due to the assumption of quadratic state costs, which, in combination with linear dynamics, does not allow to penalize the distance to the next via-point depending on the already reached via-points. One possible way to model via-point tasks with free timing is to integrate the LQR/LQG models into an outer optimization loop (similar to the parameter fitting process introduced in this article), which, e.g., identifies the minimum passage times such that every via-point is reached [138].

Following *moving targets* is commonly called pursuit tracking. Moving targets occur, e.g., in computer games. If the movement of the target can be modeled by a linear differential equation (which includes straight lines, curves, and ellipses), including moving targets in the LQR/LQG models is straightforward. Simply add the state of the target(s) (e.g., position) to the state space, and extend the system dynamics matrix A to model the dynamics of the target movement. The target dynamics can even depend on the end-effector trajectory. For example, it is possible to model a target that tries to evade the pointer. The main restriction is imposed by the linear dynamics, requiring that the target position also evolves linearly. Pre-defined target trajectories that cannot be described by a linear differential equation are more difficult to implement (in fact, they either need to be "hard-coded", i.e., each state needs to be augmented by the complete discrete-time sequence of target positions, which significantly increases the computational effort due to the curse of dimensionality, or approximated by linear dynamics).

Path following, or tracing or drawing tasks, are considerably harder to model, as they usually impose spatial-only constraints and leave the temporal profile, i.e., the movement kinematics, up

[23]In *via-point tasks*, multiple targets (the via-points) need to be reached in a pre-defined order. Usually, no timing of when each via-point needs to be reached is prescribed. Via-point tasks can be used to model pointing to several targets in a row. They have been used to model handwriting or drawing, where the via-points are chosen such that a certain letter or shape is created.

to the user. In particular, tracing can be considered the limiting case of via-point tasks (with non-determined passage times) with the distance between two via-points approaching 0. For this reason, the same issues as for via-point tasks occur. More precisely, since the cost matrices need to be specified before movement onset, the timesteps at which the end-effector should reach the desired path/via-point positions need to be known in advance. Thus, it is currently unclear whether and how LQG could be used to model path following.

The same holds for *steering tasks* (i.e., tasks with *constrained motion*), where the end-effector needs to be moved from an initial position to an end-point as quickly as possible, while keeping it inside a tunnel of possibly varying width. The most prominent examples include command selection via a hierarchical drop-down menu, parameter sliders, and scroll-bars [1, 2, 158]. While in the LQR/LQG models, a composite cost function that penalizes both the distance to target and the distance to the center of the tunnel perpendicular to the movement direction would create an incentive to move the end-effector toward the target while keeping it inside the tunnel, this intuitive approach has two major limitations.

First, the boundary constraints are implemented "softly" in the sense that the costs for being shortly outside the tunnel are only infinitesimally larger than the costs for being shortly inside. This follows directly from the LQR/LQG assumption of costs that are quadratic in the system state and thus necessarily continuous.[24]

Second, to account for the tunnel constraint, the quadratic costs require some reference position that exhibits minimum costs along the direction perpendicular to the movement direction. The most obvious choice for this minimum would be the center of the tunnel, as this corresponds to an unbiased penalization of deviations in either direction. However, empirical user studies suggest that users do not necessarily aim to follow the central path within the tunnel [6, 102]. Instead, they deliberately make use of the respective tunnel widths by adjusting their movements, e.g., to achieve higher speeds by "cutting of the corner" [107]. Regardless of the specific reference trajectory, the usage of costs that penalize the distance to *any* fixed movement trajectory that is not explicitly apparent from the task description (as it is the case for steering tasks) contradicts the *minimum intervention principle* [142], which suggests that only task-relevant deviations are being corrected.

The first issue does not necessarily impose a severe restriction to modeling plausible user behavior, since users also tend to associate a certain internal cost to fulfilling the boundary constraints, i.e., they consider staying inside the tunnel boundaries as one goal among many, rather than viewing it as an inevitable "hard" constraint [86]. The second issue, however, might constitute a serious limitation and possibly prevent a reasonable application of LQR/LQG to constrained movement tasks.

Regarding *free-hand inking* tasks (e.g., to write a certain word or draw a specific sketch), it is not clear how an appropriate cost function that includes all relevant information from the task description should look like. In addition, capturing high-level characteristics such as user-specific stroke styles or connections between individual characters might be difficult to model. However, the case of gesture-based keyboard typing has recently been successfully modeled as a via-point task with minimum jerk trajectories between two subsequent via-points [112]. It is important to keep in mind that the scope of the proposed methods clearly is to model pointing movements, while more creative tasks would require some high-level cognition process that instantiates and coordinates multiple subprocesses [156]. While we do not want to rule out the possibility that LQG can be adapted to the modeling of handwriting or drawing, further research in this regard is certainly needed.

[24]For the same reason, in the considered pointing task, it is not possible to only apply costs when being outside the target, as such costs would necessarily be discontinuous at the target boundary.

Note that each of the tasks discussed above can be solved using several input methods, e.g., mouse-, pen-, or touch-based input. Accurate modeling of the complete interaction loop, as depicted in Figure 1, thus requires to take into account device-specific properties such as a pointing transfer function or internal dynamics [22, 23]. Similarly, the Human Body Dynamics can be modeled with arbitrary granularity. For example, the fourth-order dynamics with simplified muscle activations used in this article could be replaced by complex (non-linear) biomechanical models, e.g., those implemented in state-of-the-art physics engines such as MuJoCo or OpenSim [31, 38, 59].

In general, non-linearities in the body dynamics, input devices, or interface dynamics cannot be modeled accurately using LQR/LQG. However, as long as the movements are not too big, a small signals approach can be applied and a linear approximation around an operating point can be found. To take several operating points into account, it is possible to iteratively linearize non-linear dynamics [83, 84, 143]. Further investigation into the suitability of this approach for the different dynamics of HCI is definitely needed and constitutes a promising direction for future research.

Finally, the above discussed limitations regarding the applicability to general HCI tasks only refer to the linear finite-horizon LQR/LQG case. The infinite-horizon LQR/LQG formulation [111] is less suitable for many HCI tasks, as it does not allow to take into account multiple, time-dependent objectives during optimization, which, e.g., is inevitable for via-point tasks that need to be reached in a given order, or moving targets. However, the general class of optimal control models of HCI, as discussed in Section 3, is much larger and consists of a variety of modeling approaches and solution methods, including Direct and Indirect Collocation [11], Model-Free and Model-Based RL [56, 130], (Semi-)Supervised Learning [100, 115], Model-Predictive Control [20], and mixtures of these [9, 12, 81], each of which has its own requirements on the problem, advantages, and disadvantages. While some HCI tasks might be too complex to solve using the linear methods presented in this article, the general OFC framework offers exciting opportunities to model, simulate, explore, and eventually improve the interaction between humans and computers, using a mathematically profound description.

12.4 Practical Benefits and Advice for HCI Researchers

Building on the above discussion on the applicability and generalization in the context of HCI, we clarify the concrete benefits of our proposed framework and methods to HCI researchers, using the example of the so-called *Bubble Lens*.

Previously, the *Bubble Lens* method [95] has been proposed as one of the few target acquisition techniques that explicitly takes into account kinematic movement profiles. The main idea of this method is to automatically magnify the desired target area as soon as the first corrective submovement has been detected ("kinematic triggering"). While this technique has shown to significantly outperform the standard *Bubble Cursor* [52] (the fastest pointing method at this time), the authors did not account for the fact that users might adapt their behavior once the magnification has been observed. Moreover, the criteria of when to trigger the magnification have been chosen manually, based on effectiveness and practicability. Using our proposed optimal control framework of interaction, it would be possible to analyze the effects of temporary magnification on visual input, internal estimates, predictions and subgoals, and the resulting movement trajectory (including ergonomic quantities such as muscle energy consumption or fatigue). This allows to gain a deeper understanding of *why* this technique outperforms existing methods. Finally, our unifying framework can be used to optimize the technique's remaining parameters (e.g., trigger time and duration of the lens, or visual properties such as smooth transitions), consider specific body characteristics of individual user groups, and perform simulation-based comparisons with existing methods, thus considerably improving comparability between different approaches.

As a general advice for HCI researchers, we recommend the following procedure when using our proposed optimal control framework of HCI:

(1) Make a **preliminary selection** of model(s) based on the phenomena of interest.
Not every model is suitable for every purpose. For example, the deterministic models 2OL-Eq, MinJerk, and LQR can only predict (optimal) average movement behavior, while the stochastic models LQG, E-LQG, and IC predict entire trajectory distributions. Closed-loop models can explain how humans respond to unexpected perturbations during movement. If modeling muscle activations and fatigue is of interest, then torque- or muscle-driven models of the human arm and hand are much more appropriate than kinematic models such as 2OL-Eq and MinJerk. Finally, the extended observation model included in E-LQG provides an opportunity to analyze gaze using the saccade timestep parameter.

(2) Select a **specific model** based on qualitative and quantitative criteria.
The used metrics and decision criteria crucially depend on the overarching goal of the analysis. For example, replication of user data requires a more quantitative evaluation based on some metric that incorporates all relevant aspects of the observed trajectories, whereas a comparison of models regarding their explanatory and predictive power should rather be based on qualitative results, e.g., whether well-established phenomena such as bell-shaped velocity profiles, corrective submovements, or specific eye-movement coordination patterns can be inferred. In either case, the evaluation and visualization tools from our OFC4HCI toolbox may be helpful.
Note that this stage requires choosing one (or multiple) reasonable parameter sets for each model to be compared, e.g., from the literature. Alternatively, the model parameters that "best explain" observed user data can be identified within an (outer) optimization loop, using the method presented in Section 4.2. In general, we suggest to include all parameters that are suspected to differ between independent variables (e.g., the user ID or the task condition) or which cannot be reasonably inferred from literature in the parameter fitting process.

(3) (Optional:) Fine-tune the specific **model parameters**.
If the analysis of parameters from stage 2 suggests that some parameters could be set constant, as they do not depend considerably on the user ID or task condition, another iteration of the parameter identification process could be performed. In this case, information criteria such as the *Akaike information criterion* [126], which account for the tradeoff between the goodness of fit and the simplicity of a model, might be considered.

13 CONCLUSION

In this article, we have provided an introduction to the concepts of OFC for an HCI audience and have presented a generic parameter fitting process that can be used to identify system and strategy parameters of any given control model. Using the example of mouse pointing, we have shown that both a non-trivial dynamic model of the HCI loop, which includes signal-dependent control noise, and continuously perceived noisy feedback are necessary to explain user behavior both qualitatively and quantitatively. These optimal control models show a significantly better fit to the considered user trajectories than pure dynamics models such as 2OL-Eq or pure kinematic models such as MinJerk.

The *optimal control framework for HCI* that we have proposed is *versatile*, as it can be used to model interaction with different interfaces using various input devices, and *comprehensive*, as it allows to model the complete interaction loop, including body, input device, and interface dynamics, as well as feedback properties, each depending on the task and/or the user under consideration. While the basic assumptions of LQG (linear dynamics, quadratic costs, and Gaussian noise) are relatively restrictive, we have shown its ability to replicate typical mouse movements both in terms of average behavior and between-trial variability. More importantly, we have demonstrated how the proposed framework can be used to identify characteristic differences in movement behavior between participants or task conditions. The degree to which these differences can be interpreted naturally depends on the model complexity. In particular, aggregated Human–Computer System Dynamics as used in the presented case of one-dimensional mouse pointing do not allow to simulate motion of the human body per se, but only predict movement in end-effector space (i.e., mouse cursor trajectories). For an in-depth analysis of the intrinsic characteristics and strategies of the human biomechanical and cognitive system, independent of the used interaction technique, more detailed and separate submodels of both the Human Body Dynamics and the Interface Dynamics would be required. We have also discussed the applicability of the framework to several other HCI tasks, as well as possible extensions (e.g., regarding non-linear body and interface dynamics) that remain as future work. As a more general advice for HCI researchers, we recommend first making a preliminary selection of models based on the phenomena of interest, then selecting a specific model based on qualitative and quantitative criteria, and finally fine-tuning the model parameters.

We hope that this article, along with our OFC4HCI toolbox, provides an easy-to-understand overview of how recent methods and concepts from optimal control theory can be applied to HCI using the example of mouse pointing, and encourages HCI researchers to use them in their own studies and simulations. OFC provides a concise and mathematically exact explanation of movement in interaction with computers that we hope will be useful not only for HCI research, but also for teaching HCI and ultimately for interface design.

APPENDICES

A LQR EQUATIONS

The proposed LQR model can be described as the discrete-time linear-quadratic OCP with finite horizon $N \in \mathbb{N}$

$$\begin{aligned} &\textit{Minimize} \quad J_N^{(\mathrm{LQR})}(x,u) = \sum_{n=0}^{N} x_n^\top Q_n x_n + \sum_{n=0}^{N-1} u_n^\top R_n u_n, \\ &\qquad \textit{with respect to } u = (u_n)_{n\in\{0,\ldots,N-1\}} \subset \mathbb{R}, \end{aligned} \tag{35a}$$

where $x = (x_n)_{n\in\{0,\ldots,N\}} \subset \mathbb{R}^5$ with $x_n = (p_n, v_n, f_n, g_n, T)^\top$ satisfies

$$\begin{aligned} x_{n+1} &= Ax_n + Bu_n, \quad n \in \{0,\ldots,N-1\}, \\ x_0 &= \bar{x}_0, \end{aligned} \tag{35b}$$

given some $\bar{x}_0 \in \mathbb{R}^5$, with sampling time $h > 0$ and system dynamics matrices

$$A = \begin{pmatrix} 1 & h & 0 & 0 & 0 \\ 0 & 1 & h & 0 & 0 \\ 0 & 0 & 1-\frac{h}{\tau_2} & \frac{h}{\tau_2} & 0 \\ 0 & 0 & 0 & 1-\frac{h}{\tau_1} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{h}{\tau_1} \\ 0 \end{pmatrix}, \tag{35c}$$

corresponding to the combination of a simplified second-order muscle model with time constants $\tau_1, \tau_2 > 0$ and a double integrator.

The state cost matrices are defined by

$$Q_n = \begin{pmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & \omega_v & 0 & 0 & 0 \\ 0 & 0 & \omega_f & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{5\times 5}, \quad n \in \{0, \ldots, N\}, \tag{36}$$

which implies

$$x_n^\top Q_n x_n = D_n^2 + \omega_v v_n^2 + \omega_f f_n^2, \tag{37}$$

i.e., the distance $D_n = |T - p_n|$ between mouse and target position as well as the end-effector velocity v_n and force f_n are quadratically penalized at every timestep $n \in \{0, \ldots, N\}$. In our case of one-dimensional pointing tasks, the control cost matrices are scalar and given by

$$R_n = \frac{\omega_r}{N-1} \in \mathbb{R}, \quad \omega_r > 0, \quad n \in \{0, \ldots, N-1\}, \tag{38}$$

which yields the quadratic cost terms

$$u_n^\top R_n u_n = \frac{\omega_r}{N-1} u_n^2. \tag{39}$$

It can be shown that the unique solution $u^* = (u_n^*)_{n\in\{0,\ldots,N-1\}}$ to the optimization problem (35) is given by

$$\begin{aligned} u_n^* &= \pi(x_n) = -L_n x_n, \quad n \in \{0, \ldots, N-1\}, \\ L_n &= (R_n + B^\top \mathcal{S}_{n+1} B)^{-1} B^\top \mathcal{S}_{n+1} A, \\ &\qquad n \in \{0, \ldots, N-1\}, \end{aligned} \tag{40}$$

where the symmetric matrices $\mathcal{S}_n \in \mathbb{R}^{5\times 5}$ can be determined by solving the **Discrete Riccati Equations**

$$\mathcal{S}_n = Q_n + A^\top \mathcal{S}_{n+1} A - A^\top \mathcal{S}_{n+1} B (R_n + B^\top \mathcal{S}_{n+1} B)^{-1} B^\top \mathcal{S}_{n+1} A, \tag{41a}$$

for $n \in \{0, \ldots, N-1\}$ backward in time with initial value

$$\mathcal{S}_N = Q_N. \tag{41b}$$

B SUPPLEMENTARY MATERIAL

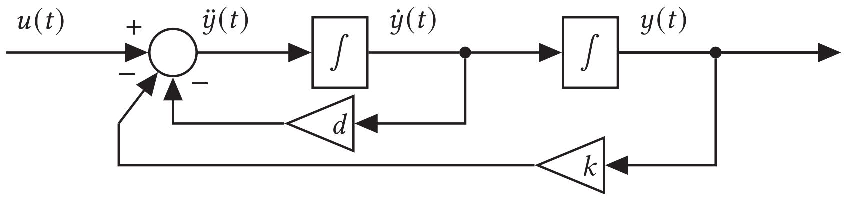


Fig. B.1. Control-flow diagram of the second-order lag (2OL).

Table B.1. Boundaries of all Model Parameters Used for Parameter fitting

Model	Parameter	Minimum	Maximum	Type
2OL-Eq	k	0	500	Continuous
	d	0	500	Continuous
MinJerk	N_{MJ}	0	N	Continuous (Relaxed)
LQR	ω_r	2e-9	20	Continuous
	ω_v	0	10e-2	Continuous
	ω_f	0	10e-4	Continuous
LQG / E-LQG	ω_r	4e-18	7e-3	Continuous
	ω_v	0	10	Continuous
	ω_f	0	10	Continuous
LQG	σ_u	10e-10	5	Continuous
	σ_s	0	5	Continuous
E-LQG	σ_u	10e-10	5	Continuous
	σ_v	0	10	Continuous
	σ_f	0	50	Continuous
	σ_e	0	5	Continuous
	γ	4e-18	100	Continuous
	n_s	0	N	Continuous (Relaxed)

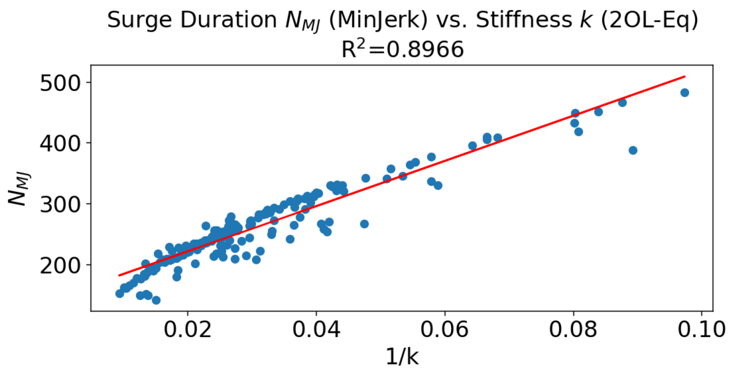


Fig. B.2. The relationship between the inverse of the stiffness parameter k in 2OL-Eq and the surge duration parameter N_{MJ} in MinJerk is captured well by a linear function.

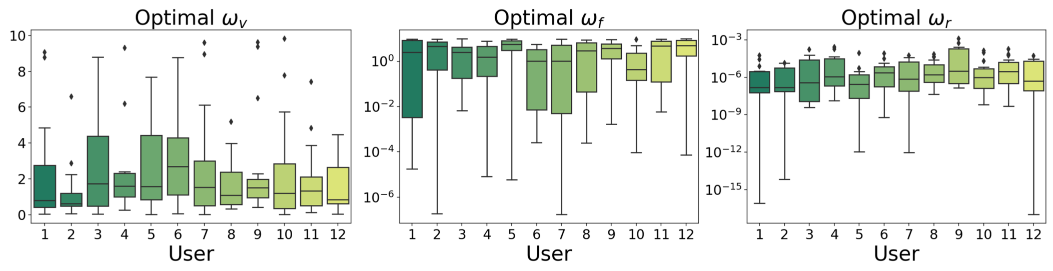


Fig. B.3. Cost weight parameters of the LQG, optimized for the trajectory sets of all participants, tasks, and directions, grouped by participants. Note that the optimal values of ω_f and ω_r are plotted on a logarithmic scale.

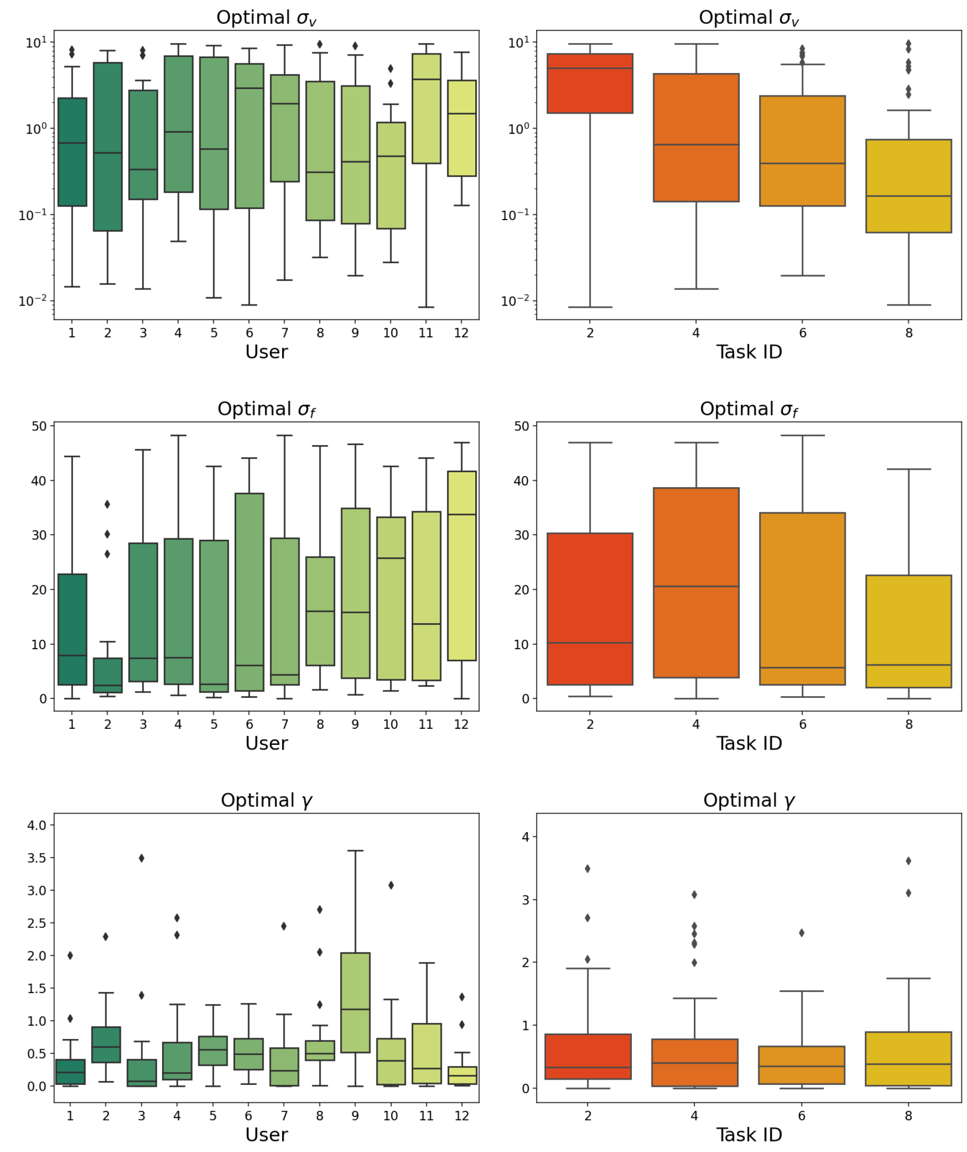


Fig. B.4. Visual observation noise parameters of the E-LQG, optimized for the mean trajectories of all participants, tasks, and directions, grouped by participants (left) and by ID (right). For better visibility, both plots for the position perception noise weight γ do not include the 2 largest outliers with values 19.7 and 55.4. Note that the optimal values of σ_v are plotted on a logarithmic scale.

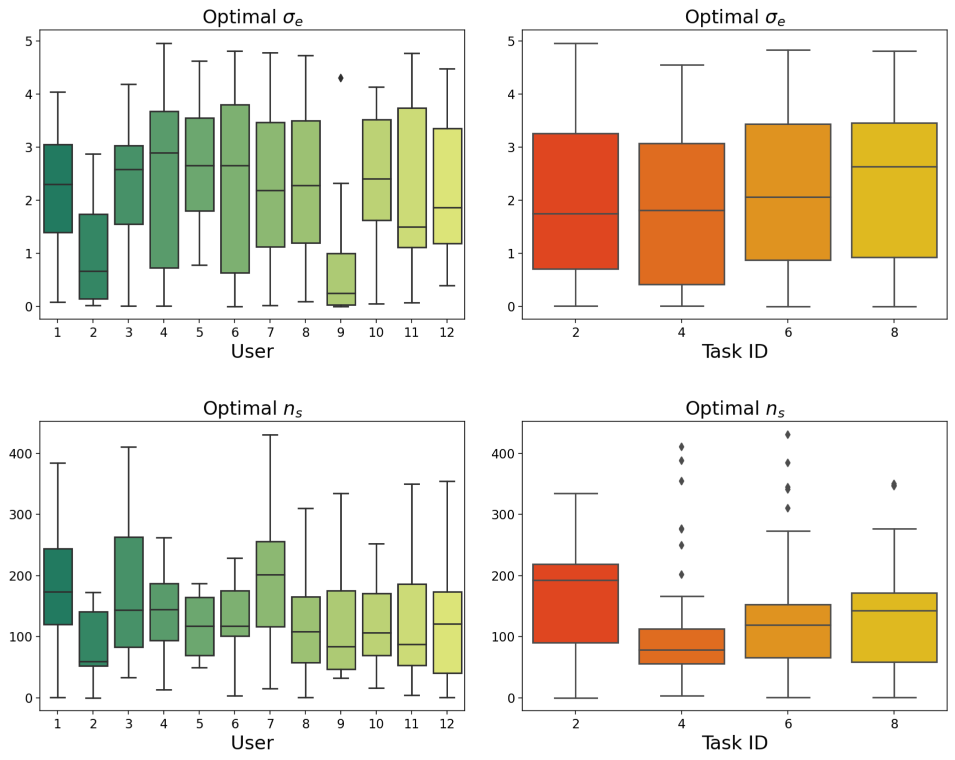


Fig. B.5. The gaze noise level σ_e and the saccade timestep n_s of the E-LQG, optimized for the mean trajectories of all participants, tasks, and directions, grouped by participants (left) and by ID (right).

ACKNOWLEDGMENTS

We would like to thank Lars Grüne for his very helpful advice and comments on a preliminary version of this article.

REFERENCES

[1] Johnny Accot and S. Zhai. 1997. Beyond fitts' law: Models for trajectory-based HCI tasks. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems.* 295–301. DOI: https://doi.org/10.1145/258549.258760

[2] Johnny Accot and Shumin Zhai. 1999. Performance evaluation of input devices in trajectory-based tasks: An application of the steering law. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (1999), 466–472. DOI: https://doi.org/10.1145/302979.303133

[3] Zvi Artstein. 1980. Discrete and continuous bang-bang and facial spaces or: Look for the extreme points. *Siam Review* 22, 2 (1980), 172–185.

[4] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. 2005. Predictive interaction using the delphian desktop. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology.* ACM, New York, NY, 133–141.

[5] Anne Auger, Nikolaus Hansen, J. M. Perez Zerpa, Raymond Ros, and Marc Schoenauer. 2009. Experimental comparisons of derivative free optimization algorithms. In *Proceedings of the International Symposium on Experimental Algorithms.* Springer, 3–15.

[6] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2016. Visual menu techniques. *ACM Computing Surveys* 49, 4, Article 60 (Dec. 2016), 41 pages. DOI: https://doi.org/10.1145/3002171

[7] Michel Beaudouin-Lafon. 2004. Designing interaction, not interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces.* Association for Computing Machinery, New York, NY, 15–22. DOI: https://doi.org/10.1145/989863.989865

[8] Bastien Berret, Enrico Chiovetto, Francesco Nori, and Thierry Pozzo. 2011. Evidence for composite cost functions in arm movement planning: An inverse optimal control approach. *PLOS Computational Biology* 7, 10 (10 2011), 1–18. DOI: https://doi.org/10.1371/journal.pcbi.1002183

[9] Bastien Berret, Adrien Conessa, Nicolas Schweighofer, and Etienne Burdet. 2021. Stochastic optimal feedforward-feedback control determines timing and variability of arm movements with or without vision. *PLOS Computational Biology* 17, 6 (06 2021), 1–24. DOI: https://doi.org/10.1371/journal.pcbi.1009047

[10] Bastien Berret and Frédéric Jean. 2016. Why Don't we move slower? The value of time in the neural control of action. *Journal of Neuroscience* 36, 4 (2016), 1056–1070. DOI: https://doi.org/10.1523/JNEUROSCI.1921-15.2016

[11] John T. Betts. 2010. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM.

[12] Tao Bian, Daniel Wolpert, and Zhong-Ping Jiang. 2020. Model-free robust optimal feedback mechanisms of biological motor control. *Neural Computation* 32, 3 (01 2020), 562–595. DOI: https://doi.org/10.1162/neco_a_01260

[13] Emilio Bizzi, Neville Hogan, F. Mussa-Ivaldi, and Simon Giszter. 1992. Does the nervous system use equilibrium point control to guide single and multiple joint movements? Behav Brain Sci. *The Behavioral and Brain Sciences* 15, 4 (12 1992), 603–13. DOI: https://doi.org/10.1017/S0140525X00072538

[14] Åke Björck. 1996. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics. DOI: https://doi.org/10.1137/1.9781611971484

[15] Hans Georg Bock and Karl-Josef Plitt. 1984. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes* 17, 2 (1984), 1603–1608.

[16] Michael Bohan, Shelby G. Thompson, and Peter J. Samuelson. 2003. Kinematic analysis of mouse cursor positioning as a function of movement scale and joint set. In *Proceedings of the International Conference on Industrial Engineering–Theory, Applications and Practice*. Wichita State University Wichita, KS, 442–447.

[17] Reinoud J. Bootsma, Laure Fernandez, and Denis Mottet. 2004. Behind Fitts' law: Kinematic patterns in goal-directed movements. *International Journal of Human-Computer Studies* 61, 6 (2004), 811–821.

[18] Daniel Bullock and Stephen Grossberg. 1988. Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review* 95, 1 (02 1988), 49–90. DOI: https://doi.org/10.1037//0033-295X.95.1.49

[19] Ronald M. Baecker and William A. S. Buxton. 1987. Readings in human-computer interaction: A multidisciplinary approach. M. Kaufmann.

[20] Eduardo F. Camacho and Carlos Bordons Alba. 2013. *Model Predictive Control*. Springer Science & Business Media.

[21] Stuart K. Card, William K. English, and Betty J. Burr. 1978. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics* 21, 8 (1978), 601–613. DOI: https://doi.org/10.1080/00140137808931762

[22] Géry Casiez and Nicolas Roussel. 2011. No more bricolage!: Methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 603–614. DOI: https://doi.org/10.1145/2047196.2047276

[23] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. 2008. The impact of control-display gain on user performance in pointing tasks. *Human–Computer Interaction* 23, 3 (2008), 215–250.

[24] Olivier Chapuis, Jean-Baptiste Labrune, and Emmanuel Pietriga. 2009. DynaSpot: Speed-dependent area cursor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1391–1400.

[25] Noshaba Cheema, Laura A. Frey-Law, Kourosh Naderi, Jaakko Lehtinen, Philipp Slusallek, and Perttu Hämäläinen. 2020. Predicting mid-air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, 1–13. DOI: https://doi.org/10.1145/3313831.3376701

[26] H. Peter Clamann. 1969. Statistical analysis of motor unit firing patterns in a human skeletal muscle. *Biophysical journal* 9, 10 (1969), 1233–1251.

[27] H. Cooper, P. Camic, D. Long, A. Panter, D. Rindskopf, and K. Sher. 2012. APA handbook of research methods in psychology, Vol 1: Foundations, planning, measures, and psychometrics.

[28] Frederic Crevecoeur, Tyler Cluff, and Stephen H. Scott. 2014. Computational approaches for goal-directed movement planning and execution. In *Proceedings of the Cognitive Neurosciences, 5th ed.*, Michael Gazzaninga and George Mangun (Eds.). MIT Press, Cambridge, MA, 461–477.

[29] E. R. F. W. Crossman and P. J. Goodeve. 1983. Feedback control of hand-movement and Fitts' law. *The Quarterly Journal of Experimental Psychology* 35, 2 (1983), 251–278.

[30] W. G. Darling, K. J. Cole, and J. H. Abbs. 1988. Kinematic variability of grasp movements as a function of practice and movement speed. *Experimental Brain Research* 73, 2 (1988), 225–235.

[31] Christopher L. Dembia, Nicholas A. Bianco, Antoine Falisse, Jennifer L. Hicks, and Scott L. Delp. 2021. OpenSim Moco: Musculoskeletal optimal control. *PLOS Computational Biology* 16, 12 (12 2021), 1–21. DOI: https://doi.org/10.1371/journal.pcbi.1008493

[32] Ashesh K. Dhawale, Maurice A. Smith, and Bence P. Ölveczky. 2017. The role of variability in motor learning. *Annual Review of Neuroscience* 40 (2017), 479–498.

[33] Jörn Diedrichsen, Reza Shadmehr, and Richard B. Ivry. 2010. The coordination of movement: Optimal feedback control and beyond. *Trends in Cognitive Sciences* 14, 1 (2010), 31–39. DOI: https://doi.org/10.1016/j.tics.2009.11.004

[34] Seungwon Do, Minsuk Chang, and Byungjoo Lee. 2021. *A Simulation Model of Intermittently Controlled Point-and-Click Behaviour.* Association for Computing Machinery, New York, NY. DOI: https://doi.org/10.1145/3411764.3445514

[35] P. Dorato and A. Levis. 1971. Optimal linear regulators: The discrete-time case. *IEEE Transactions on Automatic Control* 16, 6 (December 1971), 613–620. DOI: https://doi.org/10.1109/TAC.1971.1099832

[36] S. Edelman and T. Flash. 2004. A model of handwriting. *Biological Cybernetics* 57, 1—2 (2004), 25–36.

[37] J. D. Enderle and W. Zhou. 2010. *Models of Horizontal Eye Movements: A 3rd Order Linear Saccade Model. Part II.* Morgan & Claypool. Retrieved from https://books.google.de/books?id=XDGGBNk0e2oC.

[38] Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, and Jörg Müller. 2021. Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports* 11, 1 (2021), 1–15.

[39] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6 (1954), 381–391.

[40] Paul M. Fitts and James R. Peterson. 1964. Information capacity of discrete motor responses. *Journal of Experimental Psychology* 67, 2 (1964), 103.

[41] Tamar Flash and Ealan Henis. 1991. Arm trajectory modifications during reaching towards visual targets. *Journal of Cognitive Neuroscience* 3, 3 (07 1991), 220–230. DOI: https://doi.org/10.1162/jocn.1991.3.3.220

[42] Tamar Flash and Neville Hogan. 1985. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience* 5, 7 (1985), 1688–1703.

[43] Tamar Flash, Yaron Meirovitch, and Avi Barliya. 2013. Models of human movement: Trajectory planning and inverse kinematics studies. *Robotics and Autonomous Systems* 61, 4 (2013), 330–339.

[44] Karl Friston. 2011. What is optimal about motor control? *Neuron* 72, 3 (2011), 488–498. DOI: https://doi.org/10.1016/j.neuron.2011.10.018

[45] Peter Gawthrop, Ian Loram, Martin Lakie, and Henrik Gollee. 2011. Intermittent control: A computational theory of human control. *Biological Cybernetics* 104, 1–2 (2011), 31–51. DOI: https://doi.org/10.1007/s00422-010-0416-4

[46] Karl R. Gegenfurtner. 2016. The interaction between vision and eye movements. *Perception* 45, 12 (2016), 1333–1357.

[47] A. P. Georgopoulos, J. F. Kalaska, R. Caminiti, and J. T. Massey. 1982. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience* 2, 11 (1982), 1527–1537. DOI: https://doi.org/10.1523/JNEUROSCI.02-11-01527.1982

[48] Douglas J. Gillan, Kritina Holden, Susan Adam, Marianne Rudisill, and Laura Magee. 1990. How does fitts' law fit pointing and dragging? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* Association for Computing Machinery, New York, NY, 227–234. DOI: https://doi.org/10.1145/97243.97278

[49] J. Gordon, M. Ghilardi, and Claude Ghez. 1994. Accuracy of planar reaching movements. I. Independence of direction and extent variability. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation Cérébrale* 99, 1 (02 1994), 97–111. DOI: https://doi.org/10.1007/BF00241415

[50] Julien Gori and Olivier Rioul. 2020. A feedback information-theoretic transmission scheme (FITTS) for modeling trajectory variability in aimed movements. *Biological Cybernetics* 114, 6 (Dec 2020), 621–641. DOI: https://doi.org/10.1007/s00422-020-00853-7

[51] Julien Gori, Olivier Rioul, Yves Guiard, and Michel Beaudouin-Lafon. 2018. The perils of confounding factors: How fitts' law experiments can lead to false conclusions. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* Association for Computing Machinery, New York, NY, 1–10. DOI: https://doi.org/10.1145/3173574.3173770

[52] Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: Enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, New York, NY, 281–290. DOI: https://doi.org/10.1145/1054972.1055012

[53] Yves Guiard. 1993. On fitts's and hooke's laws: Simple harmonic movement in upper-limb cyclical aiming. *Acta Psychologica* 82, 1 (1993), 139–159.

[54] Emmanuel Guigon, Pierre Baraduc, and Michel Desmurget. 2007. Computational motor control: Redundancy and invariance. *Journal of Neurophysiology* 97, 1 (02 2007), 331–47. DOI: https://doi.org/10.1152/jn.00290.2006

[55] Simon R. Gutman and Gerald L. Gottlieb. 1992. Basic functions of variability of simple pre-planned movements. *Biological Cybernetics* 68, 1 (1992), 63–73.

[56] Adrian M. Haith and John W. Krakauer. 2013. Model-based and model-free mechanisms of human motor learning. In *Proceedings of the Progress in Motor Control.* Springer, 1–21.

[57] Kenneth B. Hannsgen. 1987. Stability and periodic solutions of ordinary and functional differential equations (T. A. Burton). *SIAM Review* 29, 4 (1987), 652–654. DOI: https://doi.org/10.1137/1029135

[58] Christopher M. Harris and Daniel M. Wolpert. 1998. Signal-dependent noise determines motor planning. *Nature* 394, 6695 (1998), 780–784. DOI: https://doi.org/10.1038/29528

[59] Lorenz Hetzel, John Dudley, Anna Maria Feit, and Per Ola Kristensson. 2021. Complex interaction as emergent behaviour: Simulating mid-air virtual keyboard typing using reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics* 27, 11 (2021), 1–1. DOI: https://doi.org/10.1109/TVCG.2021.3106494

[60] Bruce Hoff. 1994. A model of duration in normal and perturbed reaching movement. *Biological Cybernetics* 71, 6 (Oct. 1994), 481–488. DOI: https://doi.org/10.1007/BF00198466

[61] Bruce Hoff and Michael A. Arbib. 1993. Models of trajectory formation and temporal interaction of reach and grasp. *Journal of Motor Behavior* 25, 3 (1993), 175–192. DOI: https://doi.org/10.1080/00222895.1993.9942048

[62] Bruce Richard Hoff. 1992. *A Computational Description of the Organization of Human Reaching and Prehension.* Ph. D. Dissertation. Not available from Univ. Microfilms Int.

[63] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and M. Preston Mullen Jr. 1994. Integrality and separability of input devices. *ACM Transactions on Computer-Human Interaction* 1, 1 (1994), 3–26.

[64] R. Jagacinski and D. Monk. 1985. Fitts' law in two dimensions with hand and head movements. *Journal of Motor Behavior* 17, 1 (1985), 77–95.

[65] Richard J. Jagacinski and John M. Flach. 2003. *Control Theory for Humans: Quantitative Approaches to Modeling Performance.* Lawrence Erlbaum, Mahwah, New Jersey.

[66] Y. Jiang, Z. Jiang, and N. Qian. 2011. Optimal control mechanisms in human arm reaching movements. In *Proceedings of the 30th Chinese Control Conference.* 1377–1382.

[67] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. 2020. Pontryagin differentiable programming: An end-to-end learning and control framework. *Advances in Neural Information Processing Systems* 33 (2020), 7979–7992.

[68] Kelvin E. Jones, Antonia F. de C. Hamilton, and Daniel M. Wolpert. 2002. Sources of signal-dependent noise during isometric force production. *Journal of Neurophysiology* 88, 3 (2002), 1533–1544. DOI: https://doi.org/10.1152/jn.2002.88.3.1533

[69] M. Kawato. 1993. Optimization and learning in neural networks for formation and control of coordinated movement. In *Proceedings of the Attention and Performance XIV (Silver Jubilee Volume) Synergies in Experimental Psychology, Artificial Intelligence, and Cognitive Neuroscience.* 821–849.

[70] Mitsuo Kawato. 1996. Trajectory formation in arm movements: Minimization principles and procedures. *Advances in Motor Learning and Control* (1996), 225–259.

[71] Mitsuo Kawato. 1999. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology* 9, 6 (1999), 718–727.

[72] Łukasz Kidziński, Sharada Prasanna Mohanty, Carmichael F. Ong, Zhewei Huang, Shuchang Zhou, Anton Pechenko, Adam Stelmaszczyk, Piotr Jarosik, Mikhail Pavlov, Sergey Kolesnikov, Sergey Plis, Zhibo Chen, Zhizheng Zhang, Jiale Chen, Jun Shi, Zhuobin Zheng, Chun Yuan, Zhihui Lin, Henryk Michalewski, Piotr Milos, Blazej Osinski, Andrew Melnik, Malte Schilling, Helge Ritter, Sean F. Carroll, Jennifer Hicks, Sergey Levine, Marcel Salathé, and Scott Delp. 2018. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In *Proceedings of the NIPS '17 Competition: Building Intelligent Systems*, Sergio Escalera and Markus Weimer (Eds.). Springer International Publishing, Cham, 121–153.

[73] Konrad Kording. 2007. Decision theory: What "Should" the nervous system do? *Science* 318, 5850 (2007), 606–610. https://doi.org/10.1126/science.1142998

[74] Eileen Kowler. 2011. Eye movements: The past 25years. *Vision Research* 51, 13 (2011), 1457–1483. DOI: https://doi.org/10.1016/j.visres.2010.12.014. Vision Research 50th Anniversary Issue: Part 2.

[75] Richard Krauzlis, Laurent Goffart, and Ziad Hafed. 2017. Neuronal control of fixation and fixational eye movements. *Philosophical Transactions of the Royal Society B: Biological Sciences* 372, 1718 (04 2017), 20160205. DOI: https://doi.org/10.1098/rstb.2016.0205

[76] Solomon Kullback and Richard A. Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.

[77] Francesco Lacquaniti, Carlo Terzuolo, and Paolo Viviani. 1983. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica* 54, 1 (1983), 115–130.

[78] Shih-Chiung Lai, Gottfried Mayer-Kress, Jacob J. Sosnoff, and Karl M. Newell. 2005. Information entropy analysis of discrete aiming movements. *Acta Psychologica* 119, 3 (2005), 283–304. DOI: https://doi.org/10.1016/j.actpsy.2005.02.005

[79] Gary D. Langolf, Don B. Chaffin, and James A. Foulke. 1976. An investigation of fitts' law using a wide range of movement amplitudes. *Journal of Motor Behavior* 8, 2 (1976), 113–128. DOI: https://doi.org/10.1080/00222895.1976.10735061

[80] Daeyeol Lee, Nicholas L. Port, and Apostolos P. Georgopoulos. 1997. Manual interception of moving targets II. Online control of overlapping submovements. *Experimental Brain Research* 116, 3 (1997), 421–433.

[81] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics* 38, 4, Article 73 (July 2019), 13 pages. DOI: https://doi.org/10.1145/3306346.3322972

[82] Kaccie Y. Li, Pavan Tiruveedhula, and Austin Roorda. 2010. Intersubject variability of foveal cone photoreceptor density in relation to eye length. *Investigative Ophthalmology & Visual Science* 51, 12 (2010), 6858–6867.

[83] Weiwei Li and Emanuel Todorov. 2004. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the ICINCO (1)*. Citeseer, 222–229.

[84] Weiwei Li and Emanuel Todorov. 2007. Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system. *International Journal of Control* 80, 9 (2007), 1439–1453.

[85] Zhe Li, Pietro Mazzoni, Sen Song, and Ning Qian. 2018. A single, continuously applied control policy for modeling reaching movements with and without perturbation. *Neural Computation* 30, 2 (2018), 397–427. DOI: https://doi.org/10.1162/neco_a_01040. PMID: 29162001.

[86] Dan Liu and Emanuel Todorov. 2007. Evidence for the flexible sensorimotor strategies predicted by optimal feedback control. *Journal of Neuroscience* 27, 35 (2007), 9354–9368. DOI: https://doi.org/10.1523/JNEUROSCI.1110-06.2007

[87] Gerald Loeb, W. Levine, and Jiping He. 1990. Understanding sensorimotor feedback through optimal control. *Cold Spring Harbor Symposia on Quantitative Biology* 55 (02 1990), 791–803. DOI: https://doi.org/10.1101/SQB.1990.055.01.074

[88] I. MacKenzie, Tatu Kauppinen, and Miika Silfverberg. 2001. Accuracy measures for evaluating computer pointing devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 9–16. DOI: https://doi.org/10.1145/365024.365028

[89] I. Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction* 7, 1 (1992), 91–139. DOI: https://doi.org/10.1207/s15327051hci0701_3

[90] Michael J. McGuffin and Ravin Balakrishnan. 2005. Fitts' law and expanding targets: Experimental studies and designs for user interfaces. *ACM Transactions on Computer-Human Interaction* 12, 4 (2005), 388–422.

[91] Suzanne P. Mckee and Ken Nakayama. 1984. The detection of motion in the peripheral visual field. *Vision Research* 24, 1 (1984), 25–32. DOI: https://doi.org/10.1016/0042-6989(84)90140-8

[92] W. Pieter Medendorp. 2011. Spatial constancy mechanisms in motor control. *Philosophical Transactions of the Royal Society B: Biological Sciences* 366, 1564 (2011), 476–491.

[93] David E. Meyer, Richard A. Abrams, Sylvan Kornblum, Charles E. Wright, and J. E. Keith Smith. 1988. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review* 95, 3 (1988), 340.

[94] Pietro Morasso. 1981. Spatial control of arm movements. *Experimental Brain Research* 42, 2 (1981), 223–227.

[95] Martez E. Mott and Jacob O. Wobbrock. 2014. Beating the bubble: Using kinematic triggering in the bubble lens for acquiring small, dense targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 733–742.

[96] Denis Mottet, Reinoud Bootsma, Yves Guiard, and Michel Laurent. 1994. Fitts' law in two-dimensional task space. *Experimental Brain Research* 100, 1 (02 1994), 144–8. DOI: https://doi.org/10.1007/BF00227286

[97] Denis Mottet and Reinoud J. Bootsma. 1999. The dynamics of goal-directed rhythmical aiming. *Biological Cybernetics* 80, 4 (1999), 235–245.

[98] Jörg Müller, Antti Oulasvirta, and Roderick Murray-Smith. 2017. Control theoretic models of pointing. *ACM Transactions on Computer-Human Interaction* 24, 4, Article 27 (Aug. 2017), 36 pages. DOI: https://doi.org/10.1145/3121431

[99] Atsuo Murata. 1998. Improvement of pointing time by predicting targets in pointing with a PC mouse. *International Journal of Human-Computer Interaction* 10, 1 (1998), 23–32.

[100] Masaki Nakada, Tao Zhou, Honglin Chen, Tomer Weiss, and Demetri Terzopoulos. 2018. Deep learning of biomimetic sensorimotor control for biomechanical human animation. *ACM Transactions on Graphics* 37, 4, Article 56 (July 2018), 15 pages. DOI: https://doi.org/10.1145/3197517.3201305

[101] Eri Nakano, Hiroshi Imamizu, Rieko Osu, Yoji Uno, Hiroaki Gomi, Toshinori Yoshioka, and Mitsuo Kawato. 1999. Quantitative examinations of internal representations for arm trajectory planning: Minimum commanded torque change model. *Journal of Neurophysiology* 81, 5 (1999), 2140–2155. DOI: https://doi.org/10.1152/jn.1999.81.5.2140

[102] Mathieu Nancel and Edward Lank. 2017. *Modeling User Performance on Curved Constrained Paths*. Association for Computing Machinery, New York, NY, 244–254. DOI: https://doi.org/10.1145/3025453.3025951

[103] W. L. Nelson. 1983. Physical principles for economies of skilled movements. *Biological Cybernetics* 46, 2 (Feb. 1983), 135–147. DOI: https://doi.org/10.1007/BF00339982

[104] Ingram Olkin and Friedrich Pukelsheim. 1982. The distance between two random vectors with given dispersion matrices. *Linear Algebra and its Applications* 48 (1982), 257–263.

[105] Eunji Park and Byungjoo Lee. 2020. An intermittent click planning model. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[106] Phillip T. Pasqual and Jacob O. Wobbrock. 2014. Mouse pointing endpoint prediction using kinematic template matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 743–752.

[107] Robert Pastel. 2006. Measuring the difficulty of steering through corners. *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems* 1, 2 (2006), 1087–1096. https://doi.org/10.1145/1124772.1124934

[108] Y. Phillis. 1985. Controller design of systems with multiplicative noise. *IEEE Transactions on Automatic Control* 30, 10 (1985), 1017–1019. DOI: https://doi.org/10.1109/TAC.1985.1103828

[109] Rejean Plamondon. 1998. A kinematic theory of rapid human movements: Part III. Kinetic outcomes. *Biological Cybernetics* 78, 2 (1998), 133–145. DOI: https://doi.org/10.1007/s004220050420

[110] Réjean Plamondon and Adel M. Alimi. 1997. Speed/accuracy trade-offs in target-directed movements. *Behavioral and Brain Sciences* 20, 02 (1997), 279–303.

[111] Ning Qian, Yu Jiang, Zhong-Ping Jiang, and Pietro Mazzoni. 2013. Movement duration, fitts's law, and an infinite-horizon optimal feedback control model for biological motor systems. *Neural Computation* 25, 3 (2013), 697–724. DOI: https://doi.org/10.1162/NECO_a_00410

[112] Philip Quinn and Shumin Zhai. 2018. Modeling gesture-typing movements. *Human–Computer Interaction* 33, 3 (2018), 234–280. DOI: https://doi.org/10.1080/07370024.2016.1215922

[113] M. J. Richardson and T. Flash. 2002. Comparing smooth arm movements with the two-thirds power law and the related segmented-control hypothesis. *The Journal of Neuroscience* 22, 18 (2002), 8201–8211.

[114] D. A. Rosenbaum, L. D. Loukopoulos, R. G. Meulenbroek, J. Vaughan, and S. E. Engelbrecht. 1995. Planning reaches by evaluating stored postures. *Psychological review* 102, 1 (01 1995), 28–67. DOI: https://doi.org/10.1037/0033-295x.102.1.28

[115] Stuart J. Russell and Peter Norvig. 2002. Artificial intelligence - a modern approach, 2nd Edition. Prentice Hall.

[116] Fabrice R. Sarlegna and Pratik K. Mutha. 2015. The influence of visual target information on the online control of movements. *Vision Research* 110, (Part B) (2015), 144–154. DOI: https://doi.org/10.1016/j.visres.2014.07.001. On-line Visual Control of Action.

[117] Richard A. Schmidt, Timothy D. Lee, Carolee Winstein, Gabriele Wulf, and Howard N. Zelaznik. 2018. *Motor Control and Learning: A Behavioral Emphasis*. Human kinetics, Champaign, IL.

[118] Richard A. Schmidt, Howard Zelaznik, Brian Hawkins, James S. Frank, and John T. Quinn Jr. 1979. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review* 86, 5 (1979), 415.

[119] Alexander C. Schütz, Doris I. Braun, and Karl R. Gegenfurtner. 2011. Eye movements and perception: A selective review. *Journal of Vision* 11, 5 (2011), 9–9.

[120] Sofia Seinfeld, Tiare Feuchtner, Antonella Maselli, and Jörg Müller. 2020. User representations in human-computer interaction. *Human–Computer Interaction* 0, 0 (2020), 1–39. DOI: https://doi.org/10.1080/07370024.2020.1724790

[121] Reza Shadmehr, Helen J. Huang, and Alaa A. Ahmed. 2016. A representation of effort in decision-making and motor control. *Current Biology* 26, 14 (2016), 1929–1934. DOI: https://doi.org/10.1016/j.cub.2016.05.065

[122] Reza Shadmehr, Jean Jacques Orban de Xivry, Minnan Xu-Wilson, and Ting-Yu Shih. 2010. Temporal discounting of reward and the cost of time in motor control. *Journal of Neuroscience* 30, 31 (2010), 10507–10516. DOI: https://doi.org/10.1523/JNEUROSCI.1343-10.2010

[123] Reza Shadmehr and Steven P. Wise. 2005. *The Computational Neurobiology of Reaching and Pointing*. MIT Press, Cambridge, MA.

[124] David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. 2021. Reward is enough. *Artificial Intelligence* 299 (2021), 103535. DOI: https://doi.org/10.1016/j.artint.2021.103535

[125] Andrew Slifkin and Karl Newell. 1999. Noise, information transmission, and force variability. *Journal of Experimental Psychology. Human Perception and Performance* 25, 3 (07 1999), 837–51. DOI: https://doi.org/10.1037/0096-1523.25.3.837

[126] P. Stoica and Y. Selen. 2004. Model-order selection: A review of information criterion rules. *IEEE Signal Processing Magazine* 21, 4 (2004), 36–47. DOI: https://doi.org/10.1109/MSP.2004.1311138

[127] R. Storn and K. Price. 1997. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11, 4 (1997), 341–359.

[128] Hans Strasburger, Ingo Rentschler, and Martin Jüttner. 2011. Peripheral vision and pattern recognition: A review. *Journal of Vision* 11, 5 (12 2011), 13–13. DOI: https://doi.org/10.1167/11.5.13

[129] G. G. Sutton and K. Sykes. 1967. The variation of hand tremor with force in healthy subjects. *The Journal of Physiology* 191, 3 (1967), 699–711.

[130] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA.

[131] Hirokazu Tanaka, John W. Krakauer, and Ning Qian. 2006. An optimization principle for determining movement duration. *Journal of Neurophysiology* 95, 6 (2006), 3875–3886. DOI: https://doi.org/10.1152/jn.00751.2005

[132] Yuval Tassa, Nicolas Mansard, and Emo Todorov. 2014. Control-limited differential dynamic programming. In *Proceedings of the IEEE International Conference on Robotics and Automation.* DOI: https://doi.org/10.1109/ICRA.2014.6907001

[133] Benjamin W. Tatler, Mary M. Hayhoe, Michael F. Land, and Dana H. Ballard. 2011. Eye guidance in natural vision: Reinterpreting salience. *Journal of Vision* 11, 5 (2011), 5–5.

[134] C. W. Telford. 1931. The refractory phase of voluntary and associative responses. *Journal of Experimental Psychology* 14, 1 (1931), 1–36.

[135] Shelby G. Thompson, Daniel S. McConnell, Jeremy S. Slocum, and Michael Bohan. 2007. Kinematic analysis of multiple constraints on a pointing task. *Human Movement Science* 26, 1 (2007), 11–26. DOI: https://doi.org/10.1016/j.humov.2006.09.001

[136] Kurt Thoroughman and Reza Shadmehr. 2000. Learning of action through adaptive combination of motor primitives. *Nature* 407, 6805 (11 2000), 742–7. DOI: https://doi.org/10.1038/35037588

[137] Juan Camilo Vasquez Tieck, Marin Vlastelica Pogančić, Jacques Kaiser, Arne Roennau, Marc-Oliver Gewaltig, and Rüdiger Dillmann. 2018. Learning continuous muscle control for a multi-joint arm by extending proximal policy optimization with a liquid state machine. In *Proceedings of the International Conference on Artificial Neural Networks.* Springer, 211–221.

[138] Emanuel Todorov. 1998. *Studies of Goal-directed Movements.* Massachusetts Institute of Technology.

[139] Emanuel Todorov. 2002. Cosine tuning minimizes motor errors. *Neural Computation* 14, 6 (June 2002), 1233–1260. DOI: https://doi.org/10.1162/089976602753712918

[140] Emanuel Todorov. 2004. Optimality principles in sensorimotor control. *Nature Neuroscience* 7, 9 (10 2004), 907–915. DOI: https://doi.org/10.1038/nn1309

[141] Emanuel Todorov. 2005. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Computation* 17, 5 (2005), 1084–1108.

[142] Emanuel Todorov and Michael I. Jordan. 2002. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience* 5, 11 (2002), 1226–1235. DOI: https://doi.org/10.1038/nn963

[143] Emanuel Todorov and Weiwei Li. 2005. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 300–306.

[144] Y. Uno, M. Kawato, and R. Suzuki. 1989. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics* 61, 2 (01 Jun 1989), 89–101. DOI: https://doi.org/10.1007/BF00204593

[145] Francisco Valero-Cuevas. 2015. *Fundamentals of Neuromechanics*, Vol. 8. 1–194 pages. DOI: https://doi.org/10.1007/978-1-4471-6747-1

[146] Robert J. van Beers, Patrick Haggard, and Daniel M. Wolpert. 2004. The role of execution noise in movement variability. *Journal of Neurophysiology* 91, 2 (2004), 1050–1063. DOI: https://doi.org/10.1152/jn.00652.2003

[147] Frans C. T. van der Helm and Leonard A. Rozendaal. 2000. *Musculoskeletal Systems with Intrinsic and Proprioceptive Feedback.* Springer, New York, New York, NY, Chapter 11, 164–174. DOI: https://doi.org/10.1007/978-1-4612-2104-3_11

[148] Paolo Viviani and Tamar Flash. 1995. Minimum-jerk, two-thirds power law, and isochrony: Converging approaches to movement planning. *Journal of Experimental Psychology: Human Perception and Performance* 21, 1 (1995), 32.

[149] Tie Wang, Goran S. Dordevic, and Reza Shadmehr. 2001. Learning the dynamics of reaching movements results in the modification of arm impedance and long-latency perturbation responses. *Biological Cybernetics* 85, 6 (2001), 437–448.

[150] Xiaolan Wang, Mingsha Zhang, Ian S. Cohen, and Michael E. Goldberg. 2007. The proprioceptive representation of eye position in monkey primary somatosensory cortex. *Nature Neuroscience* 10, 5 (2007), 640–646.

[151] Colin Ware and Harutune H. Mikaelian. 1986. An evaluation of an eye tracker as a device for computer input. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface.* Association for Computing Machinery, New York, NY, 183–188. DOI: https://doi.org/10.1145/29933.275627

[152] John Williamson. 2006. *Continuous Uncertain Action.* Ph. D. Dissertation. Dept. of Computing Science, University of Glasgow.

[153] John Williamson, Roderick Murray-Smith, Benjamin Blankertz, Matthias Krauledat, and K.-R. Müller. 2009. Designing for uncertain, asymmetric control: Interaction design for brain–computer interfaces. *International Journal of Human-Computer Studies* 67, 10 (2009), 827–841.

[154] Robert Sessions Woodworth. 1899. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements* 3, 3 (1899), i.

[155] Charles E. Wright and David E. Meyer. 1983. Conditions for a linear speed-accuracy trade-off in aimed movements. *The Quarterly Journal of Experimental Psychology Section A* 35, 2 (1983), 279–296. DOI: https://doi.org/10.1080/14640748308402134

[156] Peter G. Yule, John Fox, David W. Glasspool, and Richard P. Cooper. 2013. *Modelling High-level Cognitive Processes*. Psychology Press.

[157] Gregory J. Zelinsky. 2008. A theory of eye movements during target acquisition. *Psychological Review* 115, 4 (2008), 787.

[158] Shumin Zhai, Johnny Accot, and Rogier Woltjer. 2004. Human action laws in electronic virtual worlds: An empirical study of path steering performance in VR. *Presence* 13, 2 (04 2004), 113–127. DOI: https://doi.org/10.1162/1054746041382393

[159] Brian Ziebart, Anind Dey, and J. Andrew Bagnell. 2012. Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*. ACM, New York, NY, 1–10. DOI: https://doi.org/10.1145/2166966.2166968

[160] J. Alberto Álvarez Martín, Henrik Gollee, Jörg Müller, and Roderick Murray-Smith. 2021. Intermittent control as a model of mouse movements. *ACM Transactions on Computer-Human Interaction (TOCHI)* 28, 5 (2021), 1–46.

Received 1 October 2021; revised 24 February 2022; accepted 4 March 2022

6 Reinforcement Learning Control of a Biomechanical Model of the Upper Extremity

Authors: Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, Jörg Müller
Status: Published in Sci Rep 11 (Scientific Reports 2021) [15]

FF conducted and JM supervised the research process. MB adjusted the biomechanical model. FF, MB, and MK worked on the MuJoCo implementation and collected simulation data. FF, MK, and AF analyzed and validated the results. FF created the plots. FF and JM wrote the main manuscript text. All authors reviewed the manuscript. FF is the corresponding author.

OPEN

Reinforcement learning control of a biomechanical model of the upper extremity

Florian Fischer✉, Miroslav Bachinski, Markus Klar, Arthur Fleig & Jörg Müller

Among the infinite number of possible movements that can be produced, humans are commonly assumed to choose those that optimize criteria such as minimizing movement time, subject to certain movement constraints like signal-dependent and constant motor noise. While so far these assumptions have only been evaluated for simplified point-mass or planar models, we address the question of whether they can predict reaching movements in a full skeletal model of the human upper extremity. We learn a control policy using a motor babbling approach as implemented in reinforcement learning, using aimed movements of the tip of the right index finger towards randomly placed 3D targets of varying size. We use a state-of-the-art biomechanical model, which includes seven actuated degrees of freedom. To deal with the curse of dimensionality, we use a simplified second-order muscle model, acting at each degree of freedom instead of individual muscles. The results confirm that the assumptions of signal-dependent and constant motor noise, together with the objective of movement time minimization, are sufficient for a state-of-the-art skeletal model of the human upper extremity to reproduce complex phenomena of human movement, in particular Fitts' Law and the $\frac{2}{3}$ Power Law. This result supports the notion that control of the complex human biomechanical system can plausibly be determined by a set of simple assumptions and can easily be learned.

In the case of simple end-effector models, both Fitts' Law and the $\frac{2}{3}$ Power Law have been shown to constitute a direct consequence of minimizing movement time, under signal-dependent and constant motor noise[1,2]. Here, we aim to confirm that these simple assumptions are also sufficient for a full skeletal upper extremity model to reproduce these phenomena of human movement. As a biomechanical model of the human upper extremity, we use the skeletal structure of the *Upper Extremity Dynamic Model* by Saul et al.[3], including thorax, right clavicle, scapula, shoulder, arm, and hand. The model has seven actuated degrees of freedom (DOFs): shoulder rotation, elevation and elevation plane, elbow flexion, forearm rotation, and wrist flexion and deviation. While the thorax is fixed in space, the right upper extremity can move freely by actuating these DOFs. To deal with the curse of dimensionality and make the control problem tractable, following van Beers et al.[4], we use a simplified second-order muscle model acting at each DOF instead of individual muscles. These second-order dynamics map an action vector obtained from the learned policy to the resulting activations for each DOF. Following van Beers et al.[4], we assume both signal-dependent and constant motor noise in the control, with noise levels 0.103 and 0.185, respectively. Multiplying these activations with constant moment arm scaling factors, which represent the strength of the muscle groups at the respective DOFs, yields the torques that are applied at each DOF independently. Further details on the biomechanical model are provided in the *Methods* section below.

The Upper Extremity Dynamic Model is significantly more complex than standard point-mass or linked-segment models. In particular, there is no explicit formula for the non-linear and non-deterministic system dynamics. Together with the objective of movement time minimization, these properties make it difficult to use classical optimal control approaches. Instead, in this paper we learn a control policy using deep *reinforcement learning (RL)*. RL algorithms, just like the optimal control methods discussed below, aim to find a policy that maximizes a given reward function. Moreover, they do not require any explicit knowledge about the underlying model. Instead, the optimal value of a certain state is estimated from sampling different actions in the environment and observing the subsequent state and obtained reward[5].

In our approach, a control policy initially generates random movements, which are rewarded with the negative time to reach randomly placed 3D targets of varying size, with the right index finger (see Fig. 1). This reward signal implies movement time minimization for aimed movements. The policy is updated using the *soft-actor-critic* algorithm (SAC)[6]. The actor and critic networks both consist of two fully connected layers with 256 neurons each,

University of Bayreuth, Bayreuth, Germany. ✉email: florian.j.fischer@uni-bayreuth.de

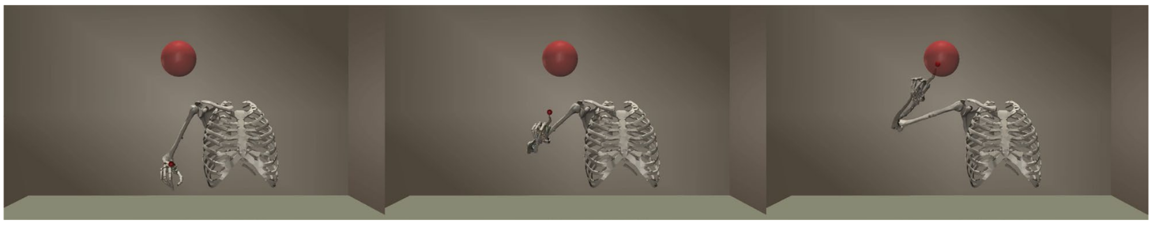


Figure 1. Synthesized reaching movement. A policy implemented as a neural network computes motor control signals of simplified muscles at the joints of a biomechanical upper extremity model from observations of the current state of the upper body. We use Deep Reinforcement Learning to learn a policy that reaches random targets in minimal time, given signal-dependent and constant motor noise.

followed by the output layer, which either returns the means and standard deviations of the action distributions (for the actor network) or the state-action value (for the critic network). Further information about the network architecture and a detailed description of all state components can be found in the *Methods* section below. To make reinforcement learning computationally feasible within a reasonable time period, a fast physics simulation is advantageous. Accordingly, we implemented the biomechanical model in MuJoCo[7].

It is important to note in this context that the assumption of minimizing total movement time does not provide any gradient information to the reinforcement learner. In particular, it is not possible to distinguish beneficial states and actions from inappropriate ones before the target has been reached, which terminates the episode and thus increases the total return. This, together with the fairly small subspace of appropriate actions relative to the number of possible control vectors, makes it very difficult to obtain a reasonable policy without additional aid. For this reason, we created an adaptive curriculum, which dynamically decreases the target diameter from 60 cm to less than 2 cm during training. This has proven to be both effective (targets with diameter around 2 cm are consistently reached by the final policy) and efficient (this minimum width was reached after 1.2M steps, while various predetermined curricula required more than 3M steps).

Related work

The question of how human arm movements are internally planned and controlled has received significant attention in the literature. Important phenomena that emerge from human arm movements include Fitts' Law and the $\frac{2}{3}$ Power Law. In this section we review related work in these areas.

Motor control.

Many models of human motor control assume that some objective function is optimized during the planning of the movement. A variety of objective functions have been proposed, including minimization of either jerk[8,9], peak acceleration[10], end-point variance[1], duration[2,11], or torque-change[12]. Moreover, combined objective functions have been used to model a trade-off between different objectives, e.g., between accuracy and effort[13,14], or jerk and movement time[15]. Extensions have been proposed that, e.g., focus on initial gating mechanisms[16] or motor synergies representing agonist and antagonist muscle groups[17].

While most of these models imply a separation between the planning and the execution stage, the *optimal feedback control* theory[18–22] assumes that sensory signals about the controlled quantity are fed back to the controller. These observations are then directly used to compute the remaining optimal control signals, resulting in a feedback loop. Extensions to infinite-horizon problems[23], which yield the optimal steady-state solution at the expense of neglecting transient behavior, and explicit non-linear time costs[24,25] have been proposed.

While many early works in motor control have modeled the biomechanics as point-mass models with linear dynamics[1,13] or linked-segment models[1], there is a growing interest in biomechanical models of increasing realism and fidelity. This is spurred by advances in biomechanical modeling[3,26,27] and simulation[28,29]. Biomechanical models allow control beyond the end-effector, for example on the level of joints[12,30–35], or muscles[36–39].

Joint-actuated models apply different optimality criteria for movement generation and coordination, minimizing, e.g., the angular accelerations with constraints[40], angular jerk[30], torque-change[12,31], mechanical energy expenditure[41], a combination of absolute work and angular acceleration[35], or some combination of accuracy and effort costs in the context of optimal feedback control[14]. The biomechanical plant in these works is usually represented as a linked-segment model, with simplified kinematic properties. In particular, the shoulder joint is commonly described as a rotation-only joint, ignoring the translatory part as well as complex movements related to the scapula and clavicle. Some of these models also include simplified muscles with simplified biomechanical attachment[14].

More recently, more complex, high-fidelity biomechanical musculoskeletal models have been introduced[36,37,42,43], where the control is muscle-based. In these models, the computation of the control values is commonly based on neural networks, particularly on deep learning and reinforcement learning. These methods have been applied successfully to predict coordinated muscle activations for multi-joint arm[42], lower body[43], and full body[36] movements. Moreover, a combination of 20 neural networks, each designed to imitate some specific part of the sensorimotor system, has recently been used to synthesize movements for such diverse sensorimotor tasks as reaching, writing, and drawing[37]. To make the control of muscle-based models feasible, these works apply multiple simplifications to the full biomechanical model, such as reducing or immobilising degrees of freedom[37,42] or even completely locking the movement to two dimensions[43], ignoring tendon's elasticity[36,37], limiting maximum passive forces[36], ignoring the muscle activation dynamics[36,37] or significantly reducing the number of independently controlled muscles[42,43]. Also, the control learning strategies differ from the pure reinforcement learning approach by applying imitation learning[36,37], or using artificial training data with simplified dynamics[37].

Up to now, these models have not been evaluated regarding the realism of the movements generated, in particular whether they exhibit features characteristic of human movements, such as *Fitts' Law* and the $\frac{2}{3}$ *Power Law*[34,36,37].

Fitts' law. Fitts' Law[44] describes the speed-accuracy trade-off of aimed movements towards a spatially defined target. Given the distance D between the initial position of the controlled end-effector (e.g., the hand or the finger) and the desired target position, and given the width W of the target, this law claims a logarithmic relationship between the distance-width ratio $\frac{D}{W}$ and the resulting movement time MT:

$$MT = a + b\log_2\left(\frac{D}{W}+1\right), \tag{1}$$

where we used the *Shannon formulation*[45]. Most works that explored possible explanations for the emergence of Fitts' Law have postulated that it results from noise in motor control. Crossman and Goodeve[46] showed that Fitts' Law emerges from the assumptions of isochronal submovements towards the target and constant error-velocity ratio. Meyer et al.[47] demonstrated that a power form of Fitts' Law emerges from the optimization of the relative duration of two submovements in order to achieve minimal movement time, assuming that the standard deviation of submovement endpoints increases proportionally with movement velocity. Fitts' Law has also been derived within the infinite-horizon optimal control framework, assuming that the target is reached as soon as the positional end-effector variance relative to the target center falls below the desired target accuracy[23].

Harris and Wolpert[1] proposed that the central nervous system (CNS) aims to minimize movement end-point variance given a fixed movement time, under the constraint of signal-dependent noise. This signal-dependent noise is assumed to be the main factor determining the end-point accuracy: Faster movements can be achieved through applying larger control signals (in the extreme, this leads to the time-minimizing *Bang-bang* control), but only at the costs of larger deviations, which in turn induce a larger end-point variance and thus a greater risk of missing the target. This trade-off has a strong neuroscientific evidence[48] and is consistent with the speed-accuracy trade-off proposed by Fitts' Law[1,2]. Moreover, in the case of arm-reaching movements, it has been shown recently that the assumptions of feed-forward control and signal-dependent noise (using dynamics of a two-link planar arm model) directly imply Fitts' Law, with coefficients a and b related to the level of signal-dependent noise[49] . Both coefficients were also shown to depend on the dynamics and kinematics, e.g., on the viscosity, or the Jacobian matrix relating the joint space and the end-effector space.

$\frac{2}{3}$ Power law. Continuous, rhythmic movements such as drawing or hand-writing, exhibit a speed-curvature trade-off described by the $\frac{2}{3}$ Power Law[50]. This law proposes a non-linear relationship between the radius of curvature ρ_n and the corresponding tangential velocity v_n,

$$v_n = k\rho_n^{1-\beta}, \tag{2}$$

$$\beta \approx \frac{2}{3}, \tag{3}$$

where the parameter k determines the velocity gain. This particularly implies that higher curvature leads to lower velocity. It has also been demonstrated that the $\frac{2}{3}$ Power Law is equivalent to constant affine velocity[51].

The $\frac{2}{3}$ Power Law has been confirmed to hold for a variety of task conditions, including hand movement[52], eye movement[53], perceptuomotor tasks[54,55], and locomotion[56]. Moreover, it has been shown to apply under the assumption of signal-dependent noise[1]. Schaal and Sternad[57] observed that the perimeter of the ellipse has a large impact on the validity of this law, with β obtained from a non-linear regression showing deviations in the order of 30–40% for large ellipses (or, alternatively, with decreasing coefficient of determination R^2, i.e., decreasing reliability of the parameter fitting). Based on these observations, Schaal and Sternad argue that the $\frac{2}{3}$ Power Law cannot be an intrinsic part of the movement planning procedure, but rather occurs as a "by-product" from the generation of smooth trajectories in intrinsic joint space[57] Following this argumentation, the non-linearities arising from the transformation from joint space to end-effector space, i.e., from a non-trivial kinematic chain, may account for scale- and direction-dependent results. Other theories see the cause of the wide applicability of the $\frac{2}{3}$ Power Law either in trajectory planning processes such as motor imagery[58] or jerk minimization[59], or directly emerging from muscle properties[60] or population vectors in motor cortical areas in the CNS[61,62].

Results

Fitts' law. In order to evaluate the trajectories resulting from our final policy for different target conditions, we designed a discrete Fitts' Law type task. The task follows the ISO 9241-9 ergonomics standard and incorporates 13 equidistant targets arranged in a circle at 50 cm distance in front of the body and placed 10 cm to the right of the right shoulder (Fig. 2). The objective is for the end-effector to reach each target and to remain inside the target for 100 ms. In this case we deem the movement successful. Although not included in the training phase, remaining inside the target seemed to be unproblematic during evaluation. If either the movement was successful, or 1.5 s have passed, the next target is given to the learned policy.

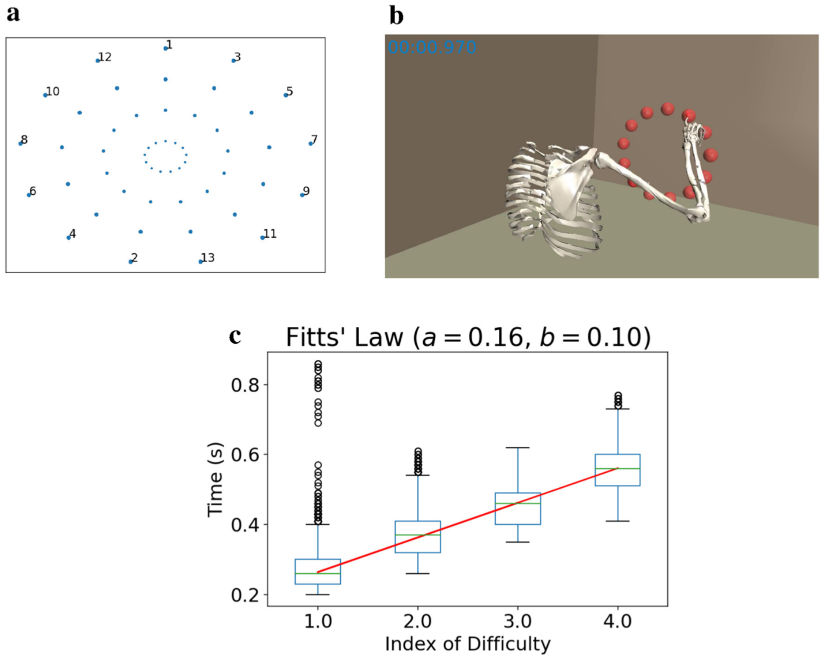


Figure 2. Fitts' Law type task. (**a**) The target setup in the discrete Fitts' Law type task follows the ISO 9241-9 ergonomics standard. Different circles correspond to different IDs and distances between targets. (**b**) Visualization of our biomechanical model performing aimed movements. Note that for each time step, only the *current* target (position and radius) is given to the learned policy. (**c**) The movements generated by our learned policy conform to Fitts' Law. Here, movement time is plotted against ID for all distances and IDs in the considered ISO task (6500 movements in total).

The Index of Difficulty (ID) of the tasks ranges from 1 to 4, where ID is computed as $\log_2(D/W + 1)$. D denotes the distance between the initial and target position, and W is the target size. We execute 50 movements for each task condition and each direction, i.e., 6500 movements in total—all were successful.

Using the trajectories from this discrete pointing task, we evaluate whether the synthesized movements follow Fitts' Law[44], i.e., whether there is a linear relationship between task difficulty (ID) and the required movement time. Figure 2c shows the total duration for each movement sorted by ID. The median movement times for each ID (green lines) are approximated by a linear function (red line, with coefficient of determination $R^2 = 0.9986$).

$\frac{2}{3}$ Power law. We evaluate whether our model exhibits the $\frac{2}{3}$ Power Law using an elliptic via-point task. To this end, we define an ellipse in 2D space (55 cm in front, 10 cm above, and 10 cm to the right of the shoulder) that lies completely within the area used for target sampling during training (ellipse radii are 7.5 cm (horizontal) and 3 cm (vertical)). Using the via-point method described in the *Methods* section below, our learned policy needs to trace the ellipse for 60 s as fast as possible . As shown in Fig. 3a, the simulation trajectories deviate from the desired ellipse, with the lower-right segment being flattened. Using these trajectories, we compute ρ_n and v_n for all time steps sampled at a rate of 100 Hz and then perform a log-log regression on the resulting values. This yields the optimal parameter values $\beta = 0.65$ and $k = 0.54$ (with correlation coefficient $R = 0.84$). Thus, the $\frac{2}{3}$ Power Law accounts for 71% of the variance observed in elliptic movements ($R^2 = 0.71$). Both the data points and the linear approximation in log-log space are shown in Fig. 3b.

Movement trajectories. In addition to Fitts' Law and the $\frac{2}{3}$ Power Law, we qualitatively analyze the movement trajectories generated by the model. Figures 4 and 5 show the position, velocity, and acceleration time series, as well as 3D movement path, Phasespace, and Hooke plots for multiple movements from the Fitts' Law type task for two representative task conditions (ID 4 respective ID 2, each with a 35 cm distance between targets) and one representative movement direction (between targets 7 and 8 shown in Fig. 2a). Apart from the 3D movement path, all plots show centroid projections of the respective trajectory onto the vector between the initial and target positions.

The movements exhibit typical features of human aimed movements, such as symmetric bell-shaped velocity profiles[63]. Movements are smooth, and gently accelerate and decelerate, as evident in the acceleration profiles and Hooke plots in Figs. 4 and 5. For high ID (Fig. 4), movements exhibit an initial rapid movement towards the target, followed by an extended phase of corrective movements. For low ID (Fig. 5), the phase of corrective movements is generally shorter.

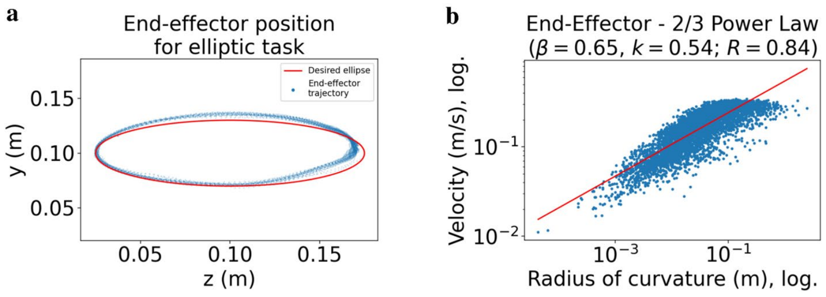


Figure 3. Elliptic via-point task. Elliptic movements generated by our learned policy conform to the $\frac{2}{3}$ Power Law. (**a**) End-effector positions projected onto the 2D space (blue dots), where targets were subsequently placed along an ellipse of 15 cm width and 6 cm height (red curve). (**b**) Log-log regression of velocity against radius of curvature for end-effector positions sampled with 100 Hz when tracing the ellipse for 60 s.

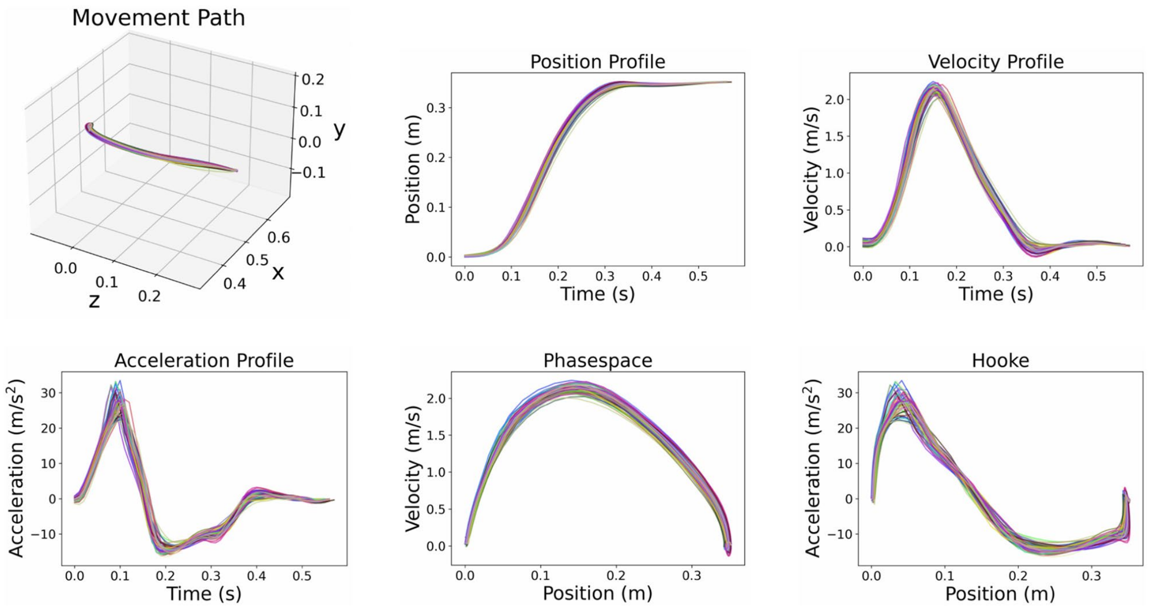


Figure 4. End-effector trajectories (ID 4). 3D path, projected position, velocity, acceleration, phasespace, and Hooke plots of 50 aimed movements (between targets 7 and 8 shown in Fig. 2a) with ID 4 and a target distance of 35 cm.

Movement trajectories towards the target are slightly curved and some of them exhibit pronounced correctional submovements at the end (see, e.g., Supplementary Fig. S1 and S2 online). The between-movement variability within one movement direction and task condition decreases with increasing ID. In particular, very simple ID 1 movements exhibit a large variability and are most prone to outliers (see, e.g., Supplementary Fig. S3 online).

For a few movement directions (mostly in ID 2 tasks), the corresponding plots seem to incorporate two different trajectory types (see, e.g., Supplementary Fig. S6 online): While some movements start with zero or even a negative acceleration and show a typical N-shaped acceleration profile, others exhibit a positive acceleration at the beginning, and their first peak is less pronounced. The reason for this behavior is corrective submovements at the end of the previous movement (see, e.g., Supplementary Fig. S4 and S5 online), leading to a different initial acceleration at the beginning of the subsequent movement. Apart from these notable features, almost all movements exhibit bell-shaped velocity and N-shaped acceleration profiles, as is typical for pointing tasks[63,64].

Discussion

Our results indicate that, under the assumption of movement time minimization given signal-dependent and constant motor noise, movement of the human upper extremity model produced by reinforcement learning follows both Fitts' Law and the $\frac{2}{3}$ Power Law. The movement times of aimed movements produced by the model depend linearly on the Index of Difficulty of the movement. For elliptic movements, the logarithm of the velocity

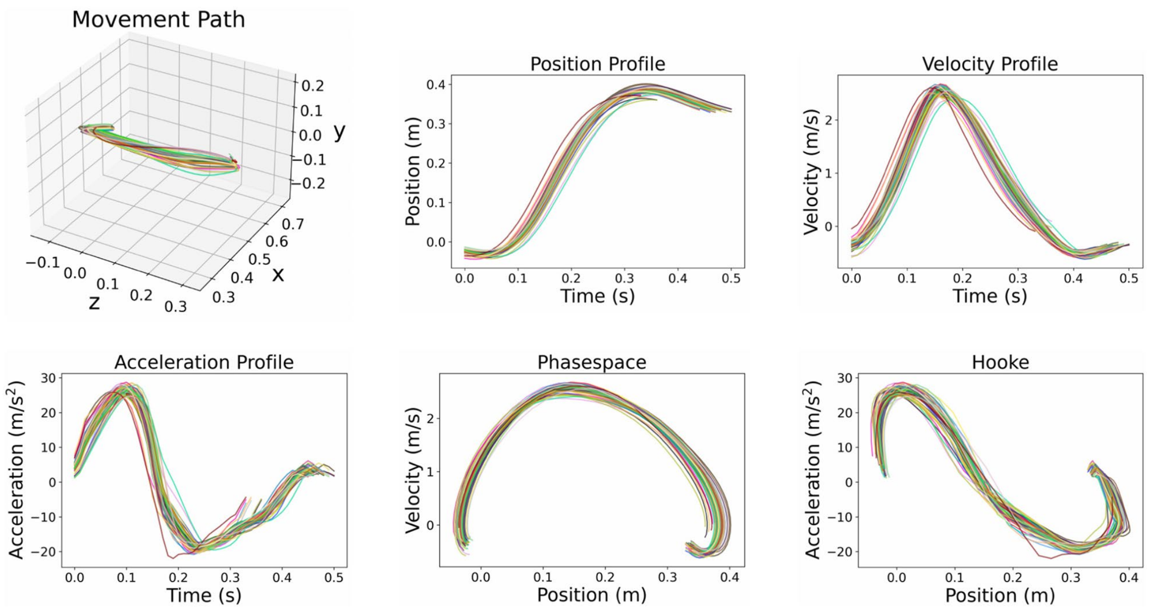


Figure 5. End-effector trajectories (ID 2). 3D path, projected position, velocity, acceleration, phasespace, and Hooke plots of 50 aimed movements (between targets 7 and 8 shown in Fig. 2a) with ID 2 and a target distance of 35 cm.

of the end-effector correlates with the logarithm of the radius of curvature[65] The optimal $\beta = 0.65$ obtained from log-log regression matches the proposed value of $\frac{2}{3}$, with a correlation coefficient of $R = 0.84$, which is consistent with previous observations, as the required ellipse has moderate size[57]. Finally, the generated trajectories exhibit features that are typical for human arm movements, such as bell-shaped velocity and N-shaped acceleration profiles.

The results confirm previous findings that demonstrated these phenomena in simpler models of the human biomechanics. In particular, the emergence of Fitts' Law and the $\frac{2}{3}$ Power Law from the assumption of signal-dependent noise has been demonstrated in the case of point-mass and linked-segment models of the human arm[1,2,49]. Our results support that insight by showing that Fitts' Law and the $\frac{2}{3}$ Power Law also emerge from those normative principles in a state-of-the-art biomechanical model of the human arm with simplified actuation.

In addition, we want to emphasize that the control signals that drive this model were obtained from RL methods. The fact that Fitts' Law and the $\frac{2}{3}$ Power Law hold for the generated trajectories provides evidence that behavior abiding these established laws of human motion can be generated using joint-actuated biomechanical models controlled by reinforcement learning algorithms. To the best of our knowledge, this has not yet been shown for state-of-the-art biomechanical models.

One limitation of our approach is the implicit assumption of *perfect observability*, as all state information (joint angles, end-effector position, etc.) are immediately available to the controller, without any disturbing noise. In the future, it will be interesting to combine state-of-the art models of sensory perception with the presented RL-based motor control approach. Promising approaches to address this problem include the usage of recurrent networks[66,67] and the internal formation of "beliefs", given the latest (imperfect) observations[68].

Another limitation is the usage of simplified muscle dynamics due to the curse of dimensionality. However, recent applications of deep learning methods, which approximate complex state-dependent torque limits[69] or muscle activation signals[37] using synthesized training data, raise hope for future approaches that efficiently combine RL or optimal control methods with state-of-the-art muscle models. It will be interesting to see whether well-established regularities such as Fitts' Law or the $\frac{2}{3}$ Power Law also emerge from such models.

Methods

Below, we first provide details on our biomechanical model. After discussing our general reinforcement learning approach, we focus on the individual components of our method, namely states, actions, scaling factors, rewards, and an adaptive target-selection mechanism. We also provide details on the implementation of our algorithm. Finally, we discuss the methods used for evaluation.

Biomechanical model of the human upper extremity. Our biomechanical model of the human upper extremity is based on the *Upper Extremity Dynamic model*[3], which was originally implemented in *OpenSim*[28]. Kinematically, the model represents the human shoulder and arm, using seven physical bodies and five "phantom" bodies to model the complex movement of the shoulder. This corresponds to three joints (shoulder, elbow, and wrist) with seven DOFs and five additional joints with thirteen associated components coupled by thirteen constraints with the DOFs. Each DOF has constrained joint ranges (see Table 1), which limits the possible movements. In contrast to linked-segment models, the *Upper Extremity Dynamic model* represents both translational and rotatory components of the movement within shoulder, clavicle, and scapula, and also within the wrist. It also contains physiological joint axis orientations instead of the perpendicular orientations in linked-segment models. The dynamics components of the musculoskeletal model are represented by the weight and inertia matrix of each non-phantom body and the default negligible masses and inertia of all phantom bodies. The dynamics properties of the model were extracted from various previously published works on human and cadaveric studies. The active components of the *Upper Extremity Dynamic Model* consist of thirty-one Hill-type muscles as well as of fourteen coordinate limit forces softly generated by the ligaments when a DOF approaches the angle range limit. Further details of this model are given in Saul et al.[3]

In order to make reinforcement learning feasible, we manually implement the *Upper Extremity Dynamic Model* in the fast MuJoCo physics simulation[7]. With respect to kinematics, the MuJoCo implementation of the model is equivalent to the original OpenSim model and contains physiologically accurate degrees of freedom, as well as corresponding constraints. We assume the same physiological masses and inertial properties of individual segments as in the OpenSim model. We do not implement muscles in the MuJoCo model, as this would significantly slow down the simulation and make reinforcement learning computationally infeasible due to the exponential growth of decision variables in the (discretized) action space when increasing the number of DOFs – the *curse of dimensionality*. In particular, computing dynamic actuator lengths (which significantly affect the forces produced by muscle activation patterns) has proven challenging in MuJoCo[70]. Instead, we implement simplified actuators, representing aggregated muscle actions on each individual DOF, which are controlled using the second-order dynamics introduced by van der Helm et al.[71] with fixed excitation and activation time constants $t_e = 30$ ms and $t_a = 40$ ms, respectively. We discretize the continuous state space system using the *forward Euler method*, which yields the following dynamics:

$$\begin{bmatrix} \sigma_{n+1}^{(q)} \\ \dot{\sigma}_{n+1}^{(q)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ \frac{-\Delta t}{(t_e t_a)} & 1 - \Delta t \frac{t_e + t_a}{t_e t_a} \end{bmatrix} \begin{bmatrix} \sigma_n^{(q)} \\ \dot{\sigma}_n^{(q)} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\Delta t}{t_e t_a} \end{bmatrix} c_n^{(q)}, \tag{4}$$

where $c_n^{(q)}$ is the applied control and $\sigma_n^{(q)}$ the resulting activation for each DOF $q \in \mathcal{Q}$, and $\mathcal{Q}$ is the set that contains all DOFs. The controls are updated every Δt=10 ms, at time steps $n \in \{0, \ldots, N-1\}$. To get more accurate results, at each time step n, we compute five sub-steps (during which the control $c_n^{(q)}$ is constant) with a sampling time of 2 ms to arrive at time step $n+1$.

We assume both signal-dependent and constant noise in the control, i.e.,

$$c_n^{(q)} = (1 + \eta_n) a_n^{(q)} + \epsilon_n, \tag{5}$$

where $a_n = (a_n^{(q)})_{q \in \mathcal{Q}}$ denotes the action vector obtained from the learned policy, and η_n and ϵ_n are Gaussian random variables with zero mean and standard deviations of 0.103 and 0.185, respectively, as described by van Beers et al.[4] The torques, which are applied at each DOF independently, are obtained by multiplying the respective activation $\sigma_n^{(q)}$ with a constant scaling factor $g^{(q)}$, which represents the strength of the muscle groups at the this DOF, i.e.,

$$\tau_n^{(q)} = g^{(q)} \sigma_n^{(q)}. \tag{6}$$

We select the scaling factors, and respectively the maximum voluntary torques for the actuators given in Table 1, based on experimental data as described below. We currently do not model the soft joint ranges in MuJoCo, as the movements the model produces do not usually reach joint limits.

The used biomechanical model provides the following advantages over simple linked-segment models:

- Phantom bodies and joints allow for more realistic movements, including both translation and rotation components within an individual joint,
- Individual joint angle and torque limits are set for each and every DOF,
- Axes between joints are chosen specifically and not just perpendicular between two segments,
- The model includes physiological body segment masses, and yields better options for scaling individual body parts, e.g., based on particular individuals.

Reinforcement learning. We define the task of controlling the biomechanical model of the human upper extremity through motor control signals applied at the joints as a reinforcement learning problem, similar to recent work from Cheema et al.[34] In this formulation, a policy $\pi_\theta(a|s)$ models the conditional distribution over actions $a \in \mathcal{A}$ (motor control signals applied at the individual DOFs) given the state $s \in \mathcal{S}$ (the pose, velocities, distance to target, etc.). The subindex θ denotes the parameters of the neural networks introduced below. At each timestep $n \in \{0, \ldots, N\}$, we observe the current state s_n, and sample a new action a_n from the current policy π_θ. The physical effects of that action, i.e., the application of these motor control signals, constitute the new state s_{n+1}, which we obtain from our biomechanical simulation. In our model, given s_n and a_n, the next state s_{n+1} is not deterministic, since both signal-dependent and constant noise are included. Hence, we denote the probability of reaching some subsequent state s_{n+1} given s_n and a_n by $p(s_{n+1}|s_n, a_n)$, while $p(s_0)$ denotes the probability of starting in s_0. Given some policy π_θ and a trajectory $T = (s_0, a_0, \ldots, a_{N-1}, s_N)$,

$$p_\theta(T) = p(s_0) \prod_{n=0}^{N-1} \pi_\theta(a_n|s_n) p(s_{n+1}|s_n, a_n) \tag{7}$$

describes the probability of realizing that trajectory. Evaluating/Sampling equation (7) for all possible trajectories $T \in \mathcal{T}$ then yields the distribution over all possible trajectories, $\varrho_\theta^{\mathcal{T}}$, induced by a policy π_θ.

We compute a reward r_n at each time step n, which allows us to penalize the total time needed to reach a given target. The total return of a trajectory is given by the sum of the (discounted) rewards $\sum_{n=0}^{N} \gamma^n r_n$, where $\gamma \in]0, 1]$ is a discount factor that causes the learner to prefer earlier rewards to later ones. Incorporating the entropy term,

$$\mathcal{H}(\pi_\theta(\cdot \mid s)) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[-\log(\pi_\theta(a \mid s))], \tag{8}$$

yields the expected (soft) return

$$J(\theta) = \mathbb{E}_{T\sim\varrho_\theta^{\mathcal{T}}}\left[\left(\sum_{n=0}^{N-1} \gamma^n \big(r_n - \alpha \log(\pi_\theta(a_n \mid s_n))\big)\right) + \gamma^N r_N\right], \tag{9}$$

which we want to maximize with respect to the parameters θ, i.e., the goal is to identify the optimal parameters θ^* that maximize $J(\theta)$. Here, the *temperature parameter* $\alpha > 0$ determines the importance of assigning the same probability to all actions that yield the same return (enforced by maximizing the entropy $\mathcal{H}$), i.e., increasing the "stochasticity" of the policy π_θ, relative to maximizing the expected total return. It thus significantly affects the optimal policy, and finding an "appropriate" value is non-trivial and heavily depends on the magnitude of the rewards r_n. For this reason, we decided to adjust it automatically during training together with the parameters θ, using dual gradient descent as implemented in the soft-actor critic algorithm (see below)[6].

It is important to note that the soft return in Equation (9) is different from the objective function used in standard reinforcement learning. The MaxEnt RL formulation, which incorporates an additional entropy maximization term, provides several technical advantages. These include the natural state-space exploration[72,73], a smoother optimization landscape that eases convergence towards the global optimum[74–76], and increased robustness to changes in the reward function[77,78]. In practice, many RL algorithms have gained increased stability from the additional entropy maximization[79–81]. Conceptually, MaxEnt RL can be considered equivalent to *probabilistic matching*, which has been used to explain human decision making[82,83]. Existing evidence indicates that human adults tend to apply probabilistic matching methods rather than pure maximization strategies[82,84,85]. However, these observations still lack conclusive neuroscientific explanation[80].

In order to approximate the optimal parameters θ^*, we use a policy-gradient approach, which iteratively refines the parameters θ in the direction of increasing rewards. Reinforcement learning methods that are based on fully sampled trajectories usually suffer from updates with high variance. To reduce this variance and thus accelerate the learning process, we choose an approach that includes two approximators: an *actor network* and a *critic network*. These work as follows. Given some state s_0 as input, the actor network outputs the (standardized) mean and standard deviation of as many normal distributions as dimensions of the action space. The individual action components are then sampled from these distributions. To update the actor network weights, we must measure the "desirability" of some action a, given some state s, i.e., how much reward can be expected when starting in this state with this action and subsequently following the current policy. These values are approximated by the critic network.

The architecture of both networks is depicted in Fig. 6. For the sake of a simpler notation, the parameter vector θ contains the weights of both networks, however these weights are not shared between the two. These two networks are then coupled with the *soft actor-critic (SAC)* algorithm[6], which has been used successfully in physics-based character motion[86]: As a policy-gradient method, it can be easily used with a continuous action space such as continuous motor signals – something that is not directly possible with value function methods like *DQN*[5]. As an off-policy method that makes use of a replay buffer, it is quite sample-efficient. This is important, since running forward physics simulations in MuJoCo constitutes the major part of the training duration. Moreover, it has been shown that SAC outperforms other state-of-the-art algorithms such as PPO[87] or TD3[88]. Supporting the observations in Haarnoja et al.[6], we also found our training process to be faster and more robust when using SAC rather than PPO. Moreover, SAC incorporates an automatic adaption of the temperature α using dual gradient descent, which eliminates the need for manual, task-dependent fine-tuning. In order to obtain an unbiased estimate of the optimal value function, we use *Double Q-Learning*[89], using a separate target critic network. The neural network parameters are optimized with the Adam optimizer[90].

States, actions, and scaling factors. Using the MuJoCo implementation of the biomechanical model described above, the **states** $s \in \mathcal{S} \subseteq \mathbb{R}^{48}$ in our RL approach include the following information:

- Joint angle for each DOF $q \in \mathcal{Q}$ in radians (7 values),
- Joint velocity for each DOF $q \in \mathcal{Q}$ in radians/s (7 values),
- Activations $\sigma^{(q)}$ and excitations $\dot{\sigma}^{(q)}$ for each DOF $q \in \mathcal{Q}$ (2×7 values),
- Positions of the end-effector and target sphere (2×3 values),
- (positional) Velocities of the end-effector and target sphere (2×3 values),
- (positional) Acceleration of the end-effector (3 values),
- *Difference vector*: vector between the end-effector attached to the index finger and the target, pointing towards the target (3 values),
- Projection of the end-effector velocity towards the target (1 value),

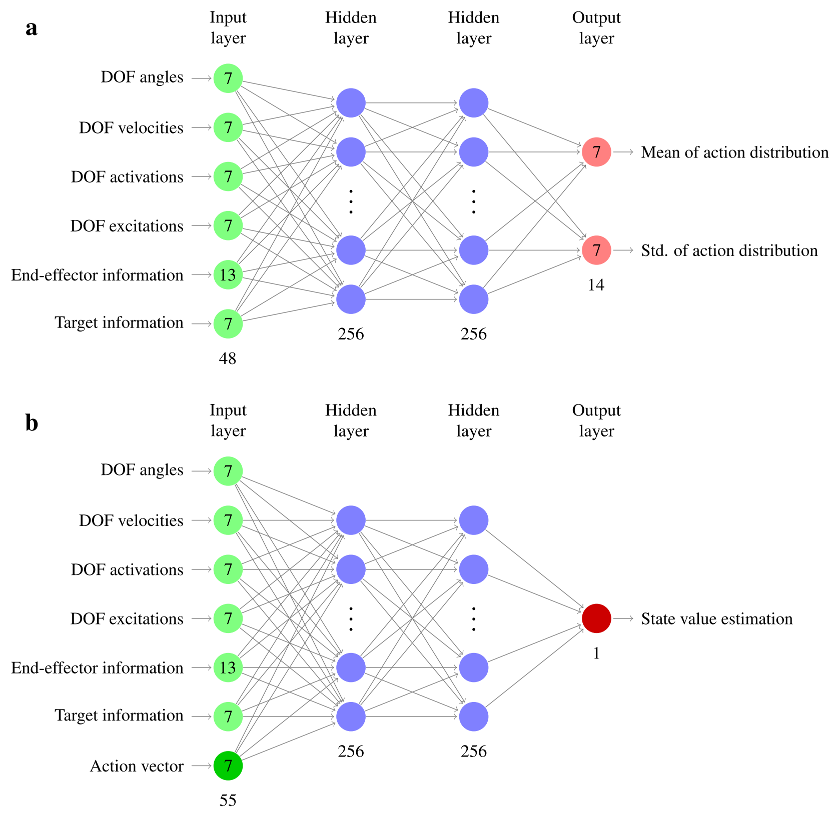


Figure 6. Neuronal network architectures. (**a**) The actor network takes a state s as input and returns the policy π_θ in terms of mean and standard deviation of the seven normal distributions, from which the components of the action vector are drawn. (**b**) The critic network takes both state s and action vector a as input and returns the estimated state-action value. Two critic networks are trained simultaneously to improve the speed and stability of learning (*Double Q-Learning*). Detailed information about the input state components are given in the *Methods* section.

- Radius of the target sphere (1 value).

We found that in our case, the target velocity (which always equals zero for the considered tasks), the end-effector acceleration, the difference vector, and the projection of the end-effector velocity can be omitted from state space without reducing the quality of the resulting policy. However, we decided to incorporate these observations, as they did not considerably slow down training and might be beneficial for more complex tasks such as target tracking or via-point tasks.

Each component $a^{(q)} \in [-1, 1]$ of the **action vector** $a = (a^{(q)})_{q\in\mathcal{Q}} \in \mathcal{A} \subseteq \mathbb{R}^7$ is used to actuate some DOF $q \in \mathcal{Q}$ by applying the torque $\tau^{(q)}$ resulting from Eqs. (4)–(6). Note that in addition to these actuated forces, additional active forces (e.g., torques applied to parent joints) and passive forces (e.g., gravitational and contact forces) act on the joints in each time step.

We determine experimentally the maximum torque a human would exert at each DOF in this task as follows. We implemented the Fitts' Law task described above in a VR environment displayed via the HTC Vive Pro VR headset. We recorded the movements of a single participant performing the task, using the Phasespace X2E motion capture system with a full-body suit provided with 14 optical markers. This study was granted ethical approval by the ethics committee of the University of Bayreuth and followed ethical standards according to the Helsinki Declaration. Written informed consent was received from the participant, which received an economic compensation for participating in the study. Using OpenSim, we scaled the *Upper Extremity Dynamic Model* to this particular person. We then used OpenSim to perform *Inverse Dynamics* to obtain the torque sequences that are most likely to produce the recorded marker trajectories. For each DOF $q \in \mathcal{Q}$, we set the corresponding **scaling factor** $g^{(q)}$ to the absolute maximum torque applied at this DOF during the experiment, omitting a small number of outliers from the set of torques, i.e., values with a distance to mean larger than 20 times the standard deviation. The resulting values are shown in Table 1.

Reward function and curriculum learning. The behavior of the policy is determined largely by the reward r_n that appears in Eq. (9). We designed the reward following Harris and Wolpert[1], who argue that there is no rational explanation as to why the central nervous system (CNS) should explicitly try to minimize previously proposed metrics such as the change in torque applied at the joints[12], or the acceleration (or jerk) of the end-effector[8]. They argue that it is not even clear whether the CNS is able to compute, store, and integrate these quantities while executing motions.

Instead, they argue that the CNS aims to minimize movement end-point variance given a fixed movement time, under the constraint of signal-dependent noise. Following Harris and Wolpert[1], t his is equivalent to minimizing movement time when the permissible end-point variance is given by the size of the target. This objective is simple and intuitively plausible, since achieving accurate aimed movements in minimal time is critical for the success of many movement tasks. Moreover, it has already been applied to linear dynamics[2].

Therefore, the objective of our model is to *minimize movement time while reaching a target of given width*.

More precisely, our **reward function** consists only of a time reward, which penalizes every time step of an episode equally:

$$r_n = -100\Delta t. \tag{10}$$

This term provides incentives to terminate the episode (which can only be achieved by reaching the target) as early as possible. Since we apply each control a_n for 10 ms, Δt amounts to 0.01 in our case, i.e., $r_n = -1$ in each time step $n \in \{0, \ldots, N\}$.

According to our experience, it is possible to learn aimed movements despite the lack of gradient provided by the reward function, provided the following requirements are met. The initial posture needs to be sampled randomly, and the targets need to be large enough at the beginning of the training to ensure that the target is reached by exploration sufficiently often in early training steps to guide the reinforcement learner. However, creating a predetermined curriculum that gradually decreases the target width during training appropriately has proved very difficult. In most cases, the task difficulty either increased too fast, leading to unnatural movements that do not reach the target directly (and often not at all), or progress was slow, resulting in a time-consuming training phase.

For this reason, we decided to use an adaptive curriculum, which adjusts the target width dynamically, depending on the recent success rate. Specifically, we define a *curriculum state*, which is initialized with an initial target diameter of 60 cm. Every 10K update steps, the current policy is evaluated on 30 complete episodes, for which target diameters are chosen, depending on the current state of the curriculum. Based on the percentage of targets reached within the permitted 1.5 s (*success rate*), the curriculum state is updated. If the success rate falls below 70%, it is increased by 1 cm; if the success rate exceeds 90%, it is decreased by 1 cm. To avoid target sizes that are larger than the initial width or are too close to zero, we clipped the resulting value to the interval $[0.1 \text{ cm}, \ 60 \text{ cm}]$.

At the beginning of each episode, the target diameter is set to the current curriculum state with probability $1 - \varepsilon$, and sampled uniformly randomly between 0.1 cm and 60 cm with probability $\varepsilon = 0.1$, which has proven to be a reasonable choice. This ensures in particular that all required target sizes occur throughout the training phase, and thus prevents forgetting how to solve "simpler" tasks (in literature, often referred to as *catastrophic forgetting*; see, e.g., McCloskey et al.[91]).

Implementation of the reinforcement learning algorithm. The actor and critic networks described in the *Reinforcement Learning* section consist of two fully connected layers with 256 neurons each, followed by the output layer, which either returns the means and standard deviations of the action distributions (for the actor network) or the state-action value (for the critic network). To improve the speed and stability of learning, we train two separate, but same-structuredidentically structured critic networks and use the minimum of both outputs as the teaching signal for all networks (*Double Q-Learning*)[6,89]. In all networks, ReLU[92] is used as non-linearity for both hidden layers. The network architectures are depicted in Fig. 6.

The reinforcement learning methods of our implementation are based on the *TF-Agents* library[93]. The learning phase consists of two parts, which are repeated alternately: *trajectory sampling* and *policy updating*.

In the trajectory sampling part, the target position is sampled from the uniform distribution on a cuboid of 70 cm height, 40 cm width, and 30 cm depth, whose center is placed 50 cm in front of the human body, and 10 cm to the right of the shoulder. The width of the target is controlled by the adaptive curriculum described above. The biomechanical model is initialized with some random posture, for which the joint angles are uniformly sampled from the convex hull of static postures that enables keeping the end-effector in one of 12 targets placed along the vertices of the cuboid described above. The initial joint velocities are uniformly sampled from the interval $[-0.005 \text{ radians/s}, \ 0.005 \text{ radians/s}]$.

In each step $n \in \{0, \ldots, N-1\}$, given the current state vector $s_n \in \mathcal{S}$ (see description above), an action is sampled from the current policy $\pi_\theta(\cdot \mid s_n)$. Next, the MuJoCo simulation uses this action to actuate the model joints. It also updates the body posture, and returns both the reward r_n and the subsequent state vector s_{n+1}. In our implementation, each episode in the learning process contains at most $N = 150$ of such steps, with each step corresponding to 10 ms (allowing movements to be longer than one and a half seconds did not improve the training procedure significantly). If the target is reached earlier, i.e., the distance between end-effector and target center is lower than the radius of the target sphere, and the end-effector remained inside the target for 100 ms, the current episode terminates and the next episode begins with a new target position and width. At the beginning of the training, 10K steps are taken and the corresponding transitions stored in a replay buffer, which has a capacity of 1M steps. During training, only one step is taken and stored per sampling phase.

In the policy updating part, 256 previously sampled transitions (s_n, a_n, r_n, s_{n+1}) are randomly chosen from the replay buffer to update both the actor network and the critic network weights. We use a discount factor of

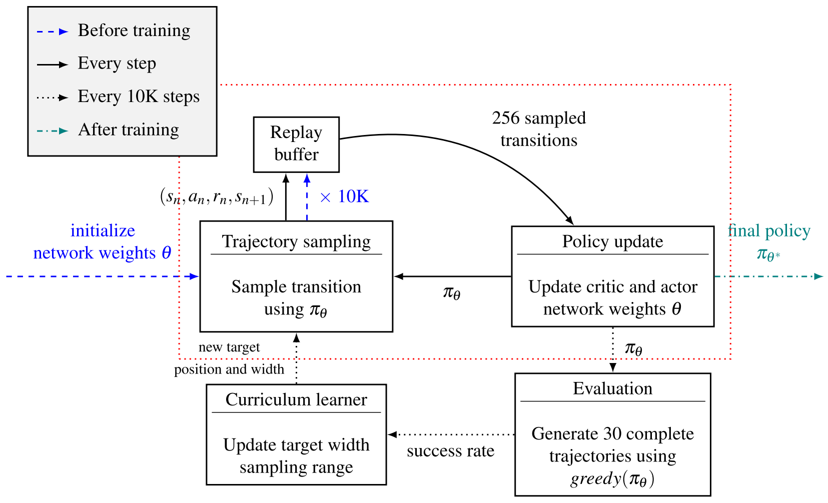

Before training
Every step
Every 10K steps
After training
Replay buffer
256 sampled transitions
(s_n, a_n, r_n, s_{n+1})
× 10K
initialize network weights θ
Trajectory sampling
Sample transition using π_θ
π_θ
Policy update
Update critic and actor network weights θ
final policy π_{θ^*}
new target position and width
π_θ
Curriculum learner
Update target width sampling range
success rate
Evaluation
Generate 30 complete trajectories using $greedy(\pi_\theta)$

Figure 7. Reinforcement learning procedure. Before training, the networks are initialized with random weights θ, and 10 K transitions are generated using the resulting initial policy. These are stored in the replay buffer (blue dashed arrows). During training (red dotted box), trajectory sampling and policy update steps are executed alternately in each step. The targets used in the trajectory sampling part are generated by the curriculum learner, which is updated every 10K steps, based on an evaluation of the most recent (greedy) policy. As soon as the target width suggested by the curriculum learner falls below 1 cm, the training phase is completed and the final policy π_{θ^*} is returned (teal dash-dotted arrow).

	Joint angle ranges (deg)		Joint torque ranges (Nm)	
Joint DOF	**Minimum**	**Maximum**	**Minimum**	**Maximum**
Elevation angle	− 90	130	− 36.01	36.01
Shoulder elevation	0	180	− 60.97	60.97
Shoulder rotation	− 90	20	− 19.37	19.37
Elbow flexion	0	130	− 12.57	12.57
Pronation/supination	− 90	90	− 1.03	1.03
Wrist deviation	− 10	25	− 2.14	2.14
Wrist flexion	− 70	70	− 1.53	1.53

Table 1. Joint ranges of individual DOFs. Angle and torque ranges of all joint DOFs, which are actuated via second-order muscle dynamics [Eq. (4)]. Moment arm scaling factors are defined as the magnitude of the torque range limits.

γ =0.99 in the critic loss function of SAC. All other parameters are set to the default values of the TF-Agents SAC implementation[93].

Both parts of our learning algorithm, the trajectory sampling and the policy update, are executed alternately until the curriculum state, i.e., the current suggested target diameter, falls below 1 cm. With our implementation, this was the case after 1.2M steps, corresponding to about four hours of training time. To evaluate a policy π_θ, we apply the action a_n^* with the highest probability under this policy for each time step (i.e., we use the corresponding *greedy* policy) and evaluate the resulting trajectory. Such an *evaluation* is done every 10K steps, for which 30 complete episodes are generated using this deterministic policy, and the resulting performance indicators are stored. After the training phase, θ^* is set to the latest parameter set θ, i.e., the final policy π_{θ^*} is chosen as the latest policy π_θ.

An overview of the complete training procedure is given in Fig. 7.

Evaluation. For an evaluation of the trajectories resulting from the learned policy for different target conditions, we designed a discrete Fitts' Law type task. This task follows the ISO 9241-9 ergonomics standard and incorporates 13 equidistant targets arranged in a circle 50 cm in front of the body and placed 10 cm right of the right shoulder (Fig. 2). As soon as a target is reached and the end-effector remains inside for 100 ms, the next target is given to the learned policy. This also happens after 1.5 s, regardless of whether or not the episode was successful.

Based on the recommendations from Guiard et al.[94], we determine different task difficulty conditions by sampling "form and scale", i.e., the *Index of Difficulty (ID)* and the distance *D* between the target centers are sampled independently, instead of using a distance-width grid. We use the *Shannon Formulation*[45] of Fitts' Law [Eq. (1)] to compute the resulting distance between the initial and target point D, given the target width W and the ID:

$$\mathrm{ID} = \log_2\left(\frac{D}{W} + 1\right). \tag{11}$$

The used combinations of distance, width, and ID can be found as Supplementary Table S1 online, and the resulting target setup is shown in Fig. 2a.

The model executes 50 movements for each task condition and each direction, i.e., 6500 movements in total. All movements reached the target and remained inside for 100 ms within the given maximum movement time of 1.5 s. Plots for all task conditions and movement directions, together with their underlying data, can be found in a public repository[95].

In addition, an adaptive "moving target" mechanism is applied to generate elliptic movements from our learned policy. During training, the policy only learned to reach a given target as fast and accurate as possible—it was never asked to follow a specific path accurately. For this reason, we make use of the following method.

Initially, we place the first target on the ellipse such that 10% of the complete curve needs to be covered clockwise within the first movement, starting at a fixed initial position (leftmost point on the ellipse). In contrast to regular pointing tasks, the target already switches as soon as the movement (or rather the projection of the movement path onto the ellipse) covers more than half of this distance. The next target is then chosen so as to again create an incentive to cover the next 10% of the elliptic curve. Thus, roughly 20 via-points in total are subsequently placed on the ellipse. As shown in Fig. 3a, this indeed leads to fairly elliptic movements.

For our evaluation, we use an ellipse with horizontal and vertical diameters of 15 cm and 6 cm (similar to the ellipse used by Harris and Wolpert[1]), with its center placed 55 cm in front, 10 cm above, and 10 cm to the right of the shoulder. The task was performed for one minute, with end-effector position, velocity, and acceleration stored every 10 ms.

Comprehensive data for all of these movements can also be found in a public repository[95].

Data availability

The datasets generated during and/or analysed during the current study are available in a public repository, https://doi.org/10.5281/zenodo.4268230.

Received: 27 November 2020; Accepted: 29 June 2021
Published online: 14 July 2021

References

1. Harris, C. M. & Wolpert, D. M. Signal-dependent noise determines motor planning. *Nature* **394**, 780–784. https://doi.org/10.1038/29528 (1998).
2. Tanaka, H., Krakauer, J. W. & Qian, N. An optimization principle for determining movement duration. *J. Neurophysiol.* **95**, 3875–3886. https://doi.org/10.1152/jn.00751.2005 (2006).
3. Saul, K. R. *et al.* Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. *Comput. Methods Biomech. Biomed. Eng.* **5842**, 1–14. https://doi.org/10.1080/10255842.2014.916698 (2014).
4. van Beers, R. J., Haggard, P. & Wolpert, D. M. The role of execution noise in movement variability. *J. Neurophysiol.* **91**, 1050–1063. https://doi.org/10.1152/jn.00652.2003 (2004).
5. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* (A Bradford Book, 2018).
6. Haarnoja, T. *et al.* Soft actor-critic algorithms and applications. arXiv:1801.01290 (2018).
7. Todorov, E., Erez, T. & Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033, https://doi.org/10.1109/IROS.2012.6386109 (2012).
8. Flash, T. & Hogan, N. The coordination of arm movements: An experimentally confirmed mathematical model. *J. Neurosci.* **5**, 1688–1703 (1985).
9. Hoff, B. & Arbib, M. A. Models of trajectory formation and temporal interaction of reach and grasp. *J. Mot. Behav.* **25**, 175–192, https://doi.org/10.1080/00222895.1993.9942048 (1993).
10. Nelson, W. L. Physical principles for economies of skilled movements. *Biol. Cybern.* **46**, 135–147. https://doi.org/10.1007/BF00339982 (1983).
11. Artstein, Z. Discrete and continuous bang-bang and facial spaces or: Look for the extreme points. *SIAM Rev.* **22**, 172–185 (1980).
12. Uno, Y., Kawato, M. & Suzuki, R. Formation and control of optimal trajectory in human multijoint arm movement—Minimum torque-change model. *Biol. Cybern.* **61**, 89–101. https://doi.org/10.1007/BF00204593 (1989).
13. Todorov, E. *Studies of Goal-Directed Movements* (Massachusetts Institute of Technology, 1998).
14. Li, W. & Todorov, E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics, (ICINCO 2004)*, vol. 1, 222–229 (2004).
15. Hoff, B. A model of duration in normal and perturbed reaching movement. *Biol. Cybern.* **71**, 481–488. https://doi.org/10.1007/BF00198466 (1994).
16. Bullock, D. & Grossberg, S. Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. *Psychol. Rev.* **95**, 49 (1988).
17. Plamondon, R. A kinematic theory of rapid human movements: Part iii. Kinetic outcomes. *Biol. Cybern.* **78**, 133–145. https://doi.org/10.1007/s004220050420 (1998).
18. Todorov, E. & Jordan, M. I. Optimal feedback control as a theory of motor coordination. *Nat. Neurosci.* **5**, 1226–1235. https://doi.org/10.1038/nn963 (2002).
19. Scott, S. Optimal feedback control and the neural basis of volitional motor control. *Nat. Rev. Neurosci.* **5**, 532–46. https://doi.org/10.1038/nrn1427 (2004).
20. Todorov, E. Optimality principles in sensorimotor control. *Nat. Neurosci.* **7**, 907–915. https://doi.org/10.1038/nn1309 (2004).
21. Shadmehr, R. & Krakauer, J. A computational neuroanatomy for motor control. *Exp. Brain Res.* **185**, 359–381 (2008).
22. Diedrichsen, J., Shadmehr, R. & Ivry, R. B. The coordination of movement: Optimal feedback control and beyond. *Trends Cognit. Sci.* **14**, 31–39. https://doi.org/10.1016/j.tics.2009.11.004 (2010).
23. Qian, N., Jiang, Y., Jiang, Z.-P. & Mazzoni, P. Movement duration, fitts's law, and an infinite-horizon optimal feedback control model for biological motor systems. *Neural Comput.*https://doi.org/10.1162/NECO_a_00410 *(2012).*

24. Shadmehr, R., De Xivry, J. J. O., Xu-Wilson, M. & Shih, T.-Y. Temporal discounting of reward and the cost of time in motor control. *J. Neurosci.* **30**, 10507–10516 (2010).
25. Berret, B. & Jean, F. Why don't we move slower? The value of time in the neural control of action. *J. Neurosci.* **36**, 1056–1070. https://doi.org/10.1523/JNEUROSCI.1921-15.2016 (2016).
26. Holzbaur, K. R., Murray, W. M. & Delp, S. L. A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. *Ann. Biomed. Eng.* **33**, 829–840 (2005).
27. Seth, A., Dong, M., Matias, R. & Delp, S. Muscle contributions to upper-extremity movement and work from a musculoskeletal model of the human shoulder. *Front. Neurorobot.* **13**, 90. https://doi.org/10.3389/fnbot.2019.00090 (2019).
28. Delp, S. L. *et al.* Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Trans. Biomed. Eng.* **54**, 1940–1950 (2007).
29. Seth, A. *et al.* Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS Comput. Biol.* **14**, 1–20. https://doi.org/10.1371/journal.pcbi.1006223 (2018).
30. Rosenbaum, D. A., Loukopoulos, L. D., Meulenbroek, R. G., Vaughan, J. & Engelbrecht, S. E. Planning reaches by evaluating stored postures. *Psychol. Rev.* **102**, 28–67. https://doi.org/10.1037/0033-295x.102.1.28 (1995).
31. Nakano, E. *et al.* Quantitative examinations of internal representations for arm trajectory planning: Minimum commanded torque change model. *J. Neurophysiol.* **81**, 2140–2155, https://doi.org/10.1152/jn.1999.81.5.2140 (1999).
32. Kawato, M. Optimization and learning in neural networks for formation and control of coordinated movement. *Attent. Perform.* 821–849 (1993).
33. Kawato, M. Trajectory formation in arm movements: Minimization principles and procedures. *Adv. Motor Learn. Control.* 225–259 (1996).
34. Cheema, N. *et al.* Predicting mid-air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, 1–13, https://doi.org/10.1145/3313831.3376701 (Association for Computing Machinery, New York, NY, USA, 2020).
35. Berret, B., Chiovetto, E., Nori, F. & Pozzo, T. Evidence for composite cost functions in arm movement planning: An inverse optimal control approach. *PLoS Comput. Biol.* **7**, 1–18. https://doi.org/10.1371/journal.pcbi.1002183 (2011).
36. Lee, S., Park, M., Lee, K. & Lee, J. Scalable muscle-actuated human simulation and control. *ACM Trans. Graph.*https://doi.org/10.1145/3306346.3322972 *(2019).*
37. Nakada, M., Zhou, T., Chen, H., Weiss, T. & Terzopoulos, D. Deep learning of biomimetic sensorimotor control for biomechanical human animation. *ACM Trans. Graph.*https://doi.org/10.1145/3197517.3201305 *(2018).*
38. Si, W., Lee, S.-H., Sifakis, E. & Terzopoulos, D. Realistic biomechanical simulation and control of human swimming. *ACM Trans. Graph.*https://doi.org/10.1145/2626346 *(2015).*
39. Fan, J., Jin, J. & Wang, Q. Humanoid muscle-skeleton robot arm design and control based on reinforcement learning. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 541–546, https://doi.org/10.1109/ICIEA48937.2020.9248350 (2020).
40. Ben-Itzhak, S. & Karniel, A. Minimum acceleration criterion with constraints implies bang–bang control as an underlying principle for optimal trajectories of arm reaching movements. *Neural Comput.* **20**, 779–812. https://doi.org/10.1162/neco.2007.12-05-077 (2008).
41. Berret, B. *et al.* The inactivation principle: Mathematical solutions minimizing the absolute work and biological implications for the planning of arm movements. *PLoS Comput. Biol.* **4**, 1–25. https://doi.org/10.1371/journal.pcbi.1000194 (2008).
42. Tieck, J. C. V. *et al.* Learning continuous muscle control for a multi-joint arm by extending proximal policy optimization with a liquid state machine. In *International Conference on Artificial Neural Networks*, 211–221 (Springer, 2018).
43. Kidziński, Ł. *et al.* Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In Escalera, S. & Weimer, M. (eds.) *The NIPS '17 Competition: Building Intelligent Systems*, 121–153 (Springer International Publishing, Cham, 2018).
44. Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *J. Exp. Psychol.* **47**, 381–391 (1954).
45. MacKenzie, I. S. A note on the information-theoretic basis for Fitts' law. *J. Mot. Behav.* **21**, 323–330. https://doi.org/10.1080/00222895.1989.10735486 (1989).
46. Crossman, E. R. F. W. & Goodeve, P. J. Feedback control of hand-movement and Fitts' law. *Q. J. Exp. Psychol.* **35**, 251–278 (1983).
47. Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E. & Keith Smith, J. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychol. Rev.* **95**, 340 (1988).
48. Matthews, P. Relationship of firing intervals of human motor units to the trajectory of post-spike after-hyperpolarization and synaptic noise. *J. Physiol.* **492**, 597–628 (1996).
49. Takeda, M. *et al.* Explanation of Fitts-law in reaching movement based on human arm dynamics. *Sci. Rep.* **9**, 19804. https://doi.org/10.1038/s41598-019-56016-7 (2019).
50. Lacquaniti, F., Terzuolo, C. & Viviani, P. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychol.* **54**, 115–130 (1983).
51. Pollick, F. E. & Sapiro, G. Constant affine velocity predicts the 13 power law of planar motion perception and generation. *Vis. Res.* **37**, 347–353. https://doi.org/10.1016/S0042-6989(96)00116-2 (1997).
52. Viviani, P. & Schneider, R. A developmental study of the relationship between geometry and kinematics in drawing movements. *J. Exp. Psychol. Hum. Percept. Perform.* **17**(1), 198–218 (1991).
53. de'Sperati, C. & Viviani, P. The relationship between curvature and velocity in two-dimensional smooth pursuit eye movements. *J. Neurosci.* **17**, 3932–3945 (1997).
54. Viviani, P. & Mounoud, P. Perceptuomotor compatibility in pursuit tracking of two-dimensional movements. *J. Mot. Behav.* **22**, 407–443. https://doi.org/10.1080/00222895.1990.10735521 (1990).
55. Viviani, P., Baud-Bovy, G. & Redolfi, M. Perceiving and tracking kinesthetic stimuli: Further evidence of motor-perceptual interactions. *J. Exp. Psychol. Hum. Percept. Perform.* **23**, 1232–1252. https://doi.org/10.1037//0096-1523.23.4.1232 (1997).
56. Hicheur, H., Vieilledent, S., Richardson, M., Flash, T. & Berthoz, A. Velocity and curvature in human locomotion along complex curved paths: A comparison with hand movements. *Exp. Brain Res.* **162**, 145–54. https://doi.org/10.1007/s00221-004-2122-8 (2005).
57. Schaal, S. & Sternad, D. Origins and violations of the 2/3 power law in rhythmic 3d movements. *Exp. Brain Res.* **136**, 60–72 (2001).
58. Karklinsky, M. & Flash, T. Timing of continuous motor imagery: The two-thirds power law originates in trajectory planning. *J. Neurophysiol.* **113**, 2490–2499. https://doi.org/10.1152/jn.00421.2014 (2015).
59. Todorov, E. & Jordan, M. I. Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *J. Neurophysiol.* **80**, 696–714 (1998).
60. Gribble, P. & Ostry, D. Origins of the power law relation between movement velocity and curvature: Modeling the effects of muscle mechanics and limb dynamics. *J. Neurophysiol.* **76**, 2853–2860. https://doi.org/10.1152/jn.1996.76.5.2853 (1996).
61. Schwartz, A. Direct cortical representation of drawing. *Science* **265**, 540–542. https://doi.org/10.1126/science.8036499 (1994).
62. Flash, T. & Handzel, A. Affine differential geometry analysis of human arm movements. *Biol. Cybern.* **96**, 577–601. https://doi.org/10.1007/s00422-007-0145-5 (2007).
63. Morasso, P. Spatial control of arm movements. *Exp. Brain Res.* **42**, 223–227 (1981).

64. Abend, W., Bizzi, E. & Morasso, P. Human arm trajectory formation. *Brain J. Neurol.* **105**, 331–348 (1982).
65. Cohen, J. *Statistical Power Analysis for the Behavioral Sciences* (Academic Press, 2013).
66. Hausknecht, M. & Stone, P. Deep Recurrent q-learning for Partially Observable MDPS. arXiv:1507.06527 (2015).
67. Liu, J., Gu, X. & Liu, S. Reinforcement learning with world model. *Adapt. Learn. Optim.* **1908**, 11494 (2020).
68. Igl, M., Zintgraf, L., Le, T. A., Wood, F. & Whiteson, S. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, 2117–2126 (PMLR, 2018).
69. Jiang, Y., Van Wouwe, T., De Groote, F. & Liu, C. K. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Trans Graph (TOG)* **38**, 1–12 (2019).
70. Ikkala, A. & Hämäläinen, P. Converting biomechanical models from opensim to Mujoco. arXiv:2006.10618 (2020).
71. van der Helm, F. C. T. & Rozendaal, L. A. Musculoskeletal systems with intrinsic and proprioceptive feedback. in *Biomechanics and Neural Control of Posture and Movement* (eds Winters, J. M. & Crago, P. E.) 164–174 (Springer New York, NY, 2000).
72. Mnih, V. *et al.* Asynchronous methods for deep reinforcement learning. arXiv:1602.01783 (2016).
73. Eysenbach, B., Gupta, A., Ibarz, J. & Levine, S. Diversity is all you need: learning skills without a reward function. arXiv:1802.06070 (2018).
74. Ahmed, Z., Le Roux, N., Norouzi, M. & Schuurmans, D. Understanding the impact of entropy on policy optimization. In Chaudhuri, K. & Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, 151–160 (PMLR, 2019).
75. Fox, R., Pakman, A. & Tishby, N. Taming the noise in reinforcement learning via soft updates. arXiv:1512.08562 (2017).
76. Vieillard, N. *et al.* Leverage the average: an analysis of KL regularization in RL. arXiv:2003.14089 (2021).
77. Eysenbach, B. & Levine, S. If maxent RL is the answer, what is the question? arXiv:1910.01913 (2019).
78. Eysenbach, B. & Levine, S. Maximum entropy RL (provably) solves some robust RL problems. arXiv:2103.06257 (2021).
79. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. & Krause, A. (eds.) *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, 1861–1870 (PMLR, 2018).
80. Abdolmaleki, A. *et al.* Maximum a posteriori policy optimisation. arXiv:1806.06920 (2018).
81. Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. arXiv:1805.00909 (2018).
82. Vulkan, N. An economist's perspective on probability matching. *J. Econ. Surv.* **14**, 101–118. https://doi.org/10.1111/1467-6419.00106 (2000).
83. Grünwald, P. D. & Dawid, A. P. Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory. *Ann. Stat.* **32**, 1367–1433. https://doi.org/10.1214/009053604000000553 (2004).
84. Weir, M. W. Developmental changes in problem-solving strategies. *Psychol. Rev.* **71**, 473 (1964).
85. Gallistel, C. R. *The Organization of Learning* (The MIT Press, 1990).
86. Peng, X. B., Abbeel, P., Levine, S. & van de Panne, M. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph* **37**, 1–14. https://doi.org/10.1145/3197517.3201311 (2018).
87. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. arXiv:1707.06347 (2017).
88. Fujimoto, S., van Hoof, H. & Meger, D. Addressing function approximation error in Actor-critic methods. arXiv:1802.09477 (2018).
89. Hasselt, H. V. Double q-learning. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S. & Culotta, A. (eds.) *Advances in Neural Information Processing Systems 23*, 2613–2621 (Curran Associates, Inc., 2010).
90. Kingma, D. P. & Ba, J. A. A method for stochastic optimization. arXiv:1412.6980 (2014).
91. McCloskey, M. & Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In Bower, G. H. (ed.) *Psychology of Learning and Motivation*, vol. 24, 109 – 165, https://doi.org/10.1016/S0079-7421(08)60536-8 (Academic Press, 1989).
92. Nair, V. & Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, 807–814 (Omnipress, Madison, WI, USA, 2010).
93. Guadarrama, S. *et al.* TF-Agents: A library for reinforcement learning in tensorflow. https://github.com/tensorflow/agents (2018).
94. Guiard, Y. The problem of consistency in the design of Fitts' law experiments: Consider either target distance and width or movement form and scale. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, 1809–1818, https://doi.org/10.1145/1518701.1518980 (Association for Computing Machinery, New York, NY, USA, 2009).
95. Fischer, F., Bachinski, M., Klar, M., Fleig, A. & Müller, J. Reinforcement learning control of a biomechanical model of the upper extremity (dataset). *Zenodo*. https://doi.org/10.5281/zenodo.

Acknowledgements
We would like to thank Aldo A. Faisal (Imperial College London) for his very helpful advice and comments on the manuscript.

Author contributions
F.F. and J.M. wrote the main manuscript text; M.B. adjusted the biomechanical model; F.F., M.B., and M.K. worked on the MuJoCo implementation and collected simulation data; F.F., M.K., and A.F. analyzed and validated the results; F.F. created the plots; F.F. conducted and J.M. supervised the research process. All authors reviewed the manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-021-93760-1.

Correspondence and requests for materials should be addressed to F.F.

7 Breathing Life Into Biomechanical User Models

Authors: Aleksi Ikkala, Florian Fischer, Markus Klar, Miroslav Bachinski, Arthur Fleig, Andrew Howes, Perttu Hämäläinen, Jörg Müller, Roderick Murray-Smith, Antti Oulasvirta
Status: Published in UIST ’22: Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology [29]

AI conducted and AO supervised the research process, with guidance from AF, AH, PH, JM, RMS. The concept and framework was developed by all authors. AI implemented the toolkit, with help from FF and MK. The biomechanical model was transferred from OpenSim to MuJoCo by AI, with improvements made by MB, FF, and MK. The RL agents were trained by AI and evaluated by AI, FF, MK, and AF. Figures were created by FF, AI, and MB. The results were discussed by all authors. All authors wrote and revised the manuscript. AI is the corresponding author.

Breathing Life Into Biomechanical User Models

Aleksi Ikkala
Aalto University
Finland

Florian Fischer
University of Bayreuth
Germany

Markus Klar
University of Bayreuth
Germany

Miroslav Bachinski[*]
University of Bayreuth
Germany

Arthur Fleig
University of Bayreuth
Germany

Andrew Howes
University of Birmingham
United Kingdom

Perttu Hämäläinen
Aalto University
Finland

Jörg Müller
University of Bayreuth
Germany

Roderick Murray-Smith
University of Glasgow
Scotland

Antti Oulasvirta
Aalto University
Finland

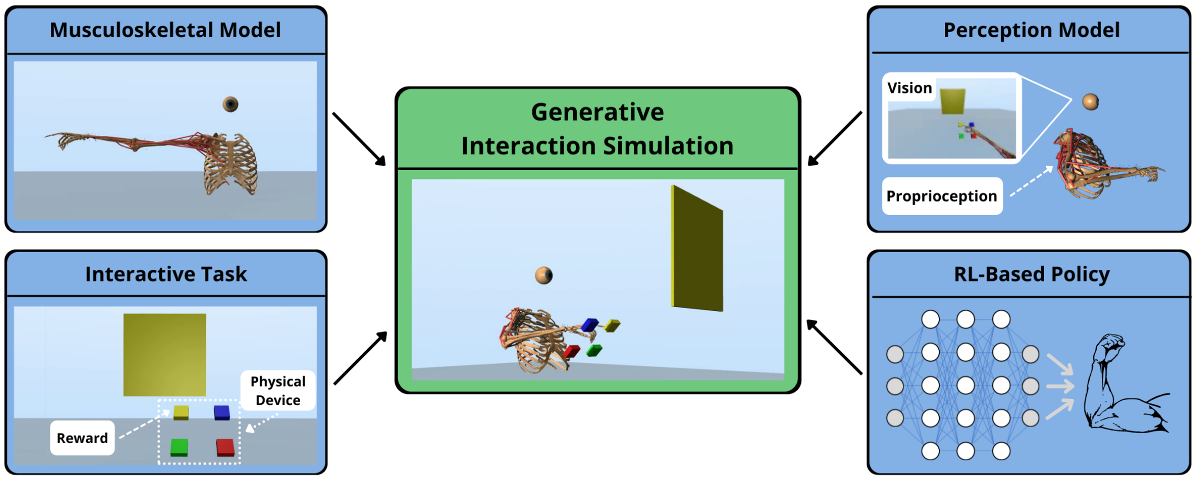


Figure 1: We present an approach for generative simulation of interaction with perceptually controlled biomechanical models interacting with physical devices. The users are modelled with a combination of muscle-actuated biomechanical models and perception models, and we use deep reinforcement learning to learn control policies by maximizing task-specific rewards. As a showcase, we apply a state-of-the-art upper body model to four HCI tasks of increasing difficulty: pointing, tracking, choice reaction, and parking a remote control car via joystick.

ABSTRACT

Forward biomechanical simulation in HCI holds great promise as a tool for evaluation, design, and engineering of user interfaces. Although reinforcement learning (RL) has been used to simulate biomechanics in interaction, prior work has relied on unrealistic assumptions about the control problem involved, which limits the plausibility of emerging policies. These assumptions include direct torque actuation as opposed to muscle-based control; direct, privileged access to the external environment, instead of imperfect sensory observations; and lack of interaction with physical input devices. In this paper, we present a new approach for learning muscle-actuated control policies based on perceptual feedback in interaction tasks with physical input devices. This allows modelling of more realistic interaction tasks with cognitively plausible visuomotor control. We show that our simulated user model successfully learns a variety of tasks representing different interaction methods, and that the model exhibits characteristic movement regularities observed in studies of pointing. We provide an open-source implementation which can be extended with further biomechanical models, perception models, and interactive environments.

[*]Also with University of Bergen.

UIST '22, October 29-November 2, 2022, Bend, OR, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9320-1/22/10.
https://doi.org/10.1145/3526113.3545689

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **User models**; *Pointing*; **Systems and tools for interaction design**.

KEYWORDS

biomechanical models, simulation models, deep reinforcement learning

ACM Reference Format:
Aleksi Ikkala, Florian Fischer, Markus Klar, Miroslav Bachinski, Arthur Fleig, Andrew Howes, Perttu Hämäläinen, Jörg Müller, Roderick Murray-Smith, and Antti Oulasvirta. 2022. Breathing Life Into Biomechanical User Models. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29-November 2, 2022, Bend, OR, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3526113.3545689

1 INTRODUCTION

Biomechanical models studied in HCI can be compared to "crash test dummies" [4, 5]. They are inactive agents, with each movement following a predefined motion plan; they do not have agency, and they cannot close the loop and explore their environments themselves. To simulate an interaction, a researcher may either manually define a set of actions, or collect motion capture data that is fit to the model using inverse simulation. What is missing is a *generative* form of simulation that would learn *realistic interaction policies* on its own through *forward simulations* in a given interactive task and without requiring prespecified motions. The ability to build a generative model that matches user behaviour is a strong test of whether we understand an interactive system. "Breathing life" into crash test dummies would open the door to wider use of these models, such as quickly evaluating prototypes before committing to an expensive experimental study. It would permit us to rapidly explore a more diverse set of user behaviours than would be feasible in experimental studies on humans, and such models could also enable parametric optimization of user interfaces. Biomechanical modeling is highly relevant for HCI research, because, with the exception of BCIs, all interfaces require physical effort from the user. Among others, physical effort is a critical consideration in the design of AR/VR interfaces, interactive surfaces, and haptic and tangible interfaces. Until recently, evaluating physical effort required running an empirical study (e.g., using NASA-TLX). However, with the advent of open access models, HCI research has recently turned to biomechanical simulations for studying and modeling interfaces as well as for developing novel interaction techniques (e.g., [4, 5, 10, 17, 29, 48]).

What has been missing is an integrative approach that would allow using reinforcement learning (RL) to learn human-like interaction policies in a designer-specified interactive task, where the task components (goals, user model, physical environment) can by flexibly changed (Figure 1). Instead, the approaches developed so far in HCI have been limited to a particular task (e.g., pointing) or a discrete combination of primitive tasks, to a particular model (e.g., upper body model), or to a particular physical environment. Moreover, with the exceptions that we discuss below, all biomechanical user models prior to this work have relied on unrealistic assumptions regarding their force actuation, ability to observe their environment, and interaction with input devices, all of which limit the realism of emerging motion patterns. First, those models operate with torque actuated joints, second, they operate without adequately perceiving their surroundings, and third, they avoid simulation of physical interaction with input devices by investigating contact-free interfaces, such as mid-air pointing (e.g., [10]). Torque actuation can be problematic because it allows for movements that are not achievable with muscles, and RL approaches will exploit these more efficient, but unrealistic, movements. Lack of visual perception can be problematic as it allows agents to, e.g., exactly know the position of a target even if it is occluded or outside the field of view. Without simulating physical interactions with input devices, the majority of interactions with computers cannot be simulated, as pure mid-air pointing is still a niche interaction technique. What is needed is a muscle-actuated user model that is able to receive and process perceptual observations of its surroundings – including vision, audition, haptics, proprioception, and so on – and physically interact with input devices. This coincides with the emerging RL challenge of learning control policies via perceptual inputs [43]. By combining perception models, musculoskeletal models, and physically simulated input devices, we can train agents to model and simulate intricate interaction tasks, such as those requiring visuomotor control. A good example of such control is presented in [51], where Nakada et al. introduced a virtual human model and used deep learning to learn reaching and tracking tasks. However, with more powerful physical simulation software, such as MuJoCo [73], we can simulate interaction steps quickly enough to use RL for more flexible problem formalisation and to allow a researcher to guide an agent's learning through reward functions.

The main contribution of this paper is twofold: 1) We integrate perception-based control and muscle-actuated biomechanical models that interact with physical input devices, and 2) we show how our RL-based approach can be leveraged to increase the scope of interactive tasks that can be simulated. The tasks present a variety of different user interfaces that require visuomotor movement control, and a user model is trained to interact with the environment through RL by rewarding desired behaviour. We show that the simulated user successfully learns to complete these interaction tasks, and the simulated movements exhibit human-like movement regularities such as Fitts' Law. This work is publicly available at https://github.com/aikkala/user-in-the-box as an extendable codebase. This implementation allows researchers to model interactive settings flexibly by changing assumptions about the musculoskeletal model, the user's task, and the perception models. It could be used in the future to e.g. aid in developing and evaluating user interfaces.

2 RELATED WORK

2.1 Biomechanical Modelling and Simulation

Biomechanical models and computer-based simulations were introduced more than four decades ago [2]. However, for a long time they were oversimplified and limited to computation of mechanical loads in static postures or using simple link-segment models [76]. Increases in computational power in the last two decades have enabled more physiologically-accurate musculoskeletal models [14, 15]. A sizable collection of such models exists now, many of which are publicly available.[1] Depending on the use case, one might opt for a model that describes the functionality of a single limb [55, 63], or go for a more comprehensive full-body model [22, 59, 74]. In these models, the force generation mechanism may rely on motors actuating individual joints, or in a more realistic use case, on muscle-tendon units [23, 45].

The main use of biomechanical models at the moment is via inverse biomechanical simulation using the OpenSim ecosystem [65]. The inverse simulation methods, namely inverse kinematics, inverse dynamics, and static optimisation or computed muscle control, allow the estimation of mechanical loads within the human musculoskeletal system, and neural controls of the muscles given motion tracking data of a given user's movement as input [15]. Such estimated variables are increasingly valuable and important in the areas of medicine, rehabilitation, and sports. On the other hand, there exists a stream of computer graphics research on biomechanical simulation and control that emphasizes visual fidelity and simulation speed rather than validation for purposes such as medical or ergonomics research [37, 38, 51, 62, 67, 69]. Forward simulation methods were also developed within the OpenSim software, however, besides their use as a component of computed muscle control, they were rarely used as standalone. They require muscle controls as inputs, which are extremely complex to measure experimentally for all required muscles, and are typically used with controls computed by inverse simulation. Standalone forward simulation has only become more useful when applied in combination with computational controllers [16, 36].

Controlling the force output of a model's actuators in forward simulations to perform a desired movement is a difficult optimisation problem. In order to find the appropriate control signals one must be able to run simulations quickly — which is often infeasible for complex models with possibly dozens of degrees-of-freedoms and a high number of (muscle) actuators. This is especially problematic for RL approaches, where finding good solutions often require millions of simulation steps. This introduces a challenge for biomechanical simulation software, which typically has not been designed for such use cases. In our work, we convert models validated by biomechanics researchers into a faster simulator [28]. An alternative would be to use a simplified simulation model and apply machine learning to predict the omitted details such as state-dependent joint actuation torque limits and muscle-based energy expenditure [30]; however, this requires generating training data using a realistic simulator, and the learned prediction model is inherently less accurate and general than the simulator itself. Previous work has also built models themselves for a faster simulator [37]. The approaches to solve this optimisation problem range from well-understood classical optimal control methods (see Section 2.2) to cutting-edge methods such as deep RL (see Section 2.3).

2.2 Classical Optimal Control Methods for HCI

Mathematically, reinforcement learning (RL) can be interpreted as a method to solve optimal control problems. Optimal control is the optimization of a cost or objective function subject to some system dynamics, such as the biomechanical model of a human and the dynamics of an interactive system. The parameter to optimize is called the control or input signal and is usually a function of time, such as muscle excitations [68]. In addition to RL, human-computer interaction has also been interpreted and simulated with classical (optimal) control methods [50]. If the cost function is quadratic and the system dynamics are linear, e.g., as in [19, Ch. 7], then the go-to method to compute the optimal control is via the linear-quadratic regulator (LQR). (Gaussian) noise, e.g., in the observation or control of the system [19, Ch. 8], can be handled by the linear-quadratic-Gaussian (LQG) regulator, an extension of LQR. More recently, event-driven Intermittent Control (IC) has been introduced as a framework to better explain relevant aspects of human movement [44]. For other cost functions and nonlinear system dynamics in particular (e.g., a state-of-the-art biomechanical model), a viable approach is to use Model Predictive Control (MPC) [33, 58]. A major difference of these approaches to the RL-based approach of our paper is that RL computes an entire policy, which maps arbitrary observations to actions and can be used to generate approximately optimal movements quickly, while classical optimal control methods compute individual optimal movements.

2.3 Reinforcement Learning -Based User Modelling

RL algorithms solve sequential decision making problems where at every timestep, an agent observes the current state, takes an action, and receives an action- and state-dependent scalar reward. The goal is to select actions that maximize expected utility defined as the sum of future rewards. RL provides a suitable framework for modelling human behaviour in a flexible way: one only needs to define the states, actions, and rewards and then RL computes the optimal policy [12, 13]. In some cases, the reward function and other parameters can be inferred from human data [6, 32]. A more detailed RL problem formulation along with our definitions for states, actions, and rewards are discussed in Section 3.1. When the reward function and state–action space, including their key limitations, are similar to a human's, increasingly human-like behavior has been shown to emerge through learning [52]. Applications in HCI include models of typing, menu selection, multitasking, and visual decision-making [52]. However, no application in HCI so far has looked at perceptual control of a biomechanical model.

Using the assumptions of signal-dependent control noise and movement time minimization, Fischer et al. [18] have shown that an RL agent can learn to generate human-like movements with a torque-actuated state-of-the-art model of the upper extremity. The generated movements were in accordance with well-established phenomena such as Fitts' Law [20] and the Two-Thirds Power Law [35]. RL-based simulation may also provide valuable information for predicting usability- and ergonomics-related criteria and to aid in interface design. For instance, Cheema et al. trained a (simplified) torque-actuated biomechanical arm model in a mid-air pointing task, and used the model to predict fatigue of real human subjects performing the task [10]. Leino et al. [39] used RL to learn policies for keystroke-level models, and used this to optimize button arrangements. In addition to learning control policies for embodied agents, RL can be used to model users performing

[1]For instance, OpenSim models https://simtk-confluence.stanford.edu:8443/display/OpenSim/Musculoskeletal+Models

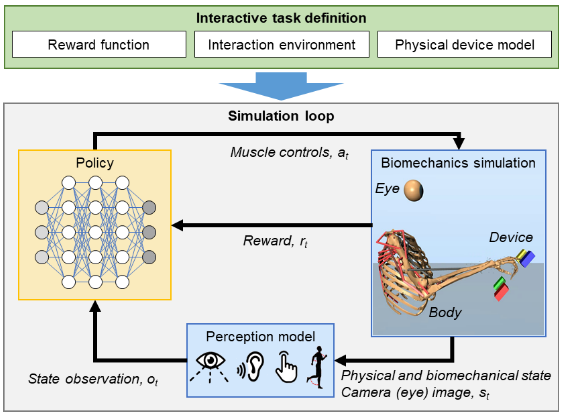


Figure 2: Our approach: The researcher specifies properties of an interactivate task (green box), including the reward function that guides the simulated user's learning process, the interaction environment, and the physical devices the user interacts with. The simulated user is defined through biomechanical and perception models, and is controlled by an RL policy based on observations from the perception models.

interactive tasks with symbolic actions. For instance, in [40, 66] RL agents interact with websites using vision and low-level mouse and keyboard actions. However, these methods rely on human demonstrations to learn policies. Recently, game companies have started applying RL-based user modelling in simulation-based game testing [34, 60, 61].

Advances in deep learning during the past decade have made it possible to scale up RL-based models. Deep RL has been used to learn control policies in increasingly complex state–action spaces — such as torque-actuated humanoids [7, 18, 53, 77] and musculoskeletal systems [37, 51], as well as eye-hand coordination in typing [31]. Hetzel et al. [25] presented an RL agent for simulating joint-controlled movement of hands in typing, however lamenting that while muscle control would have been preferable, they were not able to train a model that had muscles. Moveover, their control problem was not perceptual like ours; their agent state was a vector describing joint kinematics and the position of next target key. Nakada et al. [51] demonstrated how deep artificial neural networks (ANNs) can be leveraged to enable visuomotor control of biomechanical simulation for tasks like target tracking. However, rather than RL, they used a supervised learning approach that utilised a task- and stimuli-specific training process that is not suitable for flexibly modelling a variety of interaction tasks.

3 OVERVIEW OF APPROACH

We present User-in-the-Box (UitB), an extendable open-source implementation for biomechanical user modelling in interactive tasks in the MuJoCo [73] simulator.[2] UitB enables flexible modelling of muscle-actuated, perceptually controlled biomechanical user models with deep RL. It allows flexibly changing the physical model of the interactive device via 3D models that can be implemented in MuJoCo, or imported from another modelling software, e.g. Unity (Figure 2). The implementation can be extended with additional biomechanical models, for instance, by converting them from OpenSim [28], perception models, and interactive tasks. The user model learns to interact with the environment through a task-specific reward function.

To use UitB, one begins by defining an interactive task. This includes defining the reward function, which guides the agent's learning process, the interaction environment, and the physical devices that the user interacts with. Then, one creates the user model by choosing existing biomechanics and perception models, or implementing new ones. Models of perception can be formed by defining transducing functions from e.g. the output of an egocentrically placed camera, or other sensors available in the MuJoCo simulator. Once the interactive task and the user model are defined, the RL agent is trained using deep RL, and the simulated user's performance can be evaluated.[3] This approach forces the simulated user to adhere to important assumptions about low-level perception and movement characteristics inherent in human physiology while retaining enough flexibility to adapt the user model, through learning, to different interactive tasks. Learning control policies with RL requires an efficient forward physics simulator; in this work we use MuJoCo as a simulation environment, but ultimately this approach is simulator-agnostic.

In this section, we will first frame the problem of user modelling in interactive tasks as an RL problem and then discuss what aspects one should consider when creating an interactive task or a user model.

3.1 Modelling an Interactive Task with RL

The computational core of our framework is reinforcement learning. To utilize RL, an interactive task needs to be modelled as a Markov decision process (MDP) [70], or more generally, a partially observable MDP (POMDP). As illustrated in Figure 2, this means that at every timestep t, the agent takes an action a_t based on a state observation o_t. This results in receiving a scalar reward r_t and a new observation o_{t+1}, from which a new action a_{t+1} is performed.

We utilize a discounted RL objective with a stochastic policy and episodic learning. This means that we optimize a policy $\pi(a_t|o_t)$ to maximize the expected return $\mathbb{E}\left[\sum_{t=0}^{T}\gamma^t r_t\right]$, where the actions are sampled from the policy, $a_t \sim \pi(a_t|o_t)$. The discount factor $\gamma \in [0, 1)$ controls how much earlier rewards are preferred to distant rewards. Learning progresses through episodes, where the simulation is returned to an initial state at $t = 0$ and actions are simulated up to the time limit T.

3.1.1 Reward Function. In UitB, a key modelling aim is to select a reward function that represents whatever the user tries to achieve and values in interaction. For instance, in a pointing task, human subjects would be asked to point to a target; in an RL setting, this can be implemented by rewarding behaviour where the fingertip is brought on top of the target in minimum time and with minimum

[2]available at https://github.com/aikkala/user-in-the-box

[3]We use term *simulated user* to refer to the performance of the user model during simulation, and term *agent* — originating from the AI and RL literature — when referring to RL training or evaluation.

effort. The effort term is denoted as δ_t, and it is assumed to be a part of the reward r_t the user receives. UITB allows flexibly defining the reward function by reference to measures available in the simulation. The reward function is task-specific, and while designing it can be non-trivial, the reward functions of our four simulation tasks presented in Section 4 should provide a useful starting point.

3.1.2 Observations. To allow multisensory perception, we define our observations as a tuple $o_t = (\Omega_t, \xi_t)$, where $\Omega_t = (\omega_t^1, \omega_t^2, ...)$ is a tuple of outputs from different perception models, such as vision or proprioception, with ω_t^i being the output of i:th perception model. For the sake of clarity, we will refer to the outputs of proprioception and vision models, which are used in all of the tasks, as ω_t^P and ω_t^V, respectively. In some of the tasks we also model tactile feedback through force sensors: this tactile perception is denoted by ω_t^T. Furthermore, we introduce a stateful information observation ξ_t as a part of o_t. This quantity may contain information related to the task, e.g., how much time is left until an episode terminates. In tasks where the agent needs to, for instance, infer movement direction we may also include perception model outputs from previous timesteps in the observation, such that $o_t = (\Omega_t, \xi_t, \Omega_{t-1}, \xi_{t-1}, ..., \Omega_{t-k}, \xi_{t-k})$, where k denotes how many previous observations are included.

3.1.3 Actions. Our neural network policy does not directly output a vector of muscle controls a_t. Instead, we use *relative muscle control*, sampling relative action vectors a'_t from a Gaussian policy as

$$a'_t \sim \pi(a'_t|o_t) = \mathcal{N}(\mu_\theta(o_t), \text{diag}(\sigma^2)), \quad (1)$$

where θ are the parameters of the policy neural network, and μ and σ^2 are mean and variance vectors, respectively. The σ^2 controls the exploration/exploitation tradeoff.[4] The final muscle controls are computed as

$$a_t = clip_0^1(m_{t-1} + clip_{-1}^1(a'_t)), \quad (2)$$

where m_{t-1} are the model's internal muscle activation states for the previous step, which are included in the proprioceptive observations ω^P, and $clip_x^y$ denotes a clipping operation to range $[x, y]$. Essentially, the policy is controlling whether muscle excitation for the next step should be higher or lower than internal muscle activation in the previous step, i.e., whether the agent should increase or decrease force output of a specific muscle actuator. In particular, applying zero control results in constant internal muscle activation, which lets the body converge towards a "steady-state" posture. According to our experience, this type of control leads to a significant speed-up in the training of policies for muscle-actuated models. This is in contrast to using an *absolute muscle control*, where a policy would output the muscle excitation signals directly in range [0, 1].

3.1.4 Algorithm Choice. Depending on the POMDP formulation, one typically has multiple alternative RL algorithms to choose from. In this paper, we utilize the Stable Baselines 3 library's [57] implementation of Proximal Policy Optimization (PPO [64]), which works for both discrete and continuous states and actions, and is the most popular RL algorithm in both recent HCI works [10, 31] and high-quality human movement control papers in the computer animation literature [7, 53, 77]. Note that as per Stable Baselines 3 defaults, our standard deviations σ are optimized together with the policy network parameters θ, starting from a high value to allow thorough initial exploration of the state and action spaces.

The majority of RL algorithms, including PPO, are designed for fully observable MDPs, where the observations o_t are replaced by states s_t. The assumption is that s_t contains all the information for choosing the optimal actions. In many real-world tasks, this is not the case, but standard RL methods can still be applied, either by using a recurrent policy network [75] that can learn to infer the true state from a sequence of observations, or engineering each observation to include enough information for the inference. For instance, if a game-playing agent only observes a single frame of pixels at a time, the observation is only informative of object positions; inferring velocities can be enabled by concatenating two or more consecutive frames as the observation [47].

3.2 Interactive Task Definition

The interactive task definition includes the reward function, and models of the environment and related interaction devices.

Even when not modeling a human, the reward function is often the most difficult part of an RL problem to specify. When modeling a human, the reward function must not only be cognitively plausible but it must also facilitate efficient learning. In general interaction tasks are easier to formulate with sparse rewards, but this makes it more difficult to solve the RL optimisation problem. For instance, one could give reward only when an interaction is completed satisfactorily, like once an agent has put its fingertip inside a target in a pointing task. In practice, however, the RL problem is often made easier by using reward shaping (e.g. reward is a function of distance between fingertip and target in a pointing task), early termination (terminate an episode early if the agent is in some sense moving further from a goal), or curriculum learning (start the learning process with a simplified version of the problem).

The interaction environment is defined as a MuJoCo model that contains one or multiple interactive devices. The physical devices can be modelled as a set of MuJoCo primitives, or one could import a 3D model of a device into MuJoCo as a triangulated mesh. The device may contain physically moving parts, which need to be modelled with joints, or it may include dynamically changing content, like the color or size of an interaction device. However, some aspects of interaction may be difficult — though not impossible — to model in MuJoCo, such as the exact type of friction between contacts, or a touch screen with dynamic content.

3.3 User Model

With *user model* we refer to the combination of a biomechanical model, any set of perception models, and a control policy; all the components that are required to model and simulate a user.

The user model can be flexibly defined based on how the simulated user needs to interact with its environment. For instance, when simulating a mid-air pointing task, one mainly needs to model the movement of arm and shoulder, and vision system. The perception models can be implemented depending on the level of realism required in the modelling. For instance, the vision system could be modelled with one RGB-D camera, or two RGB cameras with overlapping fields of view. The RL policy represents a cognitive

[4] In the future, it would be interesting to add signal-dependent control noise [24] as an additional source of variance.

model that receives perceptions from the environment, and decides how to control the muscle actuators of the biomechanical model.

The perception models receive full and perfect knowledge of the simulation's physical state, biomechanical state included. The role of these models is to bound information that cannot be expected to be available to a user, and hence the user receives a transformed observation of the environment. For example, in a pointing task the user would not know the exact coordinates where a target is located, but instead has to infer the target location from visual observations. Furthermore, humans' perceptions of the world are rarely perfect; this noise modelling could be included in the perception models. However, the perception models used in our simulations are relatively simple and noise-free. One could extend the user model with more intricate perception models, for instance, e.g., by implementing foveal and peripheral vision and adding eye movements.

The user model may also include an effort term or other types of fatigue modelling to drive the agent behave in a more human-like fashion. As mentioned in Section 3.1.1, the effort term is a part of the reward the agent receives by interacting with the environment. The exact form of a suitable effort term is not known, and, as Berret et al. [8] showed in an arm movement modelling setting, the most appropriate cost function may be a combination of several different functions. In our simulations we use a neural effort term [8], which we introduce in Section 4.1. However, the UitB implementation allows these to be easily changed to investigate the effects of different effort terms.

4 SIMULATION STUDIES

In the following subsections, we show applications in a diverse and challenging set of interaction tasks (Figure 3). We first provide details of the biomechanical model used to simulate movement dynamics, and the perception models that allow the simulated user to observe its environment. Then we describe the interaction environment for the standard HCI task of mid-air pointing and analyse the simulated movements. We show that the simulated pointing movement complies with predictive models of human movement such as Fitts' Law and the Minimum Jerk model to a sufficient degree for this approach to be a valuable tool in evaluating behaviour in interactive tasks. Finally, we show that our simulated user successfully learns to perform three additional HCI tasks of varying difficulty: target tracking, choice reaction, and parking a remote control car via joystick.

4.1 Model Implementation and Training

We use *MoBL ARMS model* [63], a state-of-the-art muscle-actuated upper extremity model, originally created in OpenSim [15], to model arm movements in a set of interaction tasks. This model includes seven degrees-of-freedom (DoF) to represent the movements of shoulder, elbow, and wrist. In contrast to point-mass or linked-segment models, which are widely used to simulate user behavior [24, 71, 72], the MoBL ARMS model includes translational and rotational coupling between body segments, physiological joint axis orientations, and joint angle limits. The model is actuated by 50 Hill-type muscle-tendon actuators [45].

We converted the model to MuJoCo using the *O2MConverter* [28], which creates an approximate replica of the original OpenSim model in MuJoCo. MuJoCo allows for much faster forward simulations of the interactive environment, while including more sophisticated contact dynamics. Some of the limitations of the converted model are that MuJoCo tendons are inelastic, and tendon paths are limited to fixed sites as opposed to having dynamically moving and conditional pathpoints or wrapping objects as in OpenSim. Excluding these limitations, the MuJoCo model is anatomically as accurate; and MuJoCo and OpenSim use the same muscle activation dynamics and exhibit comparable force-length-velocity-curves.

In order to decrease the state and action space dimensionality of the RL optimisation problems, we disabled two wrist DoFs (wrist flexion and deviation), which were not instrumental in the interaction tasks. Furthermore, we disabled 24 muscle actuators that mainly actuated finger movements, which were deemed unnecessary, as there were no DoFs in the model's fingers. Therefore, 5 DoFs and 26 muscle actuators remained to represent the kinematics and dynamics of the arm. In order to ensure that our simulation can only achieve reasonable body postures, we modified the equality constraint that couples elevation angle and shoulder rotation in the MuJoCo model by adding an additional dependency on shoulder elevation (details are given in Suppl. Mat. S2). The fingers of the model were modified such that index finger is extended, while the rest are flexed.

The proprioceptive observations ω^P contain joint angles, velocities, and accelerations for the five DoFs, and muscle internal activation states m for the 26 muscle actuators. As all considered tasks require precise movement of the end-effector, we also included Cartesian coordinates of the tip of the index finger in the proprioceptive observations. The joint angles and muscle internal activations are normalised to range [-1, 1].

The visual observation ω^V is rendered from an RGB-D camera with a resolution of 120×80 pixels (Figure 4 shows an example of a visual observation). The "eye camera" was placed 20 cm above the torso, approximately where one's head would be located. For simplicity, we decided to fixate the camera position, resulting in a constant field of view in the same direction. In each of the tasks we use either one or multiple color channels with or without depth channel, depending on what kind of information the agent requires to successfully learn the task. In some tasks we also include prior visual observations to allow the agent to infer movement velocity. The image data is normalised to range [-1, 1] for each channel.

Furthermore, in choice reaction and parking tasks we have included tactile observation ω^T of the fingertip. This force sensor lets the simulated user know how much force it is exerting through contacts. The force value is a non-negative scalar.

The policy network π contains a convolutional neural network to encode the high-dimensional visual observations into lower dimensional representations. The other observations are vectors which are concatenated and passed through a separate encoder, before being concatenated with the encoded visual observations. This representation is then passed through two fully connected layers to produce the mean vectors μ_θ, which are then used to sample relative action vectors a_t' cf. (1). Network architecture details are given in Suppl. Mat. S1.

We chose to use a *neural effort* term [8] to constrain unnecessary movements, as a similar term is often used in RL when learning

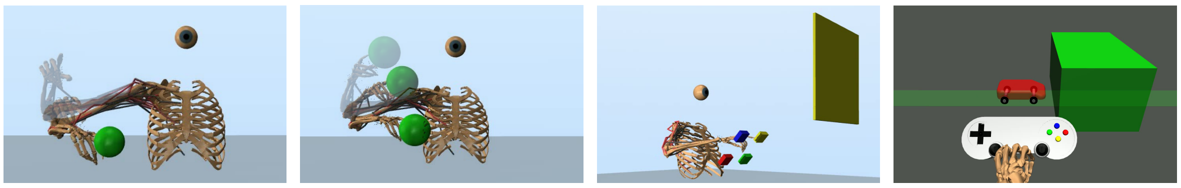

(a) Pointing (b) Tracking (c) Choice Reaction (d) Parking a Remote Control Car

Figure 3: Four interactive tasks with differing perceptual-motor requirements. The figures show the MoBL ARMS model, and the RGB-D camera that serves as a visual system.

control policies. At each timestep t we compute the effort term

$$\delta_t = \sum_{i=1}^{N} a_{t,i}^2, \tag{3}$$

which represents neural strain from controlling motor neurons of the $N = 26$ muscle actuators, cf. (2). As mentioned earlier, the effort term δ_t is part of the reward r_t, i.e. it is subtracted from the proposed reward functions.

The simulation timestep in MuJoCo is set to 2 milliseconds, and actions are sampled from the policy with a frequency of 20 Hz during training, and 100 Hz during evaluation. According to our observations, this mismatch of action frequency sampling between training and evaluation has minimal effect on the results. Training with lower sampling frequency makes the training faster and mitigates the credit assignment problem [46], while evaluation with higher action frequency is required for some of the movement analysis.

4.2 Case Study: Pointing

Pointing is one of the most intensely studied interactive tasks in HCI. In a pointing task, users are asked to move a physical or virtual end-effector towards some object, e.g., a target sphere of given size. In this case study we demonstrate perception-based muscle control in a setup that corresponds to the well-known ISO pointing task variants.

In our model of this task, the end-effector corresponds to the tip of the index finger and the target is a penetrable sphere of varying radius located in front of the simulated user. The radius and location of the target are sampled randomly during training: radius from a continuous interval [5, 15] cm, and location from a 2D plane of size 60 cm × 60 cm. The origin of the plane is located 55 cm in front of the agent's shoulder and 10 cm to the right, in order to make targets easily reachable in all areas of the plane. The agent must keep its fingertip inside a target for 500 milliseconds to successfully "hit" the target. A new target location is sampled when a target is hit, or after four seconds if the agent fails to hit the target. The new target location is sampled with rejection sampling such that the distance between two consecutive targets is typically more than 30 cm. However, a new tentative location is sampled maximum ten times, so in rare cases the distance may be less than 30 cm. A total of fourteen targets are spawned during one episode of training. The location of the target plane and dwell time of 500 milliseconds were chosen to try and match the experimental conditions of a reciprocal ISO pointing task presented in [33], which allows us to compare our simulations to their human data, specifically to user U1. In order to make the comparisons more fair, the MoBL ARMS model was anatomically scaled to better match the anatomical dimensions of user U1 (only in this task).

At time t, the simulated user observes $o_t = (\omega_t^P, \omega_t^V, \xi_t)$. The visual observation ω_t^V contains only depth information, as color information is not necessary for completing this task. The stateful information ξ_t comprises of two quantities: how many targets are left in the episode, and how many milliseconds the fingertip has been inside a target. Both quantities are normalised to range [-1, 1], and either is not necessary to learn the task, but do speed up the training process. Following the same rationale as for typical experimental instructions, we want the simulated user to complete the task of hitting 14 targets as quickly as possible. Thus, the reward function is a mixture of two components: a negative reward, shaped by the distance between the agent's fingertip and target, issued as long as the target has not been reached, and a positive bonus for hitting the target. The reward function is

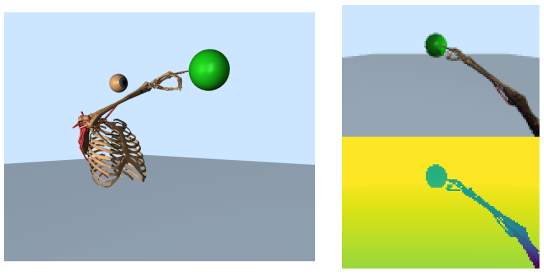

(a) (b)

Figure 4: (a) A scene during a pointing task from an external camera. (b) The same scene rendered from the RGB-D camera, RGB image on top and depth image on bottom. Both images are 120x80 pixels.

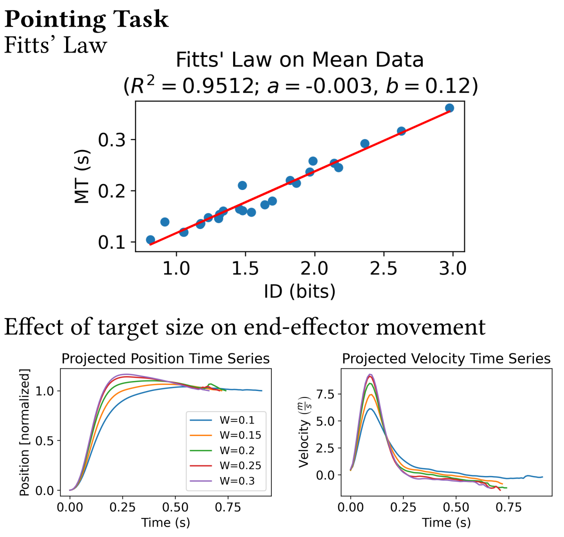


Figure 5: Simulated end-effector trajectories reproduce the Fitts' Law. That is, there is a clear effect of target size W on average peak velocity and total movement duration. Overshoot (i.e., projected position > 1) increases with target size.

$$r_t = \begin{cases} 8 - \delta_t & \text{if target is hit} \\ 0 - \delta_t & \text{if fingertip is inside target} \\ (e^{-d_t * 10} - 1)/10 - \delta_t & \text{otherwise,} \end{cases} \quad (4)$$

where d_t is the distance between fingertip and target surface and δ_t is the effort term (3) at time t.

4.2.1 Performance metrics. We collected a dataset of movements by running the policy for 100 episodes with randomly sampled target radii and locations. We denote the time between two sampled targets as one trial. During one episode the agent is presented with 14 targets, therefore we have a total of 1,400 trials in the dataset. The simulated user hit 1,395 targets (a success rate of 99.64%), and on average it took the agent 690 ms to finish one trial.

4.2.2 Fitts' Law and speed-accuracy trade-off. Fitts' Law is a well-established regularity for pointing and target acquisition tasks, claiming a linear relationship between the difficulty and the average movement time required to reach a given target [20, 42]:

$$\text{MT} = a + b\text{ID} = a + b\log_2\left(\frac{D}{W} + 1\right). \quad (5)$$

Here, D and W are the initial distance to target and the size (diameter) of the target sphere, respectively, ID denotes the Index of Difficulty in bits (using the *Shannon Formulation* [41]), and MT is the predicted movement time. In order to verify whether the end-effector trajectories produced by our simulated user follow Fitts' Law, we binned the 1,395 evaluation trials described in Section 4.2.1 into 25 groups, using 5 quantile-based partitions of each distance

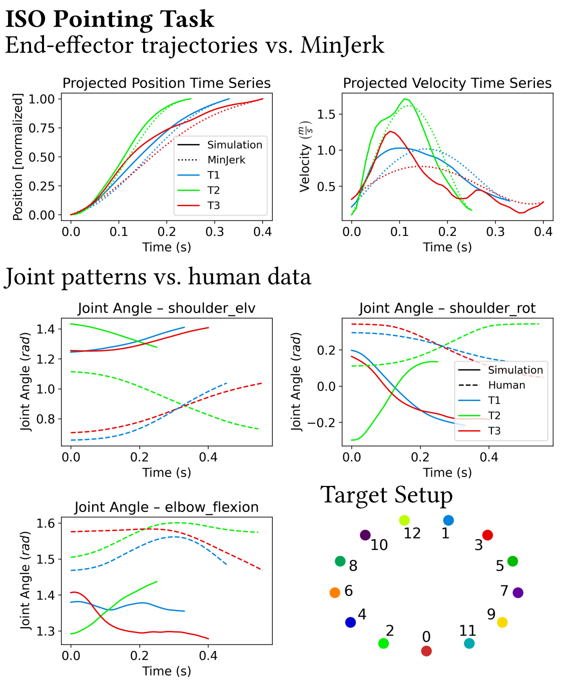


Figure 6: In our simulation of the ISO cyclical pointing task, targets are reached with a single ballistic movement (solid lines; upper plots). The projected position and velocity time series are close to the Minimum Jerk trajectories (dashed lines; upper plots). The joint angles of both shoulder and elbow resulting from our simulation (solid lines; lower plots) exhibit the same patterns as observed in the user study (dashed lines; lower plots). As expected, movement direction has a strong effect on qualitative behavior (representative movements to targets 1, 2, and 3 are shown, respectively).

and target width. For each group, we computed the average movement time per trial, and used the resulting combinations of ID and MT to identify the model parameters $a = -0.003$ and $b = 0.12$ via linear regression. As can be seen in the upper plot of Figure 5, our simulation trajectories (blue dots) are consistent with the linear relationship predicted by Fitts' Law (red line), explaining more than 95% of the between-group variance ($R^2 = 0.9512$).

To analyze the specific effect of target size on the simulation trajectories, we simulated five movements between the same two target locations for five different target sizes W. This was repeated for eight pairs of targets, resulting in a total of 40 movements per target size. To isolate the effect of target size from those of confounding variables such as movement direction or different trials, we projected each end-effector trajectory onto the respective direct path between initial and target position and then computed the *mean projection* for each target size. This was done by computing the average projected position and velocity of all 40 movements at each timestep, and concatenating the resulting position and velocity time series. As shown in the lower plots of Figure 5, there is a clear effect of target size on the mean projections, both in terms of end-effector position and velocity. As target size decreases, movements become slower, resulting in both a lower average peak velocity and a larger average movement time (9.31 m/s and 660 ms for 30 cm width vs. 6.12 m/s and 900 ms for 10 cm width). Also note that there is a clear tendency to overshoot for large target sizes. This shows that the speed-accuracy trade-off typically observed in aimed movements towards spatially constrained targets (and which is also consistent with Fitts' Law) is inherent to our simulation. Albeit the confidence intervals of these mean projections overlap (not shown in the figure), the effect is consistently seen across different movements.

4.2.3 Minimum Jerk. One of the best-known models to describe the kinematics of human aimed movements is the Minimum Jerk (MinJerk) model proposed by Flash and Hogan [21]. This model assumes that humans aim to generate smooth end-effector trajectories, which is equivalent to minimizing the change in end-effector acceleration over time, denoted as *jerk*. While the MinJerk model does not make any predictions of the underlying human body and interaction dynamics, cannot account for corrective submovements, and requires movement duration as well as initial and terminal positions, velocities, and accelerations to be known in advance, it has been successfully used for modelling perturbed reaching [26] and word-gesture keyboard typing [56].

We replicated the experimental setup of a previous user study [33], where 13 target spheres were equidistantly arranged according to the ISO 9241-9 ergonomics standard (see target setup in Figure 6). In the original experiment, all ISO targets were always visible with the active target highlighted, whereas in our version of the task only the active target was visible. As opposed to the previous task where targets were randomly sampled, now the target radius was fixed to 5 cm, and the target location was chosen according to the ISO protocol. We used the same policy as previously (trained with random targets) to control the agent in this task, as ISO pointing effectively is a subset of the more general pointing task. In Figure 6, projected simulation trajectories for three representative movements – from T0 to T1, T1 to T2, and T2 to T3 – are shown for the surge phase, i.e., since the former target was hit and until the latter target was first reached. Both the projected position and velocity time series (solid lines in upper plots) match the corresponding minimum jerk trajectories (dotted lines in upper plots) visually well, suggesting that the targets are reached with a single ballistic movement. The simulation trajectories exhibit the symmetric, bell-shaped velocity profiles during the surge phase that are characteristic of mid-air pointing [49]. More quantitative comparisons between our method and MinJerk would not be particularly meaningful due to the different assumptions and goals described above.

4.2.4 Comparison to Human Data. To identify whether body postures of our simulations are comparable to those of humans, we computed the joint angles that best explain the movements of user U1 observed in the ISO task user study in [33] via *Inverse Kinematics*, and compared them to the joint angles inferred from our simulation. Note that the trajectories which we compare, e.g. starting from T0 and ending in T1, begin when the simulated agent (or user U1) has hit T0, that is, the agent's (user U1's) fingertip is inside said target. Similarly the trajectory ends when the agent's (user U1's) fingertip is inside T1, and hence the maximum distance between the agent's and the user's initial and final fingertip positions is less than 10 cm (twice the target radius). These comparisons aim to provide qualitative evidence of the simulated agent's movements with respect to actual human movements. As can be seen in the lower plots of Figure 6, the general patterns of each shoulder elevation, shoulder rotation, and elbow flexion match considerably well between our simulation (solid lines) and the human reference (dashed lines). In particular, the body postures required to reach high targets (T1, T3; blue and red lines) are clearly different from those required for low targets (T2; green lines), which our simulation captures well. The largest differences between simulation and human data occur in terms of used joint ranges and movement duration. Both of these were expected, as we did not explicitly set our simulation to the initial body posture of the respective user, and we did not optimize the neural effort cost such that the absolute movement times would match human data.

4.3 Demonstrations: Tracking, Choice Reaction, and Parking a Remote Control Car

Here we further demonstrate that our approach is suitable to modelling and simulating a wide range of interactions that include perception and physical contact. We provide a description of each task, followed by relevant performance metrics to show that the simulated user learns

- complex muscle-actuated visuomotor control in an emergent fashion, simply based on task-specific rewards,
- to utilize prior observations to anticipate movement (tracking and parking tasks),
- to discriminate between different responses, and choose a correct response based on observed stimuli (choice reaction task),
- to control objects that have non-trivial (sixth order) dynamics (parking task using a joystick).

4.3.1 Tracking. In the tracking task the agent's objective is to follow a moving target with its fingertip as closely as possible. The environment here is very similar to the one used in pointing task: the target is confined to a 2D plane of size 60 cm × 60 cm in front of the agent, but the target is not static and target radius is fixed to 5 cm. The target follows a trajectory that is a mixture of five sine waves with varying amplitudes, frequencies, and phases. The amplitudes are uniformly sampled from interval [1, 5], and frequencies from [0, 0.5], and episode length is fixed to 10 seconds. We used a curriculum learning approach for this task, where the targets are initially fixed for the first 15 million training steps, and between 15 million and 40 million training steps the frequencies are sampled from range $[0, f_{max}]$, where f_{max} linearly increases from 0 to 0.5.

To allow the simulated user to anticipate the target's movements, we included a past visual observation as input to the policy. The observation is then $o_t = (\omega_t^P, \omega_t^V, \omega_{t-k}^V)$, where k is chosen such that the past observation is 100 milliseconds prior to the current

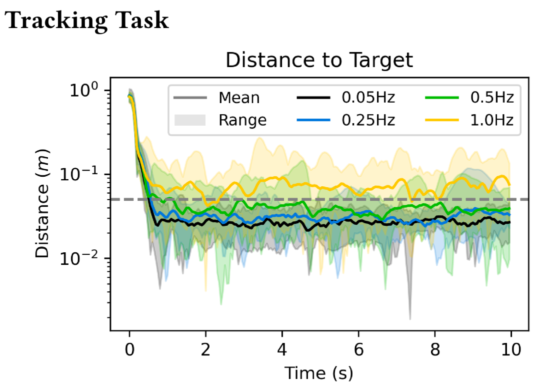


Figure 7: Distance between the agent's fingertip and target center as a function of time. After a fast initial movement towards the target, the agent is able to keep track of the target, even if the fingertip is temporarily outside the target (values above the horizontal dashed line). Each of the considered frequencies that define the movement pattern of the target is given in a different color.

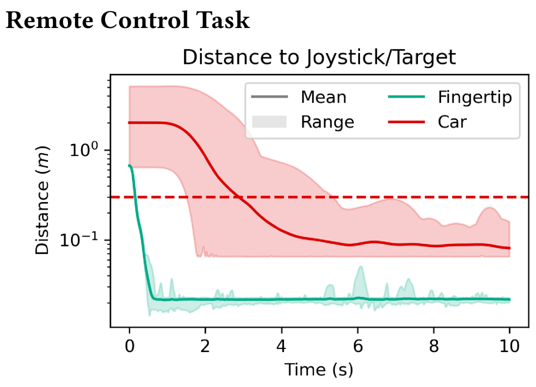


Figure 8: Distance between the agent's fingertip and the joystick (teal), as well as distance between the controlled car and target (red) as a function of time. After moving the hand towards the joystick, which takes approx. 800 milliseconds, the joystick is used to steer the car inside the target box (values below the horizontal dashed line, showing the 30 cm target size constantly used during evaluation).

one. As in the pointing task, the visual observation ω^V contains only depth information. The reward function for this task is simply

$$r_t = -d_t - \delta_t, \tag{6}$$

where d_t is the distance between fingertip and the target surface at time t, and δ_t is the effort term (3).

Figure 7 shows the distance between fingertip and target origin as a function of time on a logarithmic scale. Initially the average distance is large, as the agent's arm is besides its torso in the starting position. The distance drops quickly as the agent starts to track the target, and stays close to the target for the rest of the episode. To analyze the effect of the target speed, we considered four different frequencies f_{max} (0.05, 0.25, 0.5, and 1 Hz), with 10 episodes created for each frequency. The solid lines in Figure 7 correspond to the respective average values, the filled areas to the complete ranges. While slower targets are clearly easier to track, the agent is able to keep the fingertip mainly inside the target up to frequencies of 0.5 Hz. For 1 Hz movements, which the agent has not seen during training, the fingertip is short behind the target most of the time, resulting in a consistently small offset.

4.3.2 Choice Reaction. In a choice reaction task a participant is presented with several responses, and is required to choose between those responses when observing a stimulus. In our simulated version of this task (see Figure 3(c)), the agent is presented with four different colored buttons, and a screen to show the stimulus – all in field of view of the agent. The training procedure is similar to the pointing task: the agent has four seconds to press a button and receive a positive reward. When a button has been pressed with suitable force, or four seconds have passed, the screen changes color and indicates which button should be pressed next. The agent is presented with a stimulus and expected to choose a response ten times in one episode.

In this task the simulated user receives an observation $o_t = (\omega_t^P, \omega_t^V, \omega_t^T, \xi_t)$. The stateful information ξ_t contains one quantity: the number of targets left in the episode, normalised to range $[-1, 1]$. The reward function is akin to (4) used in the pointing task:

$$r_t = \begin{cases} 8 - \delta_t & \text{if target button is pressed} \\ (e^{-d_t * 10} - 1)/10 - \delta_t & \text{otherwise,} \end{cases} \tag{7}$$

where d_t is the distance between the fingertip and the center of the target button at time t, and δ_t is the effort term (3).

While the simulated user learns to press the appropriate buttons successfully, it does so in a rather quick manner. The average time to finish the episode is 3.94 seconds with a standard deviation of 0.91 seconds (averaged over 100 episodes). Only in four trials out of 1000 the agent was unable to respond within four seconds of observing the stimulus. While this behaviour is in the realm of possibilities for a human, it is likely faster than an average human subject would perform. One explanation for this could be the somewhat unrealistic visual model that neither models selective attention nor peripheral vision. Instead of having to alternate focus between a button and the screen, the simulated user is able to perceive all objects at the same time. It is also possible that, since the simulated user and the buttons are fixed in space, the user learns the locations of the buttons based on proprioceptive observations instead of visual observations.

4.3.3 Parking a Remote Control Car. As another interaction task, we trained the agent to steer a remote control car using a joystick (see Figure 3(d)). The goal of this interaction task is to park the car inside the green box. The initial positions of the car and the target are sampled from a green line in front of, and fully visible to, the agent. The car moves only in one dimension, along the green line, and its acceleration/deceleration is controlled by tilting the left joystick of the gamepad forward or backward. The length of an episode is fixed to ten seconds. Note that this task differs from previous tasks in terms of difficulty twofold. First, it requires fine muscle control since the joystick and required movements are relatively small. Second, in addition to controlling the biomechanical model, the agent has to learn the second-order dynamics of the car resulting in a higher complexity (sixth-order in total).

The observation for this task is $o_t = (\omega_t^P, \omega_t^V, \omega_{t-k}^V, \omega_t^T)$, where k is chosen such that the past visual observation ω_{t-k}^V is 100 milliseconds prior to the current one. To speed up the training, the visual observation contains only the red color channel. The tactile perception ω_t^T contains force reading of contact between fingertip and the joystick to aid the agent in estimating how much the joystick needs to be tilted to move the car.

The reward function is

$$\begin{aligned} r_t &= D(fingertip, joystick) + D(car, target)/10 \\ &\quad + B_{joystick,fingertip} + B_{car,target} - \delta_t, \end{aligned} \tag{8}$$

where $D(x, y) = e^{-d_t(x,y)*3} - 1$ is a function of distance d_t between x and y at time t, $B_{x,y}$ are bonus terms, and δ_t is the effort term (3). The bonus $B_{fingertip,joystick} = 0.8$ is given only once per episode, for the first time when the fingertip touches the joystick. The bonus $B_{car,target} = 8$ is granted if the car is inside the target with a velocity less than 0.1 m/s.

We evaluated the agent's performance over 50 episodes. Figure 8 shows the distances between agent's fingertip and joystick, and car and target, as functions of time.[5] The figure shows that it takes less than three seconds, on average, to move the car inside the target (values below red dashed line), and in all 50 episodes the car is successfully parked inside the target by the end of the episode.

5 SUMMARY AND DISCUSSION

We implemented perception-based muscle-actuated biomechanical models in MuJoCo, and demonstrated how an RL approach can be leveraged to simulate human-like behaviour in different interaction tasks with physically simulated input devices. The user model successfully learned to complete a variety of interaction tasks, while also producing movements that comply with predictive models of human movement, such as Fitts' Law. Such models could be used to evaluate user interfaces *in silico*, before or instead of, running evaluation studies with human subjects. Also, use of simulation may enable a more rigorous way to ensure diversity is taken into account during design. The models are available in the release of the User-in-the-Box open-source implementation, available at https://github.com/aikkala/user-in-the-box.

5.1 Reward–Model Interactions

RL provides flexibility in formulating an interaction task as an optimisation problem, but it does require experience with RL to develop an appropriate reward function, and its formulation will affect the final model outcome. As this is a new approach to generating representative models of human behaviour for HCI, there is not yet a mature workflow for refining the reward function design for a given task. Our open-source implementation introduces a new problem domain for researching the effect of a utility function, i.e., the reward function, on model behaviour. For instance, based on our observations, the scaling of a reward (not including the effort term) did not often incur major differences in the simulated user's behaviour, while e.g. the (non-)linearity of a reward function did.

In our simulation studies we employed well-known strategies for making the optimisation problem easier: early termination, reward shaping, and curriculum learning. We used a form of early termination in pointing and choice reaction tasks, where a new target was spawned every 4 seconds; reward shaping in all of the reward functions to guide the agent towards desired behaviour; and curriculum learning in the tracking task, where the agent first learns to point towards a fixed target, and eventually the target begins to move. Arzate Cruz and Igarashi's survey [1] reviews reward function design for interactive applications.

Our primary concern in this paper was to define reward functions for which policies could be learned efficiently. The reward functions in our tasks consisted of two components, a distance component and an effort component. The former guided the agent towards desired body postures and is *task-specific*, although the idea of using some sort of distance reward can be applied to many tasks. The effort term included in the reward, on the other hand, is a *task-agnostic* component that served to steer the agent to interact with the environment in a specific way, i.e., with minimum effort. A third component, time, comes into play implicitly via negative rewards. With negative rewards the agent is incentivized to finish a task quickly, if the episode length is not fixed (for example, the pointing and choice reaction tasks). If one uses only positive rewards the task completion may be unnecessarily prolonged, as there is no incentive to finish the episode promptly, especially if rewards far in the future are not heavily discounted using a low discount factor γ. To reiterate, all our reward functions share negative distance and effort components, while the positive bonus terms are connected to milestones or completion of the task.

The complexity of the task being modelled plays a significant role in finding an efficient reward function. In the pointing, tracking, and choice reaction tasks the agent learned a good control policy robustly without search for an exact scaling or parameterisation of the reward function. However, in the parking task it took us multiple iterations to find a reward function that produced a successful policy. Further study is required to find best practices for efficient reward function design.

Further, although the learned policy captures human behaviour in a number of ways (see Section 4), it is not known how well these reward functions model human subjective utility functions. Indeed, we have not tried to 'fit' the parameters of the reward function to human data. We anticipate that future work will need to take on this challenge. Future work should be inspired by what is known about human subjective utility. For example, it is known that people are sensitive to externally imposed speed/accuracy trade-offs [27, 78] but that people vary in *how* sensitive they are, with some preferring to be more accurate and others preferring to be fast.

[5]Note that the distance between car and target center cannot fall below a certain threshold, as it is always measured from the most distant wheel.

5.2 Hierarchical Controllers

Another frontier concerns computational efficiency. The required number of training steps until convergence in the first three tasks was typically 40-80 million steps, which required 24-48 hours to train with 10 parallel workers and a GPU. The agent in the last task (parking a remote control car) was trained for 130 million steps, which required 94 hours. In each of the simulation tasks we train the agent from scratch, which means that the agent must always learn again how to move its arm. We believe that training time could be significantly decreased by using a hierarchical RL approach, where the optimisation problem is made easier by first training a separate task-agnostic low-level controller to control movements of the agent, and then training task-specific higher level policies to solve the actual RL problem [3, 37, 54].

5.3 Increasing Realism

We see multiple ways to increase the realism of biomechanical and perception models for HCI research. While advancing significantly in the last decade, modern biomechanical models are developed as mechanical systems involving multiple assumptions and simplifications in comparison to the natural human body: e.g. (in order of decreasing severity) an activation optimality assumption that excludes muscle co-contraction, static muscle states that ignore fatigue effects, solid movement mechanics ignoring soft tissues, generic weight distribution based solely on rigid segment properties, continuous excitation signal instead of motor unit size-based control, or simplified hinge-based joint mechanics instead of slide & roll movement with complex 3D transformation. These simplifications can lead to the unnaturalness of generated movements, particularly ones involving fine motor control or under fatigued muscles, and deviations in predicting injury risks or fatigue. Considering the above simplifications, the modern models can simulate with reasonable accuracy most movements, except fine motorics involving co-contraction of opposite muscles, such as writing. Our biomechanical model, although more sophisticated than previous HCI models, is only a representation of the upper torso with shoulder and arm movement. There are no wrist or finger movements included. The perceptual observations were based on rather rudimentary models: the visual system was represented by a low-resolution RGB-D camera with a constant field of view, tactile perception was based on a single force sensor, and none of the perception models included noise modelling. Furthermore, MuJoCo as the chosen physics simulator might not be suitable to model some interaction devices, such as a touch screen with dynamic content. However, model development is typically allocated limited resources, and it is necessary to stop development once one has a model that works well enough for a given task. In this situation, it is important to document the qualities of the model for others to build on in future. A practical challenge is that a biomechanical model which initially appears to predict the human data well may have inaccuracies which first become apparent when used for optimisation in the RL process, as the inaccuracies are 'exploited', leading to unnatural behaviour.

While movements simulated in the pointing task shared many characteristics found in human data, the simulated movement differed in some aspects. For instance, the joint ranges of our simulated user were different from the joint ranges obtained with inverse kinematics of a human user, and the movement speed of the simulated user was slightly faster. However, it is unclear what the best choice of distance measure is between simulation states and observations of human poses, and how accurate replications of human data need to be for practical applications such as user interface evaluations. This is likely to vary based on use case.

6 CONCLUSION

We believe that perceptual control of biomechanically plausible human models is the key to more extensive use of simulations in the field of HCI. Perception has such a significant role in interaction that oversimplifying it in models may have stalled progress in studies of motor control in HCI. Linking perception and muscle-based control is necessary for understanding both low-level phenomena in HCI, such as bimanual control and eye-hand coordination when using input devices, but also higher-level phenomena, such as the emergence of fatigue-avoiding strategies in AR/VR applications.

While cognitive models have been at the heart of HCI since its inception [9] and significant progress has been made [11–13, 31, 52], one enduring limitation has been the lack of an *end-to-end* framework for predicting and explaining interaction. In these cognitive approaches, perception and biomechanics are modelled either as black box symbolic input/output functions or with mathematical laws (e.g. Fitts' Law) that do not simulate the processes of embodiment and as a consequence much of real-world interaction is left unexplained. In contrast, the approach to modelling that we have proposed in the current paper embraces perception and biomechanics as a key locus of explanation but, arguably, neglects the role of cognition. Future work should seek to combine the strengths of both approaches. For example, one could seek to explain not only how people use perception and biomechanics to point-and-click, but also how they use such skills to navigate, browse, acquire information, make decisions, and collaborate. RL, and particularly hierarchical RL, offers a framework for such an extension.

Finally, we believe that simulations of the kind discussed in this paper can help the scientific process in HCI research. The formal rigour required in creation of a simulation model, and controlling and documenting the provenance of knowledge and data used to calibrate it, makes clear the importance of many of the often poorly described aspects of context in HCI experiments. A simulation package is also easily shared with other researchers, improving reproducibility via an unambiguous implementation of the current scientific theory, the predictions of which can be validated with observed real-world data. We have made some effort to describe the weaknesses of our model, because aspects of models which at any given stage are poorly justified theoretically, are a poor fit to experimental data, or which are highly sensitive to context can be viewed as prompts to the research community about where they need better theories, more complex models, or more data. This can create improved clarity, and a shared awareness of the open problems and challenges, and can help document progress.

ACKNOWLEDGMENTS

A.I. is funded by the Academy of Finland Flagship programme "Finnish Center for Artificial Intelligence" (FCAI). R.M-S. acknowledges funding support from EPSRC grant EP/R018634/1, *Closed-loop Data Science* and the Academy of Finland via FCAI. A.O. is supported by Academy of Finland project *Human Automata*.

REFERENCES

[1] Christian Arzate Cruz and Takeo Igarashi. 2020. *A Survey on Interactive Reinforcement Learning: Design Principles and Open Challenges.* Association for Computing Machinery, New York, NY, USA, 1195–1209. https://doi.org/10.1145/3357236.3395525

[2] M. A. Ayoub, M. M. Ayoub, and A. G. Walvekar. 1974. A Biomechanical Model for the Upper Extremity using Optimization Techniques. *Human Factors* 16, 6 (1974), 585–594. https://doi.org/10.1177/001872087401600603 arXiv:https://doi.org/10.1177/001872087401600603 PMID: 4442903.

[3] Amin Babadi, Michiel Van de Panne, Caren Liu, and Perttu Hämäläinen. 2021. Learning Task-Agnostic Action Spaces for Movement Optimization. *IEEE Transactions on Visualization and Computer Graphics* (2021), 1–1. https://doi.org/10.1109/TVCG.2021.3100095

[4] Myroslav Bachynskyi, Antti Oulasvirta, Gregorio Palmas, and Tino Weinkauf. 2014. Is Motion Capture-Based Biomechanical Simulation Valid for HCI Studies? Study and Implications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 3215–3224. https://doi.org/10.1145/2556288.2557027

[5] Myroslav Bachynskyi, Gregorio Palmas, Antti Oulasvirta, Jürgen Steimle, and Tino Weinkauf. 2015. Performance and Ergonomics of Touch Surfaces: A Comparative Study Using Biomechanical Simulation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1817–1826. https://doi.org/10.1145/2702123.2702607

[6] Nikola Banovic, Tofi Buzali, Fanny Chevalier, Jennifer Mankoff, and Anind K. Dey. 2016. *Modeling and Understanding Human Routine Behavior.* Association for Computing Machinery, New York, NY, USA, 248–260. https://doi.org/10.1145/2858036.2858557

[7] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 206 (nov 2019), 11 pages. https://doi.org/10.1145/3355089.3356536

[8] Bastien Berret, Enrico Chiovetto, Francesco Nori, and Thierry Pozzo. 2011. Evidence for Composite Cost Functions in Arm Movement Planning: An Inverse Optimal Control Approach. *PLOS Computational Biology* 7, 10 (10 2011), 1–18. https://doi.org/10.1371/journal.pcbi.1002183

[9] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1983. *The psychology of human-computer interaction.* Crc Press.

[10] Noshaba Cheema, Laura A. Frey-Law, Kourosh Naderi, Jaakko Lehtinen, Philipp Slusallek, and Perttu Hämäläinen. 2020. *Predicting Mid-Air Interaction Movements and Fatigue Using Deep Reinforcement Learning.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376701

[11] Xiuli Chen, Aditya Acharya, Antti Oulasvirta, and Andrew Howes. 2021. An adaptive model of gaze-based selection. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–11.

[12] Xiuli Chen, Gilles Bailly, Duncan P Brumby, Antti Oulasvirta, and Andrew Howes. 2015. The Emergence of Interactive Behaviour: A Model of Rational Menu Search. (2015).

[13] Xiuli Chen, Sandra Dorothee Starke, Chris Baber, and Andrew Howes. 2017. A cognitive model of how people make decisions through interaction with visual displays. In *Proceedings of the 2017 CHI conference on human factors in computing systems.* 1205–1216.

[14] Michael Damsgaard, John Rasmussen, Søren Tørholm Christensen, Egidijus Surma, and Mark de Zee. 2006. Analysis of musculoskeletal systems in the AnyBody Modeling System. *Simulation Modelling Practice and Theory* 14, 8 (2006), 1100–1111. https://doi.org/10.1016/j.simpat.2006.09.001 SIMS 2004.

[15] Scott Delp, Frank Anderson, Allison Arnold, Peter Loan, A. Habib, Chand John, Eran Guendelman, and Darryl Thelen. 2007. OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement. *Biomedical Engineering, IEEE Transactions on* 54 (12 2007), 1940 – 1950. https://doi.org/10.1109/TBME.2007.901024

[16] Christopher L. Dembia, Nicholas A. Bianco, Antoine Falisse, Jennifer L. Hicks, and Scott L. Delp. 2021. OpenSim Moco: Musculoskeletal optimal control. *PLOS Computational Biology* 16, 12 (12 2021), 1–21. https://doi.org/10.1371/journal.pcbi.1008493

[17] João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, and Kaj Grønbæk. 2021. XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 290, 11 pages. https://doi.org/10.1145/3411764.3445349

[18] Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, and Jörg Müller. 2021. Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports* 11, 1 (2021), 1–15.

[19] Florian Fischer, Arthur Fleig, Markus Klar, and Jörg Müller. 2022. Optimal Feedback Control for Modeling Human-Computer Interaction. *ACM Trans. Comput.-Hum. Interact.* (2022). https://doi.org/10.1145/3524122

[20] Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47, 6 (1954), 381.

[21] Tamar Flash and Neville Hogan. 1985. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience* 5, 7 (1985), 1688–1703.

[22] Sam Hamner, Ajay Seth, and Scott Delp. 2010. Muscle contribution to propulsion and support during running. *Journal of biomechanics* 43 (10 2010), 2709–16. https://doi.org/10.1016/j.jbiomech.2010.06.025

[23] Blake Hannaford and Jack Winters. 1990. *Actuator Properties and Movement Control: Biological and Technological Models.* Springer New York, New York, NY, 101–120. https://doi.org/10.1007/978-1-4613-9030-5_7

[24] C. M. Harris and D. M. Wolpert. 1998. Signal-dependent noise determines motor planning. *Nature* 394 (09 1998), 780–4. https://doi.org/10.1038/29528

[25] Lorenz Hetzel, John Dudley, Anna Maria Feit, and Per Ola Kristensson. 2021. Complex Interaction as Emergent Behaviour: Simulating Mid-Air Virtual Keyboard Typing using Reinforcement Learning. *IEEE Transactions on Visualization and Computer Graphics* 27, 11 (2021), 4140–4149. https://doi.org/10.1109/TVCG.2021.3106494

[26] Bruce Hoff and Michael A. Arbib. 1993. Models of Trajectory Formation and Temporal Interaction of Reach and Grasp. *Journal of Motor Behavior* 25, 3 (1993), 175–192. https://doi.org/10.1080/00222895.1993.9942048 arXiv:https://doi.org/10.1080/00222895.1993.9942048 PMID: 12581988.

[27] Andrew Howes, Richard L Lewis, and Alonso Vera. 2009. Rational adaptation under task and processing constraints: implications for testing theories of cognition and action. *Psychological review* 116, 4 (2009), 717.

[28] Aleksi Ikkala and Perttu Hämäläinen. 2022. Converting Biomechanical Models from OpenSim to MuJoCo. In *Converging Clinical and Engineering Research on Neurorehabilitation IV*, Diego Torricelli, Metin Akay, and Jose L. Pons (Eds.). Springer International Publishing, Cham, 277–281.

[29] Hasan Iqbal, Seemab Latif, Yukang Yan, Chun Yu, and Yuanchun Shi. 2021. Reducing arm fatigue in virtual reality by introducing 3D-spatial offset. *IEEE Access* 9 (2021), 64085–64104.

[30] Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. 2019. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)* 38, 4 (2019), 1–12.

[31] Jussi Jokinen, Aditya Acharya, Mohammad Uzair, Xinhui Jiang, and Antti Oulasvirta. 2021. Touchscreen typing as optimal supervisory control. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–14.

[32] Antti Kangasrääsiö, Kumaripaba Athukorala, Andrew Howes, Jukka Corander, Samuel Kaski, and Antti Oulasvirta. 2017. Inferring Cognitive Models from Data Using Approximate Bayesian Computation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1295–1306. https://doi.org/10.1145/3025453.3025576

[33] Markus Klar, Florian Fischer, Arthur Fleig, Miroslav Bachinski, and Jörg Müller. 2022. Simulating Interaction Movements via Model Predictive Control. https://doi.org/10.48550/ARXIV.2204.09115

[34] Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. 2020. Estimating player completion rate in mobile puzzle games using reinforcement learning. In *2020 IEEE Conference on Games (CoG).* IEEE, 636–639.

[35] Francesco Lacquaniti, Carlo Terzuolo, and Paolo Viviani. 1983. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica* 54, 1 (1983), 115 – 130.

[36] Leng-Feng Lee and Brian R Umberger. 2016. Generating optimal control simulations of musculoskeletal movement using OpenSim and MATLAB. *PeerJ* 4 (2016), e1638.

[37] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable Muscle-Actuated Human Simulation and Control. *ACM Trans. Graph.* 38, 4, Article 73 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3322972

[38] Sung-Hee Lee and Demetri Terzopoulos. 2006. Heads up! Biomechanical modeling and neuromuscular control of the neck. In *ACM SIGGRAPH 2006 Papers.* 1188–1198.

[39] Katri Leino, Antti Oulasvirta, and Mikko Kurimo. 2019. RL-KLM: Automating keystroke-level modeling with reinforcement learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces.* 476–480.

[40] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=ryTp3f-0-

[41] I. Scott MacKenzie. 1989. A Note on the Information-Theoretic Basis for Fitts' Law. *Journal of Motor Behavior* 21, 3 (1989), 323–330. https://doi.org/10.1080/00222895.1989.10735486 arXiv:https://doi.org/10.1080/00222895.1989.10735486 PMID: 15136269.

[42] I. Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139.

[43] Richard S Marken and Warren Mansell. 2013. Perceptual control as a unifying concept in psychology. *Review of General Psychology* 17, 2 (2013), 190–195.

[44] J. Alberto Álvarez Martín, Henrik Gollee, Jörg Müller, and Roderick Murray-Smith. 2021. Intermittent control as a model of mouse movements. *ACM Transactions on Computer-Human Interaction (TOCHI)* 28, 5 (2021), 1–46.

[45] Matthew Millard, Thomas Uchida, Ajay Seth, and Scott L Delp. 2013. Flexing computational muscle: modeling and simulation of musculotendon dynamics. *Journal of biomechanical engineering* 135, 2 (2013).

[46] Marvin Minsky. 1961. Steps toward artificial intelligence. *Proceedings of the IRE* 49, 1 (1961), 8–30.

[47] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (12 2013).

[48] Roberto A Montano Murillo, Sriram Subramanian, and Diego Martinez Plasencia. 2017. Erg-O: Ergonomic optimization of immersive virtual environments. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*. 759–771.

[49] Pietro Morasso. 1981. Spatial control of arm movements. *Experimental brain research* 42, 2 (1981), 223–227.

[50] Jörg Müller, Antti Oulasvirta, and Roderick Murray-Smith. 2017. Control theoretic models of pointing. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 4 (2017), 1–36.

[51] Masaki Nakada, Tao Zhou, Honglin Chen, Tomer Weiss, and Demetri Terzopoulos. 2018. Deep Learning of Biomimetic Sensorimotor Control for Biomechanical Human Animation. *ACM Trans. Graph.* 37, 4, Article 56 (jul 2018), 15 pages. https://doi.org/10.1145/3197517.3201305

[52] Antti Oulasvirta, Jussi Jokinen, and Andrew Howes. 2022. Computational Rationality as a Theory of Interaction. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–14.

[53] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201311

[54] Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* 36, 4, Article 41 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073602

[55] E Pennestri, R Stefanelli, PP Valentini, and L Vita. 2007. Virtual musculo-skeletal model for the biomechanical analysis of the upper limb. *Journal of biomechanics* 40, 6 (2007), 1350–1361.

[56] Philip Quinn and Shumin Zhai. 2018. Modeling Gesture-Typing Movements. *Human–Computer Interaction* 33, 3 (2018), 234–280. https://doi.org/10.1080/07370024.2016.1215922 arXiv:https://doi.org/10.1080/07370024.2016.1215922

[57] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[58] J.B. Rawlings, D.Q. Mayne, and M.M. Diehl. 2017. *Model Predictive Control: Theory and Design* (2nd ed.). Nob Hill Publishing.

[59] Lei Ren, Richard K Jones, and David Howard. 2007. Predictive modelling of human walking over a complete gait cycle. *Journal of biomechanics* 40, 7 (2007), 1567–1574.

[60] Shaghayegh Roohi, Christian Guckelsberger, Asko Relas, Henri Heiskanen, Jari Takatalo, and Perttu Hämäläinen. 2021. Predicting Game Difficulty and Engagement Using AI Players. *Proceedings of the ACM on Human-Computer Interaction* 5, CHI PLAY (2021), 1–17.

[61] Shaghayegh Roohi, Asko Relas, Jari Takatalo, Henri Heiskanen, and Perttu Hämäläinen. 2020. Predicting game difficulty and churn without players. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. 585–593.

[62] Prashant Sachdeva, Shinjiro Sueda, Susanne Bradley, Mikhail Fain, and Dinesh K Pai. 2015. Biomechanical simulation and control of hands and tendinous systems. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

[63] Katherine R. Saul, Xiao Hu, Craig M. Goehler, Meghan E. Vidt, Melissa Daly, Anca Velisar, and Wendy M. Murray. 2014. Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. *Computer methods in biomechanics and biomedical engineering* 5842, May 2016 (2014), 1–14. https://doi.org/10.1080/10255842.2014.916698

[64] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17

[65] Ajay Seth, Michael Sherman, Jeffrey A. Reinbolt, and Scott L. Delp. 2011. OpenSim: a musculoskeletal modeling and simulation framework for in silico investigations and exchange. *Procedia IUTAM* 2 (2011), 212–232. https://doi.org/10.1016/j.piutam.2011.04.021 IUTAM Symposium on Human Body Dynamics.

[66] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of Bits: An Open-Domain Platform for Web-Based Agents. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 3135–3144. https://proceedings.mlr.press/v70/shi17a.html

[67] Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2014. Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–15.

[68] Eduardo D Sontag. 2013. *Mathematical control theory: deterministic finite dimensional systems*. Vol. 6. Springer Science & Business Media.

[69] Shinjiro Sueda, Andrew Kaufman, and Dinesh K Pai. 2008. Musculotendon simulation for hand animation. In *ACM SIGGRAPH 2008 papers*. 1–8.

[70] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

[71] Misaki Takeda, Takanori Sato, Hisashi Saito, Hiroshi Iwasaki, Isao Nambu, and Yasuhiro Wada. 2019. Explanation of Fitts' law in Reaching Movement based on Human Arm Dynamics. *Scientific Reports* 9 (12 2019), 19804. https://doi.org/10.1038/s41598-019-56016-7

[72] Hirokazu Tanaka, John W. Krakauer, and Ning Qian. 2006. An Optimization Principle for Determining Movement Duration. *Journal of Neurophysiology* 95, 6 (2006), 3875–3886. https://doi.org/10.1152/jn.00751.2005 arXiv:https://doi.org/10.1152/jn.00751.2005 PMID: 16571740.

[73] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[74] Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

[75] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. 2007. Solving deep memory POMDPs with recurrent policy gradients. In *International conference on artificial neural networks*. Springer, 697–706.

[76] DA Winter. 1984. Biomechanics of human movement with applications to the study of human locomotion. *Critical reviews in biomedical engineering* 9, 4 (1984), 287—314. http://europepmc.org/abstract/MED/6368126

[77] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Trans. Graph.* 39, 4, Article 33 (2020). https://doi.org/10.1145/3386569.3392381

[78] Mingrui Ray Zhang, Shumin Zhai, and Jacob O Wobbrock. 2019. Text entry throughput: Towards unifying speed and accuracy in a single performance metric. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

8 SIM2VR: Integrating Biomechanical Simulations in VR Development Environments

Authors: Florian Fischer[1], Aleksi Ikkala[1], Markus Klar, Arthur Fleig, Miroslav Bachinski, Roderick Murray-Smith, Perttu Hämäläinen, Antti Oulasvirta, Jörg Müller
Status: Submitted

FF and AI conducted and JM and AO supervised the research process, with guidance from MK, AF, MB, PH, and RMS. The idea and concept were developed by all authors. With guidance from PH, AI implemented the SIM2VR Asset, the Unity-compatible UitB extension, and the Whac-a-mole game, which were further improved by FF and MK. AI conducted the user study, and FF modeled and trained the simulated users. FF and MK conducted the evaluation and created the figures and check tools. MK and FF originated the guidelines and the walkthrough. FF and AI are the corresponding authors.

[1]Shared first authorship

SIM2VR: Integrating Biomechanical Simulations in VR Development Environments

FLORIAN FISCHER[*], University of Bayreuth, Germany
ALEKSI IKKALA[*], Aalto University, Finland
MARKUS KLAR, University of Bayreuth, Germany
ARTHUR FLEIG, University of Bayreuth, Germany
MIROSLAV BACHINSKI, University of Bergen, Norway
RODERICK MURRAY-SMITH, University of Glasgow, Scotland
PERTTU HÄMÄLÄINEN, Aalto University, Finland
ANTTI OULASVIRTA, Aalto University, Finland
JÖRG MÜLLER, University of Bayreuth, Germany

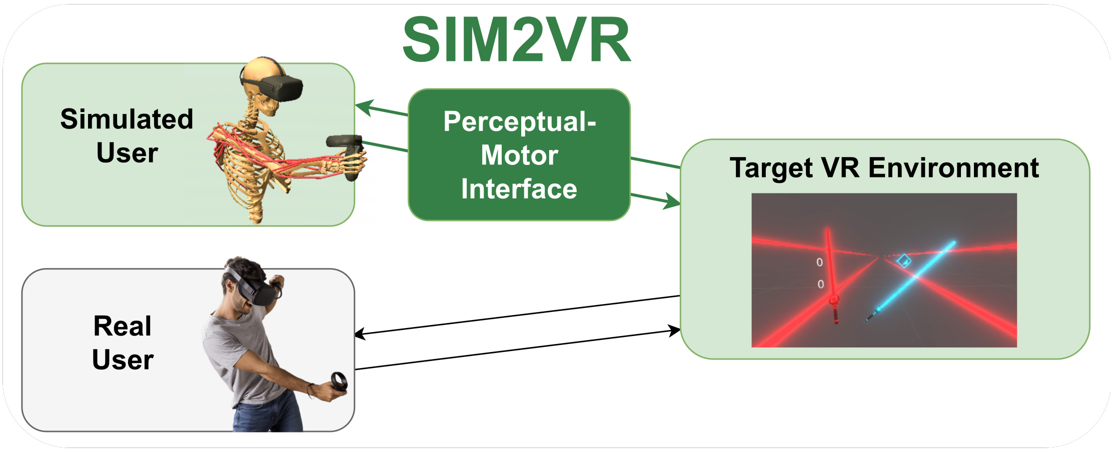


Fig. 1. SIM2VR closes the "reality gap" in computational models of user interaction. Its Perceptual-Motor Interface enables running high-fidelity biomechanical simulations directly in VR development environments, which can now be used to provide insight into users' performance, ergonomics, and movement strategies prior to user testing. Thanks to SIM2VR, the sensorimotor environments of a trained model better match those of the users, improving the accuracy of predictions. By lowering the barrier to biomechanical simulation, SIM2VR promotes the adoption of user models in VR design.

Designing VR interactions requires careful consideration of the effects of design choices on usability and ergonomics. Model-based evaluations have been shown to provide valuable insights prior to user testing. However, these models suffer from a "reality gap", meaning they are trained in simulated environments that differ from what users actually experience, which limits their accuracy and transferability. To close this gap, we introduce SIM2VR, an open-source platform for integrating biomechanical user simulations directly into VR development environments. Its key component is its Perceptual-Motor Interface, which allows models to "see" and "control" the exact same environment as humans. With two VR games, we provide recommendations for designing and selecting an appropriate biomechanical model, reward function, and curriculum for reinforcement learning, and demonstrate that SIM2VR can predict users' performance, effort, and strategy. We conclude that by lowering the barrier to biomechanical simulation, SIM2VR promotes the adoption of user models in VR design.

[*]Both authors contributed equally to this research.

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**; **Systems and tools for interaction design**; **Virtual reality**; *User models.*

Additional Key Words and Phrases: biomechanical simulation, perceptual-motor interface, interaction design, reality gap, automated testing, virtual reality, VR development environment, deep reinforcement learning

1 INTRODUCTION

Computational user models could assist in the design of VR interactions by providing insights into users' performance and experience before empirical evaluations. Recent advances in biomechanical models and deep reinforcement learning (RL) have enabled high-fidelity simulations that can be used to predict key indices of usability and ergonomics. These models include biologically plausible musculoskeletal models that are trained in physics simulators (e.g., MuJoCo [53]) to maximize a given reward function [7, 23]. In comparison to prior user models, the integration of RL and physics simulation has enabled working with more complex phenomena with fewer hand-crafted inputs. Yet, applications have been limited to relatively simple VR tasks, such as locomotion and aimed movement tasks like pointing, choice reaction, and vehicle control [8, 17, 23, 27]. We posit that, if developed further, such simulations could boost efforts in user-centered design in VR. They could allow to spot fundamental ergonomics issues such as "gorilla arm" early, even before user testing, freeing precious user testing time to evaluate the higher-level experience. This could drive the design of VR applications that explicitly minimize effort, reduce long-term fatigue, or prevent repetitive stress injuries. Simulations could also promote ability-based design in VR, because users with different physical and cognitive properties can be simulated without the risk of physical harm.

This paper attacks a critical obstacle to the wider adoption of simulation models in this space: the "reality gap". The reality gap is a known challenge in the field of robotics (e.g., [15]), where robots trained in simulations often struggle to perform well in the real world, and where several "Sim2Real" methods have been developed to address the issue. A similar issue arises in computational user modelling, where the simulation can predict how users would interact with the model of the user interface, but not the real user interface itself. To the best of our knowledge, this reality gap is not yet discussed in HCI. Currently, the environments in which user models are trained are different from the environments in which they ought to be deployed. These differences can be dramatic. Ikkala et al. [23], for example, trained their biomechanical simulations in a simulated environment that contained only the target of the aimed movement and very few or no other objects. Presently, the reality gap plagues practically every aspect of the simulation: the task environments, the rewards, the input device, the display, the feedback, the perceptual inputs of the model (what the model senses), and the control problems (what the model controls) can all be different. This is a problem, because deep RL solutions are sensitive to the state spaces they are trained in. A model trained in a contrived environment is unlikely to behave equally to users interacting in a real-world VR environment.

We present SIM2VR, which aims to close the gap by allowing training models directly in the same VR environment that users experience. It leverages the fact that VR environments already are simulations and there is no need to create a replica of them; rather, we should directly run our models in them. The key enabler in SIM2VR is the Perceptual-Motor Interface, a software component that allows simulation models to access the same camera view and control input as humans (see Figure 1). By creating a platform that can host both real users and simulated users, SIM2VR enables real and simulated users to observe and interact with exactly the same virtual environment. Compared to a user simulation interacting with a simplified low-fidelity version of the interface provided to humans, this makes the simulation more valid and directly comparable. By allowing biomechanical details of user movements to be simulated and predicted *in silico*, SIM2VR also alleviates the long-standing problem of automated testing of VR interaction, which has so far mainly focused on higher-level interaction events such as "the user pushed a UI button" [5, 20, 29, 44]. While the available biomechanical models and RL methods currently limit simulations to relatively simple sensorimotor tasks in VR, as these technologies advance, the modular nature of SIM2VR will immediately enable the direct use of such advances in computational user simulation in VR development.

In summary, we contribute the design and evaluation of a software platform that enables training biomechanical simulations directly in VR environments, providing a way to integrate user models into the real-world development workflow. We test and demonstrate our system by providing predictions of performance, ergonomics, and user strategies in a VR game with several variations, and contribute a ground truth human dataset of 18 users playing the game.

2 RELATED WORK

Below, we situate our work in relation to previous efforts in both automated testing in VR development and computational user modeling and simulation, with particular focus on simulation suites.

2.1 Automated Testing in VR Development

The academic literature on VR development tools covers a diversity of topics such as input and output technologies [11, 26, 50, 57], software toolkits providing abstractions and support for handling diverse hardware [25, 52], and interaction techniques for contexts such as expressive hand interaction [43] and VR games [16].

What has received relatively little attention is a basic problem of VR development: Compared to developing desktop and mobile applications, *iterative testing and development can be slow and cumbersome.* A VR developer typically needs to put on a VR headset and stand up and move around in space to test their work, which causes an overhead compared to testing desktop or mobile software. Because of this, one would ideally want to use *automatic testing* whenever possible. Furthermore, a particular problem VR designers face is the difficulty of predicting the end users' movements and designing for minimal fatigue and simulator sickness [1]. These are issues that traditional software testing automation does not address.

Although the need for automated testing of VR interaction was already identified in the early 2000's [5], a recent study of over 300 VR projects found that 79% of the projects did not utilize any automatic tests [44]. Test automation also followed common software testing practices without considering the user's body movements, focusing on aspects such as evaluating the correctness of a response after an event was triggered [44]. So far, even automated testing approaches designed specifically for VR do not emulate or simulate the user's moving body. Instead, they focus on moving and rotating the viewpoint to inspect VR scenes [56], or operate on higher-level action events such as "click UI button" or "grab object" which abstract away the details of the user's movements [5, 20, 29].

To the best of our knowledge, our work is the first to extend automatic testing of VR applications with a computational user model utilizing biomechanical simulation. This allows automatic testing to shed light on new aspects of user behavior and experience such as which movements a particular VR design might elicit or how difficult a movement task might be. In the domain of Natural User Interfaces (NUIs), a generative motion model has been proposed for automatic testing of gestural interaction [22], but compared to our work, the model was not embedded in a learning loop; instead, it generated random movement sequences without the capability of adapting to the tested interface.

2.2 Computational User Modelling and Simulation

Biomechanical models and computer-based simulations are well-established [2]. Increases in computational power have enabled them to evolve from simple models limited to computation of mechanical loads in static postures [59] to more physiologically-accurate musculoskeletal models [12, 13]. Biomechanical models were typically used for inverse biomechanical simulation, e.g. using the OpenSim ecosystem [48]. Such *inverse simulation methods*, namely inverse kinematics, inverse dynamics, and static optimisation or computed muscle control, allow the estimation of mechanical loads within the human musculoskeletal system, and neural controls of the muscles given motion tracking data of a given user's movement as input [13], and are typically applied in the areas of medicine, rehabilitation, and sports. Computer graphics research on biomechanical simulation and control tends to emphasize visual fidelity and simulation speed rather than scientific insight [33, 34, 41, 45, 49, 51]. *Forward simulation methods* used to be less frequently used in standalone situations, as they require muscle controls as inputs, which are extremely complex to measure experimentally. These have, however, become more useful when applied in combination with computational controllers [14, 32] and efficient physics engines [30], as we will demonstrate in this paper.

RL provides a suitable framework for modelling human behaviour in a flexible way: one only needs to define the states, actions, and rewards and then RL computes the optimal policy [9, 10]. In some cases, the reward function and other parameters can be inferred from human data [3, 28]. When the reward function and state-action space, including their key limitations, are similar to a human's, increasingly human-like behavior has been shown to emerge through learning [42]. Applications in HCI include models of typing, menu selection, multitasking, and visual decision-making [42], and the broader arguments for expanding the use of computational user simulation in HCI are presented in [18, 39]. Using the assumptions of signal-dependent control noise and movement time minimization, Fischer et al. [17] have shown that an RL agent can learn to generate human-like movements with a torque-actuated state-of-the-art model of the upper extremity. The generated movements were in accordance with well-established phenomena such as Fitts' Law [19] and the Two-Thirds Power Law [31]. RL-based simulation may also provide valuable information for predicting usability- and ergonomics-related criteria and to aid in interface design. For instance, Cheema et al. trained a (simplified) torque-actuated biomechanical arm model in a mid-air pointing task, and used the model to predict fatigue of real human subjects performing the task [8]. Leino et al. [35] used RL to learn policies for keystroke-level models, and used this to optimize button arrangements.

By combining perception models, musculoskeletal models, and physically simulated input devices, we can train agents to model and simulate intricate interaction tasks, such as those requiring visuomotor control. A good example of such control is presented in [41], where Nakada et al. introduced a virtual human model and used deep learning to learn reaching and tracking tasks.

2.3 Simulation Suites

A significant step has been the development of computationally efficient, real-time physical simulation software, such as MuJoCo [53]. With this, we can simulate interaction steps quickly enough to use RL for more flexible problem formalisations. This allows a researcher to guide an agent's learning through reward functions, as was demonstrated in [23]. With their User-in-the-Box approach, they provide a novel combination of visuomotor user models, movement-based interaction tasks, and powerful learning methods such as PPO. MyoSuite has a similar approach with a focus on dexterous hand movements, which recently added biomechanical models of hand, neck, and leg models, as well as a conversion tool for OpenSim models (MyoConverter[1]) [7, 55]. As we outline below, these recent advances in the quality, scope, and trainability of biomechanical user models can be directly leveraged by our SIM2VR platform.

3 CHALLENGES IN CLOSING THE REALITY GAP

Integrating user simulations into the VR development process is challenging, for reasons we lay out in this section.

First, simulating the user with the physics engine included in a VR development environment (e.g., the Nvidia PhysX engine implemented in Unity) is generally not feasible. This is because game engines are primarily designed to produce visually appealing rather than biomechanically plausible animations, and are thus typically limited to joint-actuated rigid body models that ignore much of the complexity of the human visuomotor system [58]. Second, reimplementing the VR interaction dynamics in the physics engine used for biomechanical modeling is tedious, highly technical, and time-consuming. In addition, such replicas are often subject to severe simplifications and (unintentional) changes of the actual application (including different game dynamics, different meshes and renderings, missing game objects, deviating movement, positions, orientations, and scalings of virtual objects, and more), which limits the validity of user simulations and increases the reality gap.

To ensure both accuracy and validity, two engines thus need to be run in parallel: The user is simulated within a physics engine that allows for biomechanically plausible motion such as MuJoCo [53] or OpenSim [47], while the VR application is running within a VR development environment such as Unity[2] or Unreal[3] or as a standalone application.

Crucially, these engines must be able to interact with each other, which requires a perceptual-motor interface that sends control input from the user simulation to the VR application and rendered images in the opposite direction. The goal in designing such an interface is to provide the model with exactly the same stimuli that a real user experiences and to provide the same control space to it, thereby closing the reality gap.

The construction of such a perceptual-motor interface poses a number of challenges:

- *Efficient transfer of input and output signals*:
  1. "Virtual" sensor data, e.g. the position and orientation of the controller and the HMD, need to be calculated based on the current user simulation state and sent to the VR environment, as if a real user were interacting with the application.
  2. Conversely, any feedback provided by the VR environment needs to be passed to the simulated user in order to model the user's sensations and perceptions during interaction.
- *Closed-loop coupling*: Updates of the VR environment (changing the state of the virtual game objects and providing a new rendered view of the scene) and the biomechanical simulation (moving the user and the input/output devices) must be coordinated.
- *Spatial alignment*: The initial pose of the simulated user needs to match that of a real user to ensure that game objects can be reached in a similar way and tasks are comparable.
- *Temporal alignment*: The frame rates of both the user simulation and the VR environment need to be aligned appropriately. Time-varying VR application frame rates (e.g., due to computational overhead) must be addressed, as well as the need for faster-than-real-time simulations (e.g., during training of an RL agent to learn interactive user behavior).

[1]https://github.com/MyoHub/myoconverter
[2]https://unity.com/
[3]https://www.unrealengine.com/

To be of practical use, the interface must be embedded in an accessible open source platform that additionally satisfies the following criteria:

- *Modifiability and evaluability*: The platform needs to support arbitrary modifications of both the VR environment and the simulated user. In addition, it should be possible to analyze and evaluate the impact of both system design choices as well as assumptions about the user's rationale and capabilities on the entire interaction loop (e.g., in terms of performance or comfort).
- *Embeddability and generalizability*: The platform needs to be easily integrable into existing workflows of VR designers and generalizable to different user models and VR interaction environments.

4 SIM2VR: A PLATFORM TO SIMULATE USER INTERACTION IN VR ENVIRONMENTS

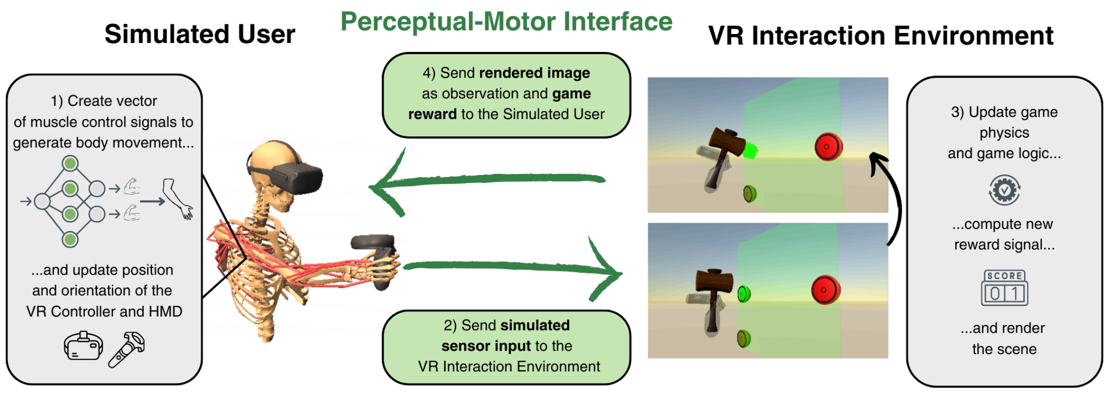

Simulated User
Perceptual-Motor Interface
VR Interaction Environment
1) Create vector of muscle control signals to generate body movement...
...and update position and orientation of the VR Controller and HMD
4) Send rendered image as observation and game reward to the Simulated User
2) Send simulated sensor input to the VR Interaction Environment
3) Update game physics and game logic...
...compute new reward signal...
SCORE
01
...and render the scene

Fig. 2. The *Perceptual-Motor Interface* establishes a continuous closed loop between the *Simulated User* and the *VR Interaction Environment*. Since simulated users see and control the exact same environment as humans, the Perceptual-Motor Interface allows to predict how users interact with a given VR application.

SIM2VR is an open-source platform for integrating biomechanical user simulations into the development process of VR interaction environments. It consists of three main components: 1) The *Simulated User*, which is a biomechanical user model capable of learning interaction tasks; 2) The *VR Interaction Environment*, which defines the virtual environment, and; 3) The *Perceptual-Motor Interface* that connects these two components.

The Simulated User controls the VR Interaction Environment in a *closed-loop* fashion, as depicted in Figure 2. First, the Simulated User generates muscle control signals based on its visual perception of the VR scene and proprioceptive information of the current body posture. Then, the biomechanical simulation is "forwarded" in the sense that the muscle forces are generated that drive movement of the body and, consequently, the hardware devices. The position and rotation of the VR controllers and HMD is then measured by virtual sensors and sent to the VR Interaction Environment using the Perceptual-Motor Interface. The VR Interaction Environment processes this input, computes a game reward based on the updated application state (e.g., the current game score), and then renders the new scene as perceived by the virtual HMD. Finally, the Perceptual-Motor Interface sends the rendered image, the reward, and optional "stateful" information (such as time remaining in a round) to the Simulated User, thus closing the interaction loop.

While the concept of the Perceptual-Motor Interface is agnostic to which physics engine and VR development environment are used, the implementation within the SIM2VR platform is tailored to user simulations running in MuJoCo and VR applications implemented with the Unity engine.

4.1 SIM2VR Components

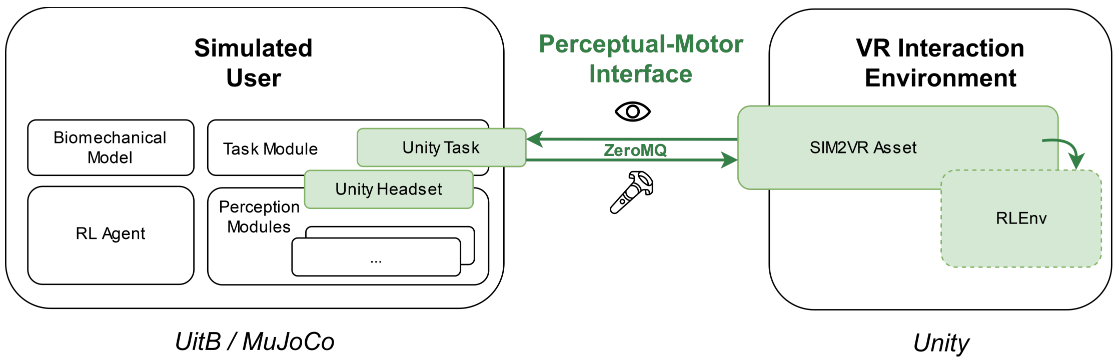


Fig. 3. The *Perceptual-Motor Interface* provided by the SIM2VR platform allows to efficiently transfer data between the *Simulated User* implemeted in the UitB framework and the *VR Interaction Environment* implemented in Unity. The components of the interface are displayed in green.

Simulated User. The Simulated User is based on the UitB framework, which offers a highly modular approach to modeling user biomechanics and interaction. More precisely, each user model consists of a biomechanical model (e.g., implemented in MuJoCo), which is augmented with one or multiple perception modules implemented in Python (see Figure 3). These perception modules define how the Simulated User perceives its surroundings, e.g., via visual or proprioceptive signals. The framework also requires choosing a task module, which sets up the interaction environment and is responsible for providing a task-dependent reward signal (i.e., information about how beneficial a given simulation state is for fulfilling the defined task). During training, this information is then used by the RL Agent to learn how to optimally control the biomechanical user model for the given interaction task and environment.

VR Interaction Environment. The VR Interaction Environment can be any existing Unity VR application or a prototype thereof, either built as a standalone app or run within the Unity Editor. However, it must be compatible with and connected to the Perceptual-Motor Interface in order to run simulations within this environment. This can be achieved by adding our *SIM2VR Asset*, as described below.

Perceptual-Motor Interface. The Perceptual-Motor Interface enables communication between the simulated user and the VR application. It consists of the UitB *Unity Task* class, a UitB vision module called *Unity Headset*, and a Unity asset called SIM2VR Asset. For an illustration of how these components relate to each other, see Figure 3.

Exploiting the modularity of UitB, the Perceptual-Motor Interface uses the concept of a task class to augment the simulated user with VR controllers and an HMD,[4] regardless of which biomechanical user model and which perception modules are otherwise used. The new Unity Task added to UitB

[4]Our platform currently comes with mesh files of the Meta Quest 1 & 2 (https://www.meta.com/quest/) however, these can be easily replaced by models of other controllers and display.

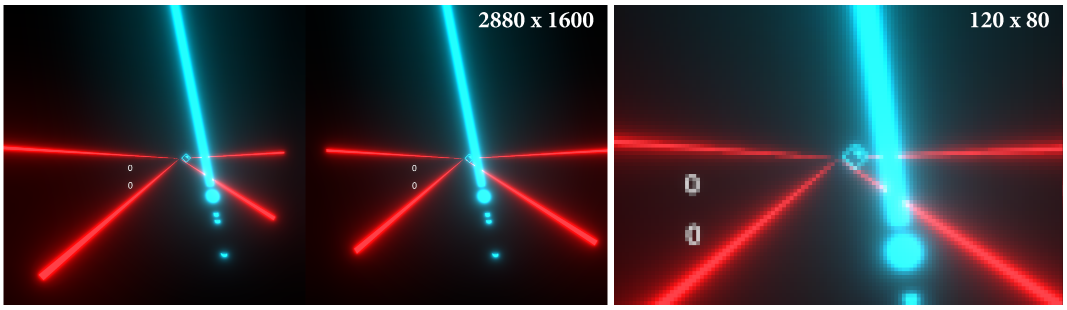


Fig. 4. With SIM2VR, simulated users for the first time perceive exactly the same visual input as humans. In practice, the only difference between the two rendered images displayed on the HMD **(left)** and the RGB-D image provided to the simulated user **(right)** is that we found it practical to reduce image size and resolution to speed up RL training.

ensures that the VR controllers and HMD are rigidly attached to the hands and head of the biomechanical model. In addition, it sends the current position and orientation of the VR controllers and the HMD to the VR Interaction environment and, conversely, makes the information received (i.e., rendered image, game reward, and other "stateful" information) available to the respective UitB modules. The Unity Headset class, which is used as default vision module, models how the rendered image is perceived from the virtual HMD. Since the VR application is agnostic to whether a virtual or "real" HMD is used, the only difference in the visual perception can be introduced by the perception module. For example, it has proven useful to reduce the image size and resolution (see Figure 4) to speed up RL training.

On the other side, the SIM2VR Asset ensures that the VR application is compatible with the simulation, i.e., it is capable of receiving virtual sensor input data and, in turn, sends its output signals back to the user simulation. The SIM2VR Asset also provides *RLEnv*, a Unity script that defines the task-specific rewards, which are then sent to the Simulated User. This RLEnv needs to be modified by the VR delevoper to adapt to the specific game dynamics and tasks under consideration (practical advice on how to define these game rewards is given in Section 5).

To model how users decide for a specific sequence of muscle control signals during interaction, a *policy* mapping perceptions to muscle control signals needs to be learned by the RL Agent. As reward signal, we use a combination of the task-specific rewards provided by the RLEnv and *effort costs* defined by the biomechanical model. This ensures that the Simulated User learns to reduce muscle exertion whenever possible, e.g., by avoiding arm movements that do not contribute to the task, and thus increases biomechanical plausibility. During the training, checkpoints that are automatically stored at a desired frequency can be used to evaluate to which extent the Simulated User has learned to interact with the given VR application, e.g., assessing its performance, ergonomics, and strategies (see Section 6).

4.2 Technical Implementation

In the following, we describe how the Perceptual-Motor Interface was implemented to address the challenges identified in Section 3.

Efficient transfer of input and output signals. We use ZeroMQ[5] to transfer data between the Simulated User implemented in UitB and the VR Interaction Environment implemented in Unity, which are

[5]https://zeromq.org/. We use Python bindings for UitB, .NET implementation for Unity.

running as separate processes, via TCP. These processes send and receive data reciprocally in turns: Unity sends the visual observation as RGB-D array along with other possible stateful information and the reward to UitB, whereas UitB computes and sends the input and output devices' position and rotation, as well as timestamps for synchronising the simulators temporally. For logging and debugging purposes, we also add the option to send any additional data from the VR application to UitB (see Section 5).

Closed-loop coupling. Updates of the Simulated User and the VR environment are synchronized by the *Unity Task* class, which waits for data from the VR environment before the MuJoCo update step is triggered, and the *Simulated User* script included in the SIM2VR Asset, which ensures that Unity frames are aligned in time (see below). The user simulation can also call the reset function of the VR application and vice versa, ensuring that both simulators are reset together when necessary.

Spatial alignment. To ensure that input and output devices are correctly aligned between the two simulators, the positions and rotations of the respective MuJoCo objects are transformed to Unity coordinates (which includes switching from a right-hand to a left-hand coordinate system) by the Unity Task class before being sent to the VR Interaction Environment. Constant offsets between the controller mesh files can also be accounted for.

Temporal alignment. While UitB runs at a constant frame rate,[6] Unity's frame rate is non-constant and typically varies with computational load. To ensure temporal alignment between the two simulators, the Unity application is restricted from sending data until the next timestamp required by the Simulated User (which is sent to Unity via the ZMQ server in advance) has passed.

To enable faster-than-real-time simulations, the Unity `timescale` parameter is set to a value greater than one (in our experience, running Unity at 5x real time has proven to be a good compromise between simulation speed and robustness). To avoid computational overhead, we also make sure that the scene is only rendered when required by the Simulated User.

Modifiability and evaluability. As SIM2VR directly integrates biomechanical user simulations into Unity, the target VR application can be edited as usual. The highly modular UitB framework allows most simulation parameters (e.g., the effort cost weight, which perceptual modules to be used, or hyperparameters of the RL agent) to be easily set from the config file. Due to being fully open-source, the SIM2VR platform can also be supplemented by new biomechanical, perceptual, or cognitive models, additional RL methods, or other features. We also provide evaluation scripts which allow to validate user models and analyze the predicted interaction behavior (see Section 5).

Embeddability and generalizability. With Unity as VR development environment, our platform can be directly integrated into existing workflows of VR designers. In addition, it can be used with arbitrary Unity applications[7] and biomechanical models implemented in MuJoCo.[8]

5 CREATING USER SIMULATIONS IN VR: DESIGN RECOMMENDATIONS

Training an RL agent to generate interactive movement in a biomechanically plausible way is generally very difficult, mainly because of the high-dimensional state-action space, which is generally prone to the *Curse of Dimensionality*, and a lot of complex computations and approximations being lumped together in the simulation process, making it difficult to identify potential flaws and errors.

[6]Note that MuJoCo is typically updated at a higher, constant frame rate to increase simulation accuracy and avoid instabilities [17, 30].

[7]The only requirement of the Unity application is that OpenXR plugin (min. version 1.5.3) is used to handle VR device interaction.

[8]Thanks to conversion tools such as the O2MConverter [24] or MyoConverter [7, 55], our platform can also be used with biomechanical models originally built in OpenSim.

To remedy this, we propose a three-step process that can be performed either prior to the training process, thus serving as design guidelines for selecting an appropriate simulation model, or when problems are observed during training that require further analysis. An overview of the different design choices VR developers face is provided in Table 1.

As an example, we consider the Beat Saber[9]-style game implemented in the *VR Beats Kit*, which is freely available on the Unity Asset Store.[10] A step-by-step guide on how to add SIM2VR to this (or other) existing VR applications can be found in Section A in the Appendix.

5.1 Biomechanical Model

First, we suggest choosing a biomechanical user model with appropriate scope. In principle, any biomechanical model implemented in MuJoCo is conceivable. Currently, this mainly includes models of the arm, elbow, hand, finger, neck, and legs [7, 24, 55]. Most of them make use of musculotendon actuators, however, torque-actuated variants of these models [17] may also be of interest, e.g. when training needs to be fast and ergonomic predictions are of minor importance. As our Beat Saber game requires movements of the VR controller, we decided for the (muscle-actuated) *MoblArmsWrist* model included in UitB. For perception, we use the basic UitB proprioception module, which allows the Simulated User to infer its joint angles, velocities, and accelerations, as well as muscle activations and index finger position, in addition to the standard Unity Headset vision module.

In the context of VR interaction, it is particularly important that the selected biomechanical user model is fundamentally capable of performing each of the movements required for the task under consideration. To achieve this, we provide **BioCheck**, a tool that visualizes the reach envelope of the selected biomechanical model along with the target positions of the VR environment. This is done the following steps. First, the position of the VR controller attached to the biomechanical user model is computed for arbitrary body postures with maximum extended arm. Second, these controller positions are transformed into Unity coordinates, using exactly the same method provided by the Perceptual-Motor Interface that is also applied during simulation. Third, the positions of the targets shown in the VR application are identified and plotted together with the reach envelope. We use BioCheck in Section 6.1 to verify the reachability of the static targets in the Whac-a-mole game (see also Figure 6 for an example plot).

Effort Costs	Task Rewards	Learning Curriculum
Neural [4]	Sparse	Uniformly Random
3CC-r [8, 37]	Dense	Manual Curriculum
Consumed Endurance [21]		Adaptive Automated Curriculum
None		

Table 1. Design choices for the Simulated User relate to task rewards (sparse, extrinsic rewards as provided by the game dynamics/task instructions, or adding continuous distance terms to ensure a dense reward function), effort costs (Neural costs, effort costs as predicted by the 3CC-r fatigue model, Consumed Endurance costs, or no effort costs at all), and learning curricula (sample all game levels or tasks with same probability, or define a curriculum either manually or using our *Adaptive Automated Curriculum*.

[9]https://beatsaber.com/

[10]https://assetstore.unity.com/packages/templates/systems/vr-beats-kit-168243

5.2 Reward Function

Second, a suitable reward function needs to be designed. Following the UitB framework, the reward function consists of an effort term related to the body movement and a task-specific reward term related to the interaction environment.

Effort costs are supposed to incentivise the reduction of muscle exertion to the extent allowed by the task. From a mathematical point of view, these effort costs should penalize a quantity directly related to the muscle control signal to act as a regularization term in the optimization problem (approximately) solved by the RL Agent. In practice, however, any effort-, energy-, exertion-, or fatigue-related cost term is conceivable, as well as a zero effort term incentivising maximum performance regardless of the effort involved. The left column of Table 1 gives an overview of the most relevant effort costs models, all of which are implemented in SIM2VR.[11]

For setting the task-specific rewards in the RLEnv class of the Unity application, an obvious first approach is to simply use the latest change in the game score, if available. While such rewards arguably incentivise the "right" goal, i.e., maximizing the number of points earned in the VR game, they are *sparse* in the sense that the rewards are provided only after the simulated user has already learned how to perform the task correctly (or has accidentally reached the goal, which usually does not happen often enough to learn a reasonable strategy from random exploration alone). To mitigate this issue often referred to as the *temporal credit assignment problem* [6, 38], *dense* rewards need to be added that guide the RL agent to learn how to achieve these goals. We anticipate that for many VR applications, distance rewards that incentivise movement of the VR controller toward the desired target object(s) will be beneficial, e.g., when game objects need to be hit within a certain amount of time or map positions must be reached. To this end, we also provide **RewardCheck**, a tool that allows to analyze how different distance reward functions develop over the course of a game round in different scenarios (worst case, best case, linear/quadratic interpolation between initial distance and zero distance, etc.). As a main benefit, this tool allows to predict and visualize the cumulative sum of each reward component, which helps in scaling these components appropriately. For example, a one-time reward (e.g., provided when successfully hitting a target) should be chosen large enough to compensate for the potentially lower reward achieved afterwards (e.g., because the distance to the next target is much larger and the agent has not learned to yet to move to this target as well). Similarly, the effort costs need to be scaled appropriately in order to effectively reduce muscle exertion whenever possible, while still being motivated to achieve the actual task encoded by the task-specific reward term. As a general rule of thumb, a ratio of 1:10 between effort costs and task-specific rewards (when fully achieved) has proven reasonable in our experience.

For Beat Saber, we use the default Neural effort costs from [23], which penalize the sum of the squared muscle control signals at each time step. As task-specific rewards we use the default sparse rewards obtained from the game scores, because they are easy to implement and not prone to unwanted biases. The effort costs are scaled accordingly.

5.3 Learning Curriculum

Third, many VR environments contain numerous tasks, variations, or game levels. Given that training a biomechanical model to perform a *single*, clearly defined interaction task is already quite challenging, this poses great challenges for the application to "real-world" applications. This issue can be addressed by defining a learning curriculum that determines how and when different conditions are sampled during the training. Two standard options are the *uniformly random task selection* (i.e., all conditions are sampled with the same probability at the beginning of each round)

[11] In contrast to [8], our variant of the 3CC-r model [37] is implemented on a muscle level, i.e., the three compartments (active, resting, and fatigued motor units) are defined and updated for each muscle individually.

and the *manual curriculum* (i.e., conditions are trained in a specific order, with some tasks being authorised only after a fixed number of training steps, or after a certain sub-goal has been achieved). However, it is also possible to define an *Adaptive Automated Curriculum*, where the performance on each task is measured during training, and tasks with a lower success rate are subsequently selected with a higher probability. This variant can be thought of as a "personal trainer" that creates a customized training plan for the Simulated User based on its current strengths and weaknesses.

If a Simulated User persistently fails to perform a given task, but the biomechanical model has shown to be principally able to, and the reward function was carefully and properly designed, it is reasonable conclude that the RL agent failed to learn the optimal policy. This can be caused by many factors, including underspecification of network capacity, an inappropriate observation or control space, bugs in the reset or update functions of either the Simulated User or the VR interaction environment, too few training steps, inappropriate hyperparameters of the chosen RL method, or an overly complex learning curriculum.

To help VR designers investigate these issues, we provide a means to easily log arbitrary variables from the Unity application during simulation. This simply requires referencing the desired variable name in the configuration file and storing the respective values in a predefined dictionary at each time frame from anywhere in the RLEnv script. The logged variables can then be inferred from the Weights and Biases dashboard[12], along with other standard metrics such as mean accumulated reward and mean length per round. For example, it is possible to observe in real time how the target hit rate evolves during training for each rail of the Beat Saber game separately. This can be used to identify errors and learning difficulties early on in training.

In addition, SIM2VR comes with several plotting and visualization tools to evaluate and compare different user simulations.

6 CASE STUDY: "WHAC-A-MOLE" VR GAME

In order to evaluate the ability of a Simulated User trained with our SIM2VR platform to behave similar to users, we developed the VR arcade game *Whac-a-mole*. Whac-a-mole is well-suited for the evaluation, as it requires non-trivial visuomotor coordination, while using a simple but widely used game logic. We implemented different game levels that allow to infer the performance, effort, and strategies of different user models. Thanks to the Perceptual-Motor Interface, it was easily possible to collect additional data from real users performing the same tasks in exactly the same VR environment.

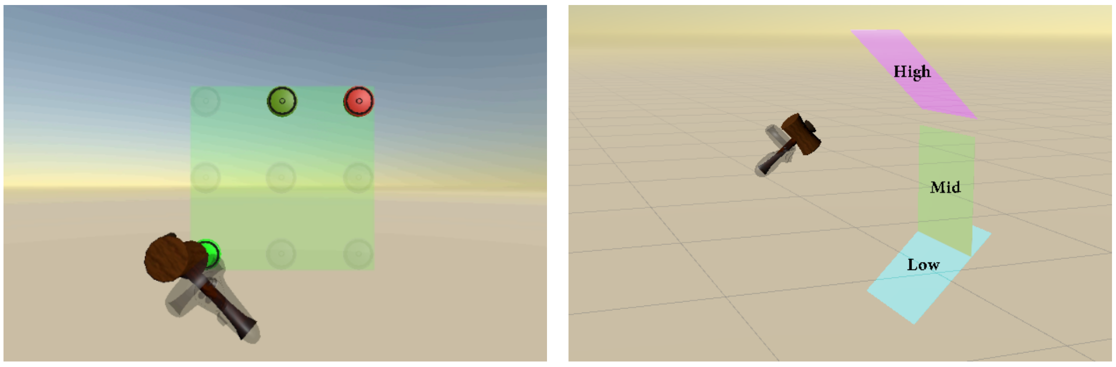
The goal of the Whac-a-mole game is to hit targets (the "moles") with a hammer to score points. Those targets appear randomly on a 3×3 grid as can be seen in Figure 5a. If a target is successfully hit within one second, it explodes and disappears (see Figure 2), and the game score is increased by one. Otherwise, the target collapses and no points are given. The game has three difficulty levels which differ in the number of simultaneously displayed targets (*easy*: 1, *medium*: 3, *hard*: 5). A second attribute, target area placement, defines the position and orientation of the target area with respect to the HMD (*low*, *mid*, *high*), as shown in Figure 5b.

We create two variants of the game: In the *constrained* variant, a velocity threshold needs to be exceeded in order to successfully hit a target, whereas in the *unconstrained* variant, no such threshold exists. Further details on the game design are provided in Section B.1 in the Appendix.

6.1 Training and Evaluating the Simulated Users

The Simulated User models trained on the Whac-a-mole game are defined as follows.

[12]https://wandb.ai/



(a) Front View with Target Grid (b) Target Area Placement Conditions

Fig. 5. (a) In Whac-a-Mole, targets appear randomly for a short time at one of 9 fixed positions and must be hit with a hammer within one second to score a point. The gray targets are shown for visualization purposes only and are not visible during the game. (b) The game allows for three different placements of the target area (low, mid, and high).

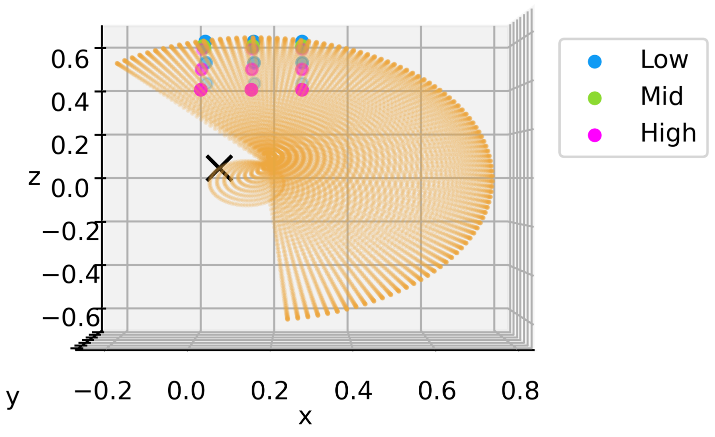


Fig. 6. For Whac-a-mole, the *BioCheck* tool provided by SIM2VR shows that all target positions can be reached by the *MoblArmsWrist* model, although some targets in the low condition may be difficult to reach. The orange lines show the hammer position for arbitrary postures with the arm fully extended, and the black cross indicates the shoulder origin. The plot shows the scene from a bird's eye view, with the user facing in the z-axis direction.

As biomechanical model, we use *MoblArmsWrist*, a MuJoCo version of the *MoBL ARMS model* [46] that includes wrist joints and muscles, resulting in a total of 7 DOFs and 32 muscles. This is a reasonable choice, given that Whac-A-Mole mainly involves movements of the right arm and wrist.[13] Using the *BioCheck* tool included in SIM2VR, we made sure that the model is able to reach all the target positions implemented in our VR game (see Figure 6). Note that performing such checks before starting training can be essential in practice, as problems with reaching certain targets could otherwise easily be (mis)attributed to poor reward design or convergence issues.

[13]Since the model does not actively model neck and head movements, the position and orientation of the HMD were kept constant throughout the simulation.

For the task rewards, we use a dense reward function which adds two terms to the game scores provided by the game logics. First, unsuccessful target contacts are rewarded with a score linearly dependent on the hitting speed (thus encouraging faster hitting speeds). Second, the distances between the hammer and each of the currently active targets are penalized (thus incentivising movement toward any target). As effort costs, we use the muscle-level version of the 3CC-r fatigue model. Using the *RewardCheck* tool included in SIM2VR, we decided to scale the sparse game score obtained from the game dynamics with a factor of 10, leave the dense distance reward and contact terms as they are, and use an effort cost weights of 0.1.

As a task selection method, we decided to sample between the low, mid, and high levels uniformly random, while the game difficulty and strategy were consistently set to *medium* and *constrained*, respectively, during training (i.e., the maximum number of simultaneous targets was always 3, and the velocity threshold was enabled). While it may be reasonable to continue training the final user models on either the Easy or Hard conditions, thereby defining a manual curriculum, we observed that the models trained solely on the Medium condition already generalize well to the remaining difficulties.[14] This Simulated User is denoted as SIM$_u$ in the following.

In addition, we created a second Simulated User SIM$_a$, with the only difference that an *Adaptive Automated Curriculum* is used instead of sampling all target positions with equal probability. In particular, with a probability of 50%, target positions are either sampled with equal probability or depending on the fail rates in the previous round (i.e., target positions that were not or hardly hit previously have a higher probability).[15]

Finally, we trained a separate instance of the SIM$_u$ model on the unconstrained version of the game, which is only used in the latter part of the evaluation.

All models were trained for 100M training steps to ensure comparability. Each of the resulting policies was then evaluated for 12 rounds in each of the three difficulty and each of the three target area placement conditions, resulting in a total of 12*6=72 rounds per Simulated User. Performance measurements for both training and evaluation can be found in Section B.2 in the Appendix.

6.2 User Study

In the user study we collected data from 18 right-handed participants; 12 participants played the constrained variant of the game, while 6 participants played the unconstrained variant. The participants were mostly local graduate or post-graduate students, or faculty members (at [redacted for submission: *name of a university*]). The average age of these participants was 28.8 years, with a standard deviation of 6.2 years; eight of the participants identified as female, nine as male, and one as non-binary. Prior to the experiment, the participants were presented with a study information sheet, a privacy notice explaining data processing and storing procedures, and all signed a consent form to agree to participate in the experiment (with the option to terminate the experiment at any time). The experiment length was approximately 30 minutes, and participants were rewarded with a [redacted for submission: *amount of money and currency*] gift card to a local restaurant.

The independent variables of the study coincide with the difficulty, target area placement, and game strategy attributes of the game described above. We estimate the effect of difficulty with the *hit rate*, i.e., the ratio of targets that were successfully hit before they disappeared to the total number of targets spawned. The effect of target area placement is estimated with the *Borg Rating of Perceived Effort (RPE)* scores reported by the participants. Finally, the effect of game strategy is estimated by qualitative movement analysis of the controller movements. In particular, we presume

[14]As described in Section 6.2, the participants of our user study were also only allowed to "train" on the Medium condition, before the experiment started.

[15]In this case, the probability of a given target position is defined as its fail rate, i.e., the percentage of missed targets that appeared in this position in the previous round, divided by the sum of all fail rates.

Independent variable	Operationalisation	Conditions	Dependent variable
Difficulty	Max. number of simult. targets	(easy, medium, hard)	Hit rate
Target area placement	Position and orientation of target area	(low, mid, high)	Borg RPE
Game strategy	Velocity threshold enabled	(true, false)	Hammer mov. traj.

Table 2. Summary of the operationalisations and conditions of the independent variables along with the dependent variables in the user study.
Abbreviations: *Simult.* stands for *simultaneous* and *mov. traj.* for *movement trajectory.*

that participants may discover a "cheat" strategy in the unconstrained version, where the hammer is kept close to the target area throughout the round, as the targets can be hit with arbitrarily low velocity.

The experiment procedure was as follows. The experiments were conducted in an office room. The participants were seated, but not restricted to the chair. Participants were asked to try to avoid bending and rotation movements of their torso (as the simulated user model has a fixed torso), and focus on their arm movements. Participants were allowed to train with the Medium/mid level. This was to reduce learning effects, and to provide the participants a chance to familiarise themselves with the notion of hitting non-physical floating targets.

In the first part of the experiment, participants performed the three difficulty conditions in counterbalanced order. After completing the first part, participants were given a rest period of at least 5 minutes and were instructed to take as long a break as necessary to ensure that their right arm was not fatigued for the second part of the experiment. During this time, they were also introduced to the Borg RPE score and completed a questionnaire. In the second part, participants performed the three target area placement conditions in counterbalanced order and reported the Borg RPE score after each condition.

For evaluation, only the data from the 12 participants interacting with the constrained variant of the game is used, unless otherwise stated.

6.3 Results

In the following, we analyze and compare how the simulated and real users from Sections 6.1 and 6.2 interact with the Whac-a-mole game in terms of performance and effort. We also discuss specific behaviors observed from both the user study and our simulations, and demonstrate that the simulated user is capable of predicting specific strategies that users may exploit when game dynamics allow.

6.3.1 Performance. To assess the performance of each user, we measure the number of target hits and misses for each of the three difficulty conditions. In Figure 7, the mean number of hits and misses from 12 rounds is shown for both simulated users SIM_u and SIM_a for each condition, along with the means of the 12 real users per condition. In the user study (solid, right bars), both the numbers of target hits and misses increase from the easy to the medium condition. However, this trend does not continue when proceeding to the hard condition. Instead, the number of target hits remains approximately constant (green), while the number of target misses (orange) increases. A similar trend can be observed for both SIM_u and SIM_a (shown as left and mid bars, respectively). In addition, the total number of hits and misses predicted by the simulated users (which can be inferred as the height of the respective bars) for each difficulty condition is well in line with the average performance of the real users, although small underestimations are visible. Here, the simulated user SIM_a matches the real user data slightly better than SIM_u.

Thus, when used in the process of developing the Whac-a-mole VR game, the simulations provided by our platform could have suggested that increasing the task difficulty beyond the

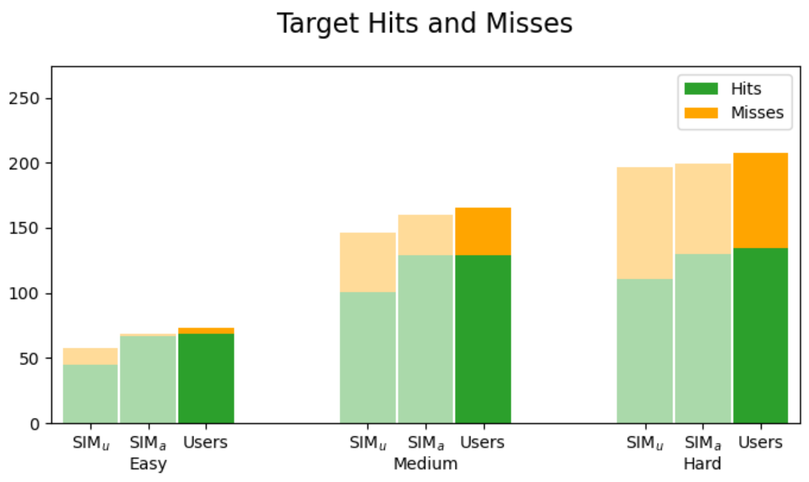


Fig. 7. The simulated users SIM$_u$ and SIM$_a$ (left and mid bars) show similar numbers of target hits and misses in the Whac-a-mole game as the 12 real users from our study on average (right bars). In particular, differences between the three difficulty conditions are predicted well.

medium condition (i.e., allowing more than three targets to appear simultaneously) would not lead to higher game scores for average users, but only to more target misses, possibly overwhelming users.

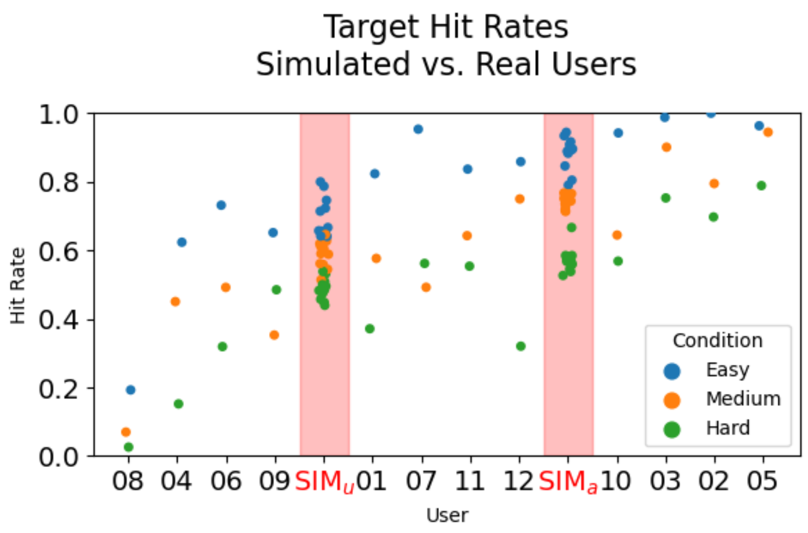


Fig. 8. Both simulated users SIM$_u$ and SIM$_a$ predict target hit rates that are within the between-user variability observed from the user study. In addition, the simulations show a similar decrease in hit rate as the game difficulty increases as most real users.

In Figure 8, the target hit rate, i.e., the percentage of spawned targets that are successfully hit within a round, is plotted separately for each of the three difficulty conditions and for all users. While there is a considerable variability in the performance of real users, both simulations consistently predict target hit rates that lie within this range. Also, a decrease in the hit rate as game difficulty increases can be observed for (almost) all real users as well as the two simulated users. Exceptions are Users 9 and 11, who achieved a higher hit rate on the hard condition than on the medium condition. Interestingly, SIM$_u$ also predicts higher hit rates for some rounds with the hard condition.

User	Variable	Condition	Mean/ Median[†]	Std./ IQR[†]	Alternative Hypothesis	Wilcoxon Signed Rank Z-score	Wilcoxon Signed Rank p-value
SIM$_u$	Max. Fatig. MUs	Low	0.232342	0.001100	Low < Mid	3.059	1.0 (n.s.)
		Mid	0.203537	0.001641	Mid < High	-3.059	0.0002 (***)
		High	0.238448	0.001772	Low < High	-3.059	0.0002 (***)
SIM$_a$	Max. Fatig. MUs	Low	0.265645	0.002151	Low < Mid	-2.51	0.0046 (**)
		Mid	0.269120	0.002360	Mid < High	3.059	1.0 (n.s.)
		High	0.229852	0.001937	Low < High	3.059	1.0 (n.s.)
User Study	Borg RPE	Low	9[†]	1.25[†]	Low < Mid	-1.508	0.0658 (n.s.)
		Mid	9[†]	3[†]	Mid < High	-2.414	0.0079 (**)
		High	9.5[†]	4.25[†]	Low < High	-2.213	0.0134 (*)

Table 3. Overview of both descriptive and inferential statistics for efforts in the Whac-a-mole game. Descriptive statistics include the mean and standard deviation for the maximum fatigued motor units per round (averaged over all muscles), as predicted by the simulation, and the median and interquartile range (IQR) for the Borg RPE as reported in the user study ([†])). Kolmogorov-Smirnov tests showed that none of the considered variables are normally distributed (all p-values < 0.001), so we used one-sided Wilcoxon Signed rank tests to infer significant differences between the three target area placement conditions.
Abbreviations: *Std.* for standard deviation, *IQR* for *interquartile range*, *Max. Fatig. MUs* for *maximum fatigued motor units*, and *n.s.* for *not significant*.

6.3.2 Effort. To assess the impact of placement of the target area on effort, we implemented three different positions (and orientations) of the target area (details are given in Section B.1). For each of these conditions, the participants in the user study were asked to report their perceived exertion after playing a 1-minute round in terms of the Borg RPE scale. The Borg RPE reported for the high condition was significantly higher than for both the low and mid conditions, while no significant difference could be found between low and mid condition (summary statistics as well as details on the statistical testing conducted for this part of the evaluation are given in Table 3). This also agrees well with qualitative remarks of some of the participants, stating that the arm felt considerably more fatigued after playing the high condition as compared to the low and mid conditions. To estimate the level of exertion from our simulation, we measured the percentage of fatigued motor units as predicted by the 3CC-r fatigue model for each simulation time frame, calculated the maximum value per round for each muscle separately, and then took the average over all 32 muscles.

As shown in Table 3, SIM$_u$ predicts a significant increase in fatigue between the mid and high as well as between the low and high conditions, showing the same trend as observed for the Borg RPE scale in the user study. Interestingly, the fatigue predicted for the low condition is considerably higher than in mid condition. Comparing the simulation videos for these conditions, it can be seen that in the low condition, SIM$_u$ follows a strategy that hardly flexes the elbow, but rather keeps the arm extended for almost the entire movement, which may lead to higher muscle fatigue over time.

For the simulated user with adaptive automated curriculum, SIM$_a$, the mid condition exhibits significantly higher fatigue values than the low condition ($p = 0.006$), whereas no significant increase from mid to high or from low to high could be found. We found that this is mainly due to different strategies for each of the three different conditions, despite being trained together and sharing the same network weights. In particular, SIM$_a$ has learned to efficiently reduce unnecessary

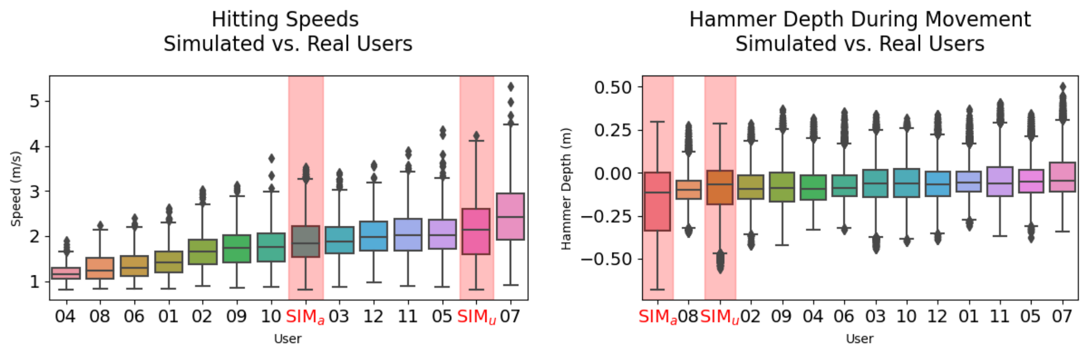


Fig. 9. **Left:** The speed, with which a target was (successfully) hit, greatly varies between users. The hitting speed predicted by the simulation is within this between-user variability. **Right:** The hammer depth during the movement show comparable values between simulated and real users. This indicates that the simulation has learned to retract the arm to a similar extent as the users in our study.

shoulder movements mainly in the high condition, resulting in a lower number of fatigued motor units than in the low and mid condition.

Our findings can be seen as an example of how small changes in the task curriculum and training process may have a considerable impact on which strategies are learned and predicted by the user simulation. Also, while the results show that characteristic differences in subjective levels of muscle exertion can be predicted in principle by our simulation-based approach, effort-related predictions must currently be treated with caution, as the effort can vary with small changes in the movement strategy.

6.3.3 Strategy. Besides predicting differences in performance and effort between different game levels, SIM2VR can also be used to infer characteristics of the movement trajectories as well as non-obvious "strategies" that users may exploit when the game dynamics allow to.

In Figure 9 (left), the speed of the hammer when hitting a target is shown for all simulated and real users. The hitting speeds predicted by the simulated users SIM$_u$ and SIM$_a$ are within the range of the real users, both in terms of mean and variance. While the predictions of SIM$_u$ are at the upper end of this range, those of SIM$_a$ are closer to the average hitting speed. This shows that both simulations generally predict reasonable hitting speeds rather than exhibiting superhuman (or subhuman) behavior.

We also found the "hammer depth", i.e. the position of the hammer in the forward-backward direction, relative to the target plane, to be a key factor for assessing the quality of a learned simulation. As shown in Figure 9 (right), all 12 real users exhibit similar depths offsets in terms of range and mean when playing the Whac-a-mole game. The hammer depths of the two simulation models SIM$_u$ and SIM$_a$ also show comparable values, indicating that the simulations retract the arm to a similar extent as real users. On the other hand, too large negative offsets can be taken as an indicator of a different, often undesired movement strategy. For example, we have found that an inappropriate dense game reward (e.g., when multiple contacts of the same target with too low a speed are rewarded more than successfully hitting the target once) or too low an effort cost often incentivise additional and/or more extensive arm movements than necessary. Similarly, a hammer depth that is consistently well below zero may signal difficulties in learning to move the controller toward the targets at all. If real user data is available (e.g., recorded by the developers

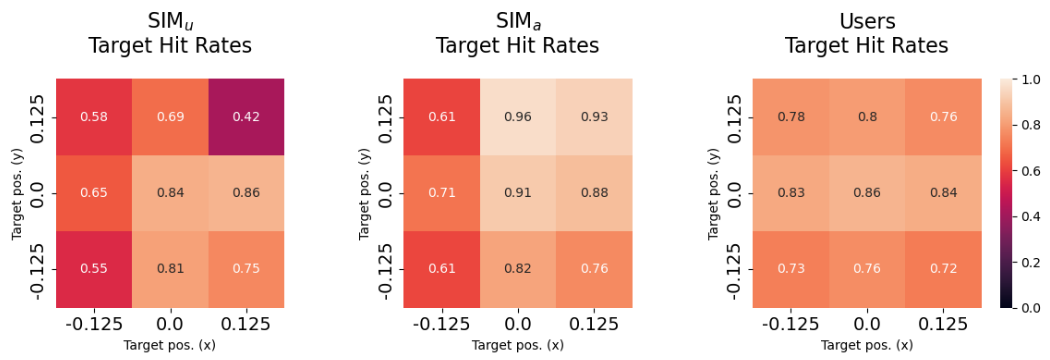


Fig. 10. The hitting rates crucially differ depending on where targets are located relative to the shoulder. In the user study (right), low targets were hit less often than mid and high targets on average. This is captured well by the simulations (left and mid), which, however, also exhibit lower hitting rates for left targets. This could indicate areas that are more difficult for the biomechanical model to reach, or that the policies did not converge to the (global) optimum.

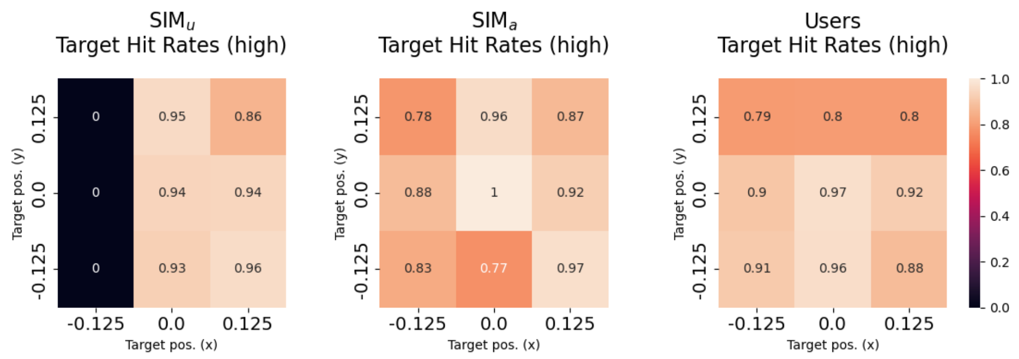


Fig. 11. While SIM$_u$ does not reach the left targets in the high condition at all (left), the simulated user trained with adaptive automated curriculum, SIM$_a$, shows a plausible average target hit rate for any target location (cf. mid and right plot).

themselves using a previous version of an application), we recommend using this data as a baseline for inferring the general plausibility of learned simulation strategies.

The Whac-a-mole game was also designed to test whether certain target positions are preferred by users. Especially in levels with more than one target displayed simultaneously, hitting *all* targets in time is often not possible, i.e., users are typically required to decide which targets are worth aiming at and which should be omitted in case of doubt. As shown in Figure 10 (right) for real users and all task conditions, the three lower targets exhibit a lower *average* hit rate than the remaining target positions. This trend is also visible from the evaluation of SIM$_u$ (left) and SIM$_a$ (mid). In addition, both simulated users show a clear preference towards targets in the mid and right column, whereas targets on the left are more frequently ignored. This strategy was not observed in the user study, but instead may be partially attributed to well-known difficulties of the biomechanical model in reaching targets on the left-front half-sphere (however, note that the preliminary reach envelope analysis in Section 6.1 as well as hit rates larger than zero suggest that all target positions are reachable in principle). Also, positional differences in the target hit rates may be attributed to

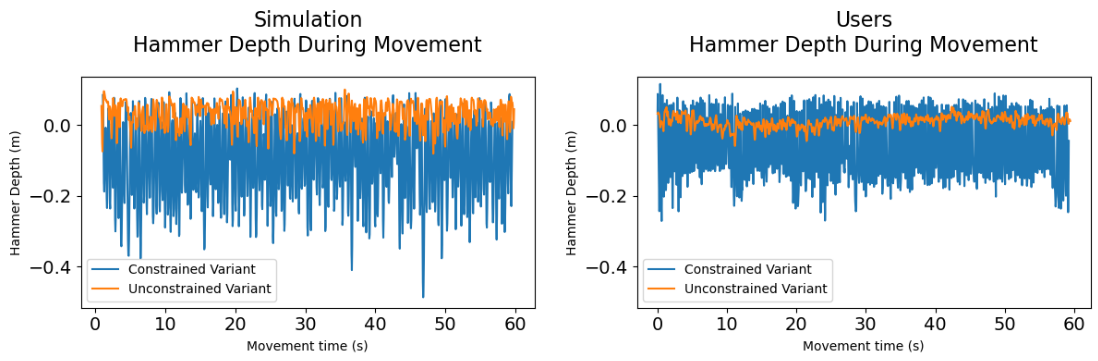


Fig. 12. In the unconstrained part of our user study, i.e., without minimum hitting velocity, our simulation predicts a different strategy, where the hammer is kept close to the target plane during the entire movement (orange line in left plot). This strategy was also observed in the user study (orange line in right plot). The hammer depth trajectories of both simulated and real users in the constrained variant are shown for comparison (blue lines). All movements are in the hard condition, but similar results were also observed for the medium condition.

policies getting stuck in local minima during training. For example, SIM$_u$ is not able to reach the left targets in the high condition at all, while SIM$_a$, which was trained with adaptive automated curriculum, has learned to do so (cf. Figure 11). Again, this highlights the importance of properly designing the simulated user with an appropriate biomechanical model, reward function, and learning curriculum, following the guidelines and tools from Section 5.

To further investigate how well simulations trained via SIM2VR can anticipate the effect of a simple game design choice on the strategies employed by users, we implemented an unconstrained variant of all five game levels in which no minimum velocity was required to successfully hit the targets (for details, see Section B.1 in the Appendix). As shown in Figure 12 (left), for the unconstrained variant, an entirely different strategy is predicted by our simulation; in particular, the unconstrained policy retracts the arm considerably less, but rather keeps the hammer relatively close to the target area during movements, which indeed allows to hit many more targets in the same time.[16] Interestingly, one of the six users from the unconstrained part of the user study exploited the same strategy, as shown in Figure 12 (right, orange line) along with a reference trajectory from the constrained part of the user study (blue line).[17]

This demonstrates the capability of SIM2VR to discover special strategies that users may exploit for a given game design. By making such insights available early in the VR development process, our platform can help find game dynamics that meet the designers' requirements for performance, effort, comfort, predictability, space limitations, and more.

7 OPEN SOURCE

In order to provide developers and researchers with a highly flexible and easily extensible platform, we release all SIM2VR code as open source. The *SIM2VR Asset* along with its source code can be found at [redacted for submission: *link to repository*], which also serves as the landing page of the SIM2VR platform. The *Simulated User* module, which extends the UitB framework to interaction

[16]Applying the two simulation strategies to the hard condition, 195 target hits were observed in the unconstrained variant, and 102 target hits and 35 target contacts were observed in the constrained variant.

[17]Another user pursued a similar strategy but hit the targets more from above, while the remaining four users followed essentially the same strategy as the users in the constrained task.

with Unity environments, is available at [redacted for submission: *link to repository*]. The *BioCheck* and *RewardCheck* tools can be found along with scripts to evaluate and plot (biomechanical) simulation trajectories at [redacted for submission: *link to repository*].

We also release the source code of the Whac-a-mole application at [redacted for submission: *link to repository*]. Finally, the data of the user study as well as the simulations and plots of this paper are publicly available at [redacted for submission: *link to dataset*].

8 DISCUSSION

So far, user simulations have been trained in special-purpose simulation environments with simplified and at-times contrived scenes, dynamics, and graphics. To close the "reality gap", and enable training models in more ecologically valid conditions, we presented SIM2VR, an open-source platform that grants biomechanical models direct access to the actual target VR environment. The Perceptual-Motor Interface enables training models in state-action spaces that can be virtually identical to what real users experience as they interact in VR. As we have demonstrated in the Whac-a-Mole study, human-like policies can be learned in Sim2VR – despite the fact that the stimuli the agent receives from the game is more realistic and therefore more complex than in previous work. The policies replicated some key facets of empirically observed behavior. The average number of target hits per round was predicted within a small error (5% with the closest model variant). Remarkably, the models also closely anticipated specific movement strategies of some individuals in the study, without having data from those users.

We expect SIM2VR to open exciting opportunities in the development of VR interactions. In particular, the ability to perform biomechanical user simulations at an early design stage can lead to more comfortable and health-promoting VR interactions, as well as VR environments that specifically consider the abilities of individual user groups. We also see great potential for the direct integration of user models in related domains, such as mobile applications, augmented reality, or mixed reality.

However, three technical challenges remain. First, more comprehensive biomechanical models are needed. The arm model we use has been developed with inverse biomechanical simulation in mind, and, e.g., can not reach all reachable areas with the same ease. Bespoke biomechanical models for forward simulation will remedy this situation. We also need to cover the full range of VR interaction methods, especially including gesture-based input and head movements. Recently released models covering finger, hand, leg, and neck movements, as well as conversion tools [24, 55] and competition tracks [54], raise hopes that a momentum is building that may also expand the range of VR applications that can be simulated in the near future. Second, modeling how users behave in higher-level interaction tasks and more complex VR applications requires more sophisticated RL models. Related concepts and frameworks from RL research such as *Hierarchical RL* [36, 40] or *Transfer Learning* [60] certainly offer a promising direction in this regard. Third, even more practical guidelines, evaluation tools, and debugging methods are needed to support VR designers without any experience in user simulations or RL to apply SIM2VR to non-standard problems. Our tools *BioCheck* and *RewardCheck* clearly provide good starting points, but much more practical knowledge needs to be gained, which can only be achieved by trying out biomechanical user simulations and subsequently sharing experiences.

9 CONCLUSION

This paper has enabled the training of RL-based user simulations in the same VR environment as users, and shown that realistic policies can be learned like this. By lowering the barrier to the use of biomechanical simulation, we hope that SIM2VR increases the adoption of automatic testing methods in VR development. Given that biomechanical user models can help identify bugs, safety risks, and health issues prior to user testing, we expect user modeling to assume a position as an indispensable part of the VR development process in the future.

REFERENCES

[1] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K Chilana. 2020. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In *[Proceedings of the 2020 CHI conference on human factors in computing systems]*. 1–13.

[2] M. A. Ayoub, M. M. Ayoub, and A. G. Walvekar. 1974. A Biomechanical Model for the Upper Extremity using Optimization Techniques. *[Human Factors]* 16, 6 (1974), 585–594. https://doi.org/10.1177/001872087401600603 arXiv:https://doi.org/10.1177/001872087401600603 PMID: 4442903.

[3] Nikola Banovic, Tofi Buzali, Fanny Chevalier, Jennifer Mankoff, and Anind K. Dey. 2016. *[Modeling and Understanding Human Routine Behavior]*. Association for Computing Machinery, New York, NY, USA, 248–260. https://doi.org/10.1145/2858036.2858557

[4] Bastien Berret, Enrico Chiovetto, Francesco Nori, and Thierry Pozzo. 2011. Evidence for Composite Cost Functions in Arm Movement Planning: An Inverse Optimal Control Approach. *[PLOS Computational Biology]* 7, 10 (10 2011), 1–18. https://doi.org/10.1371/journal.pcbi.1002183

[5] Allen Bierbaum, Patrick Hartling, and Carolina Cruz-Neira. 2003. Automated testing of virtual reality application interfaces. In *[Proceedings of the workshop on Virtual Environments 2003]*. 107–114.

[6] Alexander Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. 2015. Frame Skip Is a Powerful Parameter for Learning to Play Atari. In *[AAAI Workshop: Learning for General Competency in Video Games]*. https://api.semanticscholar.org/CorpusID:194604

[7] Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. 2022. MyoSuite–A contact-rich simulation suite for musculoskeletal motor control. *[arXiv preprint arXiv:2205.13600]* (2022).

[8] Noshaba Cheema, Laura A. Frey-Law, Kourosh Naderi, Jaakko Lehtinen, Philipp Slusallek, and Perttu Hämäläinen. 2020. Predicting Mid-Air Interaction Movements and Fatigue Using Deep Reinforcement Learning. In *[Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems]* (Honolulu, HI, USA) *[(CHI '20)]*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376701

[9] Xiuli Chen, Gilles Bailly, Duncan P Brumby, Antti Oulasvirta, and Andrew Howes. 2015. The Emergence of Interactive Behaviour: A Model of Rational Menu Search. (2015).

[10] Xiuli Chen, Sandra Dorothee Starke, Chris Baber, and Andrew Howes. 2017. A cognitive model of how people make decisions through interaction with visual displays. In *[Proceedings of the 2017 CHI conference on human factors in computing systems]*. 1205–1216.

[11] Carolina Cruz-Neira, Daniel J Sandin, Thomas A DeFanti, Robert V Kenyon, and John C Hart. 1992. The CAVE: audio visual experience automatic virtual environment. *[Commun. ACM]* 35, 6 (1992), 64–73.

[12] Michael Damsgaard, John Rasmussen, Søren Tørholm Christensen, Egidijus Surma, and Mark de Zee. 2006. Analysis of musculoskeletal systems in the AnyBody Modeling System. *[Simulation Modelling Practice and Theory]* 14, 8 (2006), 1100–1111. https://doi.org/10.1016/j.simpat.2006.09.001 SIMS 2004.

[13] Scott Delp, Frank Anderson, Allison Arnold, Peter Loan, A. Habib, Chand John, Eran Guendelman, and Darryl Thelen. 2007. OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement. *[Biomedical Engineering, IEEE Transactions on]* 54 (12 2007), 1940 – 1950. https://doi.org/10.1109/TBME.2007.901024

[14] Christopher L. Dembia, Nicholas A. Bianco, Antoine Falisse, Jennifer L. Hicks, and Scott L. Delp. 2021. OpenSim Moco: Musculoskeletal optimal control. *[PLOS Computational Biology]* 16, 12 (12 2021), 1–21. https://doi.org/10.1371/journal.pcbi.1008493

[15] Konstantinos Dimitropoulos, Ioannis Hatzilygeroudis, and Konstantinos Chatzilygeroudis. 2022. A Brief Survey of Sim2Real Methods for Robot Learning. In *[Advances in Service and Industrial Robotics]*, Andreas Müller and Mathias Brandstötter (Eds.). Springer International Publishing, Cham, 133–140.

[16] Inan Evin, Toni Pesola, Maximus D Kaos, Tuukka M Takala, and Perttu Hämäläinen. 2020. 3pp-r: Enabling natural movement in 3rd person virtual reality. In *[Proceedings of the annual symposium on computer-human interaction in play]*. 438–449.

[17] Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, and Jörg Müller. 2021. Reinforcement Learning Control of a Biomechanical Model of the Upper Extremity. *[Scientific Reports]* 11, 1 (2021), 1–15.

[18] Florian Fischer, Arthur Fleig, Markus Klar, and Jörg Müller. 2022. Optimal Feedback Control for Modeling Human-Computer Interaction. *[ACM Trans. Comput.-Hum. Interact.]* 29, 6, Article 51 (april 2022), 70 pages. https://doi.org/10.1145/3524122

[19] Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *[Journal of experimental psychology]* 47, 6 (1954), 381.

[20] Patrick Harms. 2019. Automated usability evaluation of virtual reality applications. *[ACM Transactions on Computer-Human Interaction (TOCHI)]* 26, 3 (2019), 1–36.

[21] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. 2014. Consumed Endurance: A Metric to Quantify Arm Fatigue of Mid-Air Interactions. In *[Proceedings of the SIGCHI Conference on Human Factors in Computing Systems]* (Toronto, Ontario, Canada) *[(CHI '14)]*. Association for Computing Machinery, New York, NY, USA, 1063–1072. https://doi.org/10.1145/2556288.2557130

[22] Chris J Hunt, Guy Brown, and Gordon Fraser. 2014. Automatic testing of natural user interfaces. In *[2014 IEEE Seventh International Conference on Software Testing, Verification and Validation]*. IEEE, 123–132.

[23] Aleksi Ikkala, Florian Fischer, Markus Klar, Miroslav Bachinski, Arthur Fleig, Andrew Howes, Perttu Hämäläinen, Jörg Müller, Roderick Murray-Smith, and Antti Oulasvirta. 2022. Breathing Life Into Biomechanical User Models. In *[Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology]* (Bend, OR, USA) *[(UIST '22)]*. Association for Computing Machinery, New York, NY, USA, Article 90, 14 pages. https://doi.org/10.1145/3526113.3545689

[24] Aleksi Ikkala and Perttu Hämäläinen. 2020. Converting Biomechanical Models from OpenSim to MuJoCo. , 277–281 pages. arXiv:2006.10618 [q-bio.QM] https://arxiv.org/abs/2006.10618

[25] Tommi Ilmonen and Janne Kontkanen. 2002. Software architecture for multimodal user input-FLUID. In *[ERCIM Workshop on User Interfaces for All]*. Springer, 319–338.

[26] Kazuyo Iwamoto, Satoshi Katsumata, and Kazuo Tanie. 1994. An eye movement tracking type head mounted display for virtual reality system: evaluation experiments of a prototype system. In *[Proceedings of IEEE International Conference on Systems, Man and Cybernetics]*, Vol. 1. IEEE, 13–18.

[27] Jussi Jokinen, Aditya Acharya, Mohammad Uzair, Xinhui Jiang, and Antti Oulasvirta. 2021. Touchscreen typing as optimal supervisory control. In *[Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems]*. 1–14.

[28] Antti Kangasrääsiö, Kumaripaba Athukorala, Andrew Howes, Jukka Corander, Samuel Kaski, and Antti Oulasvirta. 2017. Inferring Cognitive Models from Data Using Approximate Bayesian Computation. In *[Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems]* (Denver, Colorado, USA) *[(CHI '17)]*. Association for Computing Machinery, New York, NY, USA, 1295–1306. https://doi.org/10.1145/3025453.3025576

[29] Kadiray Karakaya, Enes Yigitbas, and Gregor Engels. 2022. Automated UX Evaluation for User-Centered Design of VR Interfaces. In *[International Conference on Human-Centred Software Engineering]*. Springer, 140–149.

[30] Markus Klar, Florian Fischer, Arthur Fleig, Miroslav Bachinski, and Jörg Müller. 2023. Simulating Interaction Movements via Model Predictive Control. *[ACM Trans. Comput.-Hum. Interact.]* 30, 3, Article 44 (jun 2023), 50 pages. https://doi.org/10.1145/3577016

[31] Francesco Lacquaniti, Carlo Terzuolo, and Paolo Viviani. 1983. The law relating the kinematic and figural aspects of drawing movements. *[Acta Psychologica]* 54, 1 (1983), 115 – 130.

[32] Leng-Feng Lee and Brian R Umberger. 2016. Generating optimal control simulations of musculoskeletal movement using OpenSim and MATLAB. *[PeerJ]* 4 (2016), e1638.

[33] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. 2019. Scalable Muscle-Actuated Human Simulation and Control. *[ACM Trans. Graph.]* 38, 4, Article 73 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3322972

[34] Sung-Hee Lee and Demetri Terzopoulos. 2006. Heads up! Biomechanical modeling and neuromuscular control of the neck. In *[ACM SIGGRAPH 2006 Papers]*. 1188–1198.

[35] Katri Leino, Antti Oulasvirta, and Mikko Kurimo. 2019. RL-KLM: Automating keystroke-level modeling with reinforcement learning. In *[Proceedings of the 24th International Conference on Intelligent User Interfaces]*. 476–480.

[36] Alexander C Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. 2019. Sub-policy adaptation for hierarchical reinforcement learning. *[arXiv preprint arXiv:1906.05862]* (2019).

[37] John M Looft, Nicole Herkert, and Laura Frey-Law. 2018. Modification of a three-compartment muscle fatigue model to predict peak torque decline during intermittent tasks. *[Journal of biomechanics]* 77 (2018), 16–25.

[38] Marvin Minsky. 1961. Steps toward Artificial Intelligence. *[Proceedings of the IRE]* 49, 1 (1961), 8–30. https://doi.org/10.1109/JRPROC.1961.287775

[39] Roderick Murray-Smith, Antti Oulasvirta, Andrew Howes, Jörg Müller, Aleksi Ikkala, Miroslav Bachinski, Arthur Fleig, Florian Fischer, and Markus Klar. 2022. What simulation can do for HCI research. *[Interactions]* 29, 6 (2022), 48–53.

[40] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. *[Advances in neural information processing systems]* 31 (2018).

[41] Masaki Nakada, Tao Zhou, Honglin Chen, Tomer Weiss, and Demetri Terzopoulos. 2018. Deep Learning of Biomimetic Sensorimotor Control for Biomechanical Human Animation. *[ACM Trans. Graph.]* 37, 4, Article 56 (jul 2018), 15 pages. https://doi.org/10.1145/3197517.3201305

[42] Antti Oulasvirta, Jussi Jokinen, and Andrew Howes. 2022. Computational Rationality as a Theory of Interaction. In *[Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems]*. 1–14.

[43] Siyou Pei, Alexander Chen, Jaewook Lee, and Yang Zhang. 2022. Hand interfaces: Using hands to imitate objects in AR/VR for expressive interactions. In *[Proceedings of the 2022 CHI conference on human factors in computing systems]*. 1–16.

[44] Dhia Elhaq Rzig, Nafees Iqbal, Isabella Attisano, Xue Qin, and Foyzul Hassan. 2023. Virtual Reality (VR) Automated Testing in the Wild: A Case Study on Unity-Based VR Applications. In *[Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis]* (Seattle, WA, USA) *[(ISSTA 2023)]*. Association for Computing Machinery, New York, NY, USA, 1269–1281. https://doi.org/10.1145/3597926.3598134

[45] Prashant Sachdeva, Shinjiro Sueda, Susanne Bradley, Mikhail Fain, and Dinesh K Pai. 2015. Biomechanical simulation and control of hands and tendinous systems. *[ACM Transactions on Graphics (TOG)]* 34, 4 (2015), 1–10.

[46] Katherine R. Saul, Xiao Hu, Craig M. Goehler, Meghan E. Vidt, Melissa Daly, Anca Velisar, and Wendy M. Murray. 2014. Benchmarking of dynamic simulation predictions in two software platforms using an upper limb musculoskeletal model. *[Computer methods in biomechanics and biomedical engineering]* 5842, May 2016 (2014), 1–14. https://doi.org/10.1080/10255842.2014.916698

[47] Ajay Seth, Jennifer L. Hicks, Thomas K. Uchida, Ayman Habib, Christopher L. Dembia, James J. Dunne, Carmichael F. Ong, Matthew S. DeMers, Apoorva Rajagopal, Matthew Millard, Samuel R. Hamner, Edith M. Arnold, Jennifer R. Yong, Shrinidhi K. Lakshmikanth, Michael A. Sherman, Joy P. Ku, and Scott L. Delp. 2018. OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *[PLOS Computational Biology]* 14, 7 (07 2018), 1–20. https://doi.org/10.1371/journal.pcbi.1006223

[48] Ajay Seth, Michael Sherman, Jeffrey A. Reinbolt, and Scott L. Delp. 2011. OpenSim: a musculoskeletal modeling and simulation framework for in silico investigations and exchange. *[Procedia IUTAM]* 2 (2011), 212–232. https://doi.org/10.1016/j.piutam.2011.04.021 IUTAM Symposium on Human Body Dynamics.

[49] Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2014. Realistic biomechanical simulation and control of human swimming. *[ACM Transactions on Graphics (TOG)]* 34, 1 (2014), 1–15.

[50] David J Sturman and David Zeltzer. 1994. A survey of glove-based input. *[IEEE Computer graphics and Applications]* 14, 1 (1994), 30–39.

[51] Shinjiro Sueda, Andrew Kaufman, and Dinesh K Pai. 2008. Musculotendon simulation for hand animation. In *[ACM SIGGRAPH 2008 papers]*. 1–8.

[52] Tuukka M Takala. 2014. RUIS: A toolkit for developing virtual reality applications with spatial interaction. In *[Proceedings of the 2nd ACM symposium on Spatial user interaction]*. 94–103.

[53] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *[2012 IEEE/RSJ International Conference on Intelligent Robots and Systems]*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[54] Guillaume Durandau Seungmoon Song Chun Kwang Tan Cameron Berg Pierre Schumacher Massimo Sartori Vikash Kumar Vittorio Caggiano, Huawei Wang. 2023. MyoChallenge 23: Towards Human-Level Dexterity and Agility. https://sites.google.com/view/myosuite/myochallenge/myochallenge-2023.

[55] Huawei Wang, Vittorio Caggiano, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. 2022. MyoSim: Fast and physiologically realistic MuJoCo models for musculoskeletal and exoskeletal studies. In *[2022 International Conference on Robotics and Automation (ICRA)]*. IEEE, 8104–8111.

[56] Xiaoyin Wang. 2022. VRTest: an extensible framework for automatic testing of virtual reality scenes. In *[Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings]*. 232–236.

[57] Colin Ware, Kevin Arthur, and Kellogg S Booth. 1993. Fish tank virtual reality. In *[Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems]*. 37–42.

[58] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. 2021. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *[arXiv preprint arXiv:2103.16021]* (2021).

[59] DA Winter. 1984. Biomechanics of human movement with applications to the study of human locomotion. *[Critical reviews in biomedical engineering]* 9, 4 (1984), 287—314. http://europepmc.org/abstract/MED/6368126

[60] Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. 2020. Transfer learning in deep reinforcement learning: A survey. *[arXiv preprint arXiv:2009.07888]* (2020).

A CREATING USER SIMULATIONS IN VR: A STEP-BY-STEP GUIDE

In the following, we demonstrate how SIM2VR can be used to generate user simulations for an existing VR application. As an example, we consider the Beat Saber[18]-style game implemented in the VR Beats Kit, which is freely available on the Unity Asset Store.[19]

A.1 Initial Steps

First, the SIM2VR Asset must be imported into the Unity project. After adding the *sim2vr* prefab as a game object to the desired scene, the *SimulatedUser* game object needs to be connected to the VR Controllers and Main Camera provided by OpenXR.

A.2 Defining the Game Reward and Reset

To run user simulations with the Unity application, appropriate reward and reset methods need to be defined. For this purpose, an application-specific class must be inherited from the *RLEnv* class provided by the SIM2VR asset. Of course, all game objects and variables relevant for the reward calculation must be accessible from this class.

The task-specific reward needs to be computed by the method `CalculateReward` and stored in the variable `_reward`. In the case of VR games that provide a game score, this can be used directly to define the reward.[20] If necessary, this "sparse" game reward can be augmented by additional, more sophisticated terms, as described in Section 5. In the Beat Saber game, we set the reward to the increase in the game score since the last frame.

The method `Reset` is called at the end of each round and therefore needs to ensure that the entire scene is reset to a (reproducible) initial state. This usually includes the destruction of game objects created during runtime and resetting private variables required to compute the game reward.[21] Preparations for the next round, such as choosing a game level or defining variables required for the reward calculations, can also be defined here or, if only need to be called once at the beginning of the game, in the method `InitialiseApplication`. For our Beat Saber game, it is sufficient to simply invoke the existing `onRestart` game event and initialise the reward to the current game score in the method *Reset*.

Finally, the Simulated User needs to be informed about whether the current round has ended. To this end, the variable `_isFinished` needs to be updated accordingly within the method `UpdateIsFinished`. For Beat Saber, we can make use of method `getIsGameRunning` of the *VR_BeatManager* instance.

A.3 Further Adjustments

Since including an application- and task-dependent time feature in the observation provided to the RL agent can be very helpful for learning interactive user behavior, the RLEnv class provides the method `GetTimeFeature`, where such a time feature can be defined. For Beat Saber, we set this to the relative in-game time[22] normalized to values between -1 and 1, as this might help the RL Agent anticipating the deterministic target sequence for a given song.

Generally, a Unity application will commence with an initial scene, such as a menu, rather than directly starting the game. As SIM2VR does not provide a generic method to switch scenes, this needs to implemented manually. In our example, we simply added a game object containing a script that selects the appropriate level and transitions to the *SaberStyle* scene.

[18]https://beatsaber.com/

[19]https://assetstore.unity.com/packages/templates/systems/vr-beats-kit-168243

[20]Note that game scores typically accumulate points throughout the round, so the reward signal should be set to the increase in that score since the last frame.

[21]All code related to resetting the game reward can be summarised in the method `InitialiseReward`.

[22]Note that this requires access to the maximum duration of the round.

Since the *MoblArmsWrist* model is limited to movements of the right arm, we modify the game to only spawn targets for the right saber. Also, since the Simulated User is unable to duck, we removed the walls moving towards the player because they would lead to a game over if touched with the HMD.

A.4 Defining the Simulated User

After preparing the VR Interaction environment for running user simulations, an appropriate Simulated User instance needs to be built.

This mainly involves the selection of a biomechanical user model (including effort costs) and perception modules other than the default Unity Headset vision module, both of which can be easily defined in the config file. Additional parameters to be (optionally) defined in the config file include, among others, the position and orientation of the HMD relative to a body part included in the biomechanical user model, arguments to be passed to the VR application (e.g., setting game level/difficulty), and RL hyperparameters (e.g., network size, time steps, batch size, etc.). For our Beat Saber example, the parameters are chosen as described in Section 5.

B DETAILS OF THE "WHAC-A-MOLE" CASE STUDY

B.1 Game Design

In our Whac-a-mole game, targets (the "moles") appear randomly for a short time at fixed positions on a plane (the "molehills") and must be hit with a hammer to score points. As shown in Figure 5a, the targets correspond to three-dimensional buttons and are spawned on a 3×3 grid, referred to as the "target area" in the following, and the hammer is controlled by the right-hand VR controller. Each target has a life span of one second, during which it linearly changes its colour from green to red. If a target is successfully hit, it explodes and disappears (see Figure 2), and the game score is increased by one. Otherwise, the target collapses after one second and no points are given. During each episode, targets are randomly spawned at variable time intervals between 0 and 0.5 seconds, unless the maximum number of simultaneous targets (which depends on the game level, see below) has already been reached. Also, if there are zero targets in the area (i.e., all targets have been hit), a new target will be spawned instantaneously.

The game levels vary in three different attributes: difficulty, target area placement, and game dynamics. The first attribute, difficulty, determines how many targets can appear simultaneously, and can have a value of 1, 3, or 5. We call these difficulty levels *Easy*, *Medium*, and *Hard*, respectively. The second attribute, target area placement, defines the position and orientation of the target area with respect to the HMD. All areas are positioned 15cm to the right of the HMD, to ensure better reachability for the right arm. We use three different settings for this attribute: in *Low* placement the area is located 30cm below and 35cm in front of the HMD, tilted 45°; in *Mid* placement the area is located 10cm below and 40cm in front of the HMD, facing the user; and in *High* placement, the area is located 20cm above and 30cm in front of the HMD, tilted -45°. The different target area placements are depicted in Figure 5b. Additionally, the third attribute, game dynamics, defines a velocity constraint for hitting the targets. In the *constrained* variant a velocity threshold needs to be exceeded in order to successfully hit a target,[23] while in the *unconstrained* variant all contacts with targets count as hits. Based on the above-mentioned attributes, we defined five different game levels for both the constrained and unconstrained variant. These levels are defined in Table 4.

Additional game parameters that were set constant for all levels include the target radius (2.5 cm), the life span of a target (one second), and the episode duration (one minute).

[23]In the constrained variant, targets need to be hit with a minimum speed of $0.8\ \mathrm{m\,s^{-1}}$ in downward direction for the Low placement and in forward direction for the Mid and the High placements.

Level Name	Difficulty	Target area Placement
Easy	easy	mid
Medium/Mid	medium	mid
Hard	hard	mid
Low	medium	low
High	medium	high

Table 4. Game levels implemented in Whac-a-mole.

B.2 Performance Measurements

Using a workstation with a AMD Ryzen Threadripper 3970X 32-core processor[24], and setting the number of parallel workers to 10, training a simulated model for 100M steps from the scratch took approximately 48-72 hours and required 3.2GB memory on the GPU (NVIDIA GeForce RTX 3090[25]). Most modern GPUs should thus be capable of 3-10 training runs in parallel, which, e.g., can be used to obtain a more robust measure of the quality of a given simulated user model by running the same training with different seeds. Evaluating a learned policy took 30-40 seconds for a one-minute round, depending on whether a video was to be created in addition to the .pickle- and .csv-log files.

[24]https://www.amd.com/

[25]https://www.nvidia.com/

9 Discussion and Future Work

This section provides a synthesis of the body of works presented in the previous chapters. I will discuss different topics therein, such as modeling (e.g., selecting the appropriate objective function) and technical challenges (e.g., computational limitations when solving the OCP for movement-based interaction). In addition to describing these obstacles, I offer several ideas for overcoming them.

9.1 Designing OCPs for Movement-Based Interaction

I proposed a generic framework for the simulation of movement-based interaction with OFC (see Chapters 3 and 5). The core of this framework lies in describing the interaction as an OCP which is then solved to create a movement simulation (as described in Section 1.2). This OCP consists of two parts: the system dynamics and the objective function.

Modeling the System Dynamics of Movement-Based Interaction

The system dynamics for modeling a movement-based interaction, in turn, can be split into three main components: 1) the human, 2) the in- and output devices, and 3) the virtual dynamics (see Fig. 1 in Chapter 3).

First, for the human aspect of the dynamics, current state-of-the-art biomechanical models serve as reference point. Using the proposed framework, I simulated human movements in various tasks, with a focus on those that require movement of only one arm, such as mid-air pointing. To achieve this, a biomechanical model of the right arm and shoulder was utilized. To account for individual users' arm lengths or determine maximum voluntary torques, I collected user data and applied the CFAT tool (refer to section 5.3 in Chapter 3 for further explanation). One benefit of this approach is that there is no necessity to gather information from a user study every time; instead, the modified biomechanical model can be utilized repeatedly for any comparable interaction simulations.

However, to be able to model other tasks, in the future, different and possibly larger biomechanical models are needed. For instance, grabbing a virtual object in VR using a hand tracking technique requires a detailed biomechanical hand model. Although new biomechanical models such as those found in MyoSuite [66] are becoming available, they include numerous additional decision variables and therefore raise the question of whether obtaining optimal control is feasible (this is further discussed in Section 9.2).

Second, input and output devices can be modeled in a simple way and eventually directly incorporated into the physics simulation. For example, instead of actually modeling a motion capturing system, I added a virtual marker to the biomechanical model simulated in MuJoCo. This virtual marker serves as the end-effector which is then fed into the transfer function to obtain the cursor position (see Section 5.1 of Chapter 3). MuJoCo can also model physical devices, such as the VR controller used in Chapter 8. However, to simplify the model, the sensor dynamics of the real controller are neglected, and instead, the position of the VR controller model in the physics simulation is used. Arguably, this approach reduces the realism of the simulation since it does not account for the processes within the controller's electronics that can alter the sensed controller position. Same goes for the model of output devices. Since they are interconnected with the perception abilities of the model, e.g., a perceived image depends on the quality of the screen and the eyesight of the user, they are often modeled together. For instance, in Chapter 7, the agent perceives a RGB-D array with a resolution of 120×80 pixels. Although the low resolution was chosen due to the high computation time of larger inputs, it can be interpreted as using a very low resolution display.

To enhance the realism of simulations, it is possible to incorporate a more realistic controller model. However, this requires knowledge of the physical and sensory characteristics of the device, such as weight, dimensions, or installed micro controllers. Similarly, the output device could be modeled in greater detail, including features such as display refresh rates. Additionally, new models for other input and output devices must be created, such as controllers with triggers or vibration feedback.

Third, to evaluate an interface with simulation, the virtual dynamics have to be known. In the works presented in this thesis, I modeled the virtual dynamics either analytically, (Chapters 3 to 5), inside of a physics simulation that can be forwarded (Chapters 6 and 7), or as an application that is controllable (Chapter 8). All of these representations can take user input and return the change of the interface.

To apply the presented approach to different interactive systems, therefore, the virtual dynamics of the interaction technique either have to be directly accessible or, at least, have to be an opaque box which provides all necessary information. This is usually the case, but can involve some work especially if the application is not available to the researcher as open-source, since additional information, such as positions of interface elements, has to be extracted. In summary, the components of the system dynamics can be modeled based on data from the literature, implementations, or preliminary studies. However, simulating various interaction techniques may still require new models of biomechanics or devices.

Incorporating User Goals into the Objective Function

Defining the correct objective function is reasonably more difficult. The objective function, when minimized (or as in DRL, maximized), should cause the model to perform a human-like movement. This means that it has to incorporate the internal goals of the user and therefore, can involve different objectives such as performance, accuracy, or ergonomics. One part of the objective function that is present in any interaction is given by the interaction task. For example, if the task is to reach an user interface element with a cursor, reaching this element should be rewarded. I have done this by adding explicit and discrete reward (e.g., by giving a single bonus upon target reach, see Chapters 7 and 8) or implicit and continuous reward (e.g., by penalizing the distance to a target in each time step, see Chapter 3). I included task completion time implicitly, by penalizing each time step before the task is solved. Additionally, I include effort terms that increase with higher control or muscle activation. These do not only depend on the task, but also on individual user preferences. Since users possibly also include internalized objective functions, I also added different cost terms that are known to lead to human-like movements such as minimizing jerk or commanded torque change.

One challenge is to balance the different components against each other. Some users may prefer less strenuous movements to faster movements, i.e., the weight for the effort term needs to be higher than for the term penalizing slow movement. To address this, I implemented inverse optimal control approaches, as explained in section 4.2 of Chapter 5 and section 3.4.3 of Chapter 3, to find weights that scale the individual cost terms. In a nutshell, inverse optimal control calculates cost weights that produce a simulation which matches given user data best. For instance, in Chapter 3, cost weights for individual user models are obtained that produce the lowest root mean squared error (RMSE) in terms of joint angles for pointing movements in five different directions (see Section 5.3 in Chapter 3).

To further understand the importance of the choice of cost weights I investigated, what influence changing the cost function weights has on the resulting simulation trajectories (see section 6.3 of Chapter 3). It is affirmed that the simulation results continuously depend on the cost weights, which raises the hope that even if these cannot be obtained perfectly through inverse optimal control, the resulting simulations are still close to the optimal result.

Although I have compared different combinations of cost terms, there are many more candidates that seem promising in the HCI context and have to be further investigated, such as cost of time [55], endpoint variance [40], or fatigue [7, 28, 38]. Moreover, when altering the objective function (e.g., when modeling a completely different task), it might be necessary to re-scale the weight parameters, which in turn requires data from a user study (if inverse optimal control is employed). This dependence on user data limits the framework's application possibilities. It is therefore crucial to find ways to generate objective functions that work for a wide range of tasks without further fine tuning.

Another way to work around the necessity of finding the perfect objective function and fitting cost weights is *Multi-Objective Optimization (MOOP)*. In MOOP, each term of the objective function is considered equally important. Solving a multi-objective optimization problem leads to a whole set of *Pareto-optimal* solutions, also called a Pareto-front, instead of a single optimal solution [71]. A solution of a MOOP is called Pareto-optimal if changing the control in any direction would increase at least one of the objective functions. MOOP is often a good solution when one does not want to or cannot decide on a weight for different objectives. Classic examples include deciding for comfort vs. price when buying a car [11] or mass vs. time of flight in space mission optimization [8]. Applied to the HCI problem of mid-air pointing, for instance, the objectives could be minimizing the distance of the cursor to the target and minimizing the effort. If enough Pareto-optimal solutions have been created, a slider that controls the weight between the objectives could be used to easily compare different simulations – ranging from slow and low effort to fast but strenuous movements.

In summary, selecting the appropriate objective function when designing the OCP is critical yet challenging. Inverse optimal control methods can aid in determining the correct cost weights, but they necessitate human data. Alternatively, MOOP is a promising approach for future research as it considers all aspects of the objective function equally.

9.2 Technical Challenges of Movement Simulation

When working with the simulation of human movements with nonlinear biomechanical models and interaction techniques, there are several technical challenges involved. First, I discuss the most crucial challenge which is given by the complexity of human motor control. Second, this section addresses the formulation of the OCP using single or multiple shot methods. Third, I describe which implications choosing the objective function has on solving the OCP. Lastly, I elaborate on the evaluation of simulations with user data.

The Complexity of High-Dimensional Muscle Control

When working with biomechanical models, it is important to consider the complexity of the human body. The model of the upper extremity I used in Chapters 7 and 8 contained a total amount of 26 controllable muscles which have to be controlled individually. In addition, many joints can be moved by different muscles which leads to considerable redundancy in the creation of movements. For instance, the biceps and brachialis both work on the flexion of the elbow. As a result, similar movements can be produced with different sets of controls, and possibly with similar cost function values. This can result in local optima, in which an optimization algorithm can get stuck.

Another challenge when using optimal control problems is the *curse of dimensionality*. This means that the size of the control space grows exponentially with the number of decision variables [2]. Imagine a system, that only has one decision variable and accepts the controls -1, 0, and 1, i.e., there are three different controls. When another decision variable with the same constraint is added, the control space will now include all possible combinations of the two variables, such as $(-1,-1)$ and $(1,0)$. This results in a total of nine different controls. The curse of dimensionality becomes increasingly severe as decision variables are added due to its exponential nature. For instance, using 26 decision variables in this simple example would result in over 2.5 trillion possible controls. In the case of using muscle activations as controls, these variables are float values which further aggravates this effect.

The computational effort is further exacerbated by the absence of derivative information, which is the case when using physics simulations such as MuJoCo. These derivative information are required for *Newton-Like* methods such as the BFGS[1] [20] used in Chapter 3. Most optimization algorithms use the derivative to find better controls by following the steepest descent. In the absence of this information, finite differences are commonly used to estimate the gradient. This means that each decision variable is varied slightly before the objective

[1]BFGS is a so-called *Quasi-Newton* method which construct second-order derivative information, i.e., the Hessian, recursively [3]. Still, first-order derivatives or approximations of these are necessary.

function is evaluated again to calculate the derivative. Increasing the number of decision variables, e.g., by adding more muscles to the biomechanical model, slows down this process. Using gradient-free algorithms like CMA-ES [26] or BOBYQA [47] does not help either because they require many evaluations of the system dynamics, and each evaluation requires expensive forward simulations. Similarly, DRL agents with complex biomechanical models take days to learn even simple interactions (the agents in Chapters 7 and 8 took up to three days on a regular machine[2]). Beyond that, to model more complex interactions that involve finger or body movements, many more muscles need to be considered (as there are over 30 muscles moving the wrist, hand and fingers).

In many cases, nonlinear problems such as the one introduced here can be locally modeled by linear dynamics. For example, the *iterative LQR (ILQR)* method can be used for nonlinear biomechanical models [37]. The concept of this approach is to linearize the system around each new state and control vector in each iteration. However, using this technique requires an analytical description of the system dynamics, which is not available when using complex biomechanical models like the ones used in this thesis. In addition, linearizations become less precise when nonlinearities have a large impact on a system.

Another solution for this challenge could be using hierarchical control or DRL methods. One promising approach to solve this challenge is *(hierarchical) distributed MPC (DMPC)* which has been successfully implemented for the control of micro grids [5, 34]. The idea of distributed MPC is to split the OCP into many smaller sub-problems that can be computed in parallel by several *sub-unit*. Each sub-unit subsequently computes their own optimal control, considering the currently planned controls of all other units as fixed. The local optimal controls are then sent to a central unit which computes a global strategy and communicates the resulting controls back to the sub-units. After a few rounds of re-optimization, the process converges to an optimal strategy for the whole system. In the case of simulating interaction movements, sub-units could be considered which control only those muscles that work on similar joints. Neurologically, DMPC is related to the theory of muscle synergies, as investigated by d'Avella et al. [9, 10]. By doing so, the computation of the optimal control can be accelerated, allowing for the use of biomechanical models with a large number of muscles. However, the actions of one muscle group are not completely independent of the actions of others. For instance, accelerating the upper arm will also apply force on the lower arm. Therefore, it remains to be shown whether this approach will converge to a sufficient solution in a reasonable time.

There also have been various advances in the research of hierarchical reinforcement learning agents [45]. The idea is to use low-level agents that learn sub-tasks and are in

[2]Such as the NVIDIA GeForce RTX 3090.

turn controlled by high-level agents to fulfill the given main goal. Training the low-level agents is faster than learning the complete control, and they can further be used for different high-level agents, depending on the task. In case of the simulation of interaction movements, a low-level agent could learn to produce postures by actuating muscles, while a high-level agent would then dictate which sequence of postures would be needed to fulfill the interaction task. The low-level agent could then also be reused for different conditions without the need of extensive retraining.

In the future, technical advances in processor performance will help mitigate these challenges. Currently, the use of more powerful GPUs can already accelerate the training process of DRL agents. However, discovering more efficient methods for solving the OCP will enable faster and more energy-efficient computation of simulations.

Single-Shooting vs. Multiple Shooting

Another technical challenge is the formulation of the OCP, i.e., how it is presented to the solving algorithm. This formulation determines the *decision variables*, i.e., the variables that the solver can adjust to find the best solution. In Chapter 3, the decision variables are the activation values, e.g., the muscle activations (see OCP (1.4)). From an HCI perspective, this is reasonable since the user controls these activations. This approach is also called *single-shooting* because to evaluate one possible realization of the decision variables, the whole state-trajectory has to be evaluated while technically ensuring that the system dynamics are maintained. In this case, the system dynamics are not defined as actual constraints during the optimization but rather result implicitly. In case of MPC, each evaluation of a control trajectory $u(\cdot)$ thus requires the simulation of the entire movement up to the MPC horizon $N \in \mathbb{N}$, i.e., evaluating $x_u(k+1, x_0) = f(x_u(k, x_0), u(k))$ for $k = 0, ..., N-1$. This results in a state trajectory $x_u(\cdot, x_0)$ which is then used to calculate the cost function value. Single-shooting is easy to implement and keeps the number of decision variables manageable. However, one technical downside of this approach is that the optimizer cannot use the structural information of the problem, e.g., that each control value only directly affects the next state.

This property can be exploited for faster computations using *multiple shooting* [4] or *collocation* [57] methods, where the OCP is discretized in the activation *and* state variables, i.e.,

$$\min_{u(\cdot),\, x(\cdot)} \sum_{k=0}^{N-1} \ell(x(k),u(k)) \tag{9.1}$$

$$\text{such that } x(k+1) = f(x(k),u(k)), \quad x(0) = x_0, \tag{9.2}$$

$$x(k) \in \mathbb{X}, u(k) \in \mathbb{U}, \quad \text{for all } k \in \mathbb{N}. \tag{9.3}$$

The system dynamics are then included in actual constraints that define how the state variables depend on the prior state and the corresponding control values, given in (9.2). A solver may decide to violate these constraints during optimization in order to find a solution more quickly, e.g., by solving for the end of the movement first. Additionally, because of the shape of the constraints (9.2), their Jacobian and Hessian are (block) sparse, which make them easier to compute. On the downside, formulating the OCP in this way significantly increases the number of decision variables which, as mentioned above, slows down the optimization - especially when no derivative information is available.

Implications of Choosing the Objective Function

From a technical perspective, some objective functions are beneficial for methods that can solve OCPs. For example, including a term that directly penalizes the control variables quadratically usually helps to improve the convergence of optimization algorithms since convexity guarantees that any local minimum is also a global minimum. In all of the presented works, such a regularization term is used. In general, optimization algorithms work better with smooth objective functions that are, optimally, of a convex shape. In particular, from an optimal control perspective, using discrete rewards as mentioned above is not recommended since they do not exhibit a gradient that leads towards the goal. Nonetheless, a reinforcement learning agent may still be able to learn reasonable movements, as I have shown in Chapters 6 to 8. Furthermore, discrete rewards make fewer assumptions about the user's intention. For instance, employing distance costs implies that the user prefers being in close proximity to a target at all times which may not be the case. Such costs represent a more restrictive rule than a one-time reward that is given upon reaching the target – regardless of the movement made. In addition, interactive systems often directly provide discrete rewards, such as scores in video games. Therefore, it is still appropriate to use discrete rewards, even though they may have negative mathematical implications. By enabling the direct integration of user simulation into VR development environments in Chapter 8, I simplify using such discrete rewards.

Challenges in Comparing Simulations to User Data

To evaluate whether simulations produce movements that are similar to that of humans,the resulting motion has to be compared to observed data from user studies. However, the acquisition of user data is cumbersome for several reasons. First, since interaction movements can be very small and fast, a motion capturing system is required that is able to record accurate high frequency motion. Second, the study participants need to wear trackers or special suits. Third, the logging of the motion capturing data and data from the experiment application (e.g., Unity[3]) have to be spatially and temporally aligned. Further, to be able to compare simulation and human data, the initial conditions must be aligned. It is especially important to align initial postures and muscle activation when comparing single movements.

Another challenge is modeling the process of users learning a new interaction technique. So far, our simulations assume that the user is an expert for the considered system. However, if a more complex interaction technique, such as the Virtual Pad from Chapter 3, is used, mastering it may require extensive training. Additionally, humans may be biased towards familiar interaction techniques, which could make learning a new system difficult. For instance, the user may be accustomed to using a tablet on a table and therefore feel more comfortable with a horizontally placed Virtual Pad. However, after some training, a tilted Virtual Pad may result in less strenuous movements. Including the learning process in simulations can aid in introducing new interaction techniques and provide insight into how new users interact with a system. However, this requires a sophisticated cognitive model of the user – in addition to the already complex biomechanical model.

9.3 Combining and Extending Models for Interaction

Combining MPC and DRL

The studies in this dissertation have investigated both DRL and OFC methods to simulate movement in HCI. DRL requires many evaluations of the system dynamics during training and thus needs long to learn, especially when using complex physics simulations as part of the system dynamics. As a benefit, once a certain task is learned, it can be sampled very quickly. MPC in contrast, does no require prior training, but needs evaluations of the system dynamics in each time step of a simulated movement.

A promising future research direction lies therefore in combining these two approaches. This approach combines the strength of the fast sampling of DRL with the explainability and optimality guarantees gained by MPC. A DRL agent could be trained to learn the underlying

[3]https://unity.com/

system dynamics, while the MPC is used to find optimal controls, using the estimation of the system dynamics [31].

Adding Improved Perception Models

The model used for the simulation with MPC in Chapter 3 assumes total observability, i.e., the simulated user knows the exact position of the cursor and the state of their body (or biomechanical system) at any time. In reality, humans perceive their environment via various senses. In our case, this perception is mostly visual and proprioceptive. However, haptic perception also plays an important role in many interaction techniques. I have investigated visual perception models for mouse pointing simulated with LQG (see Chapter 5), as well as visual and haptic perception for more complex tasks using DRL (see Chapters 7 and 8).

Augmenting also the MPC with perceptual models will increase the realism of generated movements and allow for the modeling of occlusion that can occur during interaction, as well as perceptual delays. Technically, this will require the implementation of an observer and, depending on the model, of probabilistic MPC [31]. The results could then be compared to DRL approaches that use similar perception models.

Utilizing Contact Force Models

The physics engine I used to model the biomechanics, MuJoCo [60], has a sophisticated model to calculate contact forces. Although I have used this in controlling a joystick in Chapter 7, I have not fully exploited its capabilities. Not only can it model the interaction with complex input devices like triggers or touchscreens, it can also model interactions with spatial constraints, such as mid-air interactions while sitting on an airplane. Future applications may also include devices that are capable of producing force feedback like haptic gloves or full body exoskeletons.

Creating Adaptive Interaction Techniques

It is quite possible that future advances in computing combined with improved methods will enable accurate (faster than) real-time simulation of human motion during interaction. This will allow us to create a digital twin for instant motion prediction of individual users which will be the basis for developing predictive interaction techniques for faster and more intuitive interactions. Those techniques could adapt online to changed user intentions or, in case of mixed reality applications, changed environments.

10 Conclusion

User studies are considered the gold standard for evaluating interactive systems. However, investigating movement-based interactions, which are included in most VR or AR applications, can be particularly cumbersome and costly. Instead, in silico testing provides an alternative, but previous models fail to capture user movements on a level that allows for proper evaluation of quantities such as ergonomics or effort. As a solution, I presented a framework for modeling HCI tasks that is based on OFC theory. This framework enables biomechanical simulation of interaction movements using methods from OFC and DRL. Consequently, this facilitates the automated evaluation of the ergonomics of interactive systems already at early stages of prototyping, reducing the risk of expensive fixes later on. By incorporating a model of the user's body into simulations, it becomes possible to predict difficult-to-measure quantities such as muscle activation or fatigue. This provides insight into user strategies and can help achieve a more inclusive view of HCI by enabling the evaluation of interfaces for users with different physiologies, rather than relying on a one-size-fits-all approach. Since the simulation is created by optimizing an objective function, the resulting movements resemble those of experienced users. This saves necessary training time compared to classical user testing.

I have demonstrated, how this framework can be applied to different interaction tasks such as pointing, tracking, or choice reaction, using the mouse or mid-air techniques in VR.

I have shown that OFC and DRL methods can simulate human movements in interaction scenarios in a way that provides insight into ergonomics of a system and user behavior, using state-of-the-art biomechanical models. For instance, I demonstrate that utilizing a Linear-Quadratic Regulator for simulating mouse-pointing is more effective than previous methods in interpreting user data. Using Model Predictive Control and biomechanical user models, I can predict joint movements of users during a mid-air pointing tasks. The trajectories resulting from the simulations fall within the range of between-user variance, indicating that simulations could be used as a partial replacement for user studies in this particular case. In addition to classical control-theoretic approaches, I show how reinforcement learning methods can be used to simulate user movements in different VR tasks and present a system that allows developers to run simulations while working with a development environment.

Ultimately, I aim to make my work usable for HCI researchers and practitioners. For instance, the SIM2VR platform simplifies the utilization of simulations by providing an integration of user simulations directly into the development environment. To ease the application of the proposed OFC framework, several open-source toolboxes, as well as guidelines and tutorials are provided in the corresponding works. These help to model and simulate any movement-based interactions and can be used to evaluate existing or novel interaction techniques.

In the future, the proposed approaches can be further improved by implementing larger biomechanical models or by addressing the technical challenges, as discussed in Section 9.1. In addition, the physics simulations MuJoCo includes a high-fidelity contact model that allows to simulate particular interaction scenarios, such as interaction in constrained space. If simulations are sufficiently fast, they can be used to automatically optimize interaction techniques for performance, ergonomics, or effort. Running these optimizations online could assist the interactive system in adapting to changes in the environment during interaction, which is common in mixed reality situations. For these reasons I am certain that simulating user movements will play a crucial role in the future of HCI. Leveraging these findings can support the evaluation of existing as well as novel interaction techniques and lays the foundation for automatic optimization of interactions in regards to any quantity such as performance, effort, or ergonomics.

Bibliography

[1] Baloup, M., Pietrzak, T., and Casiez, G. (2019). Raycursor: A 3d pointing facilitation technique based on raycasting. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA. Association for Computing Machinery.

[2] Bellman, R. (1957). Dynamic programming.

[3] Betts, J. T. (2020). *Practical Methods for Optimal Control Using Nonlinear Programming*, chapter 1, pages 1–46. siam.

[4] Bock, H. G., Diehl, M. M., Leineweber, D. B., and Schlöder, J. P. (2000). A direct multiple shooting method for real-time optimization of nonlinear dae processes. In Allgöwer, F. and Zheng, A., editors, *Nonlinear Model Predictive Control*, pages 245–267, Basel. Birkhäuser Basel.

[5] Braun, P., Grüne, L., Kellett, C. M., Weller, S. R., and Worthmann, K. (2016). A distributed optimization algorithm for the predictive control of smart grids. *IEEE Transactions on Automatic Control*, 61(12):3898–3911.

[6] Card, S. K. E. (1983). *The Psychology of Human-Computer Interaction (1st ed.).* CRC Press.

[7] Cheema, N., Frey-Law, L. A., Naderi, K., Lehtinen, J., Slusallek, P., and Hämäläinen, P. (2020). Predicting mid-air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA. Association for Computing Machinery.

[8] Coverstone-Carroll, V., Hartmann, J., and Mason, W. (2000). Optimal multi-objective low-thrust spacecraft trajectories. *Computer Methods in Applied Mechanics and Engineering*, 186(2):387–402.

[9] d'Avella, A. and Lacquaniti, F. (2013). Control of reaching movements by muscle synergy combinations. *Frontiers in computational neuroscience*, 7:42.

[10] d'Avella, A., Saltiel, P., and Bizzi, E. (2003). Combinations of muscle synergies in the construction of a natural motor behavior. *Nature Neuroscience*, 6(3):300–308.

[11] Deb, K. and Deb, K. (2014). *Multi-objective Optimization*, pages 403–449. Springer US, Boston, MA.

[12] Diedrichsen, J., Shadmehr, R., and Ivry, R. B. (2010). The coordination of movement: optimal feedback control and beyond. *Trends in cognitive sciences*, 14(1):31–39.

[13] Faisal, A. A., Selen, L. P., and Wolpert, D. M. (2008). Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303.

[14] Faulwasser, T., Grüne, L., and Müller, M. A. (2018). Economic nonlinear model predictive control. *Foundations and Trends® in Systems and Control*, 5(1):1–98.

[15] Fischer, F., Bachinski, M., Klar, M., Fleig, A., and Müller, J. (2021). Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports*, 11(1):1–15.

[16] Fischer, F., Fleig, A., Klar, M., Gruene, L., and Mueller, J. (2020). An optimal control model of mouse pointing using the lqr.

[17] Fischer, F., Fleig, A., Klar, M., and Müller, J. (2022). Optimal feedback control for modeling human-computer interaction. *ACM Trans. Comput.-Hum. Interact.* In Press.

[18] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381.

[19] Flash, T. and Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703.

[20] Fletcher, R. (2000). *Newton-Like Methods*, chapter 3, pages 44–79. John Wiley & Sons, Ltd.

[21] Gawthrop, P., Loram, I., Lakie, M., and Gollee, H. (2011). Intermittent control: A computational theory of human control. *Biological Cybernetics*, 104(1-2):31–51.

[22] Gori, J., Rioul, O., and Guiard, Y. (2018). Speed-accuracy tradeoff: A formal information-theoretic transmission scheme (fitts). *ACM Trans. Comput.-Hum. Interact.*, 25(5).

[23] Grüne, L. and Pannek, J. (2017). *Nonlinear Model Predictive Control. Theory and Algorithms*. Springer, London, 2nd edition.

[24] Guigon, E., Baraduc, P., and Desmurget, M. (2007). Computational motor control: Redundancy and invariance. *Journal of Neurophysiology*, 97(1):331–347. PMID: 17005621.

[25] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

[26] Hansen, N. (2016). The cma evolution strategy: A tutorial.

[27] Harris, C. M. and Wolpert, D. M. (1998). Signal-dependent noise determines motor planning. *Nature*, 394(6695):780–784.

[28] Hetzel, L., Dudley, J., Feit, A. M., and Kristensson, P. O. (2021). Complex interaction as emergent behaviour: Simulating mid-air virtual keyboard typing using reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4140–4149.

[29] Ikkala, A., Fischer, F., Klar, M., Bachinski, M., Fleig, A., Howes, A., Hämäläinen, P., Müller, J., Murray-Smith, R., and Oulasvirta, A. (2022). Breathing life into biomechanical user models. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, New York, NY, USA. Association for Computing Machinery.

[30] Jokinen, J., Acharya, A., Uzair, M., Jiang, X., and Oulasvirta, A. (2021). *Touchscreen Typing As Optimal Supervisory Control.* Association for Computing Machinery, New York, NY, USA.

[31] Kamthe, S. and Deisenroth, M. (2018). Data-efficient reinforcement learning with probabilistic model predictive control. In *International conference on artificial intelligence and statistics*, pages 1701–1710. PMLR.

[32] Kawato, M. (1993). Optimization and learning in neural networks for formation and control of coordinated movement. In *Attention and performance XIV (silver jubilee volume) synergies in experimental psychology, artificial intelligence, and cognitive neuroscience*, pages 821–849.

[33] Klar, M., Fischer, F., Fleig, A., Bachinski, M., and Müller, J. (2023). Simulating interaction movements via model predictive control. *ACM Trans. Comput.-Hum. Interact.*, 30(3).

[34] Kong, X., Liu, X., Ma, L., and Lee, K. Y. (2019). Hierarchical distributed model predictive control of standalone wind/solar/battery power system. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(8):1570–1581.

[35] Lacquaniti, F., Terzuolo, C., and Viviani, P. (1983). The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica*, 54(1):115 – 130.

[36] Lewis, F. L., Vrabie, D., and Syrmos, V. L. (2012). *Optimal control.* John Wiley & Sons.

[37] Li, W. and Todorov, E. (2004). Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics, (ICINCO 2004)*, volume 1, pages 222–229.

[38] Looft, J. M., Herkert, N., and Frey-Law, L. (2018). Modification of a three-compartment muscle fatigue model to predict peak torque decline during intermittent tasks. *Journal of biomechanics*, 77:16–25.

[39] Martín, J. A. A., Gollee, H., Müller, J., and Murray-Smith, R. (2021). Intermittent control as a model of mouse movements. *ACM Trans. Comput.-Hum. Interact.*, 28(5).

[40] Messier, J. and Kalaska, J. F. (1999). Comparison of variability of initial kinematics and endpoints of reaching movements. *Experimental brain research*, 125:139–152.

[41] Müller, J., Oulasvirta, A., and Murray-Smith, R. (2017). Control theoretic models of pointing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(4):1–36.

[42] Murray-Smith, R., Oulasvirta, A., Howes, A., Müller, J., Ikkala, A., Bachinski, M., Fleig, A., Fischer, F., and Klar, M. (2022). What simulation can do for hci research. *Interactions*, 29(6):48–53.

[43] Nakano, E., Imamizu, H., Osu, R., Uno, Y., Gomi, H., Yoshioka, T., and Kawato, M. (1999). Quantitative examinations of internal representations for arm trajectory planning: minimum commanded torque change model. *Journal of Neurophysiology*, 81(5):2140–2155.

[44] Newell, A. (1994). *Unified theories of cognition*. Harvard University Press.

[45] Pateria, S., Subagdja, B., Tan, A.-h., and Quek, C. (2021). Hierarchical reinforcement learning: A comprehensive survey. *ACM Comput. Surv.*, 54(5).

[46] Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (1996). The go-go interaction technique: Non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, UIST '96, page 79–80, New York, NY, USA. Association for Computing Machinery.

[47] Powell, M. J. et al. (2009). The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 26.

[48] Qian, N., Jiang, Y., Jiang, Z.-P., and Mazzoni, P. (2013). Movement duration, fitts's law, and an infinite-horizon optimal feedback control model for biological motor systems. *Neural computation*, 25(3):697–724.

[49] Qin, S. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764.

[50] Rawlings, J., Mayne, D., and Diehl, M. (2017). *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2nd edition.

[51] Rogers, Y. (2004). New theoretical approaches for hci. *Annual review of information science and technology*, 38(1):87–143.

[52] Rutledge, J. D. and Selker, T. (1990). Force-to-motion functions for pointing. In *Proceedings of the IFIP TC13 3rd International Conference on Human-Computer Interaction*.

[53] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

[54] Scott MacKenzie, I. and Riddersma, S. (1994). Effects of output display and control—display gain on human performance in interactive systems. *Behaviour & Information Technology*, 13(5):328–337.

[55] Shadmehr, R., de Xivry, J. J. O., Xu-Wilson, M., and Shih, T.-Y. (2010). Temporal discounting of reward and the cost of time in motor control. *Journal of Neuroscience*, 30(31):10507–10516.

[56] Soria, E., Schiano, F., and Floreano, D. (2021). Predictive control of aerial swarms in cluttered environments. *Nature Machine Intelligence*, 3(6):545–554.

[57] Tamimi, J. and Li, P. (2009). Nonlinear model predictive control using multiple shooting combined with collocation on finite elements. *IFAC Proceedings Volumes*, 42(11):703–708. 7th IFAC Symposium on Advanced Control of Chemical Processes.

[58] Todorov, E. (1998). Studies of goal-directed movements. Massachusetts Institute of Technology.

[59] Todorov, E. (2005). Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural computation*, 17(5):1084–1108.

[60] Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033.

[61] Todorov, E. and Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235.

[62] Umberger, B. R. and Miller, R. H. (2018). Optimal control modeling of human movement. *Handbook of human motion*, pages 327–348.

[63] van Beers, R. J., Haggard, P., and Wolpert, D. M. (2004). The role of execution noise in movement variability. *Journal of Neurophysiology*, 91(2):1050–1063. PMID: 14561687.

[64] Van der Helm, F. C. and Rozendaal, L. A. (2000). Musculoskeletal systems with intrinsic and proprioceptive feedback. In *Biomechanics and neural control of posture and movement*, pages 164–174. Springer.

[65] Vazquez, S., Rodriguez, J., Rivera, M., Franquelo, L. G., and Norambuena, M. (2017). Model predictive control for power converters and drives: Advances and trends. *IEEE Transactions on Industrial Electronics*, 64(2):935–947.

[66] Vittorio, C., Huawei, W., Guillaume, D., Massimo, S., and Vikash, K. (2022). Myosuite – a contact-rich simulation suite for musculoskeletal motor control. https://github.com/myohub/myosuite.

[67] Wada, Y., Kaneko, Y., Nakano, E., Osu, R., and Kawato, M. (2001). Quantitative examinations for multi joint arm trajectory planning—using a robust calculation algorithm of the minimum commanded torque change trajectory. *Neural networks*, 14(4-5):381–393.

[68] Wang, J. and Katayama, M. (2011). Optimal model for selecting human arm posture during reaching movement. In *Advances in Cognitive Neurodynamics (II)*, pages 453–458. Springer.

[69] Wentzel, J., d'Eon, G., and Vogel, D. (2020). Improving virtual reality ergonomics through reach-bounded non-linear input amplification. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA. Association for Computing Machinery.

[70] Zhou, Y., Wang, M., and Ahn, S. (2019). Distributed model predictive control approach for cooperative car-following with guaranteed local and string stability. *Transportation Research Part B: Methodological*, 128:69–86.

[71] Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multi-objective optimization. In Gandibleux, X., Sevaux, M., Sörensen, K., and T'kindt, V., editors, *Metaheuristics for Multiobjective Optimisation*, pages 3–37, Berlin, Heidelberg. Springer Berlin Heidelberg.