Elastic Geometric Shape Matching

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

von

Luise Sommer
aus Sonneberg

1. Gutachter:	Prof. Dr. Christian Knauer
2. Gutachter:	Prof. Dr. Hee-Kap Ahn
3. Gutachterin:	Prof. Dr. Maike Buchin

Tag der Einreichung:	15. September 2021
Tag des Kolloquiums:	02. Juni 2022

To my boys

Acknowledgements

This research was carried out during my employment at the Universität Bayreuth and supported by the *Deutsche Forschungsgemeinschaft (DFG)*, grants Kn 591/9-1 and Kn 591/9-2.

First of all, I would like to express my sincere gratitude to my supervisors Christian Knauer and Fabian Stehn for giving me the opportunity to work on this thesis and the wonderful research meetings, for their guidance, their support and their patience.
I also want to thank all members of the work group Algorithmen und Datenstrukturen at the Universität Bayreuth for creating such an inspiring and pleasant environment. They made this place more than just an ordinary work place and I really enjoyed being part of it. My special thanks goes to my office mate Otfried Cheong for his advice, the many books and the delicious cakes.
I want to express my gratitude to my dear friend Eleni Milona for her support, her friendship, and the help on one specific sentence.
I owe to Melanie Göpel, Cornelia Bogensperger and Benjamin Schlosser for investing their time to proofread this thesis.
Furthermore, I would like to thank my husband Andreas Sommer for his unconditional support in the last years.
Last but not least, I want to thank Gisela Roth. She always gave so much and expected nothing in return.

Abstract

In computational geometry, *geometric shape matching (GSM) problems* are among the classical and well-studied geometric optimization problems. In a conventional GSM problem, the pattern P and the model Q, both from a class $\mathcal{S}$ of geometric shapes, are given, along with a suitable distance measure $\mathrm{d} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$. The task is to compute a single transformation t from an admissible transformation class $\mathcal{T}$ acting on $\mathcal{S}$ that minimizes the distance between the transformed pattern and the model according to the distance measure d.
Problems of this flavor have many applications such as traffic sign recognition, character recognition, human-computer-interaction, etc., in different scientific fields such as robotics, computer aided medicine and drug design. Yet in cases where local distortions or complex deformations occur, a single transformation from a simple transformation class is not enough to match the pattern well to the model. A more flexible approach is needed.
Elastic geometric shape matching (EGSM) is a generalization of the conventional GSM and was designed with the intention of ensuring a both globally consistent and locally precise mapping in cases where the classical GSM approach is too restrictive.
In an EGSM problem, one is given a pattern P and a model Q along with a graph G. The pattern P is partitioned into *subshapes* and instead of a single transformation, a so-called *transformation ensemble* is computed. A transformation ensemble consists of a set of transformations, one for each subshape of P, that are individually applied to the subshapes of P with the goal to minimize the distance of the transformed pattern to the model according to a suitable distance measure. Additionally, some of the transformations are enforced to be similar with respect to a suitable similarity measure defined for the transformation class at hand. In doing so, the consistency and "continuity" of the ensemble, and consequently, also of the transformed pattern, is ensured. The graph G is called *neighborhood graph*, and encodes which transformations need to be similar.
There is a vast number of variations of EGSM problems, depending on how

the different options for, eg., $\mathcal{S}$, $\mathcal{T}$, the distance measures, and structure of G, are chosen to match the application at hand. In particular, just slight changes in the problem setup may result in the need of completely different strategies to compute a solution. In this thesis, we analyze the computational complexity of several EGSM problem variants under translations under the L_1-, the L_2- and under polygonal norms for different distance measures and graph classes. Additionally, we present exact and approximative algorithms to solve the considered problem variants.

Zusammenfassung

Geometrische Musteranpassungsprobleme (GSM Probleme) gehören zu den klassischen und gut untersuchten geometrischen Optimierungsproblemen in der algorithmischen Geometrie. In einem konventionellen GSM Problem sind das Muster P und das Modell Q, beide aus einer Klasse $\mathcal{S}$ von geometrischen Objekten, zusammen mit einem geeigneten Abstandsmaß $\mathrm{d} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$ gegeben. Das Ziel besteht darin, eine einzelne Transformation t aus einer zulässigen, auf $\mathcal{S}$ wirkenden, Transformationsklasse $\mathcal{T}$ zu berechnen, die den Abstand zwischen dem transformierten Muster und dem Modell bezüglich des Abstandsmaßes d minimiert.
Probleme dieser Art haben viele Anwendungen, wie zum Beispiel Verkehrszeichenerkennung, Texterkennung und Mensch-Computer-Interaktion, in verschiedenen wissenschaftlichen Bereichen wie der Robotik, der computergestützten Medizin und der Wirkstoffentwicklung. Allerdings ist in den Fällen, in denen lokale Verzerrungen oder komplexe Deformierungen auftreten, eine einzelne Transformation aus einer einfachen Transformationsklasse nicht genug, um das Muster gut auf das Modell abzubilden. Ein flexiblerer Ansatz wird benötigt.
Elastische Musteranpassung (EGSM) ist einer Verallgemeinerung der konventionellen GSM und wurde mit der Absicht entwickelt, eine sowohl global konsistente als auch lokal präzise Abbildung in den Fällen sicherzustellen, in denen der klassiche GSM Ansatz zu sehr einschränkt.
In einem EGSM Problem sind ein Muster P und ein Modell Q zusammen mit einem Graphen G gegeben. Das Muster P ist in *Teilmuster* unterteilt und anstatt einer einzelnen Transformation wird ein sogenanntes *Transformationsensemble* berechnet. Ein Transformationsensemble besteht aus einer Menge von Transformationen, eine für jedes Teilmuster von P, die individuell auf die Teilmuster von P mit dem Ziel angewendet werden, den Abstand zwischen dem transformierten Muster und dem Modell bezüglich eines geeigneten Abstandsmaßes zu minimieren. Zusätzlich sollen einige der Transformationen ähnlich bezüglich eines geeigneten Ähnlichkeitsmaßes sein, das für die gegebene Transformationsklasse definiert wurde. So werden die Kontinuität und die "Stetigkeit" des Ensembles, und damit auch die des transformierten Musters,

sichergestellt. Der Graph G wird *Nachbarschaftsgraph* genannt, weil er kodiert, welche Transformationen sich ähnlich sein sollen.
Es gibt eine erhebliche Anzahl an Variationen von EGSM Problemen, abhängig davon wie beispielsweise $\mathcal{S}$, $\mathcal{T}$, die Abstandsmaße und die Struktur von G gewählt werden, um das Problem auf die konkrete Anwendung anzupassen. Genauer gesagt können schon kleine Veränderungen der Problemstellung dazu führen, dass gänzlich unterschiedliche Strategien zur Lösungsfindung benötigt werden. In dieser Arbeit analysieren wir die algorithmische Komplexität einiger EGSM Probleme unter Translationen unter der L_1-, der L_2- und unter Polygonalnormen für verschiedene Abstandsmaße und Graphklassen. Darüber hinaus präsentieren wir exakte und approximative Algorithmen zur Lösung der betrachteten Problemvarianten.

Contents

Chapter 1

Introduction

Computational geometry is a field of computer science devoted to research on algorithmic solutions for geometric problems. Due to the beauty of the studied problems as well as the many applications, in which geometric algorithms play a fundamental role, computational geometry has attracted the interest of a vast number of researchers since it was originated in the early 1970s [1, 2].

1.1 Geometric Shape Matching

In computational geometry, *geometric shape matching (GSM) problems* are among the well-studied geometric optimization problems. In a conventional geometric shape matching problem, the pattern and the model, both from a class of geometric shapes, e.g., point sets, line segments or polygons, are given, along with a suitable distance measure. The pattern and the model can be seen as two geometric shapes that somehow resemble each other. The task is to compute a single transformation from an admissible transformation class, e.g., translations, rigid motions or affine transformations that minimizes the distance between the transformed pattern and the model according to the distance measure at hand.

Prominent examples of suitable distance measures are the *directed Hausdorff distance* and the *injective bottleneck distance*.

One way to state the problem is as follows:

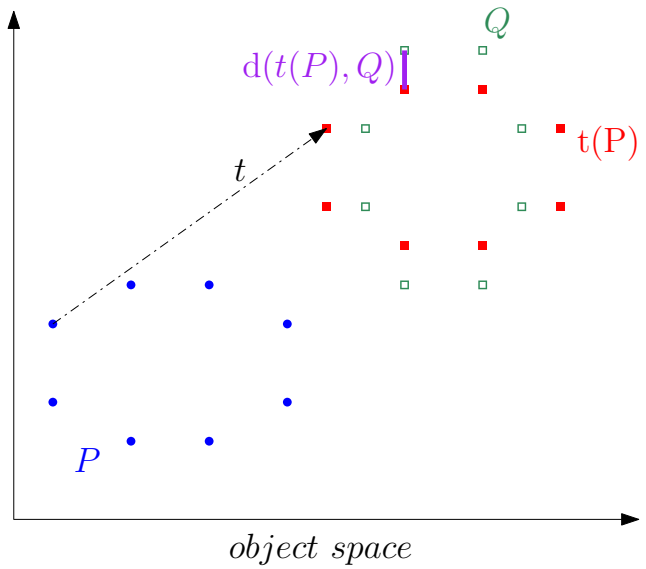

Figure 1.1: The sets P (points), Q (boxes) and $t(P)$ (filled boxes) are point sets in the plane. If only translations are allowed and the distance measure at hand is the directed L_2-Hausdorff distance, the transformation t is optimal. If the class of admissible transformations is rigid motions, there is a rigid motion that maps P exactly to Q.

Problem 1 (Geometric Shape Matching) *Given:*

$$\begin{aligned} \mathcal{S} &\quad \textit{a class of geometric shapes,} \\ \mathcal{T} &\quad \textit{a class of transformations,} \\ P \in \mathcal{S} &\quad \textit{the pattern,} \\ Q \in \mathcal{S} &\quad \textit{the model and} \\ \mathrm{d} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+ &\quad \textit{a distance measure.} \end{aligned}$$

Find: A transformation $t \in \mathcal{T}$ *, so that*

$$\mathrm{d}(t(P), Q)$$

is minimized.

See Figure 1.1 for an example.
Problems of this flavor have many applications such as traffic sign recognition as part of advanced driver assistance systems (ADAS) and in the course of autonomous driving: One of the basic uses of traffic sign recognition is the recognition of speed limit signs as part of ADAS in order to add the information to the GPS data at hand whilst driving. The information can then be displayed

in the dashboard of the car to notify the driver in case of speeding or, in the case of autonomous driving, adjust the speed of the car, see [3].
Other important applications are character recognition [4], logo detection [5], human-computer-interaction, etc., in different scientific fields such as robotics [6], computer aided medicine [7] and drug design [8]. Consequently, geometric shape matching problems have received a great amount of attention. The survey papers by Alt et al. [1] and Veltkamp et al. [9] provide an extensive overview.
In [1], some exact algorithms as well as several approximation algorithms for GSM problems are discussed. In many applications, heuristic methods, such as the *iterative closest point (ICP)* method, are used to "solve" GSM problems, where the point set A is to be matched to the point set B. In every step of the algorithm, for every point of the set A, the nearest neighbor in B, according to the L_2-distance measure, is computed. Then, the transformation that if applied to A, minimizes the maximum mean square root distance between the points in A and their nearest neighbors in B is computed and applied to point set A. The process is iterated until some stop criterion, e.g., a certain amount of iterations or the discovery of a local minimum, is met. This heuristic can be generalized to match, e.g., curves and surfaces in 3D and works in some applications. Although it computes only local minima, there is no guaranteed upper bound on the runtime. For more details, see [10].

1.2 Registration

GSM problems are closely related to *registration problems*. In registration problems the task is to align two geometric spaces, the *pattern space* and the *model space* by computing a mapping from one space into the other one. This mapping is called *registration*. Registrations are used to align, e.g., two images of the same model, taken at different times, from different angles or with different equipment. Formally stated, a registration

$$r : \mathcal{P} \to \mathcal{Q}$$

is a mapping from the pattern space $\mathcal{P}$ to the model space $\mathcal{Q}$ so that every point of the pattern space is mapped to the corresponding point of the model space. There are two major issues that can make registrations more complicated [11]:

- The pattern and the model are misaligned, e.g., due to differences in acquisitions such as different perspectives.
- The pattern is a complexly distorted version of the model, e.g., due to object movements, growth, or the influence of magnetic fields on the measuring equipment.

Usually, the cause of these deformations is not known in detail and there is not even an exact description of the model. Thus, there is no way of computing the exact registration that describes the actual deformation, but there are different approaches to find registrations that align the spaces at hand well to a certain degree.

Registration problems are often solved by reduction to a GSM problem: The same geometric object is measured in two different spaces and a matching is computed that aligns these two shapes well. This transformation minimizes the distance of the two shapes according to the distance measure at hand and is then used as the mapping from the pattern space into the model space. In doing so, a simple misalignment of the geometric shapes at hand can be adjusted by picking the right transformation. However, if the distortion of the pattern is more complex, the assumption that a single transformation from a simple transformation class is enough to align the two shapes well, is not justified.

One important example in this context are soft-tissue registrations that are needed during a computer assisted (oncological) liver surgery [12, 13], where the goal is to remove tumor tissue while preserving as much functional tissue as possible to ensure organ regeneration. Here the model space $\mathcal{Q}$ is the coordinate system of a high quality geometric 3D computer model of the liver tissue and the corresponding vascular system, constructed before the operation on the basis of CT and MRI scan data of the patient. The pattern space $\mathcal{P}$ is the coordinate system of the operation theater, where geometric references are attached to, e.g., the surgical instruments and probes, to allow for optical tracking by a surgical navigation device.

It is easy to imagine, how crucial computing a most accurate transformation is in this setting. However, if the registration is computed by reduction to a conventional GSM problem, the registration is modelled by a single transformation of a simple transformation class. If only a single transformation is allowed to align both spaces, either the average registration error or the worst registration error may be minimized. However, in most cases it is not possible to minimize both, since the deformation of the organ due to the operation process itself as well as the thorax movement due to breathing, the influence of magnetic fields on the tracking device etc., deform the pattern in a very complex way. During the operation process, several registrations are computed over time, to ensure the proper alignment of the 3D model and the patient at all times. Although, even if the registration is adjusted consistently, choosing single transformations as registrations is not enough to ensure a both globally consistent and locally precise registration. A more flexible way of dealing with complex distortions is needed.

1.3 Elastic Geometric Shape Matching

Elastic geometric shape matching (EGSM) is a generalized and more flexible variant of the conventional GSM and was first introduced in 2011 in the dissertation of Stehn [14] and a paper by Knauer, Kriegel and Stehn [15] with the intention of ensuring a both globally consistent and locally precise mapping in cases, where, e.g., local deformations may occur.
In an EGSM problem, one is given a pattern P and a model Q along with a graph G. The pattern P is partitioned into *subshapes* and instead of a single transformation, a so-called *transformation ensemble* is computed. A transformation ensemble consists of a set of transformations, one for each subshape of P, that are individually applied to the subshapes of P with the goal to minimize the distance of the transformed pattern to the model. However, the isolated conventional geometric shape matching problems for each subshape are not solved independently, but some of the transformations, e.g., transformations that act on subshapes that are geometrically close to each other, are forced to be similar with respect to a suitable similarity measure defined for the transformation class at hand. In doing so, the consistency and "continuity" of the ensemble, and consequently, also of the transformed pattern, is ensured. The graph G is called *neighborhood graph*, or *neighborhood* in short, and encodes, which transformations need to be similar.
In Figures 1.1 and 1.2, the same pattern and model are matched. In Figure 1.1, the conventional GSM approach is used while in Figure 1.2, the problem at hand is modelled as an EGSM instance.
Formally, the EGSM problem can be stated as follows:

Problem 2 (EGSM) *Given:*

$\mathcal{S}$	*a class of geometric shapes,*
$\mathcal{T}$	*a class of transformations,*
$P \in \mathcal{S}$	*the pattern with*
$\{P_1, \ldots, P_k\}$	*a partition of* P*,*
$Q \in \mathcal{S}$	*the model and*
$G = (V, E)$	*an undirected graph with* $V = \{i \mid 1 \leq i \leq k\}$ *and* $E \subseteq \{\{i, j\} \mid i, j \in V\}$.

Find: A transformation-ensemble $T = (t_1, \ldots, t_k) \in \mathcal{T}^k$*, so that*

$$\mathrm{d}(T(P), Q)$$

with $\mathrm{d} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$ *is minimized and*

$$\mathrm{sim}_G(T)$$

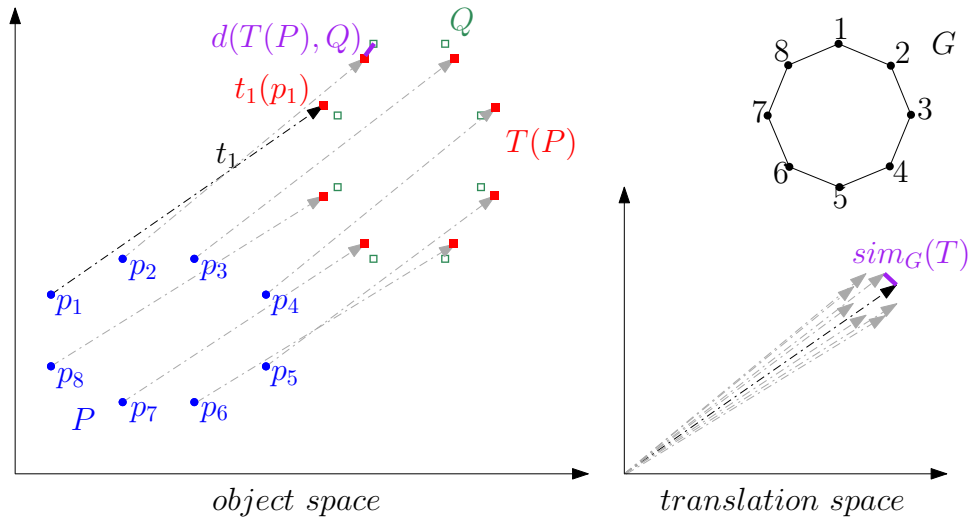

Figure 1.2: Left (object space): The sets P (points), Q (boxes) and $T(P)$ (filled boxes) are point sets in the plane. The transformation ensemble T (dash-dotted) is optimal with respect to the neighborhood graph G if only translations are allowed and the distance measure at hand is the directed L_2-Hausdorff distance. Right (translation space): The translation ensemble T. The distance measure in translation space is the L_2-distance.

with $\mathrm{sim}_G : \mathcal{T}^k \to \mathbb{R}_0^+$ *is maximized.*

We refer to a transformation ensemble that solves Problem 2 as *witness*.
There is a vast number of variants of this problem, depending on how the different options are chosen to match the application at hand. In particular, just slight changes in the problem setup result in the need of completely different strategies to compute a solution and the computational complexity thereof.

1.4 The Problem in Close Up

To get a better understanding of EGSM and its many variants, Problem 2 is now discussed in detail:

The class of geometric shapes $\mathcal{S}$. In EGSM problems, two geometric objects have to be matched. These objects are represented by the pattern P and the model Q and are usually elements of the same class of geometric shapes $\mathcal{S}$. Depending on the application, the class $\mathcal{S}$ can vary a lot. In many cases, choosing $\mathcal{S}$ to be point sets is a good option, which is why all variants of EGSM that have been considered so far, deal with this choice of $\mathcal{S}$. However,

thinking of 3D-modelling, for instance, triangulated surfaces would be suitable. Line segments or polygons could also be of interest, e.g., to describe silhouettes or characters that have to be matched.

The model space. In this thesis, everything is stated in $\mathbb{R}^2$. However, most results can also be adapted to $\mathbb{R}^3$. Considering instances in $\mathbb{R}^3$ or even higher dimensional spaces could also be fitting in a lot of applications, such as computer aided surgery and 3D-modelling.

The partitions of P. The pattern is divided into subshapes and the design of the subshapes depends on the application. In some cases, choosing every element of P to be a separate subshape may be useful while in other cases, subshapes of different sizes match the problem instance at hand better. In the course of computer assisted surgery, for instance, the parts of P encoding different organs or organ parts with different tissue structure may be separated into different subshapes, because their deformation during the operation process are likely to differ to a certain degree.

The transformation class $\mathcal{T}$. $\mathcal{T}$ encodes the class of transformations that match the partitions of P to the model Q. In [16], the authors introduced the first algorithms that solve some EGSM variants, along with an NP-hardness proof for others. In their paper, they mostly considered translations in one direction. In this thesis, $\mathcal{T}$ is the class of translations in $\mathbb{R}^2$. Considering $\mathcal{T}$ to be rigid motions or affine transformations is one of the goals of future work.

The graph G and the graph class it belongs to. The neighborhood graph encodes, which transformations should be similar and as a consequence, which subshapes of P should be transformed similarly. In many applications such as computer aided surgery it is useful to design the graph, so that the vertices corresponding to transformations for geometrically neighbored subshapes are connected by an edge. Recent research has indicated that the complexity of EGSM instances strongly depends on the graph class at hand. If G is a path, computing a solution for the given instance seems much easier in many EGSM settings. Additionally, aside from using quadratic programming techniques without provable bounds on the runtime, there are no approximation- or exact algorithms at all for graphs that contain cycles under the condition that translations in more than one direction are allowed. The computational complexity of the problem also seems to increase with the number of cycles in the neighborhood graph. The problem is be NP-complete for EGSM instances with complete neighborhood graphs.

The weight on the edges and vertices of G. In this thesis, all edges of G have the same weight, although putting weights on certain edges to encode different degrees of similarity between adjacent transformations could be useful in some applications. A prominent example are soft tissue registrations, where geometric shapes that encode the same organ should be matched in a very similar way, while shapes that encode different, neighbored organs should still be matched similarly, but to a lesser extent. By putting weights on the vertices, it is possible to encode how well the respective subshape of the pattern has to match the model. This allows different maximum distances to the model for different subshapes.

The distance measure d**.** Prominent examples of suitable distance measures are the injective bottleneck distance and the directed Hausdorff distance. In the simplest setting, the correspondence between the pattern and the model is fixed. Thus $k = |P| = |Q|$ and d measures the distance between the elements of P and the corresponding elements of Q.

The similarity measure sim_G**.** If $\mathcal{T}$ is the class of translations, measuring their similarity in translation space according to a suitable norm, e.g., the L_1-norm, the L_2-norm, the L_∞-norm or some polygonal norm is a promising approach because the geometric interpretation is clear. However, choosing $\mathcal{T}$ to be, e.g., rigid motions, makes the choice of a suitable similarity measure more complicated, as the geometric interpretation of the well-established matrix norms, such as the spectral norm in this context, is not trivial.

The tradeoff between minimizing d **and maximizing** sim_G**.** A central topic in discussing EGSM problems is the tradeoff between minimizing the distance between the transformed pattern and the model and maximizing the similarity of neighbored transformations. In this thesis, both goals are seen as equally important and are thus integrated in a common objective function with the same threshold. However, depending on the application, it may be advisable to weigh them differently instead. Regardless of the tradeoff itself, both goals should be somehow comparable, which implies using the same norm in both the distance measure d and the similarity measure sim_G if possible.

1.5 State of the Art

Several approaches that deal with non-linear geometric transformations can be found in the literature [17, 18, 19, 20, 21]. All of these strategies are

heuristics that are based on techniques such as relaxed ILP formulations [17, 18], probabilistic methods [19], or ICP formulations [20, 21]. None of these methods compute solutions that are provably optimal or optimal up to an approximation factor.
EGSM was first introduced as part of the dissertation of Stehn [14] in 2011 under the term *non-uniform geometric matching*. In his thesis, the author developed a polynomial time constant factor approximation algorithm for EGSM problems under the directed Hausdorff distance for complete neighborhood graphs.
Further results where discussed by Knauer, Kriegel and Stehn in [15] also in 2011: Several variants of EGSM problems were considered, where the pattern and the model are point sequences with a fixed correspondence between them. For these cases, efficient, i.e., polynomial time, exact algorithms based on a convex programming formulation, and approximate combinatorial algorithms were designed for different types of neighborhood graphs.
In a paper by Knauer and Stehn [16], the authors showed that EGSM is NP-hard for complete graphs under translations when considering the directed Hausdorff distance or the injective bottleneck distance under the L_2-norm. In the same paper, they give efficient exact algorithms that solve a variant of the problem for trees where only translations in a fixed direction are allowed.

1.6 The Contribution of this Thesis

In Chapters 2 to 5, everything is stated in $\mathbb{R}^2$ and translations are represented by translation vectors. A translation vector $t = (t_x, t_y)$ translates any point $p = (p_x, p_y)$ as follows:

$$t(p) \coloneqq p + t = (p_x + t_x, p_y + t_y)$$

and translates a sequence of n points $P = (p_1, \ldots, p_n)$ as follows:

$$t(P) \coloneqq \{t(p) \mid p \in P\}.$$

Further, a sequence of n translation vectors $T = (t_1, \ldots, t_n)$ translates a sequence of n points $P = (p_1, \ldots, p_n)$ as follows:

$$T(P) \coloneqq \{t_i(p_i) \mid 1 \leq i \leq n\}.$$

All discussed EGSM variants consider P and Q to be point sets in the plane and each point of P forms an individual subshape of the pattern. In most cases, the distance between the transformed pattern and the model is measured with the directed Hausdorff distance or the injective bottleneck distance:

Definition 1.1 *For two point sets $A, B \subseteq \mathbb{R}^d$, the* directed Hausdorff distance *is defined as*

$$\vec{h}(A,B) := \max_{f:A\to B} \min_{a\in A} \|a - f(a)\|,$$

and the injective bottleneck distance *is defined as*

$$\mathrm{b}(A,B) := \max_{\substack{f:A\to B \\ f \text{ injective}}} \min_{a\in A} \|a - f(a)\|.$$

If the correspondence between the points of the pattern and the points of the model is fixed, i.e., every point $p_i \in P$ is matched to a specific point $q_i \in Q$, we choose P and Q to be point sequences rather than point sets.

Definition 1.2 *For two point sequences $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ in $\mathbb{R}^2$, the* 1-to-1-distance *is defined as*

$$\max_{1\le i\le n} \|p_i - q_i\|.$$

The transformation class $\mathcal{T}$ is the class of translations and the (dis)similarity of two translation vectors is measured by the length of their difference, according to the norm at hand. The objective function considers the distance of the transformed pattern to the model in model space, and the similarity of neighboring translations in translation space equally important. The resulting variant of Problem 2 can be formally stated as:

Problem 3 *Given:*

$P = (p_1, \ldots, p_n)$	*a sequence of points (the pattern) in* $\mathbb{R}^2$
$Q = (q_1, \ldots, q_m)$	*a sequence of points (the model) in* $\mathbb{R}^2$ *and*
$G = (V, E)$	*an undirected graph with* $V = \{i \mid 1 \le i \le n\}$ *and* $E \subseteq \{\{i,j\} \mid i,j \in V\}$.

Find: A sequence of n translation vectors $T = (t_1, \ldots, t_n)$ so that

$$\max\left(\vec{h}(T(P), Q), \max_{\{i,j\}\in E} \|t_i - t_j\|\right)$$

is minimized.

Note that P and Q can either be given as point sets or point sequences, depending und the context and the specific advantages of this choice in the setting at hand.
Depending on the strategy to solve the problem, it is also promising to consider the *decision variant* of Problem 3. Here, we are given an additional parameter

$\delta > 0$ and the goal is decide if there is a sequence of n translation vectors $T = (t_1, \ldots, t_n)$ so that

$$\max\left(\vec{h}(T(P), Q), \max_{\{i,j\}\in E} \|t_i - t_j\|\right) \leq \delta.$$

Chapter 2. We address the decision variant of Problem 3 for neighborhood graphs that are trees. We consider the L_1-norm (which we also refer to as Manhattan norm), the L_2-norm, or the L_∞-norm. We present a polynomial time exact algorithm for this problem for neighborhood graphs that are trees under the L_1-norm. Also, we discuss the same problem under polygonal norms and give for any integer $k \geq 2$ a polynomial time $\left(1 + \frac{1}{k}\right)$-approximation algorithm for Problem 3 under the directed L_2-Hausdorff distance.
This research has been published in [A38, A40].

Chapter 3. We focus on Problem 3 for neighborhood graphs that are simple cycles under the Euclidean 1-to-1-distance and provide an FPTAS for this problem. It is also possible to return a witness without increasing the runtime. This research has been published in [A39].

Chapter 4. In this chapter, we discuss Problem 3 under the directed Euclidean Hausdorff distance and consider neighborhood graphs that possibly contain several cycles. We present an algorithm which, for an $\epsilon > 0$, gives a $(1 + \epsilon)$-approximation to the optimum of the objective function for this variant of the Problem 3 with a given feedback vertex set of size k_f. Also, we provide an algorithm which gives a $(1 + \epsilon)$-approximation to the optimum of the objective function for the problem with a given path or tree decomposition of width k_w.

Chapter 5. In [16], the author gives an algorithm that solves Problem 3 for tree neighborhoods in polynomial time if only translations in a fixed direction are allowed. We adapt the algorithm to the decision variant of Problem 3 for tree neighborhoods under the Euclidean 1-to-1-distance and give a non-polynomial upper bound on the runtime. We also prove that this algorithm runs in polynomial time and space if G is a complete tree or a path. We give an example illustrating the fact that solving the problem with our approach requires quadratic space at least. However, if G is a tree, the problem is more complicated. We give non-polynomial upper bounds on the time and the space required by the algorithm to solve the problem if G is a tree, which can be improved to be polynomial if the tree is also complete and give a rough description on how the difficulties in estimating the combinatorial complexity

change if the correspondence between the points of the pattern and the points of the model is not fixed.

Chapter 6. We discuss different EGSM variants other than Problem 3, the EGSM variant that is considered in the previous chapters. We give insights about EGSM under rigid motions and discuss the difficulties in designing an appropriate distance measure for rigid motions. Further, we discuss how existing strategies for EGSM problems for point sets can be modified to solve EGSM problems for line segments, triangles and triangulated surfaces. We also consider EGSM in higher dimensions, with differently weighted objectives and discuss the effect of imprecise input sets framework.

Acknowledgements. Some of the following results were developed jointly with Christian Knauer, Fabian Stehn and Otfried Cheong.

Chapter 2

Elastic Geometric Shape Matching for Translations under the Manhattan Norm

In the first part of [16] the authors showed that EGSM is NP-hard for complete graphs under translations when considering the directed Hausdorff distance or the injective bottleneck distance under the L_2-norm. However, in many applications, it may be useful to consider other graph classes, because, e.g., translations that are applied on geometrically distant subpatterns may not be required to be as similar as geometrically close ones. The hope is to find efficient algorithms for some variants of EGSM. The second part of [16] also contains efficient exact algorithms that solve variants of the problem for trees where only translations in a fixed direction are allowed. Thus considering EGSM instances for tree neighborhoods seems promising, although recent research indicates that the complexity of the EGSM instances at hand also strongly depends on the underlying norm.

In this chapter, we address the decision variant of Problem 3 for neighborhood graphs that are trees where the norm at hand is the L_1-norm, the L_2-norm, the L_∞-norm, or a polygonal norm. We present a polynomial time exact algorithm for this problem under the L_1-norm. Also, we discuss the same problem under polygonal norms and give for any integer $k \geq 2$ a polynomial time $\left(1 + \frac{1}{k}\right)$-approximation algorithm for the problem under the directed L_2-norm.

This research has been published in [A38, A40].

2.1 Problem Statement

For $p \in \{1, 2, \infty\}$ we denote the L_p-norm of a vector x by $\|x\|_p$. Since we focus on different norms in this chapter, we slightly adapt the notation of the directed Hausdorff distance:

Notation 2.1 *We write the* directed L_p-Hausdorff distance *of a point set $A \subseteq \mathbb{R}^2$ to a point set $B \subseteq \mathbb{R}^2$ as*

$$\vec{h}_p(A, B) := \max_{a \in A} \min_{b \in B} \|a - b\|_p.$$

The EGSM decision problem at hand can be formally stated as:

Problem 4 (EGSM for point sets in $\mathbb{R}^2$ under translations) *Given:*

$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$	*a point set (the pattern),*
$Q = \{q_1, \ldots, q_m\} \subset \mathbb{R}^2$	*a point set (the model),*
$G = (V, E)$	*an undirected graph with $V = \{i \mid 1 \leq i \leq n\}$ and $E \subseteq \{\{i, j\} \mid i, j \in V\}$ and*
$\delta \geq 0$	*a parameter.*

Find: A sequence of n translation vectors $T = (t_1, \ldots, t_n)$ so that

$$\max\left(\vec{h}_p(T(P), Q), \max_{\{i,j\} \in E} \|t_i - t_j\|_p\right) \leq \delta.$$

In the following, we

1. present an algorithm for Problem 4 under the directed L_1- or L_∞-Hausdorff distance for neighborhood graphs that are trees; the runtime of the algorithm is $O(n^2 m(\log m + \log n))$, see Theorem 2.9.
2. give for any integer $k \geq 2$ a $\left(1 + \frac{1}{k}\right)$-approximation algorithm for the smallest value of δ that admits a YES-instance for Problem 4 under the directed L_2-Hausdorff distance for neighborhood graphs that are trees; the runtime of the algorithm is $O(bn^3 m^2(\log m + \log n + \log b))$, where $b = \left\lceil \frac{\pi}{4}\sqrt{2(k+1)} \right\rceil$, see Theorem 2.13 and Corollary 2.14.

2.2 Deciding Problem Instances for Trees for the directed Manhattan Hausdorff distance

In this section we present an algorithm for Problem 4 for neighborhood graphs that are trees. We measure the similarity of the shapes by the directed L_1-

Hausdorff distance. The similarity of two translations is measured by the L_1-norm of the difference of their translation vectors. Consequently, the objective for this variant is to decide if there are n translation vectors $T = (t_1, \ldots, t_n)$ so that

$$\max\left(\vec{h}_1(T(P), Q), \max_{\{i,j\}\in E} \|t_i - t_j\|_1\right) \leq \delta. \tag{2.1}$$

Definition 2.2 *We call a sequence of translations T* admissible for the graph $G = (V, E)$*, if Inequality* (2.1) *holds.*

Strictly speaking, the concept of admissibility depends on δ, P and Q as well, but since they are part of the input, we refrain from including them in the notation. However, the graph at hand may vary throughout the algorithm that is introduced in the following.
The set I_p of translations that move a point $p \in P$ δ-close to some point of Q with respect to the directed L_1-Hausdorff distance is given by

$$I_p \coloneqq \bigcup_{q\in Q} \{t \in T \mid \|p + t - q\|_1 \leq \delta\}.$$

I_p is the union of m squares of side length

$$a \coloneqq \sqrt{2}\delta$$

of the same orientation. Recall that the vertices of $G = (V, E)$ represent the translations for the individual subshapes that are to be computed, and that the edges of G encode which translations are required to be similar. A variant of the following, which we refer to as Algorithm B, where only translations in a fixed direction are allowed, was first introduced in [16] to decide the problem under translations in a fixed direction for tree neighborhoods. The strategy exploits the structure of G to decide the EGSM instance at hand. To simplify the presentation, we associate the points of P with the vertex set V of G. If $v \in V$ is the vertex of G that represents the translation that is to be computed for point $p \in P$ we store the set I_p in v and refer to it also as I_v. In other words, I_v contains all admissible translations for the graph $(\{v\}, \emptyset)$.

Definition 2.3 *Let T_v be a tree rooted in v and let $p \in P$ be the point that corresponds to the vertex v. We call a translation t* admissible (for the vertex v)*, if t is part of an admissible sequence of translations for T_v (where it is assigned to p).*

We will denote the L_1-circle with radius δ centered in the origin by $\diamond$.
We start by picking an arbitrary vertex $r \in V$ and henceforth consider T_r, the tree rooted in r. For an internal vertex $v \in V$ let $c(v)_1, \ldots, c(v)_{n_v}$ be the n_v children of v. For any vertex $v \in V$ let T_v be the subtree of T_r with root v.

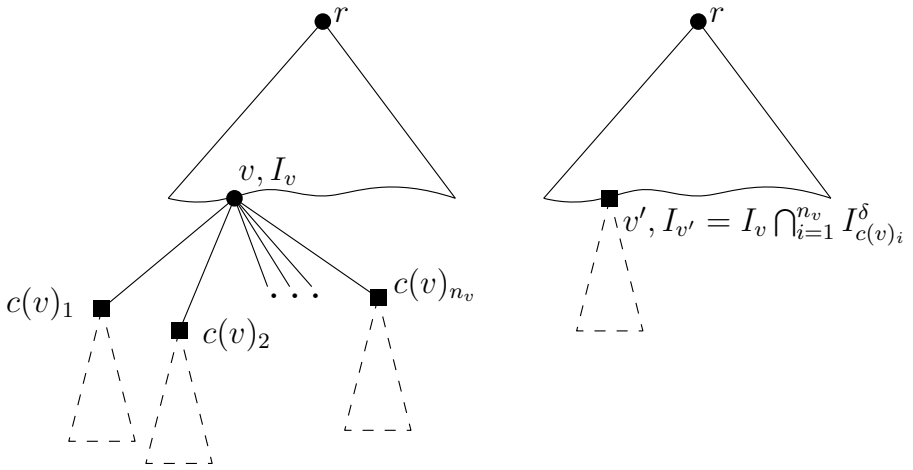

Figure 2.1: Subtree T_v before and after updating v.

The algorithm that decides whether there is a set T of translations that satisfies Equation (2.1) will be stated in detail below and has an iterative structure. The basic idea is to propagate admissible translations *bottom-to-top* through T_r by contracting a set of leaves with their common parent, while appropriately merging the sets of admissible translations of all the vertices involved. Starting with T_r, the algorithm chooses a vertex v, whose children are all leaves and conceptually replaces T_v by a single vertex v' that stores an adjusted set $I_{v'}$ of admissible translations, see Figure 2.1. The admissible regions of v and its children are merged into $I_{v'}$ in such a way that $I_{v'}$ is not empty iff there exists a set of translations that is admissible for T_v.
Moreover, if $I_{v'} \neq \varnothing$, these translations can be computed from $I_{v'}$ and the sets of admissible translations stored in the vertices of T_v. More concrete:

Algorithm 1 *We are given the point sets $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$, a tree $G = (V, E)$ and a parameter $\delta \geq 0$.*
We pick an arbitrary vertex $r \in V$ and henceforth consider T_r, the tree rooted in r.
For an internal vertex $v \in V$ let $c(v)_1, \ldots, c(v)_{n_v}$ be the n_v children of v. For any vertex $v \in V$ let T_v be the subtree of T_r with root v. In each iteration of the algorithm, we call the tree from which a vertex is selected the current tree.
Every vertex v corresponds to a point $p \in P$. At start, the set

$$I_v = I_p = \bigcup_{q \in Q} \{t \in T \mid \|p + t - q\|_1 \leq \delta\}$$

is stored in every node v of the tree.

In each iteration of the algorithm, a vertex v of the current tree is selected with the property that all children of v are leaves or vertices which already have been updated. Then, the admissible translations of v and those of the children of v are merged into a new set of admissible translations that is stored in the new vertex v', the updated version of v. To compute the set $I_{v'}$ of admissible regions for v' we proceed as follows:

1. *We* inflate *all regions $I_{c(v)_i}$ by δ for $1 \leq i \leq n_v$ which results in a set*

$$I^{\delta}_{c(v)_i} := I_{c(v)_i} \oplus \diamond,$$

   where $\oplus$ denotes the Minkowski sum operation. Note that inflating (a shifted copy of) $\diamond$ by δ leads to a square with side length $2a$, where a is the side length of $\diamond$.
2. *We compute the admissible region $I_{v'}$ for the new vertex v' as follows (see Figure 2.2):*

$$I_{v'} = \left(\bigcap_{i=1}^{n_v} I^{\delta}_{c(v)_i} \right) \cap I_v.$$

This process is repeated until one of the following cases occurs:

1. *There is a vertex v with $I_v = \varnothing$ (after a contraction):*
   The process stops and NO *is returned as the answer to Problem 4.*
2. *The root r is updated and $I_{r'} \neq \varnothing$:*
   The algorithm terminates and returns YES *as the answer to Problem 4.*

Theorem 2.4 *Algorithm 1 is correct.*

Proof. Algorithm 1 resembles Algorithm B for x-translations discussed in [16]. Hence we keep this proof short.
It is obvious that the algorithm terminates, hence it remains to prove the correctness of the answer it returns. To this end we will show the following:
After every iteration of the algorithm the following holds: Let v be a vertex of the tree that has been updated during the most recent iteration. If any $t \in I_v$ is chosen as the translation for v, then there are translations for all points of T_v so that the answer to Problem 4 is YES *with respect to the points in T_v. If $I_v = \emptyset$, there is no admissible sequence of translations for T_v and the answer to Problem 4 is* NO *with respect to T_v.*
We show this by induction on the number of updates $\#C$ of the algorithm.
The statement holds for $\#C = 0$: Before any update is performed, for any leaf v, the tree T_v only consists of one point v. Since no neighborhood constraints

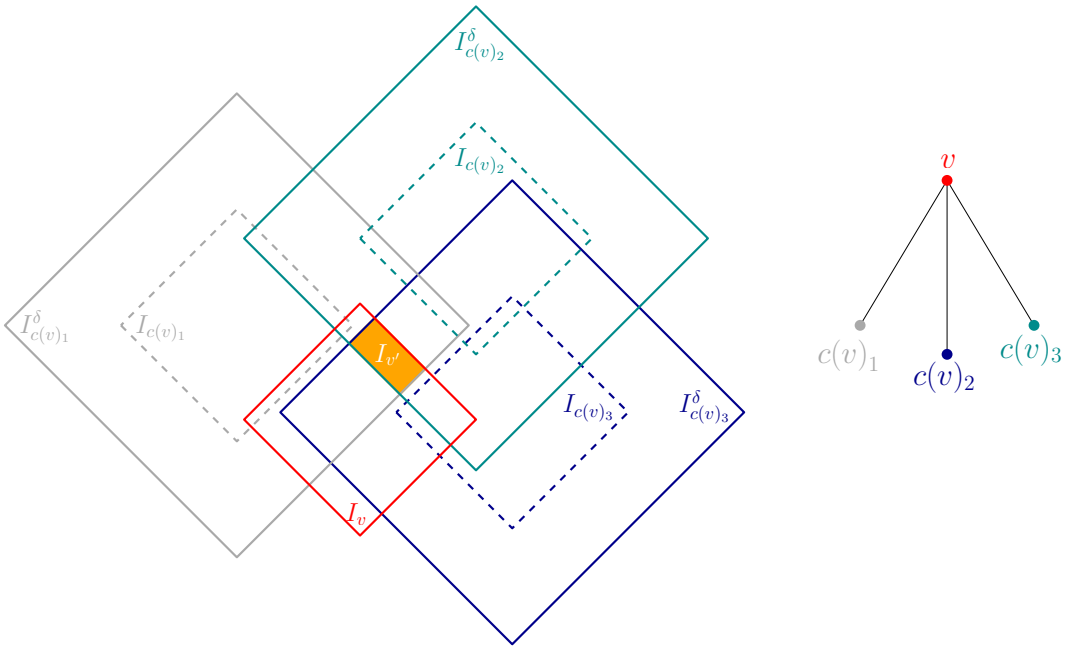

Figure 2.2: Illustration of a contraction step of a vertex v with its children. Vertex v is contracted with its three children $c(v)_1$, $c(v)_2$, and $c(v)_3$.
Left: The resulting set $I_{v'}$ of admissible translations is the common intersection of the inflated child-regions and I_v.
Right: The subtree T_v that is contracted to a single vertex v'.

have to be met, every point of I_v is admissible for the tree $\{\{v\}, \emptyset\}$, hence every point of I_v is a witness for the fact that the answer to Problem 4 is YES subject to the tree $\{\{v\}, \emptyset\}$ and $I_v \neq \emptyset$.
Induction step: Let v be a vertex of the current tree which can be updated. By induction, any translation $t \in I_{c(v)_i} \neq \emptyset$ for a child $c(v)_i$ of v can be chosen, so that there are translations for all vertices in the subtree of $T_{c(v)_i}$ in a way that the similarity constraints in translation space as well as the distance constraints in model space are met. Also, if $I_{c(v)_i} = \emptyset$ for a child $c(v)_i$ of v, there is no admissible sequence of translations for $T_{c(v)_i}$. It immediately follows that there is no admissible sequence of translations for T_v and the answer to Problem 4 is NO. Now we consider the case that $I_{c(v)_i} \neq \emptyset$ for all children $c(v)_i$ of v. Then for the vertex v', the result of the contraction of vertex v, one of the following holds:

1. $I_{v'} = \emptyset$. This implies that there is at least one child $c(v)_i$ of v so that for every $t \in I_v$, $\{t + \diamond\} \cap I_{c(v)_i} = \varnothing$, i.e., every translation that belongs

to an admissible sequence for $T_{c(v)_i}$ is more than δ-away from any $t \in I_v$. Thus, the constraints on the translations induced by T_v cannot be met and the answer to Problem 4 is NO.

2. $I_{v'} \neq \emptyset$. Then, for every $t' \in I_{v'}$, the following holds:
   (a) By construction, we have $t' \in I_v$, hence t' will bring the point corresponding to v δ-close to some point of Q.
   (b) For all $i \in \{1, \ldots, n_v\} : \ t' \in I^{\delta}_{c(v)_i}$, implying that $\{t' + \diamond\} \cap I_{c(v)_i} \neq \varnothing$. This means that every admissible region $I_{c(v)_i}$ contains a translation that is at most δ-away from t' for any $t' \in I_{v'}$ and $i \in \{1, \ldots, n_v\}$.

□

Note that this strategy also works if some of the translations are predefined, i.e., for at least one node v, a translation $t \in I_v$ is given as part of the input and thus is a fixed part of any admissible sequence of translations for the EGSM instance at hand. Then, some of the sets I_v do not contain a union of m objects, but just a single translation.

The runtime of Algorithm 1 depends on the description complexity of the sets of translations that are stored in each vertex before and after updating all vertices in G. To analyze these sets, we need to introduce some notation and basic lemmata.

Definition 2.5 *Let $B \subset \mathbb{R}^2$ be a closed connected set. A closed set $A \subset \mathbb{R}^2$ is called* B-fat *if every point $a \in A$ can be covered by a translational copy of B that is contained in A.*

Lemma 2.6 *The union of m regular k-gons of the same orientation has a description complexity of $O(km)$.*

Proof. A set of regular k-gons of the same orientation is a collection of pseudo-discs. According to [22], Theorem 2.2, the union of such m pseudo-discs has description complexity $O(km)$. □

Definition 2.7 *Let A be the union of a finite number of polygons. We call the number of vertices on the boundary of A the* description complexity *of A.*

Lemma 2.8 *Let at some point of Algorithm 1 $c(v)_i$ with $i \in \{1, \ldots, n_v\}$ be the children of a vertex v that are all leafs of the current tree. Let every $I_{c(v)_i}$ consist of m_i vertices and edges for $i \in \{1, \ldots, n_v\}$. Then, the following holds:*

1. *The set $I_{v'}$ has description complexity*

$$O(n_v(n_v m + \sum_{i=1}^{n_v} m_i)).$$

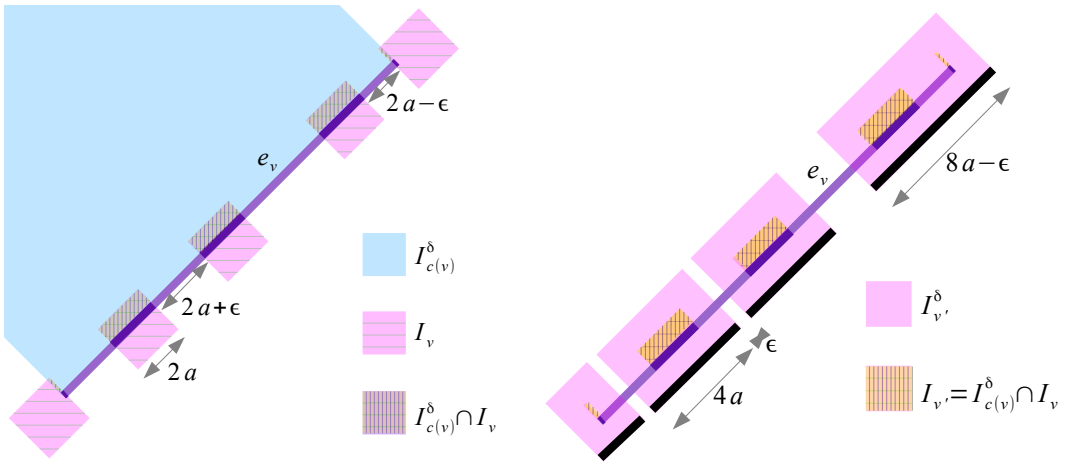

Figure 2.3: Left: Edge e_v is cut into pieces with different pairwise distances. Right: Edges originating from edge e_v in $I^{\delta}_{v'}$.

2. *The set $I^{\delta}_{v'}$ has description complexity $O(n_v m + \sum_{i=1}^{n_v} m_i)$.*

Proof. For now we assume that all vertices have pairwise different x- and y-coordinates and that no edges overlap each other. Thus all connected components of the sets that are computed during the algorithm consist of more than one vertex and the number of vertices of these sets equals the number of their edges, unless the set is empty. Hence it suffices to count the edges of these sets in order to estimate their description complexity. We first consider the regions that constitute $I_{v'}$:

1. The set I_v is a union of m squares of the same size and orientation. Since the pairwise union of two such squares has at most eight vertices, the set I_v has at most $4m$ vertices and edges in total. Each square has edge length a and so I_v has an area of at most $a^2 m$. The set I^{δ}_v consists of m squares with edge length $2a$ and thus has an area of at most $4a^2 m$.
2. Every $I_{c(v)_i}$ can be described by unions and intersections of shifted and inflated versions of $\diamond$. If $c(v)_i$ is a leaf of T_v, $I_{c(v)_i}$ has an area of at most $a^2 m$. If it is an internal vertex of T_v, it is contained in the union of m squares of area a^2 and therefore also has an area of at most $a^2 m$. The set $I^{\delta}_{c(v)_i}$ is the inflated version of $I_{c(v)_i}$ and so has an area of at most $4a^2 m$.

Each set can be described by the sequence of its boundary edges along its boundary where every edge is of one of four types: It can have a slope of ± 1 and the interior of the set can either lie above or below the edge. Every edge in

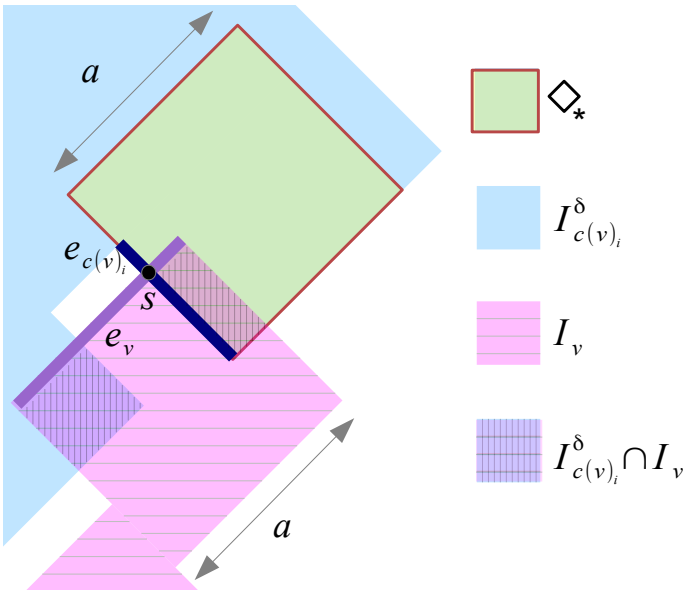

Figure 2.4: Sets $I^{\delta}_{c(v)_1}$ (blue) and I_v with focus on edges e_v and $e_{c(v)_1}$.

$I_{v'}$ either originates from an edge of I_v or from an edge of $I^{\delta}_{c(v)_i}$ for a certain i. Since the description complexity of $I_{v'}$ is bounded from above by the maximum number of its boundary edges, we can count in how many parts the edges of $I^{\delta}_{c(v)_i}$ and I_v can be split in order to determine its complexity.
The sets I_v and all $I^{\delta}_{c(v)_i}$ are obviously $\diamond$-fat. Since I_v is a collection of m pseudo-discs it has linear description complexity, see Lemma 2.6.
Part 1 of Lemma 2.8: Consider an arbitrary edge e_v of I_v: e_v can only be intersected by edges of $I^{\delta}_{c(v)_i}$ that are orthogonal to e_v, see Figure 2.3. Let $e_{c(v)_i}$ be such an edge intersecting e_v in s. Since $I^{\delta}_{c(v)_i}$ is $\diamond$-fat we can place a square $\diamond_*$ of side length a so that it has s on its boundary and is completely contained in $I^{\delta}_{c(v)_i}$. As e_v has length a and $\diamond_*$ has side length a, one endpoint of e_v has to be in $\diamond_*$, see Figure 2.4. As a consequence, e_v can be cut into at most two pieces by edges of $I^{\delta}_{c(v)_i}$. Since

$$\left(\bigcap_{i=1}^{n_v} I^{\delta}_{c(v)_i}\right) \cap e_v = \bigcap_{i=1}^{n_v} (e_v \cap I^{\delta}_{c(v)_i})$$

and the parts of e_v that remain after intersection with $I^{\delta}_{c(v)_i}$ are pairwise disjoint line segments, the maximum number of pieces of

$$\left(\bigcap_{i=1}^{n_v} I^{\delta}_{c(v)_i}\right) \cap e_v$$

is the sum of the number of pieces of all $e_v \cap I^{\delta}_{c(v)_i}$, which is $2n_v$. Hence there are at most $8n_v m$ edges in $I_{v'}$ that originate from an edge of I_v.

Since I_v is $\diamond$-fat, an edge of $I^{\delta}_{c(v)_i}$ of length l can be cut by I_v into at most $1+\lceil \frac{l}{a} \rceil$ disjoint pieces that remain as edges in $I_{v'}$, see Figure 2.5. As every $I^{\delta}_{c(v)_j}$ with $i \neq j$ is $\diamond$-fat as well, the same holds for intersecting $I^{\delta}_{c(v)_i}$ with some other $I^{\delta}_{c(v)_j}$. As $I^{\delta}_{c(v)_i}$ is $\diamond$-fat and has an area of at most $4a^2m$, the sum of the lengths of all

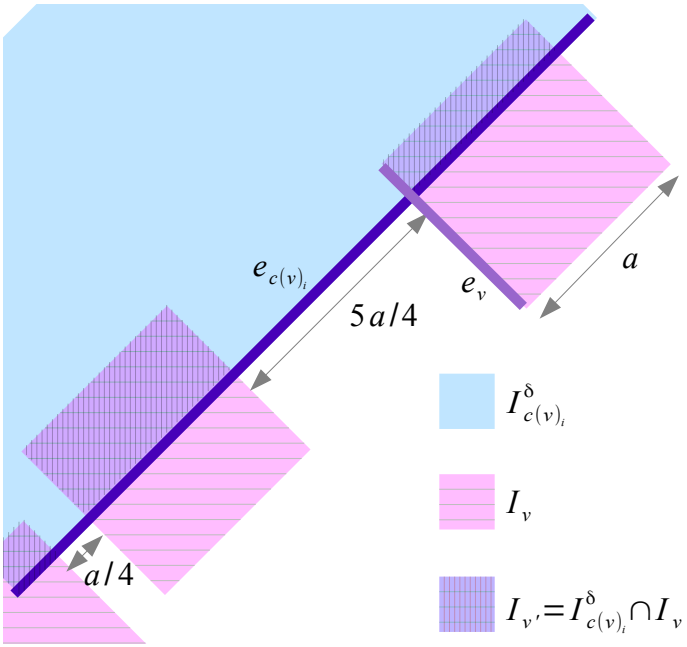

Figure 2.5: The edge $e_{c(v)_1}$ during the intersection of $I^{\delta}_{c(v)_1}$ and I_v.

edges of $I^{\delta}_{c(v)_i}$ which are of the same type is less than $\frac{4a^2m}{a} = 4am$. In order to derive an upper bound on the number of edges in $I_{v'}$ that originate from edges of $I^{\delta}_{c(v)_i}$, we look at the worst case: We assume that $I^{\delta}_{c(v)_i}$ has $m_i - 4$ very short edges each of which causes at most two edges in $I_{v'}$ (since they generate many edges without decreasing the total edge length of $I^{\delta}_{c(v)_i}$ so much) and four long edges with length at most $4am$, each of which causes at most $1 + 4m$ edges in $I_{v'}$. Therefore the maximum number of edges in $I_{v'}$ that originate from edges of a certain $I^{\delta}_{c(v)_i}$ is $n_v(2m_i + 16m - 4)$, which gives an upper bound on the

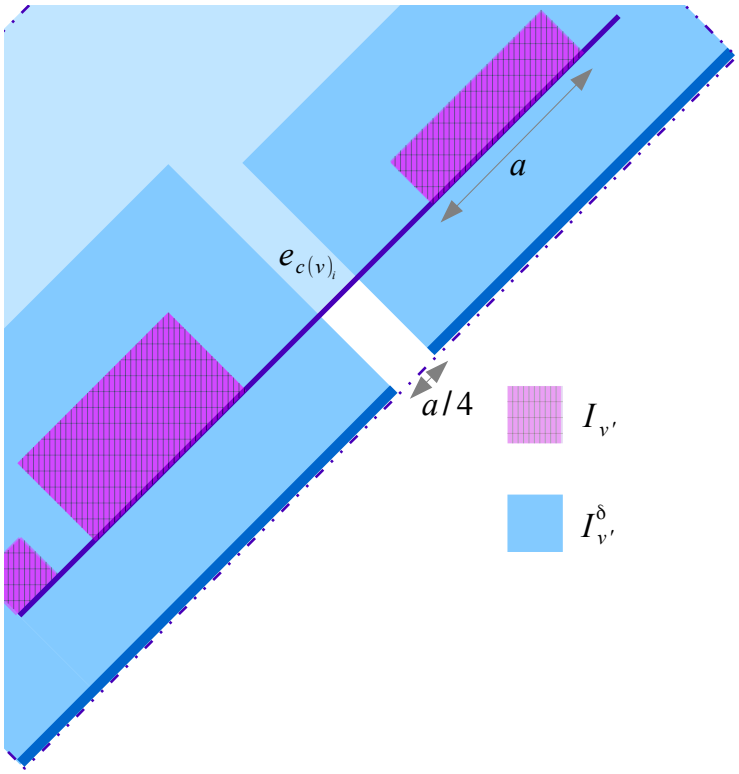

Figure 2.6: Illustration of inflating $I_{v'}$.

number of edges in $I_{v'}$ of

$$\begin{aligned} & 8n_v m + \sum_{i=1}^{n_v} n_v(2m_i + 16m - 4) \\ = \; & 8n_v m + n_v^2(16m-4) + n_v \sum_{i=1}^{n_v} m_i \\ = \; & 4n_v^2(4m-1) + n_v(8m + \sum_{i=1}^{n_v} m_i). \end{aligned} \tag{2.2}$$

Part 2 of Lemma 2.8: Let $e'_{c(v)_i}$ and $e''_{c(v)_i}$ be two edges in $I_{v'}$ which originate from the same edge $e_{c(v)_i}$ of $I^{\delta}_{c(v)_i}$. Inflating $I_{v'}$ by δ under the L_1-norm amounts to adding a shifted version of $\diamond$ centered at all points of its boundary since $I^{\delta}_{v'} = I_{v'} \oplus \diamond$. Let e_v be a certain boundary edge of $I_{v'}$. The boundary edge e'_v of $I^{\delta}_{v'}$ that originates from e_v will be shifted parallel by an amount of $\frac{a}{2}$ and will be extended by $\frac{a}{2}$ at both ends. Therefore in $I^{\delta}_{v'}$ $e'_{c(v)_i}$ and $e''_{c(v)_i}$ will merge into one edge if they are no more than a apart in $I_{v'}$ (the same arguments

hold for edges of I_v), see Figure 2.6. This fact implies that every edge of $I^\delta_{c(v)_i}$ with length l can cause at most $\lceil \frac{l}{a} \rceil$ edges in $I^\delta_{v'}$ and every edge of I_v can cause at most one edge in $I^\delta_{v'}$, no matter into how many pieces it has been divided during the last intersection step.
We again use the fact that $I^\delta_{c(v)_i}$ is $\diamond$-fat and covers an area of at most $4a^2m$: The largest number of edges in $I^\delta_{v'}$ that originate from edges of $I^\delta_{c(v)_i}$ is achieved when $I^\delta_{c(v)_i}$ has $m_i - 4$ short edges (since they generate many edges without decreasing the total edge length of $I^\delta_{c(v)_i}$ so much), each causing at most one edge in $I^\delta_{v'}$ and four long edges of length at most $4am$, each causing up to $4m$ new edges.
Summing up over all these edges results in at most $m_i - 4 + 16m$ edges that arise from edges of $I^\delta_{c(v)_i}$ for every $1 \leq i \leq n_v$ and $4m$ edges from I_v in $I^\delta_{v'}$. Hence the maximum number of edges in $I^\delta_{v'}$ is

$$4n_v(4m-1) + 4m + \sum_{i=1}^{n_v} m_i. \tag{2.3}$$

Dropping the general position assumption for the vertices of $I_{c(v)_i}$ and I_v has the effect that close by or overlapping edges of the same type may merge during the process, which only *decreases* the largest possible length of edges in the above arguments. Also, there may be some connected components that consist of one vertex. However, they can be seen as rectangles with very small edge length in general position, hence this also does not increase the description complexity. Overall, the description complexity of all sets involved is bounded from above by the description complexity of the corresponding sets in general position. □

We can now state the runtime of Algorithm 1 for EGSM instances under the directed L_1-Hausdorff distance:

Theorem 2.9 *Problem 4 can be decided in $O(n^2m(\log m + \log n))$ time under the directed L_1-Hausdorff distance for neighborhood graphs that are trees (also when reporting a witness for a* YES*-instance).*

Proof. The first step of Algorithm 1 is to compute the regions I_p for all $p \in P$, the initial admissible translations stored in the vertices of G. These sets are represented by the boundary edges of squares centered in $q - p$ for each $q \in Q$. It takes $O(m)$ time to determine I_p for a fixed p. Since the squares form a collection of pseudo-discs their union has description complexity $O(m)$ and can be computed in $O(m \log m)$ time [16, Lemma 2].

Let T_v be a subtree of T_r ($v \neq r$) consisting of k_v vertices. We let Algorithm 1 run on this subtree until it stops or vertex v has been contracted to a new vertex v' and the resulting set $I_{v'}$ has been inflated to $I^{\delta}_{v'}$. According to Equation (2.3) of Lemma 2.8, the maximum number of edges describing set $I^{\delta}_{v'}$ is given by

$$\begin{aligned} & 4n_v(4m-1) + 4m + \sum_{i=1}^{n_v} m_i \\ = \;& 4m + \sum_{i=1}^{n_v} (4(4m-1) + m_i) \end{aligned}$$

where n_v is the number of children of v and m_i is the maximum number of edges describing $I_{c(v)_i}$. We can repeatedly apply Equation (2.3) until we get to the sets stored in the leaves of T_v which consists of $4m$ edges at most. By doing so, we see that every vertex of T_v adds $4m + 4(4m-1)$ to the maximum number of edges describing set $I^{\delta}_{v'}$. Hence the term above can be simplified to

$$\begin{aligned} & 4m + \sum_{1}^{k_v - 1} (4(4m-1) + m_i) \\ \leq \;& 4(k_v - 1)(4m-1) + 4mk_v \\ = \;& 4k_v(5m-1) - 4(4m-1), \end{aligned}$$

Hence the set $I^{\delta}_{v'}$ has description complexity $O(k_v m)$.
Each of the n_r children of the root r is the root of a subtree $T_{c(r)_i}$ containing $k_{c(v)_i}$ vertices for $1 \leq i \leq n_r$. Running Algorithm 1 on $T_{c(r)_i}$ and inflating the admissible regions of its root $c(r)_i$ results in a set $I^{\delta}_{c(r)'_i}$ of translations. The set $I^{\delta}_{c(r)'_i}$ is $\diamond$-fat and has description complexity $O(k_{c(v)_i} m)$ due to Lemma 2.8, part 2.
After the last step of the algorithm, according to Equation (2.2), the maximum number of edges describing the set of admissible translations stored in r is

$$\begin{aligned} & 4n_r^2(4m-1) + 2n_r(4m + \sum_{i=1}^{n_r} m_i) \\ \overset{(2.3)}{=} \;& 4n_r^2(4m-1) + 2n_r\Big(4m + \sum_{i=1}^{n_r} \big(4k_{c(v)_i}(5m-1) - 4(4m-1)\big)\Big) \\ = \;& 4n_r^2(4m-1) + 8n_r m + 2n_r \sum_{i=1}^{n_r} 4k_{c(v)_i}(5m-1) - 2n_r^2 4(4m-1). \end{aligned}$$

Since $\sum_{i=1}^{n_r} k_{c(v)_i} = n - 1$ this can be simplified to

$$\begin{aligned} & 4n_r^2(4m-1) + 8n_r m + 8n_r(n-1)(5m-1) - 2n_r^2 4(4m-1) \\ = \; & 4n_r((n_r+2)(1-4m) + 2n(5m-1)). \end{aligned}$$

So the description complexity of the set of translations stored in r after the last step of the algorithm is $O(n_r^2 m + n_r nm) = O(n^2 m)$.
Adding up the time that is required to carry out the union/intersection operations to compute the intermediate admissible translations $I_{v'}$ for all vertices v of T_r gives the total runtime of the algorithm.
$I_{v'}$ has complexity $O(n_v^2 m + n_v nm)$ (where n_v is the number of children of v), as each $I^{\delta}_{c(v)_i}$ has description complexity $O(nm)$.
Adding up the sizes of these sets over all vertices v of T_r for which these admissible translations are computed during the algorithm results in a total complexity of $O(n^2 m)$. The overall runtime that has to be spent in each vertex to perform the union/intersection operations on sets of size $O(n_v^2 m + n_v nm)$ can be bounded from above by considering the respective runtime for a set of total size $O(n^2 m)$. By using a simple sweep line argument we can carry out these operations in $O(n^2 m(\log m + \log n))$ time. Since in case of a YES-instance, the actual set $I_{r'}$ is computed by using a bottom-to-top strategy and thus computing $I_{v'}$ and $I^{\delta}_{v'}$ for all vertices v of T_r along the way, we can conclude that the total runtime of the algorithm is $O(n^2 m(\log m + \log n))$.
Note that a witness for a YES-instance can easily be computed without changing the runtime of the algorithm by picking a translation from $I_{r'}$ and computing a sequence of admissible translations top-to-bottom from there using the already computed sets of admissible translations for the respective subtrees of T_r. □

It is easy to see that all results and arguments stated in this section also hold when considering the L_∞-norm instead of the L_1-norm. The squares that constitute the admissible regions stored in the vertices of T_r at the start of Algorithm 1 differ slightly from the ones in the L_1-case (they are rotated by $2^{-1}\pi$ and their size differs by a constant factor). However, the algorithm and the runtime analysis also hold for instances under the L_∞-norm.

On Lower Bounds

Although we do not know if the upper bounds given in Lemma 2.8 on the complexity of each $I_{v'}$ and $I^{\delta}_{v'}$ are tight, the following example shows that the bound given in Theorem 2.9 on the sum of the complexities of the sets $I^{\delta}_{v'}$ and $I_{r'}$ actually is.

Example 2.10 *Let G be a path and let the points of Q be placed as indicated in Figure 2.7, so that I_v is of the shape indicated in Figure 2.8: $(m-2)$ squares form a "staircase" with steps (which consist of two vertices) of length and height $\frac{a}{n(m-2)}$ framed by one more square on each side.*

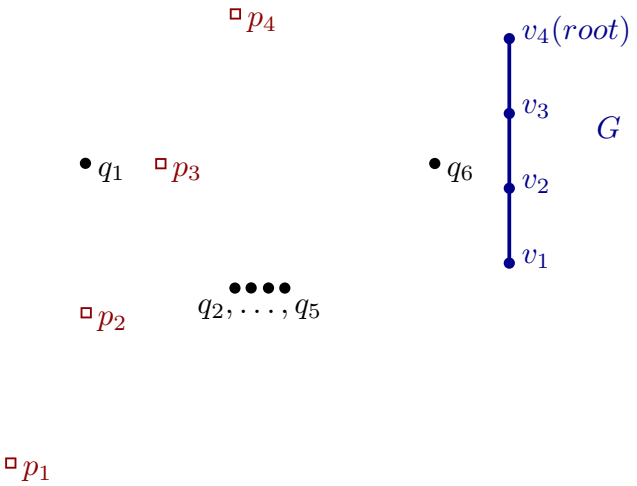

Figure 2.7: Example of the sets P (red), Q (black) and neighborhood graph G (blue) for $n = 4$ and $m = 6$.

As a result, the boundary of I_v consists of more than $2(m-2)$ vertices. Inflating I_v by δ does not change the size of the staircase and all I_v have the same geometric structure since they are translational copies of each other.

Let the points of P be placed so that the staircases of I_v and $I^{\delta}_{c(v)}$ for each vertex v of G are connected (and form one continuous staircase). As Figure 2.9 illustrates, the squares framing the staircase ensure that the staircases of I_v as well as $I^{\delta}_{c(v)}$ are part of their intersection. As a result $(m-2)$ steps are added to the staircase in each step of Algorithm 1 and the boundary of the set of admissible translations stored in the i-th vertex of the path consists of more than $2i(m-2)$ vertices after updating. Finally in last step of the algorithm a staircase that consists of $n(m-2)$ stairs is generated and therefore the total complexity of all these boundaries is $\Omega(n^2 m)$.

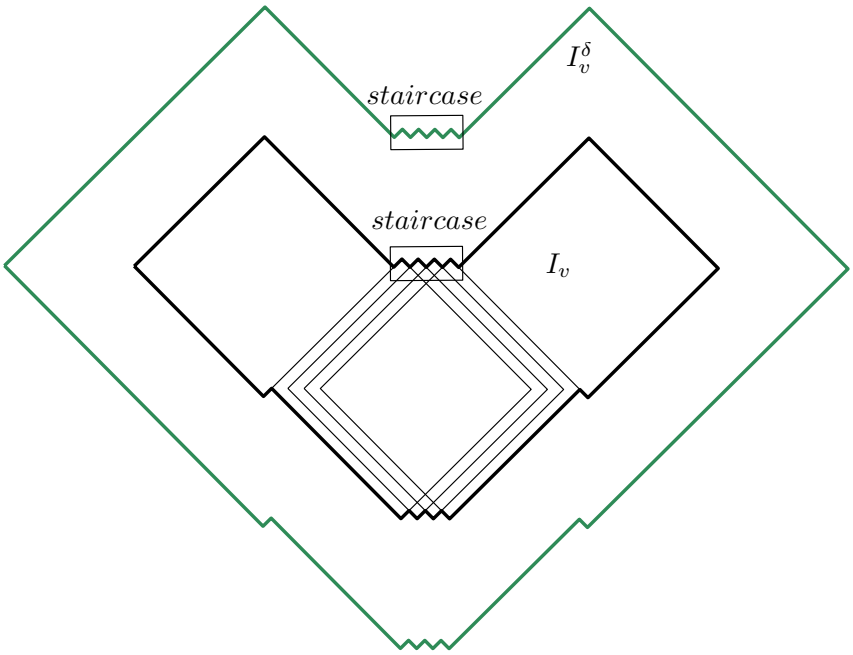

Figure 2.8: The sets I_v (black) and I_v^δ (green) for the point sequences given in Figure 2.7.

2.3 Elastic Geometric Shape Matching under Polygonal Norms

In many elastic shape matching applications, the L_2-Hausdorff distance seems to be a more natural measure than the L_1-Hausdorff distance. A very similar strategy, i.e., computing, inflating and propagating admissible regions from bottom-to-top, can be applied to decide Problem 4 in this setting. However, inflating regions with an L_2-disk by computing the Minkowski sum instead of an L_1-disk may increase the complexity of the respective regions. In this setting, admissible regions form an alternating sequence of circular arcs and vertices in which two arcs meet. When inflating an admissible region by computing its Minkowski sum with an L_2-disk the radii of the original arcs increase and vertices become arcs. Hence, the complexity of a region might double, see Figure 2.10. Although we have not been able to prove this, we expect that the complexity of the root, i.e., one end vertex of a path, becomes exponential when regions are propagated and doubled in each step along a path. For a more detailed discussion of the matter, see Section 5.3.

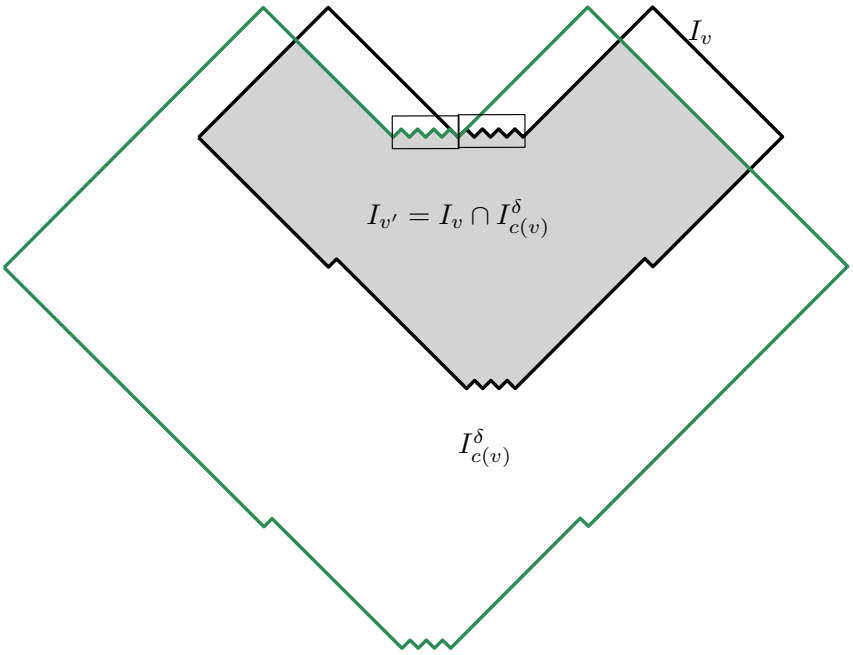

Figure 2.9: The sets $I^{\delta}_{c(v)}$ (green) and I_v (black) with $m-2$ stairs each and $I_{v'}$ (grey area).

Approximating the L_2-case

Since an L_1-disk of radius δ is contained in an L_2-disk of that radius, we can use Algorithm 1 to approximate the Euclidean setting:

Corollary 2.11 *Algorithm 1 gives a $\sqrt{2}$-approximation for Problem 4 with the same settings under the L_2-norm.*

Approximating here means that if $\delta^{opt}_{L_1}$ is the optimal (smallest, i.e., the value that permits a YES-instance in Problem 4) value of δ in the L_1 setting, the optimal value $\delta^{opt}_{L_2}$ in the Euclidean setting is bounded by

$$\delta^{opt}_{L_2} \leq \delta^{opt}_{L_1} \leq \sqrt{2}\,\delta^{opt}_{L_2}.$$

It is easy to see that, when the unit disk of the underlying metric is a regular polygon, the complexity of the admissible regions stored in the vertices of the graph at hand does not differ by more than a constant factor from the complexity of the admissible regions in for the same EGSM instance under the L_1-norm. This allows us to improve upon the approximation factor: Let $\mathcal{P}_{2b}$ ($b \in \mathbb{N}$, $b \geq 2$) be a regular polygon with $2b$ vertices, centered at the origin with radius 1 (where radius means the distance between the center and a vertex

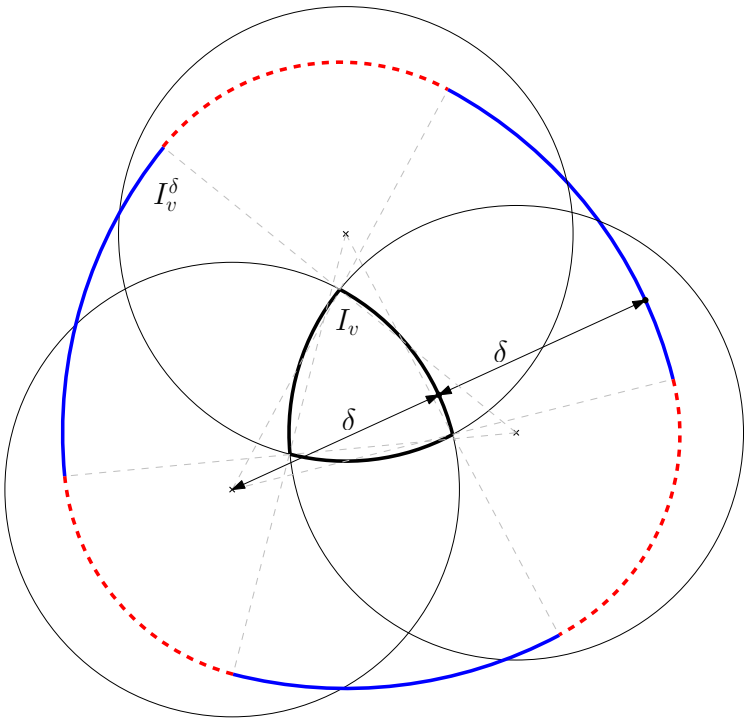

Figure 2.10: Illustration of an initial (I_v) and inflated set (I_v^δ) of admissible translations under the L_2-norm. I_v consists of three arcs, as it is the intersection of three discs of radius δ. I_v^δ consists of six arcs: The arcs in blue correspond to arcs of I_v, arcs in red are centered in vertices of the boundary of I_v.

of the polygon), see Figure 2.11. $\mathcal{P}_{2b}$ is centrally-symmetric as it has an even number of vertices and induces a norm and hence a metric to which we refer as the $L_{\mathcal{P}_{2b}}$-metric (see [23]).

For a point $c \in \mathbb{R}^2$ and $r \in \mathbb{R}^+$, let $\mathcal{P}_{2b}(c, r)$ be the $\mathcal{P}_{2b}$ disk centered in c with radius r.

We can state a version of Algorithm 1 under the $L_{\mathcal{P}_{2b}}$-metric:

Algorithm 2 *We are given the point sets* $P = \{p_1, \ldots, p_n\}$ *and* $Q = \{q_1, \ldots, q_m\}$*, a tree* $G = (V, E)$*, a parameter* $\delta \geq 0$ *and an integer* $b > 0$*.*

We pick an arbitrary vertex $r \in V$ *and henceforth consider* T_r*, the tree rooted in* r*.*

For an internal vertex $v \in V$ *let* $c(v)_1, \ldots, c(v)_{n_v}$ *be the* n_v *children of* v*. For any vertex* $v \in V$ *let* T_v *be the subtree of* T_r *with root* v*. In each iteration of the algorithm, we call the tree from which a vertex is selected the* current tree*.*

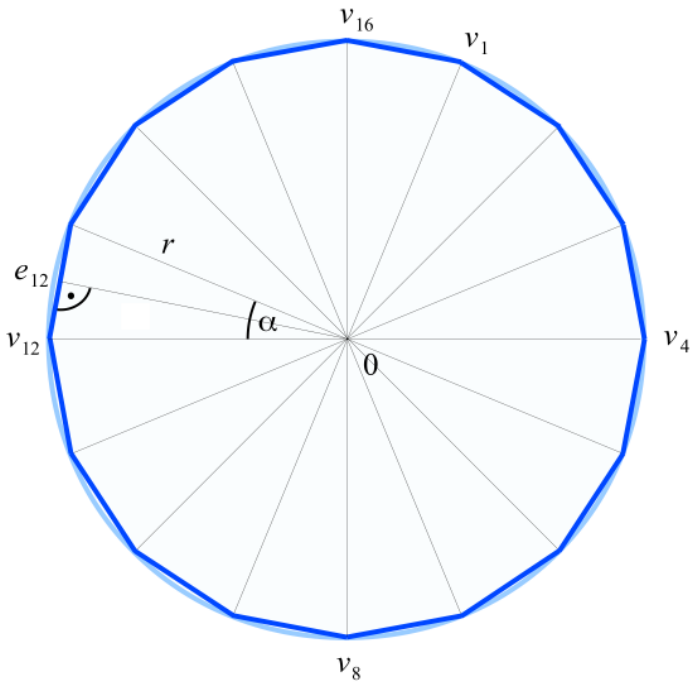

Figure 2.11: $\mathcal{P}_{16}$ with radius r.

Every vertex v corresponds to a point $p \in P$. At start, the set

$$I_v = I_p = \bigcup_{q \in Q} \{t \in T \mid \|p + t - q\|_{\mathcal{P}_{2b}} \leq \delta\},$$

where $\|\cdot\|_{\mathcal{P}_{2b}}$ denotes the $L_{\mathcal{P}_{2b}}$-norm, is stored in every node v of the tree. In each iteration of the algorithm, a vertex v of the current tree is selected with the property that all children of v are leaves or vertices which already have been updated. Then, the admissible translations of v and those of the children of v are merged into a new set of admissible translations that is stored in the new vertex v', the updated version of v. To compute the set $I_{v'}$ of admissible regions for v' we proceed as follows:

1. *We* inflate *all regions $I_{c(v)_i}$ by δ for $1 \leq i \leq n_v$ which results in a set*

$$I^{\delta}_{c(v)_i} := I_{c(v)_i} \oplus \mathcal{P}_{2b}.$$

2. *We compute the admissible region $I_{v'}$ for the new vertex v' as follows (see Figure 2.2):*

$$I_{v'} = \left(\bigcap_{i=1}^{n_v} I^{\delta}_{c(v)_i} \right) \cap I_v.$$

This process is repeated until one of the following cases occurs:

1. *There is a vertex v with $I_v = \varnothing$ (after a contraction):*
   The process stops and NO *is returned as the answer to Problem 4 under the directed $L_{\mathcal{P}_{2b}}$-Hausdorff distance.*
2. *The root r is updated and $I_{r'} \neq \varnothing$:*
   The algorithm terminates and returns YES *as the answer to Problem 4 under the directed $L_{\mathcal{P}_{2b}}$-Hausdorff distance.*

Lemma 2.12 *Given $c_1, \ldots, c_n \in \mathbb{R}^2$, $r_1, \ldots, r_n \in \mathbb{R}^+$, $2 \leq b \in \mathbb{N}$, let*

$$\mathcal{F} \coloneqq \bigcap_{i=1}^{n} \mathcal{P}_{2b}(c_i, r_i).$$

The following holds:

1. *The region $\mathcal{F}$ has description complexity $O(b)$.*
2. *The Minkowski sum $\mathcal{F} \oplus \mathcal{P}_{2b}$ has description complexity $O(b)$.*

Proof. The boundary of each $\mathcal{P}_{2b}(c_i, r_i)$ consists of $2b$ edges. Each edge has a different oriented slope and the order in which they occur along the boundary is fixed. Since all polygons are translated copies of each other, $\mathcal{F}$ consists of edges with at most $2b$ differently oriented slopes. Since $\mathcal{F}$ is constructed by intersecting different convex $\mathcal{P}_{2b}$ polygons, it is convex and its boundary consists of a subset of the initial boundary edges (or parts of edges) of the involved polygons $\mathcal{P}_{2b}$. As each slope can be present at most once in the boundary of $\mathcal{F}$, the description complexity of $\mathcal{F}$ is $O(b)$.
Since the Minkowski sum of two convex figures is convex and both, the boundary of $\mathcal{F}$ and $\mathcal{P}_{2b}$, consist of edges with $2b$ different oriented slopes in total, the description complexity of $\mathcal{F} \oplus \mathcal{P}_{2b}$ is at most $2b$. □

Theorem 2.13 *Problem 4 can be decided in $O(bn^3m^2(\log m + \log n + \log b))$ time under the directed $L_{\mathcal{P}_{2b}}$-Hausdorff distance for neighborhood graphs that are trees (also when reporting a witness for a* YES*-instance).*

Proof. We use Algorithm 2. First one has to compute the sets $I_p = \{\mathcal{P}_{2b}(q - p, \delta) \mid q \in Q\}$ for all $p \in P$ in $O(bm)$ time each. Since the polygons form a collection of pseudo-discs, their union has description complexity $O(bm)$ and can be computed in $O(bm(\log m + \log b))$ time. Each set I_p is represented by the sequences of line segments that define its boundaries, which can be stored in, e.g., a doubly connected edge list.
Intersecting nm convex objects gives at most n^2m^2 new objects. At the beginning there are $O(nm)$ polygons with description complexity $O(b)$ each,

since each set I_p consists of m polygons. According to Lemma 2.12, inflating one of these polygons or a fragment (i.e., a set formend by their intersection) does not change its description complexity. Hence the complexity of the resulting region after intersecting and joining the initial regions one-by-one is bounded by $O(bn^2m^2)$. In particular this means that each of the sets I_p in any iteration of the algorithm can be described by $O(bn^2m^2)$ vertices or line segments. Intersecting or joining two sets of polygons of size $O(bn^2m^2)$ can be done with a sweep line algorithm in $O(bn^2m^2(\log m + \log n + \log b))$ time.
There are n union operations (once for each initial I_p). Since there are n vertices in the neighborhood graph, there are at most n intersection operations. Hence the algorithm takes $O(bn^3m^2(\log m + \log n + \log b))$ time in total. □

Approximating the Optimization Version

Problem 4 can easily be changed into an optimization problem. Then, one seeks the smallest δ^{opt} such the corresponding EGSM instance is a YES-instance for all $\delta \geq \delta^{opt}$ and a NO-instance for all $\delta < \delta^{opt}$. By substituting L_2-discs by a regular polygon, the optimization version of Problem 4 for the L_2-norm can be approximated, since

$$\frac{\left|\|x\|_2 - \|x\|_{\mathcal{P}_{\sqrt{k}}}\right|}{\|x\|_2} = \left(1 + \frac{1}{O(k)}\right)$$

A careful analysis gives the following:

Lemma 2.14 *Given an integer $k \geq 2$, the result of the optimization version of Problem 4 under the $\mathcal{P}_{2b}$-norm for $b = \left\lceil \frac{\pi}{4}\sqrt{2(k+1)} \right\rceil$ gives a $\left(1 + \frac{1}{k}\right)$-approximation for the optimization version of Problem 4 with the same settings under the L_2-norm.*

Proof. Let $\delta^{opt}_{\mathcal{P}_{2b}}$ be the optimal (smallest) value in the $\mathcal{P}_{2b}$ setting and $\delta^{opt}_{L_2}$ be the optimal value in the Euclidean setting. As the values are as small as possible, some of the valid translations $t \in T$ are forced to be on the boundary of the L_2 respectively $\mathcal{P}_{2b}$ disks or arcs, thus it is sufficient to analyze this extreme setting.
Since the L_2-disk with radius $\delta^{opt}_{L_2}$ is the circumcircle of the $\mathcal{P}_{2b}$-disk with radius $\delta^{opt}_{L_2}$, the inequality

$$\delta^{opt}_{L_2} \leq \delta^{opt}_{\mathcal{P}_{2b}}$$

holds.
Moreover, the smallest L_2-disk that still touches the boundary of a $\mathcal{P}_{2b}$-disk with radius $\delta^{opt}_{\mathcal{P}_{2b}}$ is obviously its in-circle. So the radius of the in-circle of a

$\mathcal{P}_{2b}$-disk with radius $\delta^{opt}_{\mathcal{P}_{2b}}$ is also a lower bound for $\delta^{opt}_{L_2}$. The value $\delta^{opt}_{L_2}$ can be expressed as a function of $\delta^{opt}_{\mathcal{P}_{2b}}$ using the cosine rule: A segment of the polygon formed by its center and two adjacent vertices is also an isosceles triangle with angle $\alpha = \frac{\pi}{b}$ opposite to its base, see Figure 2.12. Its bisecting line has length

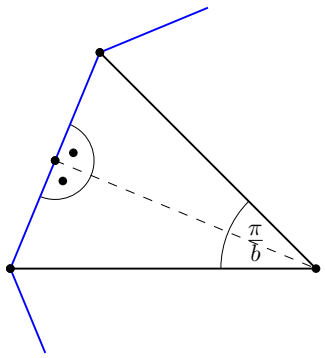

Figure 2.12: A segment of a polygon formed by its center and two adjacent vertices with angle $\frac{\pi}{b}$ opposite to its base.

$\delta^{opt}_{L_2}$, thus

$$\delta^{opt}_{L_2} \geq \delta^{opt}_{\mathcal{P}_{2b}} \cdot \cos\left(\frac{\pi}{2b}\right).$$

Writing the cosine as power series leads to

$$\begin{aligned}
\delta^{opt}_{L_2} &\geq \delta^{opt}_{\mathcal{P}_{2b}} \cdot \sum_{n=0}^{\infty} (-1)^n \frac{\pi^{2n}}{(2b)^{2n} \cdot (2n)!} \\
&= \delta^{opt}_{\mathcal{P}_{2b}} \cdot \sum_{n=0}^{\infty} \left((-1)^{2n} \frac{\pi^{4n}}{(2b)^{4n} \cdot (4n)!} + (-1)^{2n+1} \frac{\pi^{4n+2}}{(2b)^{4n+2} \cdot (4n+2)!} \right) \\
&= \delta^{opt}_{\mathcal{P}_{2b}} \cdot \sum_{n=0}^{\infty} \left(\frac{\pi^{4n}}{(2b)^{4n} \cdot (4n)!} - \frac{\pi^{4n+2}}{(2b)^{4n+2} \cdot (4n+2)!} \right) \\
&= \delta^{opt}_{\mathcal{P}_{2b}} \cdot \sum_{n=0}^{\infty} \frac{\pi^{4n}((4n+2)(4n+1)4b^2 - \pi^2)}{(2b)^{4n+2} \cdot (4n+2)!}
\end{aligned}$$

Since $b \geq 1$,

$$\pi^{4n}((4n+2)(4n+1)4b^2 - \pi^2) > 0,$$

and thus

$$\begin{aligned}
\delta_{L_2}^{opt} &\geq \delta_{\mathcal{P}_{2b}}^{opt} \cdot \sum_{n=0}^{\infty} \frac{\pi^{4n}((4n+2)(4n+1)4b^2 - \pi^2)}{(2b)^{4n+2} \cdot (4n+2)!} \\
&\geq \delta_{\mathcal{P}_{2b}}^{opt} \cdot \sum_{n=0}^{0} \frac{\pi^{4n}((4n+2)(4n+1)4b^2 - \pi^2)}{(2b)^{4n+2} \cdot (4n+2)!} \\
&= \delta_{\mathcal{P}_{2b}}^{opt} \cdot \left(1 - \frac{\pi^2}{8b^2}\right) \\
\Leftrightarrow \delta_{\mathcal{P}_{2b}}^{opt} &\leq \left(1 - \frac{\pi^2}{8b^2}\right)^{-1} \cdot \delta_{L_2}^{opt}.
\end{aligned}$$

Choosing $b = \left\lceil \frac{\pi}{4}\sqrt{2(k+1)} \right\rceil$ finally results in

$$\delta_{\mathcal{P}_{2b}}^{opt} \leq \left(1 + \frac{1}{k}\right) \cdot \delta_{L_2}^{opt},$$

which proofs the statement. □

In other words, Algorithm 2, or $A2$ in short, gives a $\left(1 + \frac{1}{k}\right)$-approximation for the optimization version of Problem 4 with the same settings under the L_2-norm by approximating every L_2-disk with a regular polygon with $O(\sqrt{k})$ vertices. This algorithm will be also used in Chapter 4.

2.4 On Possible Modifications

Problem 4 is a restricted instance of the *general* EGSM framework, where, e.g., different translation classes and objectives are also discussed, see Section 1.4. We want to briefly mention two aspects of the general formulation that are not present in Problem 4 but that can easily be incorporated into the algorithms that are discussed so far.

1. **The objective**. Minimizing the maximum of two distances, one between pattern and model and one in translation space, results in an equilibrium of the two, i.e., for the smallest value of δ that permits a YES-instance we have that both distances will be equal to δ. Depending on the application at hand, it might be reasonable to weigh these two aspects differently. Algorithm 1 can incorporate this by using different radii for the initial admissible translation discs and for the discs of the *inflation step*.

2. **The neighborhood graph**. In Problem 4 the graph G that encodes the similarity constraints on the translations is a tree without weights. By putting weights on the edges, it is possible to specify *how* similar two translations have to be. By putting weights on the vertices, it is possible to encode how well the respective subshape of the pattern has to match the model (this allows different directed Hausdorff distances for different subshapes). Both variants can be incorporated into the algorithm, again by simply using discs of different radii for each individual vertex and edge of the neighborhood graph.

In all these cases, the proof of correctness in Theorem 2.4 can easily be adapted for the modified algorithm. However, the runtime analysis of Algorithm 1 relies on upper bounds on the combinatorial complexity of certain geometric structures that are constructed by the algorithm. These bounds, amongst other things, depend on the fact that central building blocks used by the algorithm have the same size. This is crucial for Lemma 2.8, where upper bounds on description complexity of admissible regions that appear during Algorithm 1 are given. Since this is no longer true in the modified algorithm(s), we do not have (non-trivial) bounds for the runtime of these variants.

Chapter 3

An FPTAS for an Elastic Shape Matching Problem with Cyclic Neighborhoods

In Chapter 2 we considered a variant of the problem for trees under the L_1-, the L_∞-norm and under polygonal norms, and presented a polynomial time algorithm that gives a $(1+\epsilon)$-approximation to the optimization variant of the same problem under the L_2-norm. However, if the neighborhood graph contains at least one cycle, the strategy of the algorithms introduced in Chapter 2 does not work and at this point there are no results regarding problem instances where the neighborhood graph contains cycles. In particular, there is no literature that deals with efficient exact or approximation algorithms for EGSM instances where G contains a cycle. Moreover, we do not know if the problem is NP-hard.
In the following we focus on Problem 3 for neighborhood graphs that are simple cycles under the Euclidean 1-to-1-distance and provide an FPTAS for this problem. It is also possible to return a valid sequence of translations without increasing the runtime. The results of this chapter are stated in $\mathbb{R}^2$ can also be extended to $\mathbb{R}^d$ for any $d \in \mathbb{N}$ with $d > 2$.
This research has been published in [A39].

3.1 Problem Statement

In the following $\|\cdot\|$ denotes the L_2-norm in $\mathbb{R}^2$. Also, all index arithmetic is modulo n.
We consider the following variant of the EGSM problem:

Problem 5 *Given:*

$$\begin{aligned} P = (p_0, \ldots, p_{n-1}) \quad & \text{a sequence of points (the pattern) and} \\ Q = (q_0, \ldots, q_{n-1}) \quad & \text{a sequence of points (the model).} \end{aligned}$$

Find: A sequence of translations $T = (t_0, \ldots, t_{n-1})$, so that the function

$$\gamma(T, P, Q) := \max\left(\max_{0 \le i < n} \|q_i - (p_i + t_i)\|, \max_{0 \le i < n} \|t_i - t_{i+1}\|\right)$$

is minimized.

The neighborhood graph is given implicitly through the constraints on the translations, which form a simple cycle. Note that in this chapter we name the points of pattern and model starting with index 0 instead of 1 since this allows us to simplify the presentation using all index artihmetic modulo n.

Measuring the distance of the points $(t_i + p_i)$ and q_i in model space is the same as measuring the distance of the points t_i and $q_i - p_i$ in translation space. This is why Problem 5 can be studied in translation space entirely:
Let

$$\begin{aligned} c_i \quad & := q_i - p_i \text{ for } 0 \le i < n \text{ and} \\ C \quad & := (c_0, \ldots, c_{n-1}). \end{aligned}$$

The function $\gamma(T, P, Q)$ can be rewritten as

$$\gamma(T, C) := \max\left(\max_{0 \le i < n} \|c_i - t_i\|, \max_{0 \le i < n} \|t_i - t_{i+1}\|\right).$$

We refer to translations in translation space, i.e., translations, simply as points. First, we introduce some notation:

Notation 3.1 *Let $c, u, v \in \mathbb{R}^2$ and $r > 0$.*

1. *$D_r(c)$ denotes the disk with radius r centered in c and $\partial D_r(c)$ denotes its boundary.*
2. *We define*
$$I_r(c, u, v) := D_r(c) \cap D_r(u) \cap D_r(v).$$

Notation 3.2 *For a given sequence $C = (c_0, \ldots, c_{n-1})$, we define*

$$\delta^* := \min_T \gamma(T, C).$$

Definition 3.3 *We call a sequence of translations* $T = (t_0, \ldots, t_{n-1})$ δ-admissible (for C)*, iff* $\gamma(T, C) \leq \delta$*. A sequence that is* δ^**-admissible is called an* optimal *sequence. We will use the symbol* T^* *to denote an optimal sequence. A translation* t *is called* (δ, i)-admissible (for C)*, iff there is a* δ*-admissible sequence* $T = (t_0, \ldots, t_i = t, \ldots, t_{n-1})$*.*

Strictly speaking, δ^* and the concept of δ-admissibility depend on C, but since C is part of the input and does not vary throughout the subproblems, we refrain from including C in the notation.

3.2 The Algorithm

Observe that there is a simple 3-approximation to δ^*:

Notation 3.4 *We define*

$$\delta^{(3)} := \gamma(C, C).$$

Lemma 3.5 C *gives a* 3*-approximation to* δ^**, i.e.,*

$$\delta^{(3)} \leq 3\delta^*.$$

Proof. The sequence C is $\delta^{(3)}$-admissible by construction. Consider an optimal sequence $T^* = (t_0^*, \ldots, t_{n-1}^*)$. Since $\|c_i - t_i^*\| \leq \delta^*$, $\|t_i^* - t_{i+1}^*\| \leq \delta^*$ and $\|c_{i+1} - t_{i+1}^*\| \leq \delta^*$, it follows that $\|c_i - c_{i+1}\| \leq 3\delta^*$ for all $0 \leq i < n$, and as a consequence

$$\delta^* = \gamma(T^*, C) \geq \frac{1}{3} \max_{0 \leq i < n} \|c_i - c_{i+1}\| = \frac{1}{3}\gamma(C, C) = \frac{1}{3}\delta^{(3)}.$$

□

Lemma 3.5 is the basis for the construction of our FPTAS, since it implies that every $(\delta^*, 0)$-admissible point lies within the disk $D_{\delta^{(3)}}(c_0)$. A simple way to get a $(1+\epsilon)$-approximation to δ^* for some $\epsilon > 0$ is to sample t_0 from a dense enough ϵ_{grid}-grid, e.g., a grid where the distance between samples is at most ϵ_{grid} that covers $D_{\delta^{(3)}}(c_0)$. We call the points of this grid *translation-samples*. Here, $\epsilon_{\text{grid}} = \Theta\left(\epsilon\delta^{(3)}\right)$, so the grid consists of $\Theta(\epsilon^{-2})$ points. We also sample the value δ of the objective function on the interval $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ with sample-distance $\epsilon_{\text{obj}} = \Theta\left(\epsilon\delta^{(3)}\right)$. We call the samples on $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ *radius-samples*, see Figure 3.1. We then test for every radius-sample δ, whether there exists a solution T' so that $\gamma(T', C) \leq \delta$ and a translation-sample is the 0th component of T'. This test is a variant of a problem that has already been studied in [16], where the authors give an algorithm, which we refer to as Algorithm B, that

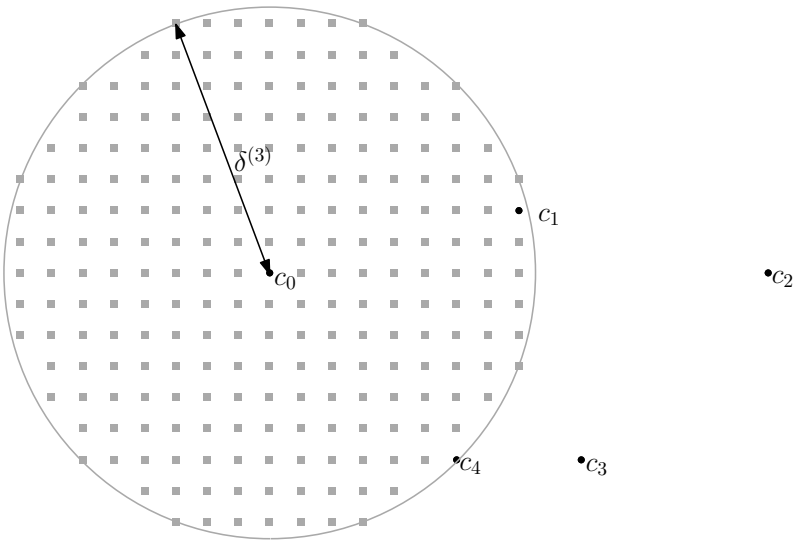

Figure 3.1: For a given set $C = (c_0, \ldots, c_4)$ and radius-sample $\delta^{(3)}$, t_0 is sampled from a grid that covers the disk $C_{\delta^{(3)}}(c_0)$.

solves this problem for paths and the case that only translations in a fixed direction are allowed in $O(n^2 \log n)$ time. We use a slightly modified version of Algorithm B, see Section 3.3. Consequently, this simple FPTAS runs in $O\left(\epsilon^{-3} n^2 \log n\right)$ time.

This result can be improved in several ways. The first obvious improvement is to perform a binary search on $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$, which improves the run-time to $O\left((\log \epsilon^{-1})\, \epsilon^{-2} n^2 \log n\right)$.

The second idea is based on Lemma 3.8 below, which says that for every $\delta \geq \delta^*$, there is a δ-admissible sequence T containing a point t_i that lies on $\partial D_\delta(c_i)$ for some i. Consequently, we do not have to sample the whole disk $D_{\delta^{(3)}}(c_i)$ for the current radius-sample δ, but to only sample $\partial D_\delta(c_i)$. Unfortunately, there is no way to identify the disks (the c_i) with this property, hence it is no longer possible to pick an arbitrary disk and sample it, but we have to sample the boundary of all disks. This changes the run-time to $O\left((\log \epsilon^{-1})\, \epsilon^{-1} n^3 \log n\right)$. Of course, this is only an improvement if $\epsilon^{-1} \gg n$. On the other hand, this strategy enables us to apply another modification: We can approximate each $\partial D_\delta(c_i)$ by a regular polygon with $O\left(\epsilon^{-1/2}\right)$ vertices, see Figure 3.2. Due to the convexity of the problem, we can then perform a binary search on the edges of this polygon and get an FPTAS that runs in $O\left(\epsilon^{-1/2} \left(\log \epsilon^{-1}\right)^2 n^3 \log n\right)$ time.

This gives us the following tradeoff between precision and input size:

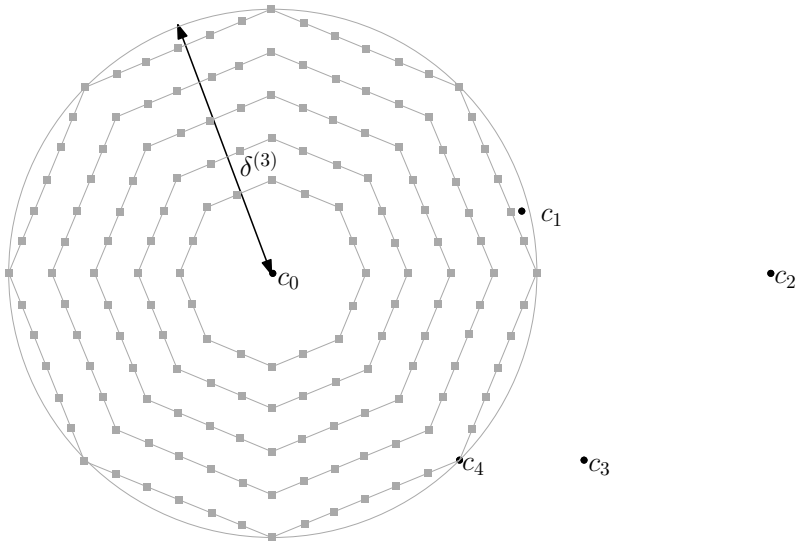

Figure 3.2: For a given set $C = (c_0, \ldots, c_4)$ and different radius-samples $\delta \in [3^{-1}\delta^{(3)}, \delta^{(3)}]$, t_0 is sampled from the boundary of inscibing polygons of $C_\delta(c_0)$.

Theorem 3.6 *We can compute a $(1+\epsilon)$-approximation to δ^* in*

$$O\left(\left(\log \epsilon^{-1}\right) \epsilon^{-2} n^2 \log n\right) \text{ time and in}$$
$$O\left(\epsilon^{-1/2} \left(\log \epsilon^{-1}\right)^2 n^3 \log n\right) \text{ time.}$$

Since it is clear how to implement the approximation when sampling the interior of $D_{\delta^{(3)}}(c_0)$, we elaborate on the improvements of the second strategy.

3.3 A Detailed Description of the Algorithm

First, we state the actual algorithm:

Algorithm 3 *We are given a sequence of points $C = (c_0, \ldots, c_{n-1})$ and an $\epsilon > 0$.*
For every index $0 \leq i < n$, we repeat the following strategy:
We sample the value δ of the objective function by performing a binary search on the interval $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ up to accuracy $\epsilon_{\text{obj}} = \Theta\left(\epsilon\delta^{(3)}\right)$.
For every radius-sample δ, we approximate $\partial D_\delta(c_i)$ by an inscribing regular polygon with $O\left(\epsilon^{-1/2}\right)$ vertices and sample the translation t_i from the boundary of the polygon by performing a binary search on every of its edges up to accuracy $\epsilon_{\text{edge}} = \Theta\left(\epsilon\delta^{(3)}\right)$.

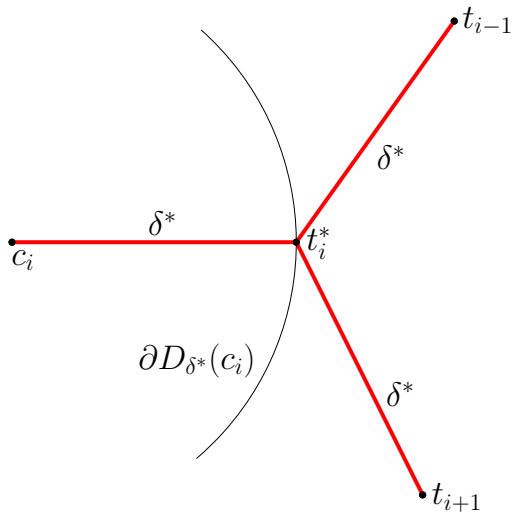

Figure 3.3: Illustration of the key-point t_i^*, a (δ^*, i)-admissible point of $T^* = (t_0^*, \ldots, t_{n-1}^*)$ with $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*) = \{t_i^*\}$ and $t_i^* \in \partial D_{\delta^*}(c_i)$.

We test for every pair of radius-sample δ and translation-sample t_i, whether there exists a solution T' so that $\gamma(T', C) \leq \delta$ and the translation-sample t_i is the ith component of T'. This test, called Algorithm 4, is a variant of Algorithm B*, and is elaborately described in Section 3.3.*

On (δ, i)-Admissible Points

The reason why it suffices to sample the boundaries of all disks rather than sampling the interior of one disk with a grid is that any optimal solution T^* contains a *key-point*:

Definition 3.7 *A (δ^*, i)-admissible point t_i^* of $T^* = (t_0^*, \ldots, t_{n-1}^*)$ is called a* key-point*, iff $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*) = \{t_i^*\}$ and $t_i^* \in \partial D_{\delta^*}(c_i)$, see Figure 3.3.*

Lemma 3.8 *For every optimal sequence $T^* = (t_0^*, \ldots, t_{n-1}^*)$, there is an index $0 \leq i < n$ so that t_i^* is a key-point.*

Before we can prove Lemma 3.8, we need to consider certain characteristics of δ-admissible sequences $T = (t_0, \ldots, t_{n-1})$. Each of the points t_i of T appears in exactly three of the constraints induced by $\gamma(T, C)$:

$$\begin{aligned} \|t_i - c_i\| &\leq \delta, \\ \|t_i - t_{i-1}\| &\leq \delta \text{ and} \\ \|t_i - t_{i+1}\| &\leq \delta. \end{aligned} \tag{3.1}$$

for all $0 \leq i < n$. These constraints on t_i can be interpreted as disks of radius δ centered in t_{i-1}, t_{i+1} and c_i. Consequently t_i needs to lie in the common intersection $I_\delta(c_i, t_{i-1}, t_{i+1})$ of these three disks, see Figure 3.4.

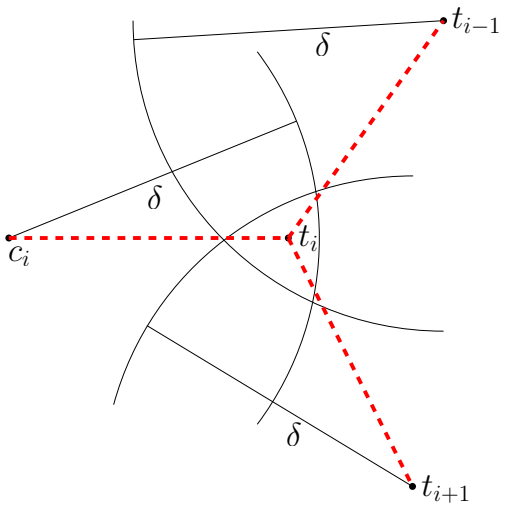

Figure 3.4: The set $I_\delta(c_i, t_{i-1}, t_{i+1})$ is the intersection of $D_\delta(c_i)$, $D_\delta(t_{i-1})$ and $D_\delta(t_{i+1})$. The three constraints of (3.1) in which t_i appears, are visualised by fat line segments. They are not tight, which is indicated with dashed lines.

Proposition 3.9 *Let $T = (t_0, \ldots, t_{n-1})$ be a δ-admissible sequence. The set $I_\delta(c_i, t_{i-1}, t_{i+1})$ consists of exactly one point t_i iff one of the following cases holds:*

1. *The intersection of two of the three disks $D_\delta(c_i)$, $D_\delta(t_{i-1})$ and $D_\delta(t_{i+1})$ consists of one point, i.e., t_i, that in turn lies inside the third disk. Therefore t_i lies on the midpoint of the line segment between the centers of the first two disks, see Figure 3.5. In this case we call t_i a* segment-type *point.*
2. *The intersection between any two of the three disks $D_\delta(c_i)$, $D_\delta(t_{i-1})$ and $D_\delta(t_{i+1})$ consists of more than one point, but the intersection of all three is one point, i.e., t_i, and t_i lies in the interior of the triangle with vertices c_i, t_{i-1} and t_{i+1}. In this case, we call t_i a* point-type *point.*

Proof of Proposition 3.9. "$\Leftarrow$": trivial.
"$\Rightarrow$": The set $I_\delta(c_i, t_{i-1}, t_{i+1})$ is the intersection of three disks. Since $I_\delta(c_i, t_{i-1}, t_{i+1}) = \{t_i\}$, either the intersection of two of the three disks consists of only one point, which equals Part 1, or none of the intersections of two of three disks consists of one point, but the intersection of all three does. In

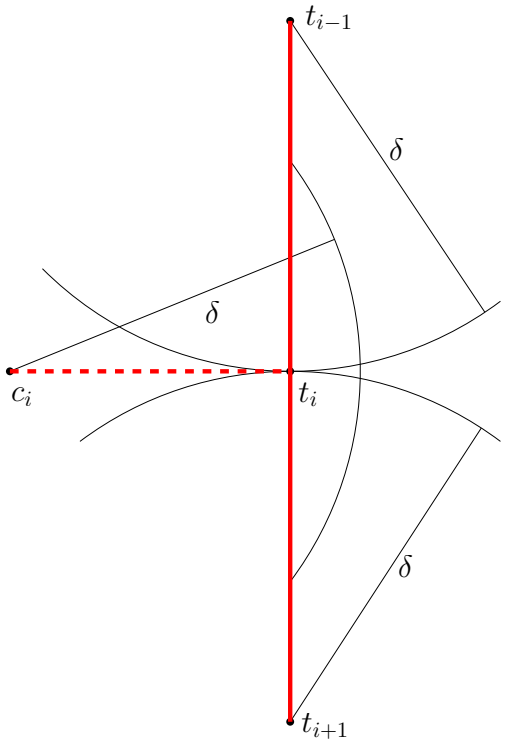

Figure 3.5: The set $I_\delta(c_i, t_{i-1}, t_{i+1})$ consists of one point. The solid fat lines indicate tight constraints.

the latter case, t_i is equally far from c_i, t_{i-1} and t_{i+1}, which implies that t_i is the circumcenter of the triangle with vertices c_i, t_{i-1} and t_{i+1}. Suppose t_i lies outside this triangle; this means that the triangle is obtuse. Let w.l.o.g. the obtuse angle be at c_i, so t_i and c_i are on opposing sides of the line segment $\overline{t_{i-1}t_{i+1}}$, see Figure 3.6. Let t_i' be the midpoint of $\overline{t_{i-1}t_{i+1}}$, so $t_i \neq t_i'$. Then t_i' is closer to c_i, t_{i-1} and t_{i+1} than t_i, which implies that t_i' is (δ, i)-admissible, so $t_i' \in I_\delta(c_i, t_{i-1}, t_{i+1})$, which is a contradiction. □

With this, we can now prove Lemma 3.8:

Proof of Lemma 3.8. Observe that for every sequence $T = (t_0, \ldots, t_{n-1})$ and for all $0 \leq i < n$, the set $I_\delta(c_i, t_{i-1}, t_{i+1})$ is the intersection of three disks and thus (the boundary of) $I_\delta(c_i, t_{i-1}, t_{i+1})$ has a characteristic structure that either resembles a triangle, a lens, or a point. For easier referencing, we associate the different kinds of structures $I_\delta(c_i, t_{i-1}, t_{i+1})$ may have with its appearance. We differentiate between three different kinds of structures:

1. The boundary of $I_\delta(c_i, t_{i-1}, t_{i+1})$ consists of three circular arcs that are connected by vertices. We call this shape *triangle-shape*.

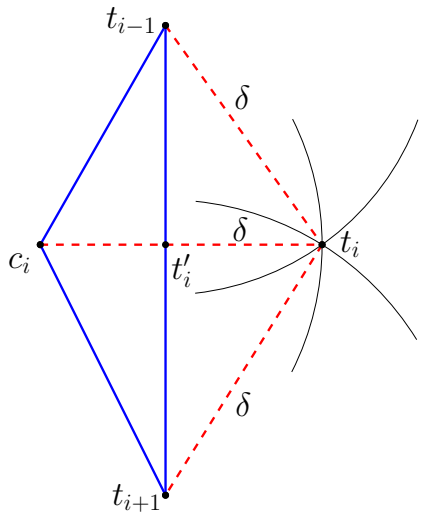

Figure 3.6: Illustration the obtuse triangle between c_i, t_{i-1} and t_{i+1} (solid lines) constructed during the proof of Proposition 3.9.

2. The boundary of $I_\delta(c_i, t_{i-1}, t_{i+1})$ consists of two circular arcs that are connected by vertices. We call this shape *lens-shape*.
3. The set $I_\delta(c_i, t_{i-1}, t_{i+1})$ consists of one point. We call this shape *point-shape*.

Let $T^* = (t_0^*, \ldots, t_{n-1}^*)$ be an optimal sequence.
For the sake of contradiction, we assume that none of the constraints of (3.1) are tight for T^*. As a consequence, δ^* can be reduced by

$$\min\left(\min_{0\le i<n}\left(\delta^* - \|c_i - t_i^*\|\right), \min_{0\le i<n}\left(\delta^* - \|t_i^* - t_{i+1}^*\|\right)\right),$$

which is a contradiction to the optimality of δ^*. Consequently, every T^* contains at least one (δ^*, i)-admissible point that is involved in at least one tight constraint in (3.1).
Suppose that every point of T^* occurs in at most two tight constraints in (3.1) and that $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$ has either triangle- or lens-shape for every index $0 \le i < n$. Then for every $0 \le i \le n-1$, the point t_i^* lies either in the interior of $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$ or on a circular arc (if it occurs in one tight constraint) or on a vertex (if it occurs in two tight constraints) of the boundary of $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$. Let t_i^* be a point of T^* that lies on the boundary of $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$, and let t_i' be a point in the interior of $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$. Then the sequence $(t_0^*, \ldots, t_{i-1}^*, t_i', t_{i+1}^*, \ldots, t_{n-1}^*)$ is optimal and $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$ has either triangle- or lens-shape for all $0 \le i < n$. Since this strategy can be

applied to every point of T^* that occurs in up to two tight constraints, there is an optimal sequence so that none of its points occurs in a tight constraint, which is a contradiction.
Hence, T^* contains at least one point t_i^* so that $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*) = \{t_i^*\}$, e.g., t_i^* has point-shape. According to Proposition 3.9, t_i^* then is either a segment-type point (if it occurs in two tight constraints) or a point-type point (if it occurs in three tight constraints). One of the following two cases holds:

1. The point t_i^* is a segment-type point and $\|c_i - t_i^*\| = \delta^*$ is one of the two tight constraints, or t_i^* is a point-type point. Then, t_i^* lies on $\partial D_{\delta^*}(c_i)$ and hence is a key-point.
2. For every t_i^* with $\{t_i^*\} = I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$ the following holds: t_i^* is a segment-type point and it lies on midpoint of the line segment $\overline{t_{i-1}^* t_{i+1}^*}$. If t_{i+1}^* or t_{i-1}^* occur in only one tight constraint (the constraint in which t_i^* occurs), both points can be shifted towards t_i^* so that $\|t_i^* - t_{i-1}^*\| < \delta^*$ and $\|t_i^* - t_{i+1}^*\| < \delta^*$. As a consequence, δ^* is not minimal, which is a contradiction. Hence $\{t_{i-1}^*\} = I_{\delta^*}(c_{i-1}, t_{i-2}^*, t_i^*)$ and $\{t_{i+1}^*\} = I_{\delta^*}(c_{i+1}, t_i^*, t_{i+2}^*)$. If either of them is a segment-type point and satisfies $\|t_j^* - c_j\| = \delta^*$ for one $j \in \{i-1, i+1\}$, or one of them is a point-type point, a point that lies on $\partial D_{\delta^*}(c_j)$ is found. If both points are segment-type points and $\|t_j^* - c_j\| < \delta^*$ for $j \in \{i-1, i+1\}$, t_{i-2}^* and t_{i+2}^* have to be analysed in the same manner, i.e., their type has to be identified the same way we did for t_i^*. This strategy can be carried forward from point to point along C until one point-type point is found, or all points have been proven to be segment-type points. This implies that all $t_i^* \in T^*$ lie on a straight line and successive points have distance δ^*, which is a contradiction to the fact, that the neighborhood graph is a cycle: Of all points we placed on the straight line, the both furthest away from each other also need to have distance at most δ^*, which is not possible if $n > 2$.

□

Lemma 3.8 contains more information than we need in order to prove that we can sample the boundaries of all disks instead of completely sampling one of the disks. The information that $I_{\delta^*}(c_i, t_{i-1}^*, t_{i+1}^*)$ consists of one point for at least one index i is required later in the arguments for Lemma 3.19.

Approximating δ^*

There is at least one index $0 \leq i < n$ for every $T^* = (t_0^*, \ldots, t_{n-1}^*)$, so that t_i^* is a key-point, which implies that $t_i^* \in \partial D_{\delta^*}(c_i)$. Since we have no way of determining the index i, so that t_i^* is a key-point, the boundaries of all disks

have to be sampled in order to find a suitable approximation to t_i^*. Since we do not know the optimal radius δ^* either, we have to sample the boundary of all disks for dense enough radius-samples in $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$. In order to verify that the binary search on $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ is successful, we have to prove that there actually is a sample-radius $\delta \in [\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ for all $0 \leq i < n$, so that there is a (δ, i)-admissible point on $\partial D_\delta(c_i)$.

Lemma 3.10 *There is a $\delta^{(3)}$-admissible sequence $T = (t_0, \ldots, t_{n-1})$ so that every $(\delta^{(3)}, i)$-admissible point of T lies on $\partial D_{\delta^{(3)}}(c_i)$.*

Proof. Consider the $\delta^{(3)}$-admissible sequence C. If the sequence is shifted $\delta^{(3)}$-far in an arbitrary direction, e.g., $T \coloneqq (t_0, \ldots, t_{n-1})$ with $t_i \coloneqq c_i + (0, -\delta^{(3)})$ for all $0 \leq i < n$, the sequence remains $\delta^{(3)}$-admissible and $t_i \in \partial D_{\delta^{(3)}}(c_i)$ for all $0 \leq i < n$, see Figure 3.7. □

It is easy to see that Lemma 3.10 does not hold for every $\delta \geq \delta^*$, see Figure 3.8.

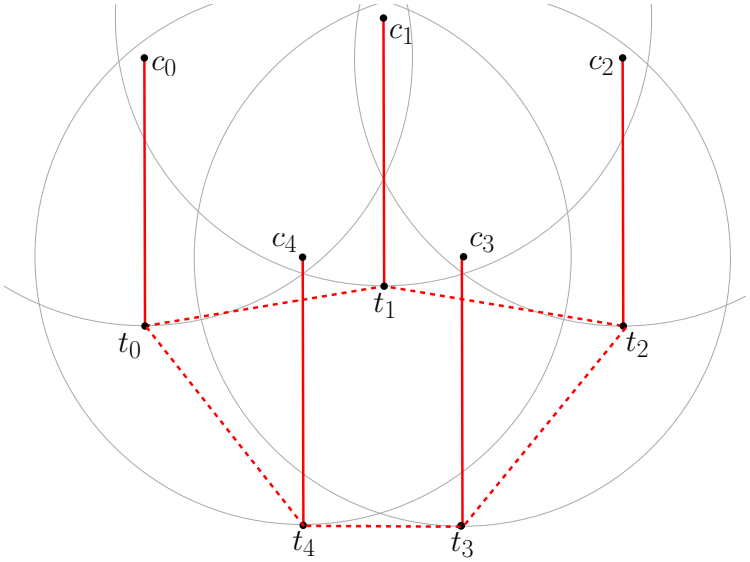

Figure 3.7: Set T as constructed in the proof of Lemma 3.10 with tight constraints (solid fat lines) and constraints that are met with inequality (dashed fat lines).

Notation 3.11 *For every index i, let δ_i^* be the smallest value, so that there is a (δ_i^*, i)-admissible point $t_i \in \partial D_{\delta_i^*}(c_i)$.*

Note that δ_i^* is not necessarily a radius-sample.
It follows from Lemma 3.10 that $\delta_i^* \leq \delta^{(3)}$ for all $0 \leq i < n$. In order to compute δ^* from the values $\delta_0^*, \ldots, \delta_{n-1}^*$, we need the following observation:

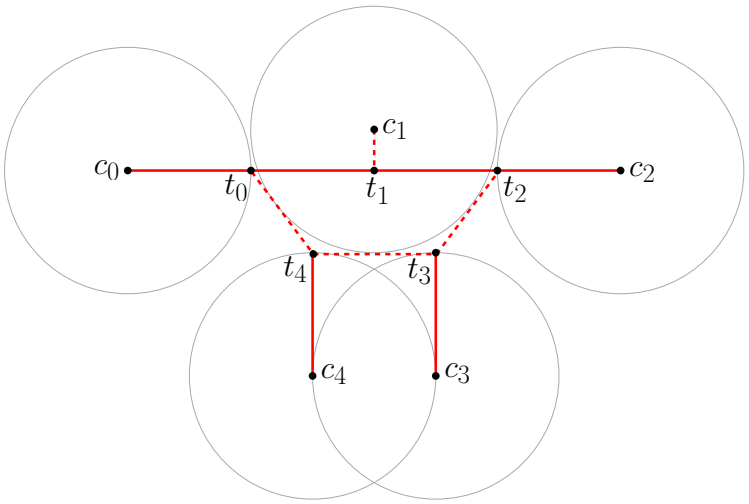

Figure 3.8: The same sequence C as in Figure 3.7 and the sequence T, where t_1 cannot lie on $\partial D_{\delta^*}(c_1)$, since all points on $\partial D_{\delta^*}(c_1)$ are more than δ^* apart from t_0 or t_2. The tight constraints (solid fat lines) are δ^* long.

Lemma 3.12 *The following equation holds:*

$$\delta^* = \min_{0 \le i < n} \delta_i^*.$$

Also, $\delta_i^* \in [\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ *for all* $0 \le i < n$.

Proof. Every δ_i^* meets the inequality $\delta_i^* \ge \delta^*$ and δ^* can not be smaller than the smallest of all δ_i^*: According to Lemma 3.8, there is an index $0 \le j < n$ so that t_j^* is a key-point for every optimal sequence $T^* = (t_0^*, \ldots, t_{n-1}^*)$. In particular, $t_j^* \in \partial D_{\delta^*}(c_j)$, which is a contradiction to the definition of δ_j^*. Hence, $\delta^* = \min_{0 \le i < n} \delta_i^*$.
The second part of Lemma 3.12 follows from Lemma 3.10. □

Consequently, in order to find δ^*, it suffices to compute δ_i^* for all $0 \le i < n$.

Notation 3.13 *For an* $\epsilon > 0$, *let* $T^{(\epsilon)}$ *be a solution so that*

$$\delta^{(\epsilon)} := \gamma(T^{(\epsilon)}, C) \le (1+\epsilon)\delta^*.$$

Let $t_i^{(\epsilon)}$ *denote a* $(1+\epsilon)$*-approximation to a* (δ_i^*, i)*-admissible point* t_i *with* $t_i \in \partial D_{\delta_i^*}(c_i)$. *Let* $\delta_i^{(\epsilon)}$ *be the radius-sample of* $t_i^{(\epsilon)}$.

We already know that $\delta_i^* \in [\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ for all i. In order to prove that a binary search for δ_i^* on $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ works for every index i, we need one more characteristic of (δ, i)-admissible points.

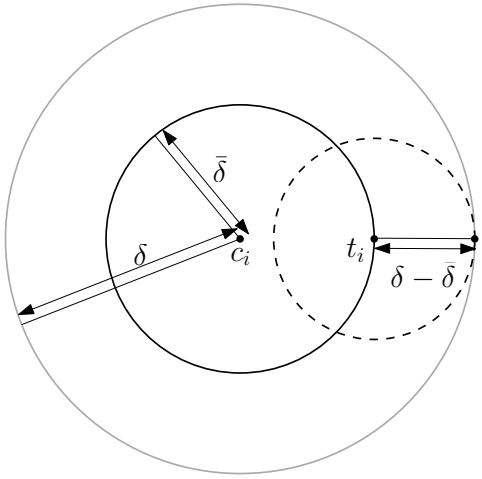

Figure 3.9: The sets $\partial D_\delta(c_i)$ and $\partial D_{\delta-\bar{\delta}}(t_i)$ share exactly one point.

Lemma 3.14 *Let $\bar{\delta} \geq 0$ be so that there is a $(\bar{\delta}, i)$-admissible point $t_i \in \partial D_{\bar{\delta}}(c_i)$. Then there is at least one (δ, i)-admissible point on $\partial D_\delta(c_i)$ for all $\delta \geq \bar{\delta}$.*

Proof. Let $\delta > \bar{\delta}$. Then the point t_i is (δ, i)-admissible, but does not lie on $\partial D_\delta(c_i)$. Also, there is a disk $D_{\delta-\bar{\delta}}(t_i)$ so that every point $t \in D_{\delta-\bar{\delta}}(t_i)$ is still (δ, i)-admissible, and by construction $\partial D_\delta(c_i)$ and $\partial D_{\delta-\bar{\delta}}(t_i)$ share exactly one point, see Figure 3.9. □

Consequently, for every index $0 \leq i < n$ a binary search for δ_i^* on $[\frac{1}{3}\delta^{(3)}, \delta^{(3)}]$ can be carried out and it remains to determine a suitable sample-distance ϵ_{obj}: Since we aim for a $(1+\epsilon)$-approximation, we have to guarantee that $\delta^{(\epsilon)} \leq (1+\epsilon)\delta^*$. Hence, it suffices to find some $\delta_i^{(\epsilon)} \in [\delta_i^*, \delta_i^* + \epsilon\delta_i^*]$ for each $0 \leq i < n$ and we have to choose ϵ_{obj} (the density of the radius-samples) subject to ϵ and $\delta^{(3)}$. The analysis on how ϵ_{obj} has to be chosen exactly will be carried out in Lemma 3.19, since it also depends on our final improvement, in particular on the polygons that will be used to approximate the boundaries of all disks. In order to describe the final improvement in more detail, we need to briefly explain Algorithm B, the 'propagation along the path' decision algorithm of [16] that, given a radius-sample δ, a translation-sample t_i and an index i, decides, whether t_i is (δ, i)-admissible.

An Algorithm for Paths

In Section 3.2, we mentioned that due to the convexity of Problem 5, the run-time of the simple PTAS introduced in Section 3.2 can be improved by approximating the boundary of each disk by the boundary of a regular convex

polygon and performing a binary search on the edges of that polygon. Then, for every sample-point t and sample-radius δ it is tested if t is (δ, i)-admissible. For a given sequence C and index i there already exists an algorithm to test if a given point $t_i \in D_\delta(c_i)$ is (δ, i)-admissible: The point t_i is (δ, i)-admissible iff there are points $t_{i+1}, \ldots, t_{n-1}$, $t_0, \ldots, t_{i-1}$ so that $\gamma(T, C) \leq \delta$ for $T = (t_0, \ldots, t_{n-1})$. Deciding this is the same as solving the following problem:

Problem 6 *Given:*

$$\begin{aligned} C' = (c'_1, \ldots, c'_{n-1}) \quad & \textit{a sequence of points} \\ t \quad & \textit{a point, and} \\ \delta \quad & \textit{a parameter.} \end{aligned}$$

Find: A sequence $T' = (t'_1, \ldots, t'_{n-1})$ *with*

$$\begin{aligned} \|c'_j - t'_j\| \leq \delta \textit{ for every } 1 \leq j < n, \\ \|t'_j - t'_{j+1}\| \leq \delta \textit{ for every } 1 \leq j < n-1, \\ \|t - t'_1\| \leq \delta \textit{ and} \\ \|t'_{n-1} - t\| \leq \delta. \end{aligned} \tag{3.2}$$

In the following, $c'_j \coloneqq c_{i+j}$ and $t'_j = t_{i+j}$ for all $1 \leq j < n$, and $t = t_i$.
In [16] the authors introduced Algorithm B that solves Problem 6 in $O(n^2 \log n)$ time and space under translations along a fixed direction. In Chapter 2, we already used a variant of this algorithm for translations in $\mathbb{R}^2$. In this chapter, we again use the same strategy, although in a different setting and we need some insights about how this algorithm works and what the geometric properties of Problem 6 are in order to motivate and prove the correctness of the approach to approximate the boundaries of all disks mentioned above. Hence, we will now explain this algorithm in short with our notation and refer to Chapter 2 for more details along with a proof of correctness.
We already defined $c'_1, \ldots, c'_{n-1}$ above. Additionally, let

$$\begin{aligned} c'_0 \quad & \coloneqq t, \\ c'_n \quad & \coloneqq t \text{ and} \\ I_j(t) \quad & \coloneqq D_\delta(c'_j), \end{aligned}$$

the set of points t'_j so that $\|t'_j - c'_j\| \leq \delta$ for $1 \leq j < n$. Let

$$\begin{aligned} I_0(t) \quad & \coloneqq \{t\} \text{ and} \\ I_n(t) \quad & \coloneqq \{t\}. \end{aligned}$$

We call the set $I_j(t)$, or I_j in short, the *jth admissible region* and every $t' \in I_j$ an *admissible point*. Note that every admissible region is convex. We define the sequences

$$\begin{aligned} S &:= (t = c'_0, c'_1, \ldots, c'_{n-1}, t = c'_n) \text{ and} \\ S_j &:= (c'_0, \ldots, c'_j) \end{aligned}$$

for $0 \leq j \leq n$.
The algorithm that decides whether there is a sequence T that satisfies (3.2) has an iterative structure. The basic idea is to propagate admissible points starting with $I_0 = \{t\}$ along S until c'_n is reached by appropriately merging the admissible regions of the successive points of S.
Starting with I_1, the algorithm updates I_j to I'_j by replacing I_j with an adjusted set I'_j of admissible points until in the last step I_n is updated to I'_n. The jth and $(j-1)$th admissible region are merged into I'_j in such a way that I'_j is not empty iff there is a sequence of points that is δ-admissible for S_j.
Also, if $I'_j \neq \varnothing$, a δ-admissible sequence for S_j can be computed from $I'_0, \ldots, I'_j$.
More concrete:

Algorithm 4 *We are given the point sequence $C = (c_0, \ldots, c_{n-1})$, a translation t, a parameter $\delta \geq 0$ and an index $0 \leq i < n$.*
First, we set $c'_0 = t$, $c'_n = t$, $I_0(t) = \{t\}$, $I_n(t) = \{t\}$, $c'_j = c_{i+j}$ and $I_j = D_\delta(c'_j)$ for $1 \leq j < n$.
Starting with I'_1, we proceed as follows to compute the sets I'_j for all $1 \leq j \leq n$ in ascending order: We inflate *the region I'_{j-1} by δ which results in a set*

$$I'^{\delta}_{j-1} := I'_{j-1} \oplus D_\delta,$$

where $\oplus$ denotes the Minkowski sum and D_δ is the disk with radius δ centered in the origin. The admissible region I'_j is given by

$$I'_j = I'^{\delta}_{j-1} \cap I_j.$$

This process is repeated until one of the following cases occurs:

1. *There is an index j with $I'_j = \varnothing$ after an intersection operation:*
   The process stops and NO *is returned along with the tuple $(k(t), \mu(t))$, where $k(t)$ is the index of the first vertex of S that was not reached, and $\mu(t)$ is the L_2-distance between the inflated version of the last non-empty admissible region and its succeeding admissible region.*
2. *I_n is updated and $I'_n \neq \varnothing$:*
   The algorithm terminates and returns YES.

Approximating the Boundary of a Disk with a Polygon

The simplest approach that tests if there is a $(1+\epsilon)$-approximation to a key-point on $\partial D_\delta(c_i)$ is to pick $k = \Theta\left(\epsilon^{-1}\delta^{(3)}\right)$ suitably distributed translation-samples on the circle $\partial D_\delta(c_i)$ and propagate all of them with Algorithm 4. In that way, k propagations, i.e., calls of Algorithm 4, have to be carried out. This number can be reduced to $O(k^{1/2}\log k)$ by exploiting the convex structure of the admissible regions that occur during the propagation process: The main idea is to approximate, i.e., inscribe, $\partial D_\delta(c_i)$ by a regular polygon with $O(k^{1/2})$ vertices and to perform a binary search on each of its edges with a sample-distance that depends on ϵ and $\delta^{(3)}$. Note that we could also use circumscribing polygons. However, using inscribing polygons turns out to be more convenient during the analysis of Algorithm 3 we carry out later in the current section.
At first, we will discuss how many vertices the polygon needs to have and what the minimum sample-distance for sample-points on each of the edges is in order to guarantee that for every point t on $\partial D_\delta(c_i)$ there is a sample point on the edges of the polygon that serves as a $(1+\epsilon)$-approximation to t, i.e., is at most $3^{-1}\delta^{(3)}\epsilon$-far from t.

Notation 3.15 *Let $P_{\delta,p}(c_i)$, or $P_\delta(c_i)$ in short, denote the inscribing regular polygon of $\partial D_\delta(c_i)$ with p vertices, where one of the vertices is the point $(c_i - (0,\delta))$ (a point on $\partial D_\delta(c_i)$). By a slight abuse of notation, we identify $P_\delta(c_i)$ with its boundary, since we solely operate on the boundary of the polygons at hand.*

Note that since all such polygons contain the lowermost point of the circle they inscribe, they are all concentric.

Lemma 3.16 *Let*

$$p \coloneqq \left\lceil 3^{1/2}\pi\epsilon^{-1/2}\right\rceil$$

and let the edges of $P_\delta(c_i)$ be sampled with sample-distance ϵ_{edge} so that $\epsilon_{\text{edge}} \leq \frac{1}{3}\delta^{(3)}\epsilon$. Then, there is a translation-sample $t \in P_\delta(c_i)$ for every point $u \in \partial D_\delta(c_i)$ so that

$$\|t - u\| \leq \frac{1}{3}\delta^{(3)}\epsilon.$$

Proof. An illustration of the following is given in Figure 3.10.
Let v and v' be two successive vertices of $P_\delta(c_i)$ and let e be the edge with endpoints v and v'. The points v, v' and c_i define an isosceles triangle with base e and apex angle

$$\alpha \coloneqq 2\pi p^{-1}.$$

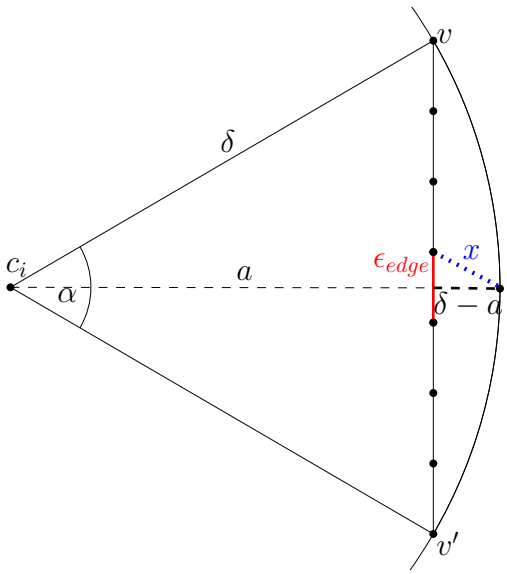

Figure 3.10: The disk $D_\delta(c_i)$ with an edge of the inscribed polygon $P_\delta(c_i)$ with endpoints v and v'. The line segments of length x (dotted line) and ϵ_{edge} (solid fat line) indicate an isosceles triangle, where the line segment of length a (dashed fat line) is the bisector.

The length of e can be estimated by using the Taylor series expansion of $\sin(\alpha/2)$ as

$$|e| = 2\delta \sin\left(\frac{\alpha}{2}\right) = 2\delta \sin\left(\frac{\pi}{p}\right) \leq \frac{2\delta\pi}{p}.$$

The maximum distance between a point on $\partial D_\delta(c_i)$ and $P_\delta(c_i)$ is $\delta - |a|$, where a is the apothem of $P_\delta(c_i)$:

$$\delta - |a| = \delta - \delta\cos\left(\frac{\alpha}{2}\right) \leq \delta\left(1 - \left(1 - \frac{\pi^2}{2p^2}\right)\right) = \delta\left(\frac{\pi^2}{2p^2}\right),$$

because (using the Taylor series expansion) $\cos\left(\frac{\alpha}{2}\right)$ can be estimated as $\cos(\alpha/2) \leq (1 - \pi^2(2p^2)^{-1})$. Let the maximum distance between a point on $\partial D_\delta(c_i)$ and the closest translation-sample on $P_\delta(c_i)$ be named x. Since the maximum distance of a point on $\partial D_\delta(c_i)$ and $P_\delta(c_i)$ is $\delta - |a|$ and the samples that describe the edges of $P_\delta(c_i)$ have distance ϵ_{edge}, the application of Thales' theorem leads to

$$x^2 = (\delta - |a|)^2 + \left(\frac{\epsilon_{\text{edge}}}{2}\right)^2.$$

Since $\epsilon_{\text{edge}} \leq \frac{1}{3}\delta^{(3)}\epsilon$, it follows that

$$\begin{aligned} x^2 &\leq \left(\frac{\delta\pi^2}{2p^2}\right)^2 + \left(\frac{\epsilon_{\text{edge}}}{2}\right)^2 \\ &= \left(\frac{\delta\pi^2}{2(\lceil 3^{1/2}\pi\epsilon^{-1/2}\rceil)^2}\right)^2 + \left(\frac{\epsilon_{\text{edge}}}{2}\right)^2 \\ &\leq \left(\frac{\delta\pi^2}{2(3^{1/2}\pi\epsilon^{-1/2})^2}\right)^2 + \left(\frac{\epsilon_{\text{edge}}}{2}\right)^2 \\ &\leq \left(\frac{\epsilon\delta}{6}\right)^2 + \frac{\epsilon_{\text{edge}}{}^2}{4} \\ &\leq \frac{(\epsilon\delta^{(3)})^2}{36} + \frac{(\epsilon\delta^{(3)})^2}{36} \leq \left(\frac{1}{3}\epsilon\delta^{(3)}\right)^2 \\ \Leftrightarrow x &= \epsilon\frac{1}{3}\delta^{(3)}. \end{aligned}$$

□

Striktly speaking, p depends on ϵ, but we refrain from including ϵ in the notation.

In the remainder, we will show that the binary search among the samples on one edge of $P_\delta(c_i)$ can be carried out in $O(\log \epsilon^{-1} n^2 \log n)$ time. Here $O(n^2 \log n)$ is the time that is needed to carry out the propagation process for a single translation-sample t by Algorithm 4. Since this approach builds on several properties of the tuple $(k(t), \mu(t))$ returned by Algorithm 4, we have to introduce some of them first: The following lemma describes the dependency of the tuple on the translation-samples of one edge of $P_\delta(c_i)$.

Lemma 3.17 *Let s and s' be two* NO*-instances of Algorithm 4,* A4 *in short, for a given radius-sample δ, point sequence C and index i, i.e.,*

$$\begin{aligned} \text{A4}(s, \delta, i) &= (\text{NO}, (k(s), \mu(s))) \text{ and} \\ \text{A4}(s', \delta, i) &= (\text{NO}, (k(s'), \mu(s'))), \end{aligned}$$

and let $t \in \overline{ss'}$. Then, one of the following holds:

$$\begin{aligned} \text{A4}(t, \delta, i) &= \text{YES}, \\ \text{A4}(t, \delta, i) &= (\text{NO}, (k(t), \mu(t))). \end{aligned}$$

In the latter case the tuple $(k(t), \mu(t))$ has the following properties:

1. $$k(t) \geq \min(k(s), k(s')),$$

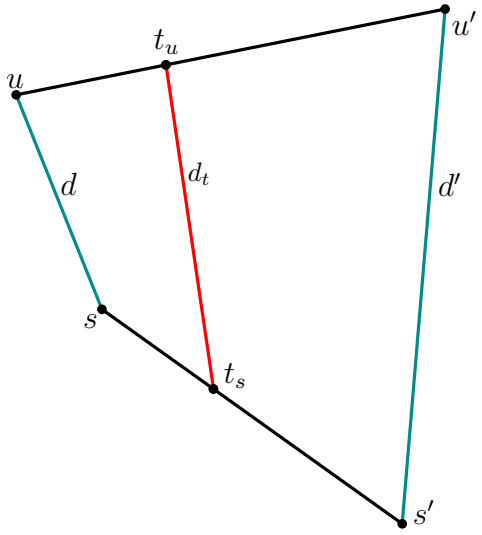

Figure 3.11: Example of the quadrilateral considered in Proposition 3.18.

2. *if* $k(t) = k(s) = k(s')$, *then*

$$\mu(t) \leq \max(\mu(s), \mu(s')).$$

Moreover, if $k(p) = k(s) = k(s')$ *for all points* $p \in \overline{ss'}$, *the function*

$$\begin{array}{ll} f & \to [0,1] \text{ with} \\ x & \mapsto \mu((1-x)s + xs') \end{array}$$

is strictly convex.

Note that the points s and s' in Lemma 3.17 are not necessarily translation-samples on an edge of a polygon at hand. However, we apply Lemma 3.17 to translation samples only in the following.
In order to prove Lemma 3.17, we need one more observation:

Proposition 3.18 *Given two line segments* $\overline{ss'}$ *and* $\overline{uu'}$ *in the plane. If* $\|s - u\| \leq d$ *and* $\|s' - u'\| \leq d'$ *for* $d, d' \in \mathbb{R}^+$, *then for every point* $t_s \in \overline{ss'}$ *there is a point* $t_u \in \overline{uu'}$ *so that*

$$d_t := \|t_s - t_u\| \leq \max(d, d')$$

and vice versa.

Proof. Every point $t_s \in \overline{ss'}$ can be expressed as $t_s = (1-x)s + xs'$ for $x \in [0,1]$,

see Figure 3.11. With $t_u \coloneqq (1-x)u + xu'$, the following holds:

$$\begin{aligned} d_t = \|t_s - t_u\| &= \|(1-x)s + xs' - ((1-x)u + xu')\| \\ &= \|(1-x)(s-u) + x(s'-u')\| \\ &\leq (1-x)\|s-u\| + x\|s'-u'\| \\ &= (1-x)d + xd' \leq \max(d, d'). \end{aligned}$$

□

With this, we can now prove Lemma 3.17:

Proof of Lemma 3.17. Let $C' = (c'_1, \ldots, c'_{n-1})$ with $c'_j \coloneqq c_{i+j}$ for all $1 \leq j < n$. Part 1: Let w.l.o.g. $k(s) \leq k(s')$. As mentioned in the description of Algorithm 4, $k(s)$ is the index of the first vertex that was not reached while propagating s. Obviously, all vertices with smaller indices were reached. Suppose $s_{k(s)-1}$ is the point on $D_\delta(c'_{k(s)-1})$ that establishes $\mu(s)$, i.e., the point in $I_{k(s)-1}(s)$ that is closest to $D_\delta(c'_{k(s)})$. Then there is a sequence $(s = s_0, s_1, \ldots, s_{k-1})$ with $s_i \in D_\delta(c'_i)$ for $1 \leq i \leq k(s)-1$ and $\|s_i - s_{i+1}\| \leq \delta$ for all $0 \leq i \leq k(s)-2$. The point $s'_{k(s')-1}$ and the sequence $(s'_0, \ldots, s'_{k(s')-1})$ are defined in a similar way. We introduce the following notation for certain line segments:

$$\begin{aligned} S_i &\coloneqq \overline{s_i s'_i} && \text{for all } 0 \leq i \leq k(s)-1, \\ R_i &\coloneqq \overline{s_i s_{i+1}} && \text{for all } 0 \leq i \leq k(s)-2 \text{ and} \\ R'_i &\coloneqq \overline{s'_i s'_{i+1}} && \text{for all } 0 \leq i \leq k(s)-2. \end{aligned}$$

For all $0 \leq i \leq k(s)-2$, the four line segments S_i, S_{i+1}, R_i and R'_i form a quadrilateral, where the two opposing sides R_i and R'_i have a length of at most δ. As illustrated in Figure 3.12, the quadrilaterals are connected and form a kind of sequence where the line segments S_i are the connection between two successive quadrilaterals. According to Proposition 3.18, there is a point on S_{i+1} for every point on S_i so that the distance of both points is at most δ and vice versa. Given point $t \in S_0$, it follows that there is a sequence $(t = t'_0, t'_1, \ldots, t'_{k(s)-1})$ with $t'_i \in D_\delta(c'_i)$ for $1 \leq i \leq k(s)-1$ and $\|t'_i - t'_{i+1}\| \leq \delta$ for $0 \leq i \leq k(s)-2$. Hence

$$k(t) \geq \min(k(s), k(s')).$$

Part 2: Suppose $k(t) = k(s) = k(s')$ and w.l.o.g. $\mu(s) \geq \mu(s')$; also, let $(t = t'_0, t'_2, \ldots, t'_{k(s)-1})$ be the sequence defined above. Note that $t'_{k(s)-1}$ lies on $S_{k(s)-1}$ and is not necessarily the point defining $\mu(t)$. We add one quadrilateral

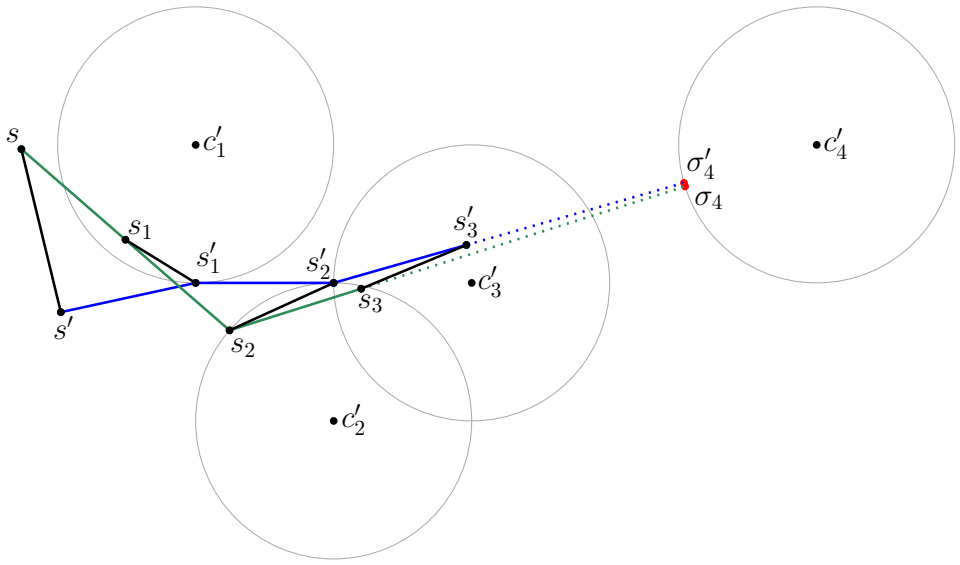

Figure 3.12: The sequences $(s_0, \ldots, s_3, \sigma_4)$ and $(s'_0, \ldots, s'_3, \sigma'_4)$ form a sequence of quadrilaterals.

to the sequence: Let $\sigma_{k(s)}$ be the point on $D_\delta(c'_{k(s)})$ with distance $\mu(s) + \delta$ to $s_{k(s)-1}$ and let $\sigma'_{k(s)}$ be defined in a similar way. Then:

$$\begin{aligned} S_{k(s)} &:= \overline{\sigma_{k(s)}\sigma'_{k(s)}}, \\ R_{k(s)-1} &:= \overline{s_{k(s)-1}\sigma_{k(s)}} \text{ and} \\ R'_{k(s)-1} &:= \overline{s'_{k(s)-1}\sigma'_{k(s)}}, \end{aligned}$$

Again, $S_{k(s)-1}, S_{k(s)}, R_{k(s)-1}$ and $R'_{k(s)-1}$ form a quadrilateral, but this time, the two opposing sides have lengths $\mu(s)+\delta$ and $\mu(s')+\delta$. Due to Proposition 3.18, there is a point $\tau'_{k(s)}$ on $S_{k(s)}$ with

$$\|t'_{k(s)-1} - \tau'_{k(s)}\| \leq \max\left(\mu(s) + \delta, \mu(s') + \delta\right).$$

Now let $k(t) = k(s) = k(s')$ for all points $t \in \overline{ss'}$ and let t be expressed as $t = (1-x)s + xs'$. The sequence $(t = t'_0, \ldots, t'_{k(s)-1}, t'_{k(s)} = \tau'_{k(s)})$ with $t'_i = ((1-x)s_i + xs'_i)$ satisfies $t'_i \in D_\delta(c'_i)$, since every point t'_i is located on S_i and due to Proposition 3.18 two successive points have a distance of at most δ exept for $t'_{k(s)-1}$ and $\tau'_{k(s)}$. As a consequence, we have:

$$\begin{aligned} \mu(t) &\leq \|t'_{k(s)-1} - \tau'_{k(s)}\| - \delta \\ &\leq (1-x)(\delta + \mu(s)) + x(\delta + \mu(s')) - \delta \\ &= (1-x)\mu(s) + x\mu(s'). \end{aligned}$$

Since this holds for every choice of s and s' as well as every choice of $t \in \overline{ss'}$, function f is convex.
Strictly speaking

$$\mu(t) < (1-x)\mu(s) + x\mu(s'),$$

i.e., f is strictly convex, because $S_{k(s)}$ is the chord of a disk and therefore the distance between $t'_{k(s)-1}$ and $\tau'_{k(s)}$ is greater than the distance between $t'_{k(s)-1}$ and the disk that contains $\tau'_{k(s)}$. □

Note that Lemma 3.17 holds for any line segment contained in $D_\delta(c_i)$.
As a consequence of the convexity of the function f, in order to test if there is a (δ, i)-admissible point on the line segment $\overline{ss'}$ (which means that there is a point on $t \in \overline{ss'}$ so that the propagation of t with radius-sample δ is successful) a binary search can be carried out among the samples along the line segment $\overline{ss'}$.
The runtime depends on the number of sample-points that have to be propagated, which is $O\left(\log {\epsilon_{\text{edge}}}^{-1}\right)$ for sample-distance ϵ_{edge}. Since every propagation takes $O(n^2 \log n)$ time, the procedure runs in $O\left(\log {\epsilon_{\text{edge}}}^{-1} n^2 \log n\right)$.
We already know from Lemma 3.16 that the length of an edge of $P_\delta(c_i)$ is at most $2\delta\pi p^{-1}$. Up to here, we gave an upper bound for ϵ_{edge}, i.e., $\epsilon_{\text{edge}} \leq 3^{-1}\delta^{(3)}\epsilon$. Now we also establish a lower bound by chosing ϵ_{edge} from the interval $[12^{-1}3^{1/2}\delta^{(3)}\epsilon, 3^{-1}\delta^{(3)}\epsilon]$. With that and and $p = \left\lceil 3^{1/2}\pi\epsilon^{-1/2}\right\rceil$, the number of translation-samples that have to be propagated, is

$$\begin{aligned}
\log\left(\frac{2\delta\pi}{\epsilon_{\text{edge}}p} + 1\right) &\leq \log\left(\frac{12 \cdot 2\delta\pi}{3^{1/2}\delta^{(3)}\epsilon p} + 1\right) \\
&\leq \log\left(\frac{24\pi}{3^{1/2}\epsilon p} + 1\right) \\
&= \log\left(\frac{24\pi}{3^{1/2}\epsilon\left\lceil 3^{1/2}\pi\epsilon^{-1/2}\right\rceil} + 1\right) \\
&\leq \log\left(\frac{24\pi}{3^{1/2}\epsilon(3^{1/2}\pi\epsilon^{-1/2})} + 1\right) \\
&= \log\left(\frac{24\epsilon^{1/2}}{3\epsilon} + 1\right) \\
&= \log\left(\frac{8}{\epsilon^{1/2}} + 1\right) \in O\left(\log\frac{1}{\epsilon^{1/2}}\right),
\end{aligned}$$

which leads to a runtime of $O\left(\epsilon^{-1/2}\log\epsilon^{-1}n^2\log n\right)$ in total for propagating all sample-points of one polygon and a fixed radius-sample.
For technical reasons, we also need the following insight:

Lemma 3.19 *Let $D_{\delta^*}(c_i)$ be a disk so that $\partial D_{\delta^*}(c_i)$ contains the key-point t_i^*. Also, let*

$$\bar{\delta} := \delta^* + \epsilon_{\text{obj}}$$

and let the sample-distance of the points on the edges of $P_{\bar{\delta}}(c_i)$ be

$$\begin{aligned} \epsilon_{\text{edge}} &:= \frac{\sqrt{3}}{12}\epsilon\delta^{(3)} \text{ and} \\ \epsilon_{\text{obj}} &:= \frac{\sqrt{3}}{12}\epsilon\delta^{(3)}. \end{aligned}$$

The following holds:

1. *The area of $(\bar{\delta}, i)$-admissible points on $\partial D_{\bar{\delta}}(c_i)$ is a circular arc with endpoints s and s' and*
$$\|s - s'\| \geq \epsilon_{\text{obj}}.$$
2. *There is a translation-sample on $P_{\bar{\delta}}(c_i)$ that is a $(1+\epsilon)$-approximation to t_i^*.*

Proof. Since $D_{\delta^*}(c_i)$ is a disk so that $\partial D_{\delta^*}(c_i)$ contains the key-point t_i^*, this point is also the only (δ^*, i)-admissible point in $D_{\delta^*}(c_i)$. Also, t_i^* lies inside the triangle defined by c_i, t_{i-1} and t_{i+1}, see Proposition 3.9. This implies that either c_i and t_{i-1} or c_i and t_{i+1} lie on opposing sides of the tangent of $D_{\delta^*}(c_i)$ at t_i^*. If we increase δ^* by ϵ_{obj}, the intersection of $D_{\bar{\delta}}(t_{i-1})$ and $D_{\bar{\delta}}(t_{i+1})$ changes from just one point to a convex set in the shape of a lens. The circular arc of $(\bar{\delta}, i)$-admissible points on $\partial D_{\bar{\delta}}(c_i)$ is the intersection of this lens with $\partial D_{\bar{\delta}}(c_i)$ and a short geometric inspection shows that this circular arc with endpoints s and s' is shortest if the distance between t_{i-1} and t_{i+1} is $2\delta^*$ and both points lie on the tangent of $D_{\bar{\delta}}(c_i)$ at t_i^*:
Let

$$\begin{aligned} d &:= \overline{ss'}, \\ h &:= \overline{\frac{1}{2}(c_i - t_{i+1}), t_i^*}, \\ z &:= \overline{t_i^* s'} \text{ and} \\ k &:= \overline{t_i^*, \frac{1}{2}(s - s')}, \end{aligned}$$

see Figure 3.13. The application of Thales' theorem provides the following

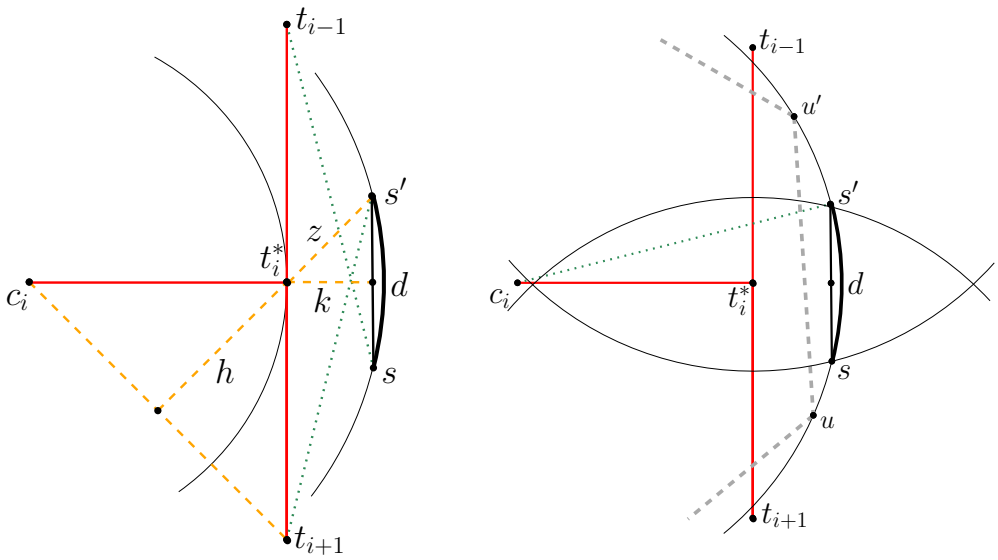

Figure 3.13: Left: The circular arc of $(\bar{\delta}, i)$-admissible points on $\partial D_{\bar{\delta}}(c_i)$ (fat) with line segments of length δ^* (solid fat lines) and $\bar{\delta}$ (dotted fat lines). Right: $P_{\bar{\delta}}(c_i)$ (dashed fat lines), $D_{\bar{\delta}}(c_i)$ with edge $\overline{uu'}$ that intersects k.

equations:

$$|h| = \frac{1}{\sqrt{2}}\delta^*, \tag{3.3}$$

$$|z| = \sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - |h|^2} - |h|, \tag{3.4}$$

$$|k| = \sqrt{|z|^2 - \frac{1}{4}|d|^2}, \text{ and} \tag{3.5}$$

$$|k| = \sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{4}|d|^2} - \delta^*. \tag{3.6}$$

Squaring the equation (3.5)=(3.6) results in

$$|z|^2 - \frac{1}{4}|d|^2 = (\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{4}|d|^2 + (\delta^*)^2 - 2\delta^*\sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{4}|d|^2}.$$

Now Equation (3.4) can be applied and both sides of the equation can be

simplified:

$$\begin{aligned}
&\Leftrightarrow \quad \left(\sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - |h|^2} - |h|^2\right)^2 \\
&\qquad = (\delta^* + \epsilon_{\text{obj}})^2 + (\delta^*)^2 - 2\delta^* \sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{4}|d|^2} \\
&\overset{(3.3)}{\Leftrightarrow} \quad \sqrt{2}\sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{2}(\delta^*)^2} + \delta^* \\
&\qquad = 2\sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{4}|d|^2}.
\end{aligned}$$

Again, we square both sides of the equation and simplify the resulting terms:

$$\Leftrightarrow |d|^2 = 2(\delta^* + \epsilon_{\text{obj}})^2 - 2\sqrt{2}\delta^* \sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{2}(\delta^*)^2}.$$

Suppose $|d|^2 < {\epsilon_{\text{obj}}}^2$. Then

$$\begin{aligned}
&2(\delta^* + \epsilon_{\text{obj}})^2 - {\epsilon_{\text{obj}}}^2 < 2\sqrt{2}\delta^* \sqrt{(\delta^* + \epsilon_{\text{obj}})^2 - \frac{1}{2}(\delta^*)^2} \\
\Leftrightarrow \quad &4(\delta^* + \epsilon_{\text{obj}})^4 - 4{\epsilon_{\text{obj}}}^2(\delta^* + \epsilon_{\text{obj}})^2 + {\epsilon_{\text{obj}}}^4 < 8(\delta^*)^2(\delta^* + \epsilon_{\text{obj}})^2 - 4(\delta^*)^4 \\
\Leftrightarrow \quad &12(\delta^*)^2{\epsilon_{\text{obj}}}^2 + 8\delta^*{\epsilon_{\text{obj}}}^3 + {\epsilon_{\text{obj}}}^4 < 0,
\end{aligned}$$

which is a contradiction to the fact that $\delta^* \geq 0$ and $\epsilon_{\text{obj}} > 0$. Thus, $|d|^2 \geq {\epsilon_{\text{obj}}}^2$ and with that $d \geq \epsilon_{\text{obj}}$, which proves part 1.
Since the maximum distance between a point on $P_{\bar{\delta}}(c_i)$ and the boundary $\partial D_{\bar{\delta}}(c_i)$ is smaller than ϵ_{obj}, one of the following two cases holds:

1. There is a vertex of $P_{\bar{\delta}}(c_i)$ that is located on the circular arc of $(\bar{\delta}, i)$-admissible points for $t_i^* \in \partial D_{\bar{\delta}}(c_i)$.
2. There is an edge of $P_{\bar{\delta}}(c_i)$ that intersects with k. Let the vertices of this edge be called u and u'.

If case 1 holds, then the vertex of $P_{\bar{\delta}}(c_i)$ on the circular arc gives a $(1+\epsilon)$-approximation to t_i^*. If case 2 holds, then $\overline{uu'}$ and $I_{\bar{\delta}}(c_i, t_{i-1}, t_{i+1})$ intersect. The resulting line segment is shortest if $\overline{uu'}$ is parallel to $\overline{t_{i-1}t_{i+1}}$; also, the line segments $\overline{t_{i-1}t_{i+1}}$ and d are parallel by construction. The intersection of $D_{\bar{\delta}}(t_{i-1})$ and $D_{\bar{\delta}}(t_{i+1})$ (the lens) is an axisymmetric convex object and $\overline{t_{i-1}t_{i+1}}$, which has length $2\epsilon_{\text{obj}}$, lies on its axis of symmetry. Hence,

$$|\overline{t_{i-1}t_{i+1}}| \geq |\overline{ab}| \geq |d|$$

for every line segment $\overline{ab}$ that is parallel to $\overline{t_{i-1}t_{i+1}}$ and $\overline{ab}$ lies inside the vertical stripe within the lens that is bounded by $\overline{t_{i-1}t_{i+1}}$ and d, which is why the length of the line segment of (δ^*, i)-admissible points that lies inside $\overline{uu'}$ is at least ϵ_{obj} long. The sample-distance we use on the edges of $P_{\bar{\delta}}(c_i)$ is

$$\frac{\sqrt{3}}{12}\epsilon\delta^{(3)} < \frac{1}{3}\delta^{(3)}\epsilon,$$

which proves part 2. □

Analysis of Algorithm 3

We first discuss the runtime of Algorithm 3: At the start, the value of a 3-approximation to δ^* is computed in $O(n)$ time. Except for basic arithmetic operations, Algorithm 3 consists of four nested loops: The first loop iterates over all of the n input points of the sequence C. For each of these points a binary search for $\delta \in [3^{-1}\delta^{(3)}, \delta^{(3)}]$ up to accuracy $\epsilon_{\text{obj}} = 12^{-1}\sqrt{3}\delta^{(3)}\epsilon$ is carried out; this takes $O(\log \epsilon^{-1})$ steps. In each step of this binary search all $p = \lceil 3^{1/2}\pi\epsilon^{-1/2} \rceil \in O(\epsilon^{-1/2})$ edges of $P_\delta(c_i)$ are inspected, and on each of them a binary search among $2\delta\pi{\epsilon_{\text{edge}}}^{-1} \in O(\epsilon^{-1})$ translation-samples is performed. Each translation-sample is propagated with Algorithm 4, which takes $O(n^2 \log n)$ time per call. This gives a total runtime of $O(\epsilon^{-1/2}(\log \epsilon^{-1})^2 n^3 \log n)$.

Algorithm 3 is correct: If $\delta^{(\epsilon)} < \delta^{(3)}$ is returned, it permits a YES-instance of Problem 5 since there was a translation-sample that has been propagated successfully and therefore is part of a $\delta^{(\epsilon)}$-admissible sequence T. This also means that the very translation-sample that establishes $\delta^{(\epsilon)}$ is propagated and together with the intermediate steps of the propagation gives a T, which then serves as a witness, i.e., is $\delta^{(\epsilon)}$-admissible. If there was no successful propagation, $\delta^{(\epsilon)} = \delta^{(3)}$ is returned and we know from Lemma 3.5 that there is always a $\delta^{(3)}$-admissible sequence.

Now we analyse the precision of the approximation Algorithm 3 computes: The precision of the binary search on δ is $\epsilon_{\text{obj}} < 3^{-1}\delta^{(3)}\epsilon$.
We know from Lemma 3.8 that for every optimal sequence $T^* = (t_0^*, \ldots, t_{n-1}^*)$, there is an index $0 \leq i < n$ so that t_i^* is a key-point, i.e., $t_i^* \in D_{\delta_i^*}(c_i)$. Recall, that for every index i, δ_i^* denotes the smallest value, so that there is a (δ_i^*, i)-admissible point $t_i \in \partial D_{\delta_i^*}(c_i)$. We know from Lemma 3.12, that

$$\delta^* = \min_{0 \leq i < n} \delta_i^*.$$

Algorithm 3 computes for every $0 \leq i < n$ the smallest radius-sample δ so that there is a translation-sample on the boundary of the inscribed polygon of

$\partial D_d(c_i)$ that is propagated successfully. Hence it is enough in terms of analyzing the precision of the approximation of the algorithm to analyze the precision of the approximation subject to one of the disks that contain a key-point. Hence in the following we consider the fixed index $0 \leq i < n$ with property that $D_{\delta^*}(c_i)$ is a disk so that $\partial D_{\delta^*}(c_i)$ contains the key-point t_i^* and $\delta_i^* = \delta^*$.
Also, all polygons are concentric by construction. If $P_\delta(c_i)$ and $P_{\delta+\epsilon_{\text{obj}}}(c_i)$ are two polygons with circumradii that differ by ϵ_{obj}, the distance between any point on $P_\delta(c_i)$ and $P_{\delta+\epsilon_{\text{obj}}}(c_i)$ is at most ϵ_{obj} and vice versa. Every edge of these two polygons is sampled with points of distance ϵ_{edge}, and with Thales' theorem it follows that for every translation-sample on $P_\delta(c_i)$ there is a translation-sample on $P_{\delta+\epsilon_{\text{obj}}}(c_i)$ with distance $3^{-1}\delta^{(3)}\epsilon$ or less and vice versa. Combined with Lemma 3.16, we have that for every δ there is a translation-sample in $D_\epsilon(t_i)$ for every (δ, i)-admissible point $t_i \in \partial D_\delta(c_i)$. According to Lemma 3.19, one of the following two cases holds:

- For every $\delta \geq \delta^* + 3^{-1}\delta^{(3)}\epsilon$ there is at least one (δ, i)-admissible translation-sample on $P_\delta(c_i)$ so that the line segment of all (δ, i)-admissible points on one of the edges of this polygon is at least $3^{-1}\delta^{(3)}\epsilon$ long.
- One vertex of the polygon is a (δ, i)-admissible point and since all polygons are concentric, this vertex is (δ, i)-admissible for every $\partial D_\delta(c_i)$ with $\delta \geq \delta^*$.

We consider the radius-samples $\bar{\delta}$, $\bar{\delta} + \epsilon_{\text{obj}}$ and $\bar{\delta} + 2\epsilon_{\text{obj}}$, where $\bar{\delta} \coloneqq \delta^* + \zeta - \epsilon_{\text{obj}}$ for some $0 < \zeta < \epsilon_{\text{obj}}$. Since $\bar{\delta} < \delta^*$, none of the propagations for this δ are successful. The following holds:

$$\begin{aligned}
\bar{\delta} + \epsilon_{\text{obj}} &< \delta^* + \epsilon_{\text{obj}} \\
&= \bar{\delta} - \zeta + \epsilon_{\text{obj}} + \epsilon_{\text{obj}} \\
&< \bar{\delta} + 2\epsilon_{\text{obj}} \\
&< \delta^* + \frac{\sqrt{3}}{6}\delta^{(3)}\epsilon \\
&< \delta^* + \frac{1}{3}\delta^{(3)}\epsilon.
\end{aligned}$$

Due to Lemma 3.19, this means that for radius-sample $\bar{\delta}+2\epsilon_{\text{obj}}$ the two endpoints of the circular arc of $(\bar{\delta}+2\epsilon_{\text{obj}}, i)$-admissible points in $D_{\bar{\delta}+2\epsilon_{\text{obj}}}(c_i)$ have a distance of at least ϵ_{obj}, which is why there is at least one translation-sample on the inscribed polygon of the disk at hand that is propagated successfully and the algorithm returns

$$\delta^{(\epsilon)} = \bar{\delta} + 2\epsilon_{\text{obj}} < \delta^* + \frac{1}{3}\delta^{(3)}\epsilon < (1+\epsilon)\delta^*$$

as the approximation to δ^*. Hence the algorithm computes a $(1+\epsilon)$-approximation to δ^* for Problem 5.
It also returns a $(\delta^{(\epsilon)}, i)$-admissible point $t^{(\epsilon)}$ from which a $\delta^{(\epsilon)}$-admissible sequence $T^{(\epsilon)}$ can be computed in $O(n^2 \log n)$ time: Since $t^{(\epsilon)}$ is $(\delta^{(\epsilon)}, i)$-admissible, it has been propagated successfully by Algorithm 4. During a run of Algorithm 4, a sequence of admissible regions was computed. A $\delta^{(\epsilon)}$-admissible sequence $T^{(\epsilon)}$ can easily be computed by starting with $t^{(\epsilon)}$, inflating it with $\delta^{(\epsilon)}$ and intersecting the resulting disk with the given admissible region I_{i+1} and picking a translation from this set. This translation is inflated again and the resulting disk intersected with the next admissible region. This strategy is then repeated until the last admissible region is processed.

3.4 The Strategy for Paths Does not Work for Cycles

In the beginning of this chapter, we mentioned that the strategies of Algorithm 1 and Algorithm 2 introduced in Chapter 2 do not work if the neighborhood graph at hand contains at least one cycle. Suppose, we use a variant of Algorithm 4, which is customized for paths, for the case of G being a simple cycle and a fixed radius-sample δ. In this variant, the first and last translation are not fixed but $I_0 = I_n = D_\delta(c_0)$. Starting with $I_0 = D_\delta(c_0)$, admissible regions are propagated along the cycle until the algorithm returns NO or $I'_0 = (I'^{\delta}_{n-1} \cap D_\delta(c_0)) \neq \emptyset$. If the algorithm returns NO, there is no admissible set of translations for δ.
If $I'_0 \neq \emptyset$, we know that for every point $t'_0 \in I'_0 \subseteq D_\delta(c_0)$ there is a sequence of translations $T = (t_0, \ldots, t_{n-1}, t'_0)$ with

$$\begin{aligned} \|t_i - c_i\| &\leq \delta \text{ for } 0 \leq i \leq n-1, \\ \|t'_0 - c_0\| &\leq \delta, \\ \|t_i - t_{i+1}\| &\leq \delta \text{ for } 0 \leq i \leq n-2 \text{ and} \\ \|t_{n-1} - t'_0\| &\leq \delta. \end{aligned}$$

However, generally $t_0 \neq t'_0$. The last intersection operation reduces $D_\delta(c_0)$ to I'_0, but we do not know if there is a sequence of translations T with $t_0 \in I'_0$ for any $t'_0 \in I'_0$, since the algorithm started with $I_0 = D_\delta(c_0)$ and as a result, it may be that $t_0 \in (D_\delta(c_0) \setminus I'_0)$, see Figure 3.14.
By applying the above mentioned variant of Algorithm 4 on this new setting, i.e., the propagation process starts with I'_0, the invalid choices for t_0 could be filtered out. However, after one more round of propagation, I'_n may be reduced to $I_n'' \subseteq I'_n$. Then, the procedure may have to be repeated again and it is not known if this strategy terminates, see the following illustrating series of figures:

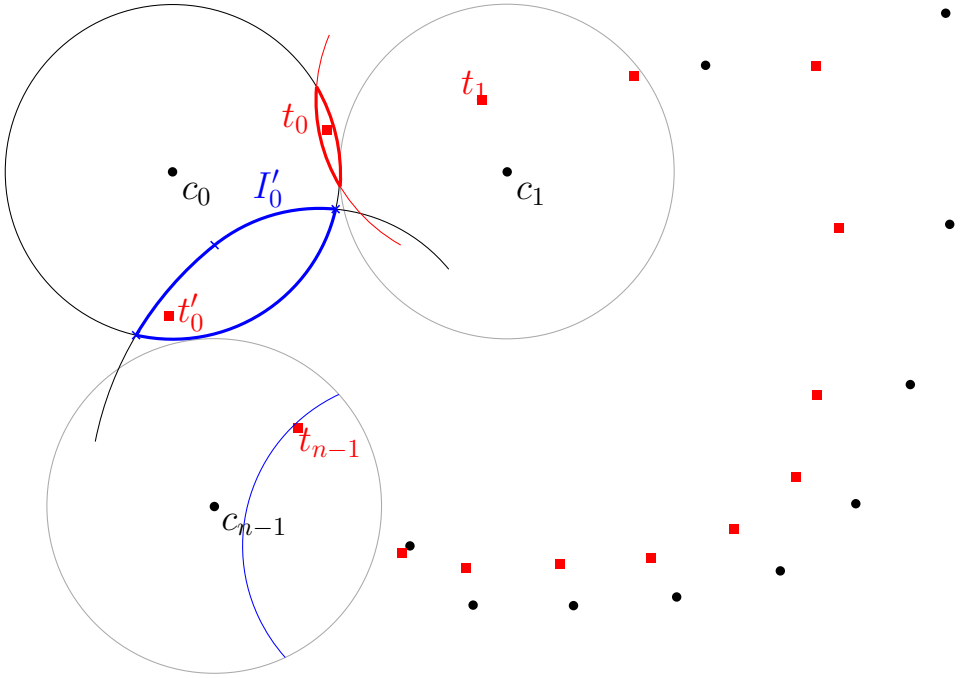

Figure 3.14: Applying the variant of Algorithm 4 on the set C results in the set I'_0 (blue) and for t'_0, all possible choices of t_0 with $\|t_0 - t_1\| \leq \delta$ are in $D_\delta(c_0) \setminus I'_0$ (red).

Figures 3.15 to 3.19 illustrate how the above mentioned variant of Algorithm 4 works on the point sequence $C = (c_0, \ldots, c_4)$ for a given δ. The edges of the neighborhood graph G are indicated as line segments. Admissible regions are depicted step by step as intermediate results of the algorithm.

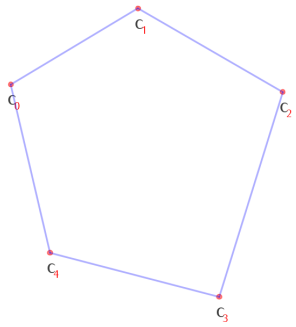

Figure 3.15: The set $C = \{c_0, \ldots, c_4\}$ (red). The edges of the corresponding neighborhood graph are indicated as line segments (purple).

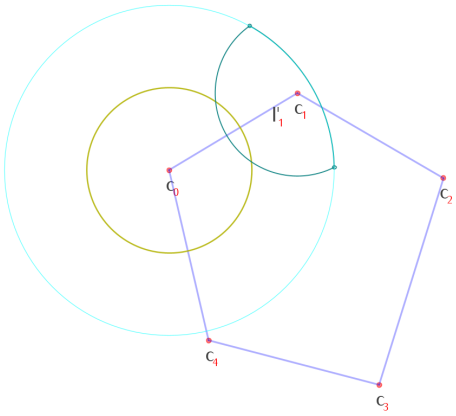

Figure 3.16: The admissible region $I'_1 = D_{2\delta}(c_0) \cap D_\delta(c_1)$ is computed (shades of blue and turquoise).

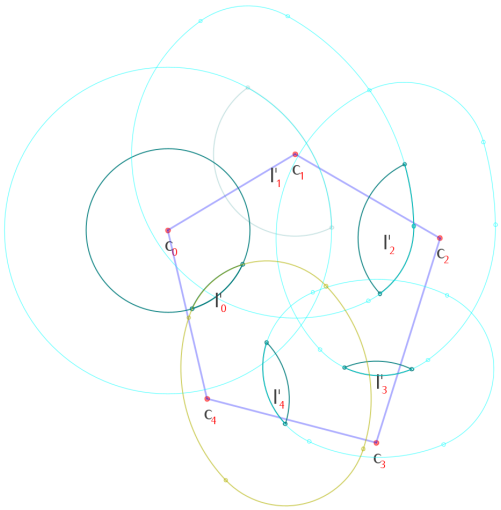

Figure 3.17: After one round of propagation, $I'_0 \subsetneq D_\delta(c_0)$ is computed.

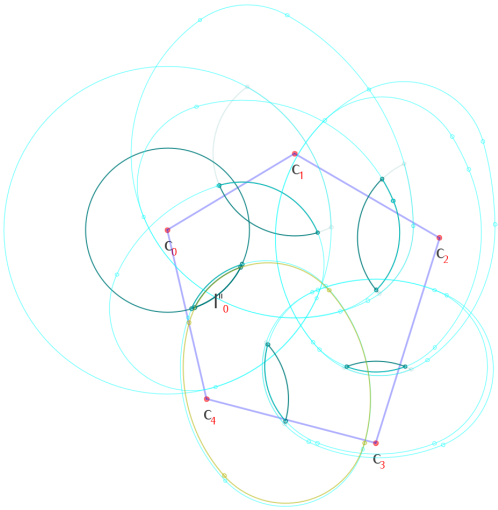

Figure 3.18: After two rounds of propagation, $I''_0 \subsetneq I'_0$ is computed.

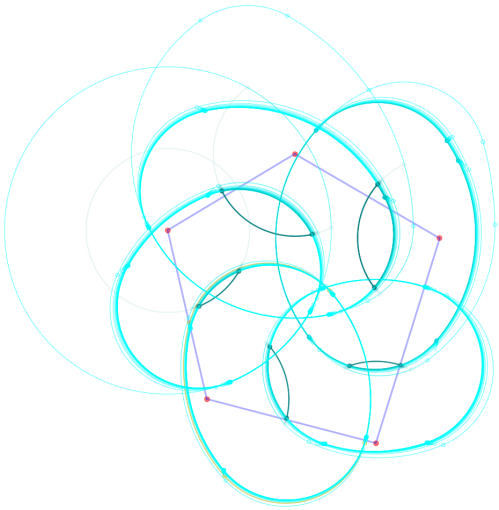

Figure 3.19: The same scene after 10 rounds of propagation (the labels have been omitted in order to provide a better view).

Chapter 4

Elastic Geometric Shape Matching on Neighborhoods that Contain Cycles

In the previous chapter we constructed Algorithm 3, an FPTAS for an EGSM instance where the neighborhood graph is a simple cycle. This strategy can also be used to design algorithms that apply to EGSM instances under neighborhood graphs that contain more than one cycle, even though no efficient exact algorithms are known for this EGSM problem variant. However, the approach and the runtime of this algorithm strongly depends on the number of cycles as well as other structural features of G.

Two ways to classify the structure of a cycle-containing graph $G = (V, E)$ is considering its *feedback vertex set (FVS)*, i.e., a subset X of V so that the subgraph of G induced by $V \setminus X$ is cycle-free, or its *path- or tree decomposition*. A path- or tree decomposition of G is a mapping of G into a path or a tree T, respectively, so that every vertex, also called *bag* of T is associated with a subset of V so that the vertices of G are represented as subtrees in T and the vertices of G are adjacent in T if the corresponding subtrees intersect. The *path- or treewidth* of a path- or tree decomposition is the size of the largest bag minus 1. Both checking for a given graph G if there is a FVS of size k and if there is a path- or tree decomposition of width k is NP-complete.

In the following, we consider EGSM instances under the directed L_2-Hausdorff distance under translations, where either a FVS or a path- or tree-decomposition of the neighborhood graph is given for which we present $(1 + \epsilon)$-approximation algorithms for any $\epsilon > 0$.

4.1 Problem Statement

For the most part, we use the same notation as in Chapter 3:
Recall that for a point $c \in \mathbb{R}^2$ and some $r > 0$, $D_r(c)$ denotes the disk with radius r centered in c. For any closed set $A \in \mathbb{R}^2$, ∂A denotes the boundary of A. Also, $\|\cdot\|$ denotes the Euclidean norm.
We consider the following variant of the EGSM problem:

Problem 7 *Given:*

$$\begin{aligned} P = \{p_1, \ldots, p_n\} \quad & \textit{a point set (the pattern),} \\ Q = \{q_1, \ldots, q_m\} \quad & \textit{a point set (the model), and} \\ G = (V, E) \quad & \textit{an undirected graph with } V = \{i \mid 1 \leq i \leq n\} \textit{ and} \\ & E \subseteq \{\{i, j\} \mid i, j \in V\}. \end{aligned}$$

Find: A sequence of translations $T = (t_1, \ldots, t_n)$, so that the function

$$\gamma(P, Q, T, G) := \max\left(\vec{h}(T(P), Q), \max_{\{i,j\} \in E} \|t_i - t_j\|\right)$$

is minimized.

Note that Problem 7 is similar to Problem 5 discussed in Chapter 3, apart from the fact that in Problem 7 the distance measure in object space is chosen to be the directed L_2-Hausdorff distance, i.e., the correspondence between the points of the pattern and the points of the model is not known.

First, we introduce some notation and definitions. Let

$$\begin{aligned} C_i \quad & := \{q_j - p_i \mid 1 \leq j \leq m\} \text{ and} \\ \mathcal{C} \quad & := (C_1, \ldots, C_n) \end{aligned}$$

for $1 \leq i \leq n$. Problem 7 can be considered entirely in transformation space. The objective function can be written as

$$\gamma(\mathcal{C}, T, G) := \max\left(\max_{1 \leq i \leq n} \vec{h}(t_i, C_i), \max_{\{i,j\} \in E} \|t_i - t_j\|\right).$$

Notation 4.1 *For a given sequence $\mathcal{C} = (C_1, \ldots, C_n)$, we define*

$$\delta^* := \min_T \gamma(\mathcal{C}, T, G).$$

Definition 4.2 *Let $\delta \geq 0$. A sequence of translations $T = (t_1, \ldots, t_n)$ is called* δ-admissible (for sequence $\mathcal{C}$ and neighborhood graph G)*, iff $\gamma(\mathcal{C}, T, G) \leq \delta$. A sequence that is δ^*-admissible is called an* optimal *sequence. We use the symbol T^* to denote an optimal sequence.*
A translation t is called (δ, i)-admissible (for sequence $\mathcal{C}$ and graph G)*, iff there is a δ-admissible sequence $T = (t_1, \ldots, t_i = t, \ldots, t_n)$.*

Recall that δ^* and the concept of δ-admissibility depend on $\mathcal{C}$ and G, but since $\mathcal{C}$ and G are both part of the input and do not vary throughout the subproblems, we refrain from including them in the notation.

Notation 4.3 *Let $C \in \mathcal{C}$ and $\delta \geq 0$. We define*

$$D_\delta(C) \coloneqq \bigcup_{c \in C} D_\delta(c).$$

Note that every (δ, i)-admissible translation is forced to lie in $D_\delta(C_i)$. Note that every C_i for $1 \leq i \leq n$ is a translate of Q. Therefore, every $D_\delta(C_i)$ is the union of m L_2-disks with radius δ. Solving Problem 7 is then equivalent to finding the smallest radius δ, so that there is a sequence of translations $T = (t_1, \ldots, t_n)$ so that $t_i \in D_\delta(C_i)$ for all $1 \leq i \leq n$ and T meets all restrictions induced by G.

In this chapter, we focus on Problem 7 under the directed Hausdorff distance and consider two graph classes that (potentially) contain several cycles. For now, suppose that the correspondence between the points of the pattern and the points of the model is known, i.e., $n = m$ and p_i is matched to q_i for all $1 \leq i \leq n$.
In Section 4.2, we present an algorithm which, for an $\epsilon > 0$, gives a $(1+\epsilon)$-approximation to δ^* for Problem 7 with a given feedback vertex set of size k_f in $O(\epsilon^{-2k_f - 1/2} \log \epsilon^{-1} k_f n)$ time and $O(\epsilon^{-1/2} k_f n + \epsilon^{-2k_f})$ space. In Section 4.3, we give an algorithm that gives a $(1+\epsilon)$-approximation to δ^* for Problem 7 with a given path or tree decomposition of width k_w in $O(\epsilon^{-2(k_w+1)} \log \epsilon^{-1} {k_w}^2 n)$ time and $O(\epsilon^{-2(k_w+1)} n)$ space.
The algorithms described in this chapter still work under unknown correspondence. However, the runtime and required space change. The algorithm introduced in Section 4.2 then takes

$$O\left(\left(\frac{1}{\epsilon^2} n^2 m\right)^{k_f} \left(\log n + \log \frac{1}{\epsilon}\right) T_{A2}\left(\frac{\epsilon}{n}, k_f n, m\right)\right)$$

time and

$$O\left(\left(\frac{1}{\epsilon^2} n^2 m\right) k_f T_{A2}\left(\frac{\epsilon}{n}, k_f n, m\right)\right)$$

space, where

$$T_{A2}(\epsilon_{\mathrm{A2}}, n, m) = \frac{1}{\sqrt{\epsilon_{\mathrm{A2}}}} n^3 m^2 \left(\log n + \log m + \log \frac{1}{\epsilon_{\mathrm{A2}}} \right).$$

The algorithm introduced in Section 4.3 takes

$$O\left(\left(\frac{1}{\epsilon^2} n^2 m \right)^{(k_w+1)} \left(\log \frac{1}{\epsilon} + \log n \right) {k_w}^2 n \right)$$

time and

$$O\left(\left(\frac{1}{\epsilon^2} n^2 m \right)^{(k_w+1)} \right)$$

space for instances under unknown correspondence.

Note that none of the algorithms presented in this chapter are FPT algorithms except for the special case that the feedback vertex sets or path- or tree decompositions at hand have constant size.

4.2 Solving Instances with Given Feedback Vertex Sets

First, we need the following definition:

Definition 4.4 *For a given undirected graph $G = (V, E)$, a* feedback vertex set (FVS) *X of G is a subset of V so that the subgraph of G induced by $V \setminus X$ is cycle-free. A* minimum feedback vertex set (MFVS) *of G is one with minimal cardinality.*

Checking for a given graph G if there is a FVS of size k, is NP-complete. An algorithm that determines in $O(k^k n)$ time, whether there is a FVS of cardinality k for a given graph with n vertices, is given in [24]. The algorithm is also able to return a FVS of size k in case of a YES-instance. In this chapter we presume that a suitable FVS is given.
In the following, we introduce basic tools that are required for designing the main algorithm of this section, i.e., Algorithm 5.

A Basic FPTAS for Cycle-Free Neighborhoods

Suppose G is cycle-free. Algorithm 2, or $A2$ in short, described in Section 2.3 computes a $(1 + \epsilon_{\mathrm{A2}})$-approximation to the decision variant of Problem 7 in

time and space $O(T_{A2}(\epsilon_{\mathrm{A2}}, n, m))$ with

$$T_{A2}(\epsilon_{\mathrm{A2}}, n, m) = \frac{1}{\sqrt{\epsilon_{\mathrm{A2}}}} n^3 m^2 \left(\log n + \log m + \log \frac{1}{\epsilon_{\mathrm{A2}}} \right)$$

for any $\epsilon_{\mathrm{A2}} > 0$. Here, all L_2-disks that describe sets of admissible translations in the Euclidean setting are approximated by regular inscribed polygons with $O({\epsilon_{\mathrm{A2}}}^{-1/2})$ vertices. Given a parameter $\delta > 0$, the sequence $\mathcal{C}$ and the graph G, Algorithm 2 decides if there is a δ-admissible sequence T by computing, inflating and propagating sets of admissible translations along the given graph. Additionally, a witness, i.e., a sequence of translations that solves the problem at hand, can be returned in case of a YES-instance. Also, the proof of correctness of Algorithm 2 still holds, if we alter it slightly by predefining some of the translations, i.e., at start, initialize some of the admissible regions as one single translation instead of a polygon. Then, the algorithm decides if there is a δ-admissible sequence T containing all predefined translations. Note that predefining some of the translations does not degrade the runtime of the algorithm.
In this chapter, referring to Algorithm 2 means referring to the variant of Algorithm 2 mentioned above, where some translations can be predefined.

Cutting Cycles

The underlying idea of Algorithm 5, the $(1+\epsilon)$-approximation algorithm for Problem 7 for a given FVS, is based on the fact that there is a $3(n-1)$-approximation to δ^*, regardless of the structure of G: w.l.o.g., let G be connected (if G is not connected, the following strategy is applied on all connected components of G separately). For now, suppose that the correspondence between the points of the pattern and the points of the model is fixed that is, p_i is matched to q_i and $n = m$. According to the results in Chapter 3, there is a 3-approximation to δ^*:

Notation 4.5 *Let*

$$\begin{aligned} c_i &= q_i - p_i \text{ for all } 1 \leq i \leq n \text{ and} \\ C &= \{c_1, \ldots, c_n\}. \end{aligned}$$

We define

$$\begin{aligned} T^{(3)} &= C, \text{ and} \\ \delta^{(3)} &= \gamma(C, T^{(3)}, G). \end{aligned}$$

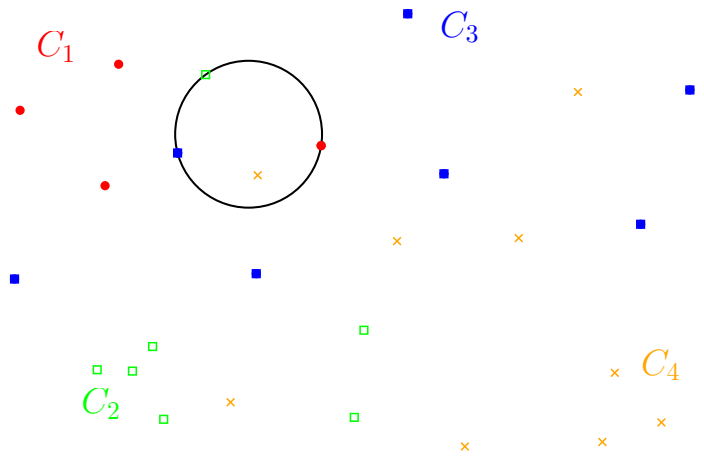

Figure 4.1: Point sets $C_1, \ldots, C_4$ and the smallest multi-color-ball.

Then, $T^{(3)}$ is a 3-approximation to δ^*, see Lemma 3.5 for details.
This idea can be extended to Problem 7 under the directed Hausdorff distance where the translations t_i are restricted to be chosen from C_i for all $1 \leq i \leq n$: Finding a sequence of translations T minimizing $\gamma(\mathcal{C}, T, G)$ is equivalent to picking a point t_i from every set C_i so that the maximum distances between translations corresponding to neighboring vertices in G are minimized.
A closely related problem, as described in the following, is the *Multi-Color-Ball-Problem*:
Given n sets of m points each, the task is to find the disk with the smallest radius that contains at least one point of every set. An illustration of the multi-color-ball of 4 sets is given in Figure 4.1. There are two differences between the Multi-Color-Ball-Problem and the problem of finding a sequence of translations minimizing $\gamma(\mathcal{C}, T, G)$: The first is that in the Multi-Color-Ball-Problem n sets of m points instead of m L_2-disks are considered and the second is that in the Multi-Color-Ball-Problem, not just the distance between the chosen points that are adjacent according to the neighborhood graph but the pairwise distance of all chosen points is considered, which equals an EGSM setting where G is complete. Given an arbitrary graph G, the radius of the smallest multi-color-ball is an approximation to the optimum of $\gamma(\mathcal{C}, T, G)$ and the quality of the approximation depends on the graph diameter (i.e., the length of the longest shortest path in G).
Although the Multi-Color-Ball-Problem is NP-hard under the L_2-metric, see [25], a 2-approximation to the problem is given in [26], where the authors present an algorithm that computes a disk with at most twice the optimal radius in $O(nm \log m + m \log n)$ time. By using this result, in the following, we show that there is a $(n-1)2^{-1}$-approximation to δ^*:

Notation 4.6 *For a point $c \in \mathbb{R}^2$ and $r > 0$ let $D_r(c)$ be the smallest multi-color-ball for a given sequence $\mathcal{C}$ and let $\bar{c}_i$ be a point of C_i that lies in $D_r(c)$ for all $1 \leq i \leq n$. We write*

$$\begin{aligned} T^{\text{app}} &= (\bar{c}_1, \ldots, \bar{c}_n) \text{ and} \\ \delta^{\text{app}} &:= \gamma(\mathcal{C}, T^{\text{app}}, G). \end{aligned}$$

Lemma 4.7 *For a point $c \in \mathbb{R}^2$ and $r > 0$ let $D_r(c)$ be a smallest multi-color-ball for the sequence $\mathcal{C}$ and let $\bar{c}_i$ be a point of C_i that lies in $D_r(c)$ for all $1 \leq i \leq n$. Then, δ^{app} is a $(n-1)2^{-1}$-approximation to δ^*.*

Proof. The points $\bar{c}_i$ and $\bar{c}_j$ lie within $D_r(c)$ for all $1 \leq i, j \leq n$ and $\|\bar{c}_i - \bar{c}_j\| \leq 2r$, thus

$$\delta^* \leq \max_{\{i,j\} \in E} \|\bar{c}_i - \bar{c}_j\| = \delta^{\text{app}} \leq 2r.$$

We prove $\delta^* \geq 2r(n-1)^{-1}$ by contradiction:
Suppose $\delta^* < 2r(n-1)^{-1}$. The constraint encoded in the edge $\{i, j\} \in E$ is equivalent to the L_2-distance between $\bar{c}_i$ and $\bar{c}_j$, since $t_i = \bar{c}_i$ for all $1 \leq i, j \leq n$. The graph G is connected, and the graph diameter is at most $n-1$. Now let v_i and v_k be the vertices at the end of a longest shortest path in G. The geometric interpretation of this path is a sequence of line segments from $\bar{c}_i$ to $\bar{c}_k$ so that in an optimal solution every line segment has length at most δ^*. As illustrated

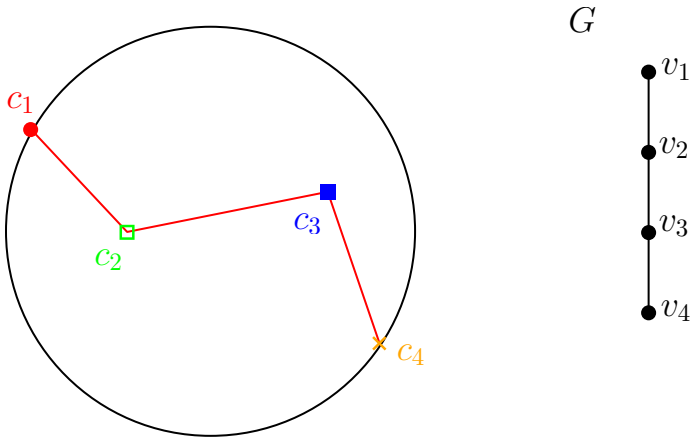

Figure 4.2: The points $c_1, \ldots, c_4$ within the smallest multi-color-ball, the graph G and its geometric interpretation as a sequence of line segments.

in Figure 4.2, since there are at most $n-1$ edges between v_i and v_k,

$$\begin{aligned} \|\bar{c}_i - \bar{c}_k\| &\leq (n-1)\delta^* \\ &< (n-1)\frac{2r}{(n-1)} \\ &= 2r. \end{aligned}$$

As a consequence, the diameter of the smallest enclosing disk of T^{app} is strictly less than $2r$. This implies that $D_r(c)$ is not the smallest multi-color-ball for the sequence $\mathcal{C}$, which is a contradiction. Thus,

$$\delta^* \geq 2r\frac{1}{(n-1)} = \frac{2}{n-1}\delta^{\text{app}}.$$

□

Combining the results of Lemma 3.5 and Lemma 4.7 instantly shows that δ^{app} is a $2^{-1}3(n-1)$-approximation to δ^*. In other words,

$$\frac{2}{3(n-1)}\delta^{\text{app}} \leq \delta^* \leq \delta^{\text{app}}. \tag{4.1}$$

Unfortunately, the Multi-Color-Ball-Problem is NP-hard. So there is no way of computing δ^{app} in polynomial time. However, the 2-approximation to the problem given in [26] can be computed $O(nm \log m + m \log n)$ time. Let $\delta^{2\text{app}}$ be approximation to δ^{app} computed with the algorithm described in [26]. Then, δ^{app} is a $3(n-1)$-approximation to δ^*.
As mentioned earlier this holds for graphs with more than one connected component as well, since the strategy can then be applied on each of the connected components separately.

The Algorithm

Before we state the actual algorithm that, for an $\epsilon > 0$, gives a $(1+\epsilon)$-approximation to δ^* for Problem 7 with a given feedback vertex set, we consider a simpler EGSM setting for illustrating the technique that will be used.

In Chapter 3, Problem 7 under cyclic neighborhoods and under fixed correspondence has already been discussed in detail. For most parts, the approach presented in Chapter 3 can easily be adapted to Problem 7 under cyclic neighborhoods (with unknown correspondence):
Let G be a simple cycle and suppose that a value δ and a translation t are given. Consider the following decision problem:

Is there is a δ-admissible sequence $T = (t_1 = t, \ldots, t_n)$?
Since t_1 is fixed, solving this decision variant of Problem 7 equals solving the decision variant of Problem 7 where G is a path with $n+1$ vertices in order $v_1, \ldots, v_n, v_1$ and fixed translations on both ends of the path. This means, we can cut the cycle and modify it to be a path by guessing one translation for a given value of δ.
Recall that every (δ^*, i)-admissible translation t_i lies within the set

$$D_{\delta^{2\text{app}}}(C_i) = \bigcup_{c \in C_i} D_{\delta^{2\text{app}}}(c) :$$

If $\gamma(\mathcal{C}, T, G) \leq \delta^{2\text{app}}$ for a sequence of translations T, this implies that $\vec{h}(t_i, C_i) \leq \delta^{2\text{app}}$ for all $1 \leq i \leq n$ and all $t_i \in T$.
A simple idea to get a $(1+\epsilon)$-approximation to Problem 7 is to sample t_1 from a dense enough ϵ_{grid}-grid that covers $D_{\delta^{2\text{app}}}(C_1)$ (we call the points of this grid *translation-samples*) and to sample δ on the interval $[3^{-1}(n-1)^{-1}\delta^{2\text{app}}, \delta^{2\text{app}}]$ (we call the samples of this interval *radius-samples*) with a suitable sample rate ϵ_{obj}. Here, $\epsilon_{\text{grid}}, \epsilon_{\text{obj}} = \Theta(\epsilon\delta^{2\text{app}})$. Then, a binary search on the interval $[3^{-1}(n-1)^{-1}\delta^{2\text{app}}, \delta^{2\text{app}}]$ is carried out and in every step of the binary search, Algorithm 2 is used to test for the radius-sample δ at hand if one of the translation-samples is part of a δ-admissible sequence.

This approach can be applied to other graphs that contain exactly one cycle: W.l.o.g., let $v_1 \in V$ be the vertex so that the graph induced by $V' = V \setminus \{v_1\}$ is cycle-free.

Notation 4.8 *Let $G = (V, E)$ be a graph. We denote the degree of the vertex $v \in V$ with* $\deg(v)$.

Instead of transforming G into a path, we transform G into a forest G_{dg} by copying v_1 $\deg(v) - 1$ times and replacing every endpoint v_1 of the edges of G with a different copy of v_1, see Figure 4.3.
Additionally, for every call of Algorithm 2, we use the same fixed translation-sample for every copy of v_1 and sample-radius δ. Note that we need the same translation-sample for each copy of v_1, because they represent the same vertex in the original graph G, which means that they correspond to the same translation.
A generalization of this approach of graphs that contain more than one cycle is described in the following.

Let X be a FVS of G so that $|X| = k_f > 1$ and for $V = \{v_1, \ldots, v_n\}$ let w.l.o.g. $X = \{v_1, \ldots, v_{k_f}\}$. We can transform G into a forest: We create a set of new

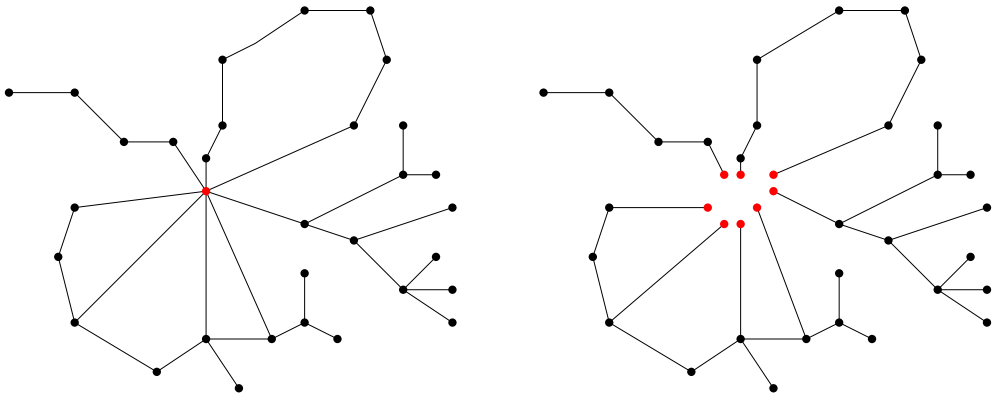

Figure 4.3: Left: Neighborhood graph G with a FVS of cardinality 1 (red). Right: Neighborhood graph G after it has been transformed into a tree. The red vertices are the eight copies of the red vertex of G before the transformation.

vertices of size $\deg(x)$ for every $x \in X$:

$$V_{\text{Copy}(x)} \coloneqq \{x_i \mid 1 \leq i \leq \deg(x)\},$$

the set of copies of x. Let E_x be the set of edges in E that are adjacent to x, so

$$E_x \coloneqq \{\{x, v\} \in E \mid v \in V\}.$$

We create a new set of edges $E_{\text{Copy}(x)}$ that is similar to E_x, but in every edge x is replaced with a different copy of x, i.e., with a different vertex of $V_{\text{Copy}(x)}$:

$$E_{\text{Copy}(x)} \coloneqq \left\{\{x_i, v\} \mid \{x, v\} \in E_x \text{ and } x_i \in V_{\text{Copy}(x)}\right\}.$$

As illustrated in Figure 4.4, with this definition, we can construct the desired graph, i.e., a dissected version of G, as follows:

$$\begin{aligned} G_{\text{dg}} &\coloneqq (V_{\text{dg}}, E_{\text{dg}}) \text{ with} \\ V_{\text{dg}} &\coloneqq (V \setminus X) \cup \left(\bigcup_{x \in X} V_{\text{Copy}(x)} \right) \text{ and} \\ E_{\text{dg}} &\coloneqq (E \setminus E_x) \cup \left(\bigcup_{x \in X} E_{\text{Copy}(x)} \right), \end{aligned} \tag{4.2}$$

With this, we can now state Algorithm 5:

Algorithm 5 *We are given the point sets $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$, a undirected graph $G = (V, E)$ and a FVS X of G so that $|X| = k_f > 1$ and for $V = \{v_1, \ldots, v_n\}$ let w.l.o.g. $X = \{v_1, \ldots, v_{k_f}\}$.*

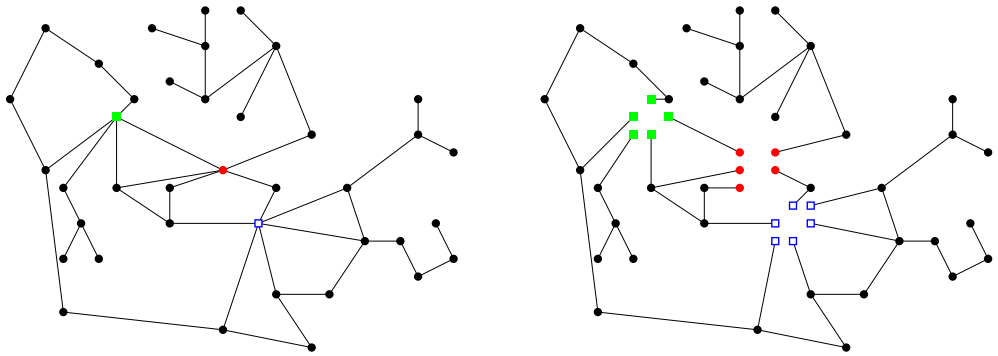

Figure 4.4: Left: Neighborhood graph G with a FVS of cardinality 3 (green, red and blue).
Right: Neighborhood graph G after it has been transformed into a tree. The colored vertices are the copies of the vertices of the same color of G before the transformation.

First, we construct $G_{\mathrm{dg}} = (V_{\mathrm{dg}}, E_{\mathrm{dg}})$*, dissected version of* G*, as elaborately described above.*
Now a binary search on the interval $[3^{-1}(n-1)^{-1}\delta^{2\mathrm{app}}, \delta^{2\mathrm{app}}]$ *is carried out, until a* $(1+\epsilon)$*-approximation to* δ^* *is found. For every radius-sample* $\delta \in [3^{-1}(n-1)^{-1}\delta^{2\mathrm{app}}, \delta^{2\mathrm{app}}]$*, we sample* t_i *from a dense enough* ϵ_{grid}*-grid that covers* $D_\delta(C_i)$ *for all* $1 \leq i \leq k_f$*.*
Then, we test by applying Algorithm 2 on G_{dg}*,* δ *and the* k_f*-tuple* $(t_1, \ldots, t_{k_f})$ *of fixed translation-samples at hand if there is a solution*

$$\begin{aligned} T' &= (t_1 = t'_1, \ldots, t_{k_f} = t'_{k_f}, t'_{k_f+1}, \ldots, t'_n) \text{ with} \\ \gamma(\mathcal{C}, T', G_{\mathrm{dg}}) &\leq \delta. \end{aligned}$$

Finally, the smallest radius-sample that permits a YES*-instance of Algorithm 2 is returned along with a withness* T'*.*

Note that for any $v_i \in X$ we assign the same translation-sample to each vertex in $V_{\mathrm{Copy}(v_i)}$ in G_{dg}, since all vertices in $V_{\mathrm{Copy}(v_i)}$ represent the same vertex in the original graph G, which means that they correspond to the same translation. Since the translation-samples for the different vertices in X are chosen independently from each other, every possible combination of them has to be tested by a run of Algorithm 2. Let $l_i(\delta) \in \mathbb{N}$ be the number of translation-samples that constitute the grid that covers $D_\delta(C_i)$ for any $1 \leq i \leq n$, then $\Pi_{1 \leq i \leq k_f} l_i(\delta)$ k_f-tuples of translation-samples have to be tested for every fixed radius-sample δ.
In particular, this number can slightly be reduced in most cases. For the radius-sample at hand, it is enough to find one k_f-tuple of translation-samples

that permits a YES-instance of Algorithm 2. Also, a clever choice of which k_f-tuples of translation-samples are tested first, could be promising: In Section 3.3, we rated translation-samples that constitute a NO-instance of the algorithm at hand by counting the steps of the propagation algorithm until the first intersection operation results in an empty set. However, it is not known if this strategy also improves the upper bound on the runtime of Algorithm 5.

Lemma 4.9 *For every $1 \leq i \leq n$, the following equation holds:*

$$\delta^* = \min_{t \in D_{\delta^{2\text{app}}}(C_i)} \quad \min_{T \text{ with } t_i = t} \gamma(\mathcal{C}, T, G).$$

Proof. Since $\delta^* \leq \delta^{2\text{app}}$, $t_i \in D_{\delta^{2\text{app}}}(C_i)$ for every optimal sequence of translations and thus

$$\min_{t \in D_{\delta^{2\text{app}}}(C_i)} \quad \min_{T \text{ with } t_i = t} \gamma(\mathcal{C}, T, G) = \min_{T} \gamma(\mathcal{C}, T, G) = \delta^*.$$

□

Theorem 4.10 *For an $\epsilon > 0$, let*

$$\begin{aligned} \epsilon_{\text{grid}} &:= \frac{1}{\sqrt{2}^5} \frac{1}{3(n-1)} \epsilon \delta^{2\text{app}}, \\ \epsilon_{\text{obj}} &:= \frac{1}{4} \frac{1}{3(n-1)} \epsilon \delta^{2\text{app}} \text{ and} \\ \epsilon_{\text{A2}} &:= \frac{1}{2} \frac{1}{3(n-1)} \epsilon \delta^{2\text{app}}. \end{aligned}$$

For a given neighborhood graph $G = (V, E)$ with n vertices and a FVS $X \subseteq V$ of cardinality $k_f \geq 1$, Algorithm 5 computes a $(1+\epsilon)$-approximation to δ^* in

$$O\left(\left(\frac{1}{\epsilon^2} n^2 m\right)^{k_f} \left(\log n + \log \frac{1}{\epsilon}\right) T_{A2}\left(\frac{\epsilon}{n}, k_f n, m\right)\right)$$

time and

$$O\left(\left(\frac{1}{\epsilon^2} n^2 m\right) k_f T_{A2}\left(\frac{\epsilon}{n}, k_f n, m\right)\right)$$

space.

Proof. The proof of correctness directly follows from the proof of correctness of Algorithm 2 in Chapter 2 together with Lemma 4.9.
We now estimate the runtime of Algorithm 5: First, $\delta^{2\text{app}}$ is computed in $O(nm \log m + m \log n)$ time and G_{dg} is constructed in $O(k_f n)$ time. Note

that in the worst case, the endpoints of $O(k_f n)$ edges in G get replaced by a copy of their endpoints in G_{dg}, so V_{dg} consists of $O(k_f n)$ vertices at most. Every translation that corresponds to a vertex of the FVS is sampled by an ϵ_{grid}-grid of $O(\epsilon^{-2}n^2m)$ translation-samples, since there are m disks of radius $\delta^{2\text{app}}$ to be covered with an ϵ_{grid}-grid and $\delta^* \in [3^{-1}(n-1)^{-1}\delta^{2\text{app}}, \delta^{2\text{app}}]$. Calling Algorithm 2 for all possible k_f-tuples results in $O((\epsilon^{-2}n^2m)^{k_f})$ calls. For every k_f-tuple, a binary search on the interval $[3^{-1}(n-1)^{-1}\delta^{2\text{app}}, \delta^{2\text{app}}]$ is carried out up to accuracy ϵ_{obj} to find a suitable radius-sample. This takes $(\log n + \log \epsilon^{-1})$ time per k_f-tuple. Since V_{dg} consists of $O(k_f n)$ vertices at most, Algorithm 2 itself requires $T_{A2}(\epsilon n^{-1}, k_f n, m)$ time per call. This results in a total runtime of $O((\epsilon^{-2}n^2m)^{k_f}(\log n + \log \epsilon^{-1})T_{A2}(\epsilon n^{-1}, k_f n, m))$.
In every step of Algorithm 5, the best possible sample-radius along with the corresponding k_f-tuple is stored, which requires $O(\epsilon^{-2k_f})$ space. Algorithm 2 requires $T_{A2}(\epsilon n^{-1}, k_f n, m)$ space per call, which directly follows from the results given in Chapter 2. Hence, Algorithm 2 requires $O(T_{A2}(\epsilon n^{-1}, k_f n, m))$ space. This makes $O((\epsilon^{-2}n^2m)k_f + T_{A2}(\epsilon n^{-1}, k_f n, m))$ space in total.

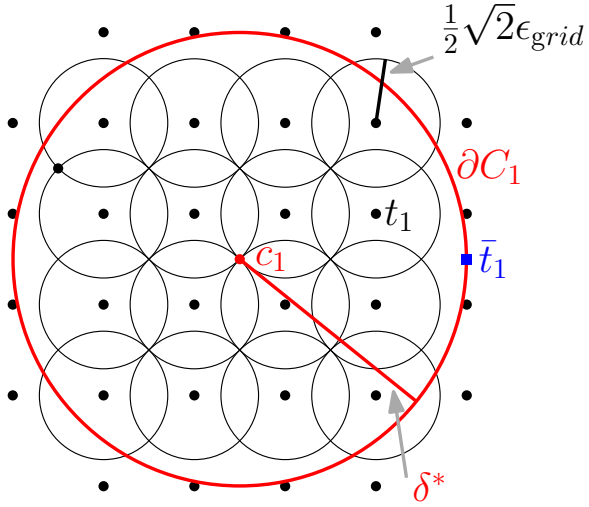

Figure 4.5: One disk from $D_{\delta^*}(C_1)$ (red) and a grid of translation-samples along with the collection of disks (black), where every translation-sample in $D_{\delta^*}(C_1)$ is the center of a disks and all disks have radius $2^{-1}\sqrt{2}\epsilon_{\text{grid}}$. The point t_1 is the translation-sample, which is closest to $\bar{t}_1$.

Now we prove the quality of the approximation: There are three inaccuracies that add up: ϵ_{A2}, ϵ_{grid} and ϵ_{obj}. Suppose, we know δ^* and let w.l.o.g. $D_{\delta^*}(C_1)$ be covered with an ϵ_{grid}-grid. Consider the collection of disks, where every translation-sample in $D_{\delta^*}(C_1)$ is the center of a disks and all disks have radius $\sqrt{2}\epsilon_{\text{grid}}$. Then, $D_{\delta^*}(C_1)$ is completely contained in this collection of disks,

whereas for a radius of $2^{-1}\sqrt{2}\epsilon_{\text{grid}}$ the collection would not neccessarily cover $D_{\delta^*}(C_1)$ completely, see Figure 4.5.
In particular, any point in $D_{\delta^*}(C_1)$ which is furthest from any grid-point of $D_{\delta^*}(C_1)$ in the worst case is located on $\partial D_{\delta^*}(C_1)$. Let one of these points be called $\bar{t}_1$ and let the closest grid point be called t_1. Then, in the worst case,

$$\begin{aligned} \|t_1 - \bar{t}_1\| &< \sqrt{2}\epsilon_{\text{grid}} \\ &\overset{(4.1)}{=} \frac{1}{4}\frac{1}{3(n-1)}\epsilon\delta^{2\text{app}} \\ &\leq \frac{1}{4}\epsilon\delta^*. \end{aligned}$$

Since we have to add ϵ_{A2} and ϵ_{obj} to this, the total relative error is no more than

$$\begin{aligned} \frac{1}{4}\epsilon\delta^* + \frac{1}{2}\frac{1}{(3(n-1))}\epsilon\delta^{2\text{app}} + \frac{1}{4}\frac{1}{(3(n-1))}\epsilon\delta^{2\text{app}} &\leq \\ \frac{1}{4}\epsilon\delta^* + \frac{1}{2}\epsilon\delta^* + \frac{1}{4}\epsilon\delta^* &= \epsilon\delta^*. \end{aligned}$$

□

On Feedback Vertex Sets and Fixed Correspondence

The runtime of Algorithm 5 improves significantly if the correspondence between the points of the pattern and the points of the model is known that is, $n = m$ and w.l.o.g. p_i is matched to q_i for all $1 \leq i \leq n$. Let

$$\begin{aligned} c_i &:= q_i - p_i \text{ for } 1 \leq i \leq n \text{ and} \\ C &:= (c_1, \ldots, c_n). \end{aligned}$$

Then, every (δ, i)-admissible translation lies within the L_2-disk $D_\delta(c_i)$. Also, there is a 3-approximation to δ^*, see Lemma 3.5. Therefore, let $\delta^{(3)}$ be defined as in Lemma 3.5 ($\delta^{(3)} = \gamma(C, C)$). This means, $\delta^* \in [3^{-1}\delta^{(3)}, \delta^{(3)}]$ in this section.
With that, we can proof the following theorem:

Theorem 4.11 *Let $n = m$ and let p_i be matched to q_i for all $1 \leq i \leq n$. Let*

$$\begin{aligned} \epsilon_{\text{grid}} &:= \frac{1}{3\sqrt{2}^5}\epsilon\delta^{(3)}, \\ \epsilon_{\text{obj}} &:= \frac{1}{12}\epsilon\delta^{(3)} \text{ and} \\ \epsilon_{\text{A2}} &:= \frac{1}{6}\epsilon\delta^{(3)}. \end{aligned}$$

For a given neighborhood graph $G = (V, E)$ with n vertices and a FVS $X \subseteq V$ of cardinality $k_f \geq 1$, Algorithm 5 computes a $(1+\epsilon)$-approximation to δ^* in $O(\epsilon^{-2k_f-1/2} \log \epsilon^{-1} k_f n)$ time and $O(\epsilon^{-1/2} k_f n + \epsilon^{-2k_f})$ space.

Proof. The proof of correctness directly follows from the proof of correctness of Theorem 4.10.
At the start, the value of a 3-approximation to δ^* is computed in $O(n)$ time and G_{dg} is constructed in $O(k_f n)$ time. In the worst case, the endpoints of $O(k_f n)$ edges in G get replaced by a copy of their endpoints in G_{dg}, which is why V_{dg} consists of $O(k_f n)$ vertices at most. Every translation that corresponds to a vertex of the FVS is sampled by an ϵ_{grid}-grid of $O(\epsilon^{-2})$ translation-samples, since there is just one disk of radius $\delta^{(3)}$ to be covered with an ϵ_{grid}-grid and $\delta^* \in [3^{-1}\delta^{(3)}, \delta^{(3)}]$. As a consequence, calling Algorithm 2 for all possible k_f-tuples results in $O(\epsilon^{-2k_f})$ calls. For every k_f-tuple, a binary search on the interval $[3^{-1}\delta^{(3)}, \delta^{(3)}]$ is carried out up to accuracy ϵ_{obj} to find a suitable radius-sample. This binary search needs $O(\log \epsilon^{-1})$ steps per k_f-tuple.
Now, we need to analyse the runtime of Algorithm 2 for the special case of known correspondence. At the start, Algorithm 2 approximates each (δ, i)-admissible set of translations with a regular convex polygon with $O(\epsilon^{-1/2})$ vertices. Since each of these regions is convex, their intersection is also convex and can therefore be described with $O(\epsilon^{-1/2})$ vertices, see Lemma 2.12. According to Lemma 2.12, the Minkowski sum of such an object and the inscribed polygon of the unit disk in the algorithm does not change the description complexity of the object. Since V_{dg} consists of $O(k_f n)$ vertices, Algorithm 2 works with $k_f n$ objects of description complextity $O(\epsilon^{-1/2})$ each. Intersecting and inflating convex polygons can be done in linear time, which is why one call of Algorithm 2 takes $O(\epsilon^{-1/2} k_f n)$ time. This results in a total runtime of $O\left(\epsilon^{-2k_f-1/2} \log \epsilon^{-1} k_f n\right)$.
We now analyze the space required by Algorithm 5 for the case of known correspondence. In every step of Algorithm 5, the best possible sample-radius along with the corresponding k_f-tuple is stored, which requires $O(\epsilon^{-2k_f})$ space. Algorithm 2 requires $O(\epsilon^{-1/2} k_f n)$ space per call, which directly follows from the analysis above. Hence, Algorithm 5 requires $O(\epsilon^{-1/2} k_f n + \epsilon^{-2k_f})$ space in total.
We now prove the quality of the approximation: There are three inaccuracies that add up: ϵ_{A2}, ϵ_{grid} and ϵ_{obj}. First we consider ϵ_{grid}: In the worst case,

$$\begin{aligned} \|t_1 - \bar{t}_1\| &< \sqrt{2}\epsilon_{\text{grid}} \\ &= 2^{-2}3^{-1}\epsilon\delta^{(3)} \\ &\leq 2^{-2}\epsilon\delta^*, \end{aligned}$$

see Theorem 4.10 for details. Since we have to add ϵ_{A2} and ϵ_{obj} to this, the

total absolute error is no more than

$$\begin{aligned} \frac{1}{4}\epsilon\delta^* + \frac{1}{6}\epsilon\delta^{(3)} + \frac{1}{12}\epsilon\delta^{(3)} &\leq \\ \frac{1}{4}\epsilon\delta^* + \frac{1}{2}\epsilon\delta^* + \frac{1}{4}\epsilon\delta^* &= \epsilon\delta^*. \end{aligned}$$

□

4.3 Solving Instances with Bounded Pathwidth or Treewidth

In Section 4.2 we saw that we can approximate the solution to Problem 7, even if the neighborhood graph G contains several cycles, although the runtime of the $(1+\epsilon)$-approximation algorithm strongly depends on the size of a (M)FVS, which is also related to the number of cycles in G. However, if we know that the cycles are somehow separated into disjoint or almost disjoint groups, the runtime can be improved by exploiting the structural features of a given *path-* or *tree decomposition* of the neighborhood graph at hand:

Pathwidth

The pathwidth of a given graph $G = (V, E)$ is defined as follows:

Definition 4.12 *A* path decomposition *of a graph $G = (V, E)$ is a sequence $\mathcal{P} = (X_1, \ldots, X_r)$ of subsets of V, called* bags*, with $X_i \subseteq V$ for all $1 \leq i \leq r$ so that*

1. $\bigcup_{i=1}^{r} X_i = V$.
2. *For every $\{u, v\} \in E$, there is an $1 \leq l \leq r$ so that the bag X_l contains u and v.*
3. *For every $u \in V$, if $u \in X_i \cap X_k$ for some $i \leq k$, then $u \in X_j$ for every $i \leq j \leq k$.*

The width *of a path decomposition $\mathcal{P} = (X_1, \ldots, X_r)$ is defined as $\max_{1 \leq i \leq r} |X_i| - 1$. The* pathwidth *of a graph G is the smallest possible width of any path decomposition of G.*

An approximation algorithm that, given a graph with n vertices, determines in $O(8^k k^2 n^2)$ time, whether the path- or treewidth of the graph is at most k, is presented in [24]. This algorithm is able to return a path or tree decomposition of width at most $4k + 4$ in case of a YES-instance.

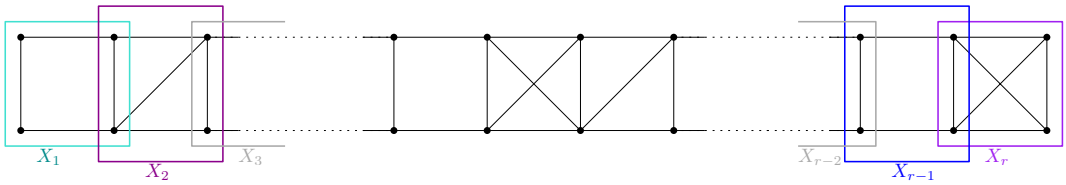

Figure 4.6: Neighborhood graph G of pathwidth 3.

In the following, let $\mathcal{P} = (X_1, \ldots, X_r)$ be a given path decomposition of G of width k_w, see Figure 4.6. Note that checking for a given graph G if there is a path decomposition of width k_w is NP-complete.

Notation 4.13 *For a path decomposition $\mathcal{P} = (X_1, \ldots, X_r)$ of the graph $G = (V, E)$, $G_i := (X_i, E_i)$ is the subgraph of G induced by the bag X_i for all $1 \leq i \leq r$, i.e., $E_i := \{\{u, v\} \in E \mid u, v \in X_i\}$.*
Also, we denote the cardinality of X_i with x_i for all $1 \leq i \leq r$.
The subset of $\mathcal{C}$ that corresponds to the vertices in X_i is denoted with $\mathcal{C}_i$.

Note, that $x_i \leq k_w + 1$ for all $1 \leq i \leq r$.

Problem 7 for a given path decomposition is a special case of Problem 7 for a given tree decomposition, which will be discussed in the next section. The approach to compute a $(1+\epsilon)$-approximation to δ^* is the same in both cases. However, to illustrate the strategy that is used, we first focus on the case that the given decomposition of G is a path decomposition. The proof of correctness and the analyses of run-time, space requirements and approximation quality are included in the next section.

The main idea to get a $(1+\epsilon)$-approximation to δ^* for the variant of Problem 7 with given path decomposition is to sample δ from a suitable interval and then for a fixed δ start at one end of the path decomposition and solve the decision variant of the problem separately bag by bag along the path until the other end of the path is reached.
The $3(n-1)$-approximation $\delta^{2\text{app}}$ introduced in Section 4.2 is the basis for the following strategy: We perform a binary search on the interval $[3^{-1}(n-1)^{-1}\delta^{2\text{app}}, \delta^{2\text{app}}]$ up to accuracy ϵ_{obj} to determine the smallest radius-sample that permits a YES-instance for the decision variant of Problem 7, where all sets of admissible translations are approximated by grids of translation-samples:
Let the radius-sample at hand be called δ. In each iteration, the following strategy is applied: Starting with bag X_1, and then bag by bag until we reach X_r, we compute an approximate solution to the decision variant of Problem 7 restricted to the bag at hand, i.e., the subgraph induced by it, until we either

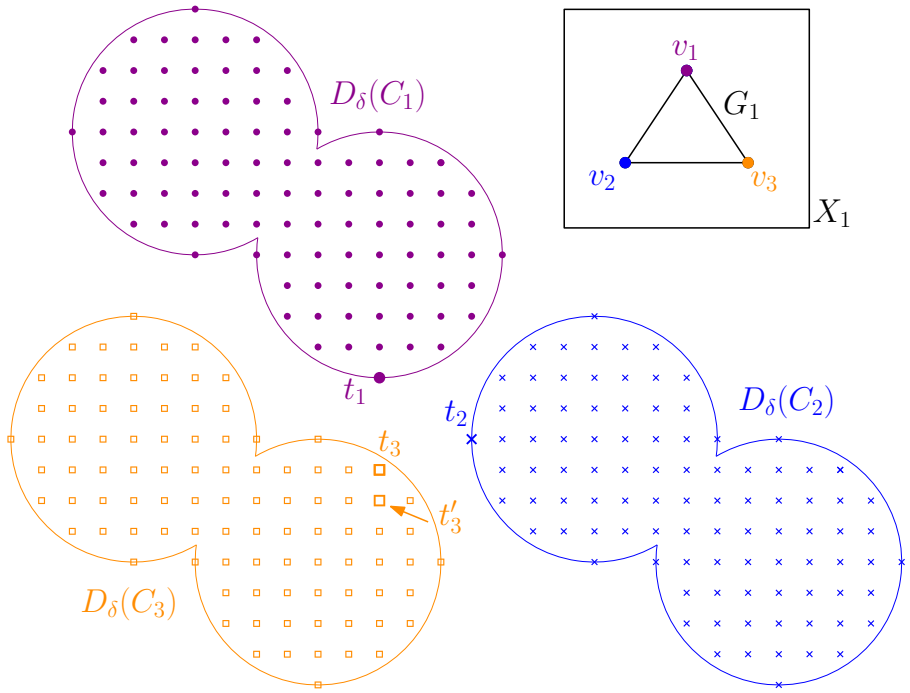

Figure 4.7: Bag X_1 contains the graph G_1, a complete graph with 3 vertices. The sets $D_\delta(C_1)$, $D_\delta(C_2)$ and $D_\delta(C_3)$ are covered by grids of translation-samples. The 3-tuples of translation samples $T_1 = (t_1, t_2, t_3)$ and $T'_1 = (t_1, t_2, t'_3)$ are the only ones that are δ-admissible for the bag X_1, i.e., $\gamma(\mathcal{C}_1, T_1, G_1) \leq \delta$ and $\gamma(\mathcal{C}_1, T'_1, G_1) \leq \delta$.

find a subgraph that is a NO-instance (for the radius-sample δ) or the subgraph G_r induced by X_r permits a YES-instance (for δ): Recall that every vertex in G and with that, every vertex $v_j \in X$ corresponds to a set of m disks of radius δ, $D_\delta(C_j)$ for $1 \leq j \leq x = |X|$.

We start with X_1. For every vertex $v_j \in X_1$, we sample $D_\delta(C_j)$ with a dense enough ϵ_{grid}-grid, see Figure 4.7. This results in x_1 grids of translation-samples. The goal is to test if there is a sequence of translation samples $\bar{T} = (t_1, \ldots, t_{x_1})$ so that $\gamma(\mathcal{C}_1, \bar{T}, G_1) \leq \delta$. Note that the translation-samples for different vertices are chosen independently from each other. This results in $O((\epsilon^{-2}n^2m)^{x_1})$ x_1-tuples of translation-samples that may be δ-admissible for the bag X_1. Note that we sample $D_\delta(C_j)$, for every vertex $v_j \in X_1$ (and later for all other bags of the path decomposition), since the structure of the subgraph of G with vertices in bag X_1 is arbitrary. Thus we do not know beforehand if it is enough to sample the corresponding sets of a certain subset of vertices in order to cut all cycles involved. For certain graph classes, there may also be a more

sophisticated way to search for valid tuples of translation-samples.
For each x_1-tuple $\bar{T}$, we test, if all constraints encoded in G for the vertices contained in X_1 are met, i.e., we test, if $\gamma(\mathcal{C}_1, \bar{T}, G_1) \leq \delta$. This means that we can decide, for every x_1-tuple and for a fixed sample-radius δ, whether the x_1-tuple is δ-admissible (and with this whether it is a witness for a YES-instance) restricted to bag X_1 or if it is not δ-admissible. This information can be stored in a table. If none of the x_1-tuples turns out to be δ-admissible, we conclude that δ was chosen too small and proceed with the next step of the binary search on the radius-samples.
However, if there are any δ-admissible x_1-tuples, we proceed with the next step and iteratively repeat the same step bag by bag for all bags $X_2, \ldots, X_r$: Let $1 \leq i < r$. The translations of the unions of disks corresponding to $X_i \cap X_{i+1}$ have already been sampled in the ith step of the algorithm. Therefore, only the $|X_i \cap X_{i+1}|$-tuples of translation-samples that belong to a δ-admissible x_i-tuple of the previous bag may be part of a δ-admissible sequence of translations and hence are considered as candidates for the next bag. In doing so, all x_{i+1}-tuples for bag X_{i+1} in question are checked and the information of δ-admissible translation-samples is carried on through the path, until either none of the x_i-tuples of bag X_i is δ-admissible for some $1 \leq i \leq r$ or at least one x_r-tuples of bag X_r turns out to be δ-admissible, which implies that there is a sequence of translation-samples so that all constraints encoded in G are met for the considered sample-radius δ.

Treewidth

The treewidth of a graph $G = (V, E)$ is defined as follows:

Definition 4.14 *A* tree decomposition *of a graph $G = (V, E)$ is a pair $\mathcal{T} = (T, \{X_t\})_{t \in V(T)}$, where T is a tree so that every vertex t of T is assigned to a* bag *$X_t \subseteq V$ with*

1. *$\bigcup_{t \in V(T)} X_t = V$.*
2. *For every $\{u, v\} \in E$, there is a vertex t of T so that the bag X_t contains u and v.*
3. *For every $u \in V$, the set $T_u = \{t \in V(T) \mid u \in X_t\}$ induces a subtree of T.*

We define

$$n_v \coloneqq |V(T)|.$$

The width *of a tree decomposition $\mathcal{T} = (T, \{X_t\})_{t \in V(T)}$ is*

$$\max_{t \in V(T)} |X_t| - 1.$$

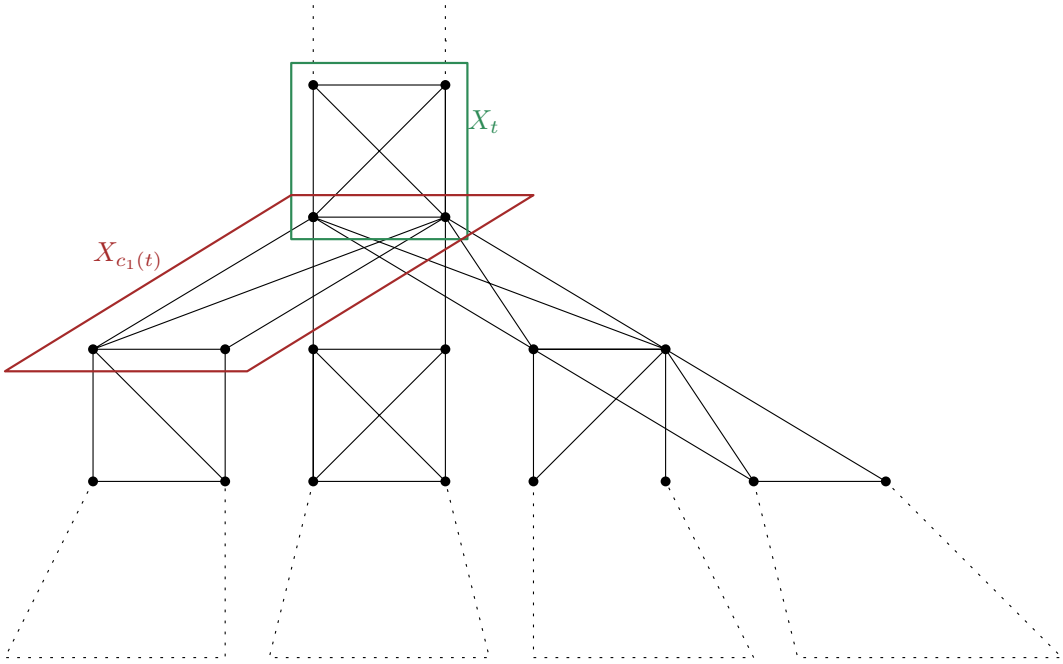

Figure 4.8: Neighborhood graph G of treewidth 3.

The treewidth *of a graph G is the smallest possible width of any tree decomposition of G.*

In the following, let $\mathcal{T} = (T, \{X_t\})_{t \in V(T)}$ be a given tree decomposition of G of width k_w, see Figure 4.8. Checking for a given graph G if there is a tree decomposition of width k_w is NP-complete.

Notation 4.15 *For a tree decomposition $\mathcal{T} = (T, \{X_t\})_{t \in V(T)}$ of the graph $G = (V, E)$, $G_t := (X_t, E_t)$ is the subgraph of G induced by the bag X_t for all $t \in T$, i.e., $E_t := \{\{u, v\} \in E \mid u, v \in X_t\}$.*
Also, we denote the cardinality of X_t with x_t for all $t \in T$.
The subset of $\mathcal{C}$ that corresponds to the vertices in X_t is denoted with $\mathcal{C}_t$.

The strategy that can be applied in case of a neighborhood graph with given treewidth, is just a slight modification of the one for given pathwidth:

Algorithm 6 *We are given the point sets $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$, a undirected graph $G = (V, E)$ and a tree decomposition $\mathcal{T} = (T, \{X_t\})_{t \in V(T)}$ of G.*
First, we pick an arbitrary vertex r of T as the root and from now on consider T_r, the tree rooted in r. We perform a binary search on the interval $[3^{-1}(n-1)^{-1}\delta^{2\text{app}}, \delta^{2\text{app}}]$ up to accuracy ϵ_{obj} to determine the smallest radius-sample

that permits a YES*-instance for the decision variant of Problem 7, where all sets of admissible translations are approximated by grids of translation-samples:*

Let the sample-radius at hand be named δ. In every iteration, we propagate δ-admissible translation-samples from bottom to top through T_r by computing an approximate solution to the decision variant of Problem 7 restricted to the bag at hand, i.e., the subgraph induced by it, until we either find a subgraph that is a NO*-instance (for the radius-sample δ) or the subgraph G_r induced by X_r permits a* YES*-instance (for δ):*

We start with the leafs of T_r and the bags they correspond to. Let l be a leaf of T_r. Just as we did for the case that a path decomposition is given, for every vertex $v_j \in X_l$ with $1 \leq j \leq x_l$, we sample $D_\delta(C_j)$ with a dense enough ϵ_{grid}-grid. This results in x_l grids of translation-samples. The goal is to test if there is a sequence of translation samples $\bar{T} = (t_1, \ldots, t_{x_l})$ so that $\gamma(\mathcal{C}_l, \bar{T}, G_l) \leq \delta$. Note that the translation-samples for different vertices are chosen independently from each other. This results in $O((\epsilon^{-2}n^2m)^{x_l})$ x_l-tuples of translation-samples that may be δ-admissible for the bag X_l. Note that we sample $D_\delta(C_j)$, for every vertex $v_j \in X_l$ (and later for all other bags of the tree decomposition), since the structure of the subgraph of G with vertices in bag X_l is arbitrary. Thus we do not know beforehand if it is enough to sample the corresponding sets of a certain subset of vertices in order to cut all cycles involved. For certain graph classes, there may also be a more sophisticated way to search for valid tuples of translation-samples. For each x_l-tuple $\bar{T}$, we decide if it is δ-admissible (so it is a witness for a YES*-instance) for the problem restricted to bag X_l or if it is not δ-admissible by testing, if $\gamma(\mathcal{C}_l, \bar{T}, G_l) \leq \delta$. This information can be stored in a table.*

Then we proceed with the inner vertices of T_r: For every inner vertex t of T_r, let n_t be the number of children of t and let $\mathfrak{c}_1(t), \ldots, \mathfrak{c}_{n_t}(t)$ be the children of t. We call the bags $X_{\mathfrak{c}_i(t)}$ child-bags *of X_t for $1 \leq i \leq n_t$ and X_t the* parent-bag *of $X_{\mathfrak{c}_i(t)}$ for $1 \leq i \leq n_t$. All child-bags of X_t have already been handled in a previous iteration of the algorithm. In particular, all translations corresponding to vertices in $\left(\bigcup_{i=1}^{n_t} X_{\mathfrak{c}_i(t)}\right) \cap X_t$ have been sampled. As a consequence, only the $\left|\left(\bigcup_{i=1}^{n_t} X_{\mathfrak{c}_i(t)}\right) \cap X_t\right|$-tuples of translation-samples that belong to a* YES*-instance in all of the child-bags they appear in, are considered as candidates for the parent-bag. In doing so, all x_t-tuples for bag X_t in question are tested and the information of δ-admissible translation-samples is carried on through the tree, until all x_t-tuples of a bag X_t are not δ-admissible for some $t \in V(T_r)$ or at least one x_r-tuple of bag X_r turns out to be δ-admissible, which implies that there is a δ-admissible sequence of translations for the sample-radius δ.*

Theorem 4.16 *For an $\epsilon > 0$, let*

$$\begin{aligned} \epsilon_{\text{grid}} &:= \frac{1}{\sqrt{2}^3}\frac{1}{3(n-1)}\epsilon\delta^{2\text{app}} \text{ and} \\ \epsilon_{\text{obj}} &:= \frac{1}{2}\frac{1}{3(n-1)}\epsilon\delta^{2\text{app}}. \end{aligned}$$

For a given neighborhood graph $G = (V, E)$ with n vertices and a given tree-decomposition $\mathcal{T} = (T, \{X_t\})_{t \in V(T)}$ with width $k_w \geq 1$, Algorithm 6 computes a $(1+\epsilon)$-approximation to δ^* in

$$O\left(\left(\frac{1}{\epsilon^2}n^2m\right)^{(k_w+1)}\left(\log\frac{1}{\epsilon}+\log n\right){k_w}^2 n\right)$$

time and

$$O\left(\left(\frac{1}{\epsilon^2}n^2m\right)^{(k_w+1)}\right)$$

space.

Proof. The proof of correctness equals the proof of correctness for Algorithm 1 given in Chapter 2, Theorem 2.4 with the slight difference that the vertices of the tree become bags. Therefore, we refer to Theorem 2.4 for the proof of correctness.
The binary search on the interval $[3^{-1}(n-1)^{-1}\delta^{2\text{app}}, \delta^{2\text{app}}]$ takes $O(\log\epsilon^{-1}+\log n)$ time. In each step of the binary search, the whole graph is analysed: There are $O(n)$ bags. We focus on one specific bag X_t with $t \in T_r$ and let the radius-sample at hand be named δ. There are up to $k_w + 1$ vertices and $2^{-1}(k_w+1)(k_w+2)$ edges in X_t and every vertex and every edge in X_t encodes one constraint in the objective function. Hence it takes

$$O((k_w+1)+2^{-1}(k_w+1)(k_w+2)) = O({k_w}^2)$$

time to test for a fixed tuple $\bar{T}$ of translation-samples, if $\gamma(\mathcal{C}_t, \bar{T}, G_t) \leq \delta$. Each tuple of translation-samples for bag X_l consists of $x_l \leq k_w + 1$ translation-samples. For every vertex $v_i \in X_t$, the corresponding translation-sample of the tuple at hand is chosen from a dense enough ϵ_{grid}-grid that covers $D_\delta(C_i)$. Since in the worst case, the largest radius-sample is $3(n-1)\delta^*$ covering a disk of this radius with an ϵ_{grid}-grid results in $O(n^2{\epsilon_{\text{grid}}}^{-1})$ translation-samples. $D_\delta(C_i)$ consists of m disks, thus $O(n^2m{\epsilon_{\text{grid}}}^{-1})$ translation-samples are needed to cover $D_\delta(C_i)$. Since the translation samples corresponding to different vertices in X_t can be chosen independently from each other, there are $O((\epsilon^{-2}n^2m)^{x_t})$

x_t-tuples of translation-samples that are possibly a solution to the decision variant of Problem 7, restricted to bag X_t. Thus, processing one bag takes $O((\epsilon^{-2}n^2m)^{(k_w+1)}{k_w}^2)$ time.
Computing all YES-tuples of one parent-bag from its n_t child-bags takes $O(n_t(\epsilon^{-2}n^2m)^{(k_w+1)})$ time. Since there are $O(n)$ child-bags in total, the whole updating process takes $O(n(\epsilon^{-2}n^2m)^{(k_w+1)})$ time. This results in a total runtime of $O((\epsilon^{-2}n^2m)^{(k_w+1)}(\log\epsilon^{-1}+\log n){k_w}^2n)$.
The space that is required is bounded by the size and number of tables that have to be managed. There are $O(n)$ tables of size $O((\epsilon^{-2}n^2m)^{(k_w+1)})$ each.
Now we prove the quality of the approximation: There are two inaccuracies that add up: ϵ_{grid} and ϵ_{obj}. First we consider ϵ_{grid}: In the worst case,

$$\begin{aligned}\|t_1-\bar{t}_1\| &< \sqrt{2}\epsilon_{\text{grid}}\\ &= 2^{-1}(3(n-1))^{-1}\epsilon\delta^{2\text{app}}\\ &\leq 2^{-1}\epsilon\delta^*,\end{aligned}$$

see Theorem 4.10 for details. Since we have to add ϵ_{obj} to this, the total absolute error is no more than

$$2^{-1}\epsilon\delta^*+2^{-1}(3(n-1))^{-1}\epsilon\delta^{2\text{app}}\leq\epsilon\delta^*.$$

□

On Path- or Tree Width and Fixed Correspondence

If the correspondence between the points of the pattern and the points of the model is known that is, $n=m$ and p_i is matched to q_i for all $1\leq i\leq n$, there is a much faster approach to get a $(1+\epsilon)$-approximation to the optimum of Problem 7. We use Algorithm 6 with one slight alternation:
Instead of using the $3(n-1)$-approximation $\delta^{2\text{app}}$ to δ^*, we can now use the 3-approximation $\delta^{(3)}$ to δ^* given in Lemma 3.5. As a consequence, the interval that contains δ^* shrinks to $[3^{-1}\delta^{(3)},\delta^{(3)}]$, which is why the binary search on this interval takes $O(\log\epsilon^{-1})$ time. Since p_i is matched to q_i for all $1\leq i\leq n$, every (δ,i)-admissible translation lies within the single L_2-disk $D_\delta(c_i)$ with $c_i\coloneqq p_i-q_i$ (instead of the union of m disks in the case for unknown correspondence). Additionally, the largest radius-sample is $3\delta^*$ and thus we need no more than $O(\epsilon^{-2})$ translation-samples, to sample this disk. This reduces the time to process one bag to $O((\epsilon^{-2})^{(k_w+1)}{k_w}^2)$ and the whole updating process takes $O(n(\epsilon^{-2})^{(k_w+1)})$ time.
Summing all that up, we see that Algorithm 6 runs in $O(\epsilon^{-2(k_w+1)}\log\epsilon^{-1}{k_w}^2n)$ time and $O(\epsilon^{-2(k_w+1)}n)$ space under fixed correspondence under the usage of the 3-approximation $\delta^{(3)}$.

4.4 Combining both Approaches

Solving EGSM problems under translations for neighborhood graphs that contain cycles is a complex task, in particular, because the algorithmic complexity of the problem seems to depend on the number these cycles. The strategies discussed in this chapter for given FVSs or given path- or tree decomposition have a lot in common: Both "cut" the cycles of G by sampling some sets of admissible transformations at hand and use the structure of the given graph G in order to keep the number of regions that have to be sampled as small as possible. Also, both approaches can be combined:
Suppose we are given a decomposition $\mathcal{G} = (G', \{X_t\})_{t \in V(G')}$ of G which is defined analog to a tree decomposition of G but is not necessarily cycle-free and suppose $\mathcal{G}$ has a small (given) feedback vertex set $X \subset \bigcup_{t \in V(G')} \{X_t\}$ (i.e., X contains bags of $\mathcal{G}$). Then we can transform $\mathcal{G}$ into a tree decomposition $\mathcal{G}_{\text{dg}}$ by following the strategy of Equation (4.2) and then solve the problem approximately as described in Section 4.3 by computing feasible tuples of translation-samples bag by bag and bottom-to-top through $\mathcal{G}_{\text{dg}}$ while using the same translation-samples for every bag of $\mathcal{G}_{\text{dg}}$ that corresponds to a bag in the FVS of $\mathcal{G}$.

4.5 Discussion

Overall, the runtimes and space requirements of the algorithms introduced in this chapter seem somehow disappointing, especially for instances under unknown correspondence, and are therefore to some extend not of use in real-life applications. Regarding Algorithm 6, the algorithm for bounded path- or treewidth, the cause of this is the combinatorial complexity of the structures inside the particular bags, which does not depend on the path- or treewidth of the graph but rather the structure of the graph restricted to the bags, which is unknown in our setting. By adding more information beforehand,Algorithm 6 could be customized in order to get reasonably faster.
For some instances, e.g., instances where G is a planar graph, it could be even faster to just run Algorithm 6 under fixed correspondence for all m^n possible matchings of the points of P to the points of Q instead of using Algorithm 6 for instances under unknown correspondence, since in this setting, the runtimes of processing one bag under unknown and under fixed correspondence differ by a factor of $O((n^2m)^{k_w+1}) = O((n^2m)^{\sqrt{n}+1})$.

One question that arrises when looking critically at the results displayed in this chapter is the question of what happens if we use Algorithm 6 on an

instance, where the neighborhood graph is an actual tree or path, i.e., $k_w = 1$. Suppose, the correspondence of the points of the pattern to the points of the model is fixed and G is a path. Then, Algorithm 6 takes $O((\epsilon^{-2})^2 \log \epsilon^{-1} n)$ time, where $O(\epsilon^{-2})$ is the number of samples needed to sample each of the n disks involved. Appendix A contains Algorithm 11, a comparable algorithm that runs in $O(\epsilon^{-1} n)$ time if the decision variant is considered and can easily be adapted to solve the optimization version of the problem in $O(\epsilon^{-1} \log \epsilon^{-1} n)$ time.
Obviously, the runtimes of both algorithms differ by a factor of $O(\epsilon^{-3})$. One reason is that Algorithm 11 approximates every disk by just sampling its boundary, hence $O(\epsilon^{-1})$ samples are needed for each of the n disks involved. Algorithm 6 can be adjusted accordingly, resulting in a runtime of $O((\epsilon^{-1})^2 \log \epsilon^{-1} n)$. Note that this method only works for instances under fixed correspondence where the neighborhood graph is a tree or a path. However, Algorithm 6 is still slower by a factor of $O(\epsilon^{-1})$ after the adjustment. During a call of Algorithm 6, every bag is processed separately. Since every bag contains exactly two vertices, all possible 2-tuples of samples contained in each bag are considered and we test for the value δ at hand if they are δ-admissible restricted to the corresponding bag. This takes $O(\epsilon^{-2})$ time. This step is crucial if $k_w > 1$ in order to "cut" all possible cycles contained in the bags at hand. However, if G is a tree or a path, there are no cycles and this step is unneeded, which explains the conceptual difference between both algorithms.

Chapter 5

The Combinatorial Complexity of Admissible Regions under the Euclidean Distance

A key operation used in Chapter 3 and Chapter 4 in order to compute exact solutions for EGSM problem instances under the L_2-norm is computing the Minkowski sum of an L_2-disk and a set A that originated from intersecting and merging L_2-disks of different radii (we also call it *inflating* the *admissible regions*). Usually, the number of circular arcs that define the boundary of the set gained from inflating A is greater than the number of circular arcs defining the boundary of A, which is one major reason for considering other norms than the L_2-norm and the design of Algorithm 2, the approximation algorithm given in Chapter 2. In this chapter, we provide some insights about the issue of estimating the combinatorial complexity of such regions and give upper and lower bounds for the following special EGSM instances.

5.1 Problem Statement

In this chapter, $\|\cdot\|$ denotes the L_2-norm.

Problem 8 *Given:*

$P = (p_1, \ldots, p_n)$	*a sequence of points (the pattern),*
$Q = (q_1, \ldots, q_n)$	*a sequence of points (the model),*
$G = (V, E)$	*a cycle-free graph with* $V = \{v_1, \ldots, v_n\}$ *and* $E \subseteq \{\{v_i, v_j\} \mid v_i, v_j \in V\}$, *and*
$\delta \in \mathbb{R}^+$	*a parameter.*

Find: A sequence of translations $T = (t_1, \ldots, t_n)$ *so that*

$$\max\left(\max_{1\leq i\leq n} \|q_i - (p_i + t_i)\|, \max_{\{v_i,v_j\}\in E} \|t_i - t_j\|\right) \leq \delta. \tag{5.1}$$

Let

$$\begin{aligned} &c_i \coloneqq q_i - p_i \text{ for } 1 \leq i \leq n \text{ and} \\ &C \coloneqq (c_1, \ldots, c_n). \end{aligned} \tag{5.2}$$

Recall that measuring the distance of the points $(t_i + p_i)$ to q_i in model space is the same as measuring the distance of the points t_i and c_i in translation space, which is why Problem 8 can be studied in translation space entirely. Then, inequality (5.1) changes to

$$\gamma(T, C, G) \coloneqq \max\left(\max_{i\in V} \|c_i - t_i\|, \max_{\{i,j\}\in E} \|t_i - t_j\|\right) \leq \delta. \tag{5.3}$$

In [16] the authors give an algorithm, called Algorithm B, that solves Problem 8 in $O(n^2 \log n)$ time if only translations in a fixed direction are allowed. In the first part of this chapter, we adapt the algorithm to Problem 8, then called Algorithm 7, and prove that it runs in $O(n^2 \log n)$ time and space if G is a path and arbitrary translations in the plane are allowed. We give an example illustrating the fact that solving the problem using Algorithm 7 requires $\Omega(n^2)$ space. However, if G is not a path, the problem seems to be more complicated. In the second part of this chapter, we give (non-polynomial) upper bounds on the time and the space required by Algorithm 7 to solve the problem if G is a tree, which can be improved to be polynomial if G is a complete, not neccesarily binary, tree. In the last part of this chapter, we give a rough description of the difficulties in estimating the combinatorial complexity that arise if the correspondence between the points of the pattern and the points of the model is not fixed.
Note that the neighborhood graph G is not necessarily connected. However, if G consists of more than one connected component, the problem at hand can be solved by computing the solution for every tree within the forest separately. Then, the given EGSM instance is a YES-instance, iff there is a valid sequence of translations for every tree in G.

First, we introduce some notation:

Recall that for a closed set $A \subset \mathbb{R}^2$, ∂A denotes its boundary. Also, for $c \in \mathbb{R}^2$ and $r > 0$, $D_r(c)$ denotes the disk with radius r centered in c. D_δ denotes the disk with radius δ centered in the origin.

Notation 5.1 *Let $c \in \mathbb{R}^2$ and $r > 0$, and let s and e be two points on $\partial D_r(c)$. Then (r, c, s, e) denotes the circular arc of $\partial D_r(c)$ with startpoint s and endpoint e (clockwise).*

Also recall from Defintion 2.5 that for a closed connected set $B \subset \mathbb{R}^2$ a closed set $A \subset \mathbb{R}^2$ is called *B-fat* if every point $a \in A$ can be covered by a translated copy of B that is fully contained in A.

Definition 5.2 *We call a point t* admissible (for δ, C and G)*, iff t is part of a sequence T so that*

$$\gamma(T, C, G) \leq \delta.$$

Strictly speaking, the concept of admissibility depends on δ and C, but since δ and C are part of the input, we refrain from including them in the notation. Let $c \in \mathbb{R}^2$. Recall that I_c denotes the set of translations that are at most δ-far from c:

$$I_c := \{t \mid \|c - t\| \leq \delta\} = D_\delta(c).$$

To simplify the presentation, we associate the points of C with the vertex set V, since every point in C corresponds to one vertex of V. For a vertex $v \in V$ that represents the translation that is to be computed for point $c \in C$, we store the set of translations I_c that are admissible for $(\{v\}, \varnothing\})$ in v and refer to it simply as I_v.

5.2 The Algorithm

In [16] the authors give an algorithm, here called Algorithm B, that solves Problem 8 in $O(n^2 \log n)$ time and space if only translations in a fixed direction are allowed. This algorithm has already been used as the basis for solving Problem 8 under different norms and with unknown correspondence between the points of the pattern and the points of the model in Chapter 2. Algorithm B can easily be adapted to solve Problem 8, although the runtime changes due to the more complex geometry of the objects that are processed. Chapter 2 contains a detailed description of this algorithm for problem instances under the L_1-, the L_∞-, or under polygonal norms along with a proof of its correctness. Since we need some information on how the algorithm works under the L_2-norm in order to evaluate its runtime for solving Problem 8, we give a short description of a variant of this algorithm which is adjusted to solve Problem 8 at this point:

Algorithm 7 *We are given a point sequence $C = (c_1, \ldots, c_n)$, a tree $G = (V, E)$ and a parameter $\delta \geq 0$.*

Every vertex v corresponds to a point $c \in C$ and we can identify v with c. Hence, at start, the set

$$I_v = I_c = D_\delta(c)$$

is stored in every node v of the tree.
We then pick an arbitrary vertex $r \in V$ and henceforth consider T_r, the tree rooted in r. For an internal vertex v let $\mathfrak{c}_1(v), \ldots, \mathfrak{c}_{n_v}(v)$ be the n_v children v and let T_v be the subtree in T_r rooted in v.
In each step of the algorithm, a vertex v of the current tree is selected with the property that all children of v are leaves or vertices which already have been updated. Then, the vertex v is updated to a new vertex v' and the set $I_{v'}$ of admissible regions for v' is computed as follows: First, we inflate *all regions $I_{\mathfrak{c}_i(v)}$ by δ for $1 \leq i \leq n_v$ which results in a set*

$$I^\delta_{\mathfrak{c}_i(v)} \coloneqq I_{\mathfrak{c}_i(v)} \oplus D_\delta,$$

where $\oplus$ denotes the Minkowski sum. The admissible region $I_{v'}$ for the new vertex v' is given by

$$I_{v'} = \left(\bigcap_{i=1}^{n_v} I^\delta_{\mathfrak{c}_i(v)} \right) \cap I_v.$$

This process is repeated until one of the following cases occurs:

1. *There is a vertex v with $I_v = \varnothing$ (after a contraction):*
   The process stops and NO *is returned as the answer to Problem 8.*
2. *The root r is updated and $I_{r'} \neq \varnothing$:*
   The algorithm terminates and returns YES *as the answer to Problem 8.*

Note that, if $I_{v'} \neq \varnothing$, the sequence of translations T can be computed from $I_{v'}$ and the sets of admissible translations stored in the vertices of T_v.
Figure 5.1 illustrates the computed admissible regions for a given point sequence $C = (c_1, \ldots, c_8)$ and δ^*.

5.3 Minkowski Sums of Admissible Regions and their Combinatorial Complexity

Inflating admissible regions with an L_2-disk increases the number of circular arcs that compose the boundaries of the corresponding regions at hand, see Figure 5.2. To this point there are no results on what the combinatorial complexity of these regions is or even if it is polynomial. Computing reasonable upper and lower bounds of these objects appears to be challenging. In the

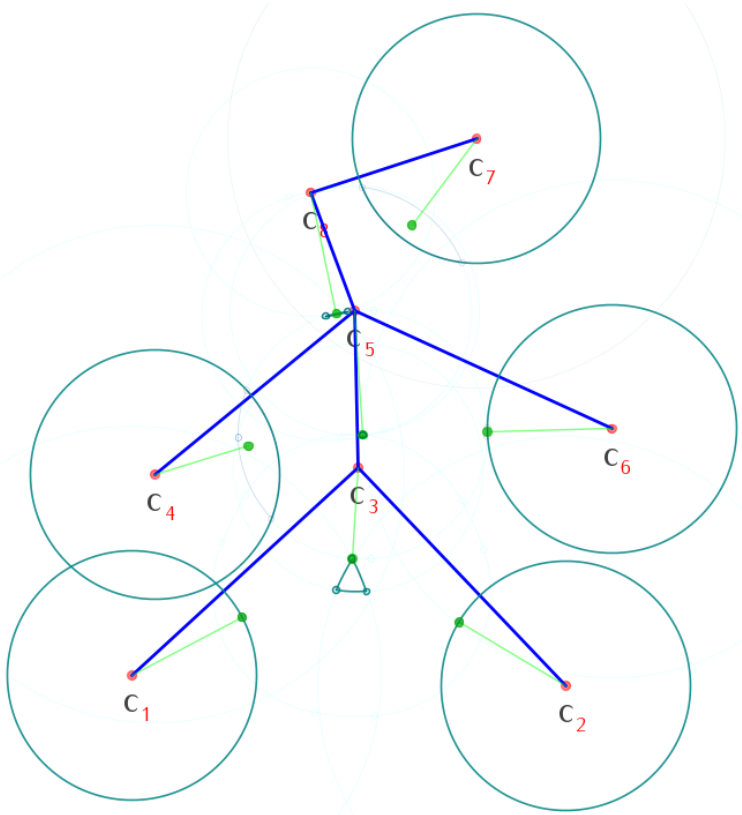

Figure 5.1: The computed admissible regions (turqouise) for a given point sequence $C = (c_1, \ldots, c_8)$ and δ^*. The edges of the corresponding neighborhood graph are indicated as line segments (blue). The figure also contains a witness for an optimal sequence of translations (green points).

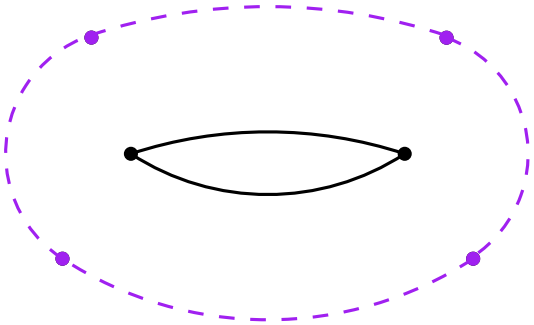

Figure 5.2: An admissible region with a boundary that consists of two circular arcs (solid line). Its inflated version (dashed line) consists of four circular arcs.

following, we take a closer look at these regions and their complexity as they are constructed as intermediate results during a run of Algorithm 7 if applied to Problem 8 under the 1-to-1-distance. Each of these regions can be described by a sequence of circular arcs:

Definition 5.3 *A (not necessarily convex or connected) set $B \subset \mathbb{R}^2$ is called* arc-set *if the boundary of B can be described by a sequence of circular arcs; $k(B)$ denotes number of these arcs.*
We call $k(B)$ the description complexity *of B.*
If two successive circular arcs share the same tangent in their common endpoint, this endpoint point is called smooth. *We call ∂A* smooth *if all endpoints of ∂A are smooth.*

One of the goals in this chapter is to find an upper bound on the description complexity of given arc-sets. Since the number of arcs of an arc-set that consists of at least zwo circular arcs equals the number of its endpoints, we can either estimate the number of circular arcs or the number of endpoints of the arc-set at hand in order to get an upper bound for its description complexity.

Observation 5.4 *Let $A \subset \mathbb{R}^2$ be a convex arc-set, so that ∂A consists of l circular arcs $a_1, \ldots, a_l$ (i.e., $k(A) = l$) with radii r_i, centers m_i and endpoints e_i, hence*

$$a_i = (r_i, m_i, e_i, e_{i+1})$$

for all $1 \leq i \leq l$ with $e_{l+1} = e_1$, see Figure 5.3 for an example.
Then $\partial A^\delta = \partial(A \oplus D_\delta)$ is given by the sequence of circular arcs

$$(b_1, \bar{a}_1, b_2, \bar{a}_2, \ldots, b_l, \bar{a}_l)$$

with

$$\begin{aligned} b_i &= \left(\delta, e_i, \frac{\delta}{r_{i-1}}(e_i - m_{i-1}) + e_i, \frac{\delta}{r_i}(e_i - m_i) + e_i\right) \text{ and} \\ \bar{a}_i &= \left(r_i + \delta, m_i, \frac{\delta}{r_i}(e_i - m_i) + e_i, \frac{\delta}{r_i}(e_{i+1} - m_i) + e_{i+1}\right). \end{aligned} \tag{5.4}$$

The common endpoint e_{i+1} of the circular arcs a_i and a_{i+1} is smooth if the rays $[m_i e_{i+1}$ and $[m_{i+1} e_{i+1}$ are subsets of the same straight line. In this case the circular arc b_{i+1} of ∂A^δ framed by $\bar{a}_i$ and $\bar{a}_{i+1}$ has length 0. Also, ∂A^δ is smooth by construction.
The following equation holds:

$$l \leq k\left(A^\delta\right) \leq 2l. \tag{5.5}$$

Furthermore, $k\left(A^\delta\right) = k\left((A^\delta)^\delta\right)$. For more information about Minkowski sums, we refer to [2].

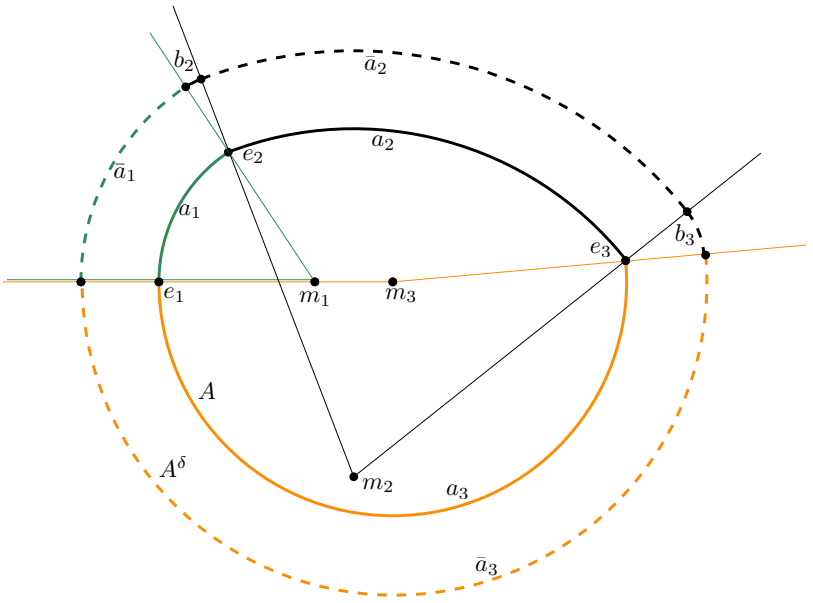

Figure 5.3: The set ∂A (solid fat lines) consists of the three arcs a_1, a_2 and a_3. ∂A^δ (dashed fat lines) consists of the five arcs $\bar{a}_1, b_2, \bar{a}_2, b_3$ and $\bar{a}_3$. The endpoint e_1 on ∂A is smooth and creates one endpoint of ∂A^δ, while the endpoints e_2 and e_3 on ∂A are not smooth and each create two endpoints on ∂A^δ.

In other words, each endpoint of ∂A creates one or two endpoints on ∂A^δ, depending on wether or not the endpoint of ∂A is smooth, see Figure 5.3. Note that this only holds if A is convex.

On Path Neighborhoods

In the following, let $G = (V, E)$ be a path with $V = \{v_1, \ldots, v_n\}$ and $E = \{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n-1\}$.

Theorem 5.5 *Algorithm 7 solves Problem 8 in $O(n^2 \log n)$ time if the neighborhood graph G is a path with n vertices.*

Proof. Let G be rooted in vertex v_n. After the initialization step, the admissible region stored in vertex v_i for $1 \leq i \leq n$ is $I_{v_i} = D_\delta(c_i)$.
In the next step of Algorithm 7, the vertex v_2 is updated to a new vertex v_2' (since it is the only vertex whose child v_1 is a leaf) by inflating I_{v_1} and intersecting it with I_{v_2}. The new vertex v_2' then represents the region

$$I_{v_2'} = I_{v_2} \cap I_{v_1}^\delta = D_\delta(c_2) \cap D_{2\delta}(c_1),$$

hence $k\left(I_{v_2'}\right) \leq 2$.

The same procedure is repeated iteratively until the root is updated (or $I'_{v_i} = \varnothing$ at some point). Therefore, in the ith step of Algorithm 7, the set

$$I_{v'_{i+1}} = I_{v_{i+1}} \cap I^{\delta}_{v'_i} = D_\delta(c_{i+1}) \cap I^{\delta}_{v'_i}$$

is computed. The set $I_{v_{i+1}}$ is a disk of radius δ and $I^{\delta}_{v'_i}$ is a convex set that is D_δ-fat, since it is the Minkowski-sum of a convex set and D_δ. Hence the boundaries of $I_{v_{i+1}}$ and $I^{\delta}_{v_i}$ intersect in at most two points and

$$k\left(I_{v'_{i+1}}\right) \leq k\left(I^{\delta}_{v'_i}\right) + 2$$

for every $2 \leq i < n$.
In the next step, $I_{v'_{i+1}}$ is inflated. As we know from Observation 5.4, every endpoint of $\partial I^{\delta}_{v_i}$ is smooth and hence every endpoint of $\partial I^{\delta}_{v_i}$ that is now part of $\partial I_{v'_{i+1}}$ causes one endpoint in $I^{\delta}_{v'_{i+1}}$. In the worst case there are two endpoints on $I_{v'_{i+1}}$ that originated from the intersection of $I_{v_{i+1}}$ and $I^{\delta}_{v'_i}$. They are not necessarily smooth and therefore can create at most 4 endpoints in $I^{\delta}_{v'_{i+1}}$. Hence,

$$k\left(I^{\delta}_{v'_{i+1}}\right) \leq k\left(I^{\delta}_{v'_i}\right) + 4$$

and

$$k\left(I_{v'_n}\right) \leq 2 + k\left(I^{\delta}_{v'_{n-1}}\right) \leq 2 + \sum_{i=2}^{n-1} 4 = 2 + 4(n-2).$$

The intersection operation on each vertex can be done in $O(n \log n)$ time by using a sweep line strategy, which leads to a total runtime of $O(n^2 \log n)$. □

We now give an example, where $k\left(I'_{v_n}\right) = 2(n-1)$. For the construction of this example, we need the following lemma.

Lemma 5.6 *Let $D_{r_1}(c_1)$ and $D_{r_2}(c_2)$ be two disks so that their boundaries intersect in exactly two points e and $\bar{e}$. We define*

$$\begin{aligned} f_e: \quad & \mathbb{R}_0^+ \to \mathbb{R}^2 \text{ with} \\ x \quad & \mapsto c_1 + x(e - c_1) \text{ and} \\ f_{\bar{e}}: \quad & \mathbb{R}_0^+ \to \mathbb{R}^2 \text{ with} \\ x \quad & \mapsto c_1 + x(\bar{e} - c_1). \end{aligned}$$

The distance $\|f_e(x) - f_{\bar{e}}(x)\|$ increases linearly with respect to x and so does the length of the circular arc defined by $D_x(c_1)$, $f_e(x)$ and $f_{\bar{e}}(x)$.

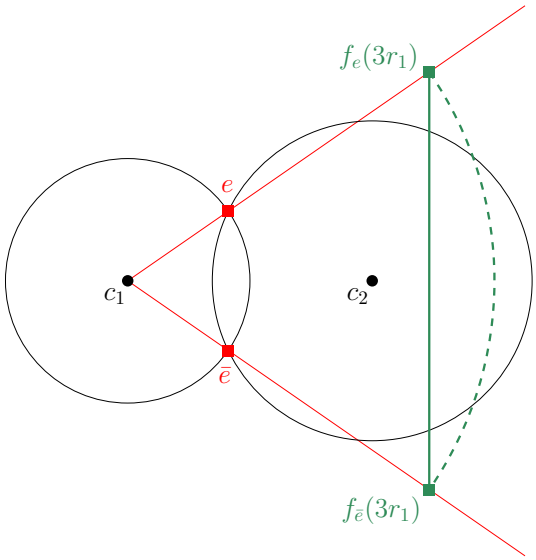

Figure 5.4: Illustration of Lemma 5.6 with $\|f_e(3r_1) - f_{\bar{e}}(3r_1)\|$ (solid fat line) and the circular arc $(3r_1, c_1, f_e(3r_1), f_{\bar{e}}(3r_1))$ (dashed fat line).

Proof. See Figure 5.4 for a small example of this proof. The application of Thales' theorem leads to

$$\begin{aligned}\|f_e(x) - f_{\bar{e}}(x)\| &= \|c_1 + x(e - c_1) - (c_1 + x(\bar{e} - c_1))\| \\ &= \|x(e - \bar{e})\| \\ &= x\|e - \bar{e} + 2c_1 - 2c_1\| \\ &= x\|f_e(1) - f_{\bar{e}}(1)\|.\end{aligned}$$

Let α be the interior angle of the triangle $\Delta(c_1, f_e(x), f_{\bar{e}}(x))$ at vertex c_1. The circular arc $(x, c_1, f_e(x), f_{\bar{e}}(x))$ has length

$$\frac{\alpha}{2\pi} 2x\pi = x\alpha.$$

□

Example 5.7 *Illustrations of this example can be found in Figure 5.5 and Figure 5.6.*
We define $c_1 := (0, 0)$ and $\delta := 1$. For $1 < i \leq n$ the points c_i are placed on the positive y-axis with $c_i.y < c_j.y$ for all $i < j$. The instance is therefore symmetric with respect to the y-axis. In the following, the exact placement of the points of C is described step by step mainly by using Thales' theorem.
The boundaries of the admissible regions can each be described as a sequence of circular arcs and we will focus on the endpoints of these arcs in the description.

We will denote the endpoints with negative x-component and the endpoints with positive x-component on the boundary of every admissible region with $e_{i,j}$ *and* $\bar{e}_{i,j}$ *for* $1 \leq i, j \leq n$*, respectively. Here, the index* i *indicates that* $e_{i,j}$ *originated from the intersection* $I^{\delta}_{v'_i} \cap D_1(c_{i+1})$ *and the index* j *indicates the* L_2*-distance of* $e_{i,j}$ *to the corresponding centerpoint* c_i*:* $\|c_i - e_{i,j}\| = j + 1$*.*
Hence, let $e_{1,1}$ *and* $\bar{e}_{1,1}$ *be the intersection points of* $\partial I^1_{v_1} = \partial D_2(c_1)$ *and* $\partial I_{v_2} = \partial D_1(c_2)$*. For all* $1 < j < n$*, let*

$$e_{1,j} \coloneqq e_{1,1} + \frac{j-1}{2}(e_{1,1} - c_1)$$

be the point on the ray $[c_1 e_{1,1}$ *with* $\|c_1 - e_{1,j}\| = j + 1$ *and let* $\bar{e}_{1,j}$ *be defined similarly. Also, let*

$$d_{1,j} \coloneqq \|e_{1,j} - \bar{e}_{1,j}\|.$$

Note that for $1 < j \leq n$ *the points* $e_{1,j}$ *are the endpoints on* $I^{\delta}_{v'_j}$ *that originated from the intersection between the circular arcs with center* c_1 *and center* c_2*, see Observation 5.4, Equation* (5.4)*. Let* c_2 *be placed on the y-axis so that*

$$d_{1,1} = 2\frac{1}{n^2}.$$

Then, $k\left(I^{\delta}_{v'_2}\right) = 4$*. Let* c_3 *be placed on the y-axis so that* $e_{1,2}$ *and* $\bar{e}_{1,2}$ *are contained in* $D_1(c_3)$ *and the boundaries of* $I^1_{v'_2}$ *and* $D_1(c_3)$ *intersect in two points* $e_{2,1}$ *and* $\bar{e}_{2,1}$ *with*

$$d_{2,1} \coloneqq \|e_{2,1} - e_{1,2}\| = \frac{1}{n^2}.$$

Note that as a consequence $e_{2,1}$ *and* $\bar{e}_{2,1}$ *are placed on* $\partial D_1(e_{1,1})$ *and* $\partial D_1(\bar{e}_{1,1})$*. For all* $1 < j < n$*, let*

$$e_{2,j} \coloneqq e_{2,1} + (j-1)(e_{2,1} - e_{1,1})$$

and let

$$\bar{e}_{2,j} \coloneqq \bar{e}_{2,1} + (j-1)(\bar{e}_{2,1} - \bar{e}_{1,1}).$$

Let

$$d_{2,j} \coloneqq \|e_{2,j} - e_{1,j+1}\|.$$

We iteratively place all of the c_i *for* $2 < i \leq n$ *the same way: We place every* c_i *on the y-axis so that* $e_{i-2,2}$ *and* $\bar{e}_{i-2,2}$ *are contained in* $D_1(c_i)$ *and the boundaries of* $I^1_{v'_{i-1}}$ *and* $D_1(c_i)$ *intersect in two points* $e_{i-1,1}$ *and* $\bar{e}_{i-1,1}$ *with*

$$d_{i-1,1} \coloneqq \|e_{i-1,1} - e_{i-2,2}\| = \frac{1}{n^2}.$$

Note that as a consequence $e_{i-1,1}$ *and* $\bar{e}_{i-1,1}$ *are placed on* $\partial D_1(e_{i-2,1})$ *and* $\partial D_1(\bar{e}_{i-2,1})$*. For all* $1 < j < n$*, let*

$$e_{i-1,j} \coloneqq e_{i-1,1} + (j-1)(e_{i-1,1} - e_{i-2,1})$$

and let

$$\bar{e}_{i-1,j} \coloneqq \bar{e}_{i-1,1} + (j-1)(\bar{e}_{i-1,1} - \bar{e}_{i-2,1})$$

Let

$$d_{i-1,j} \coloneqq \|e_{i-1,j} - e_{1-2,j+1}\|,$$

see Figure 5.5.
After $n-1$ *iterations* $\partial I'_{v_n}$ *consists of several circular arcs. Instead of counting these arcs, we can count the number of endpoints on* $\partial I'_{v_n}$*. These endpoints are given by the sequence*

$$E_n \coloneqq (e_{n-1,1}, e_{n-2,2}, \ldots, e_{1,n-1}, \bar{e}_{1,n-1}, \ldots, \bar{e}_{n-1,1})$$

of length $2(n-1)$*, due to construction, see Figure 5.6. Let*

$$E_i \coloneqq (e_{i-2,2}, \ldots, e_{1,i-1}, \bar{e}_{1,i-1}, \ldots, \bar{e}_{i-2,2})$$

for $2 \leq i \leq n$*. In order to prove that the construction is actually valid, it remains to show that the sequence of circular arcs with endpoints* E_i *is completely contained in* $D_1(c_i)$*:*
According to Lemma 5.6 every $d_{i,n-i}$ *for* $1 \leq i < n$ *has length at most* n^{-1}*, and so*

$$\|e_{i-1,1} - \bar{e}_{i-1,1})\| < 2.$$

Also, the curvature of every circular arc in between the endpoints is at least the curvature of a disk with radius 1*, and all endpoints are smooth, since* $I^1_{v'_{i-1}}$ *is an inflated set. Hence, the projection of the boundary part of* $I^1_{v'_{i-1}}$ *described by* E_i *on the y-axis is an interval of length less than* 1*. As a consequence, there is a line segment* $\overline{ab}$ *with* $a, b \in \mathbb{R}^2$*,* $a.x = b.x = 0$ *an* $a.y < b.y$ *so that every endpoint of* E_i *on* $\partial I^1_{v'_{i-1}}$ *is completely contained in any disk* $D_1(c)$ *with* $c \in \overline{ab}$*. Thus every centerpoint* c_i *we placed is actually part of the corresponding line segment on the y-axis and henceforth valid.*

On Tree Neighborhoods

In the following, we consider instances, where the neighborhood graph $G = (V, E)$ is a tree rooted in $r \in V$, i.e., $G = T_r$.

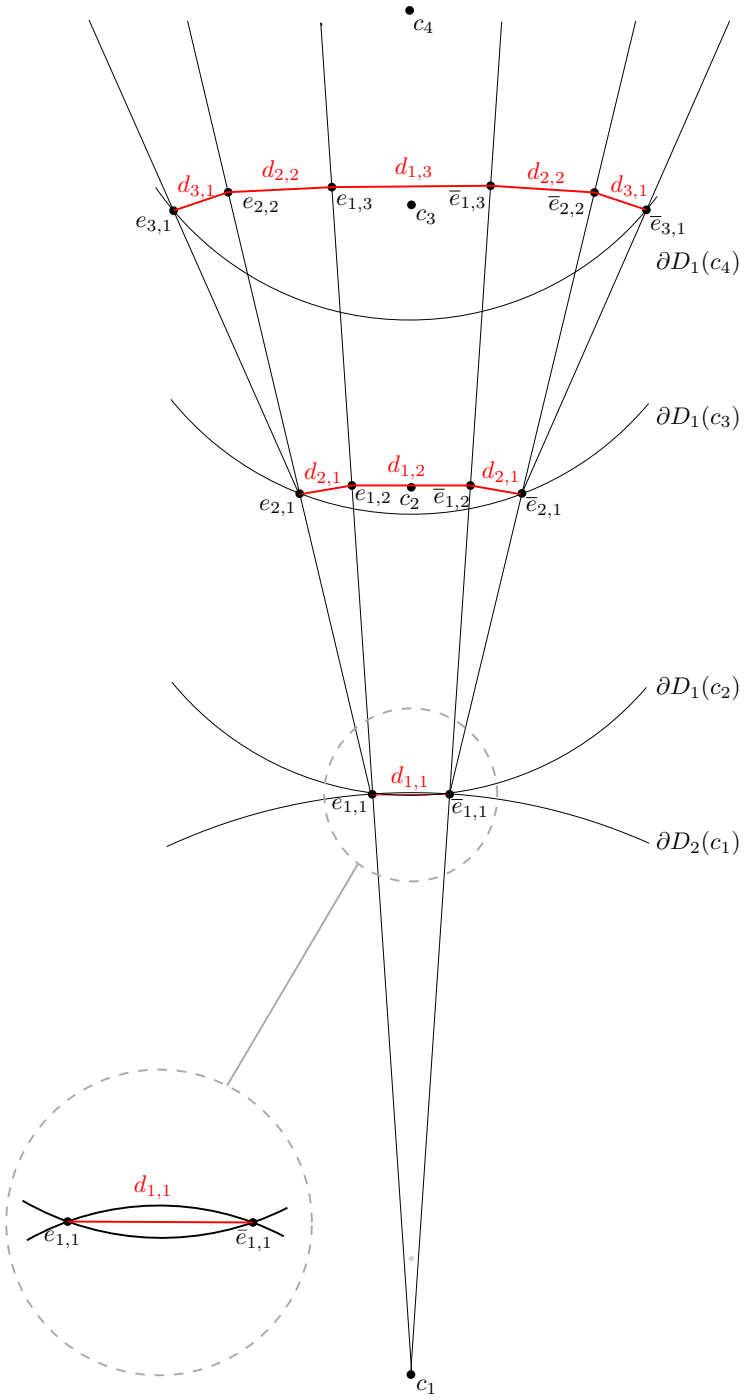

Figure 5.5: Illustration of Example 5.7 with $n = 4$.

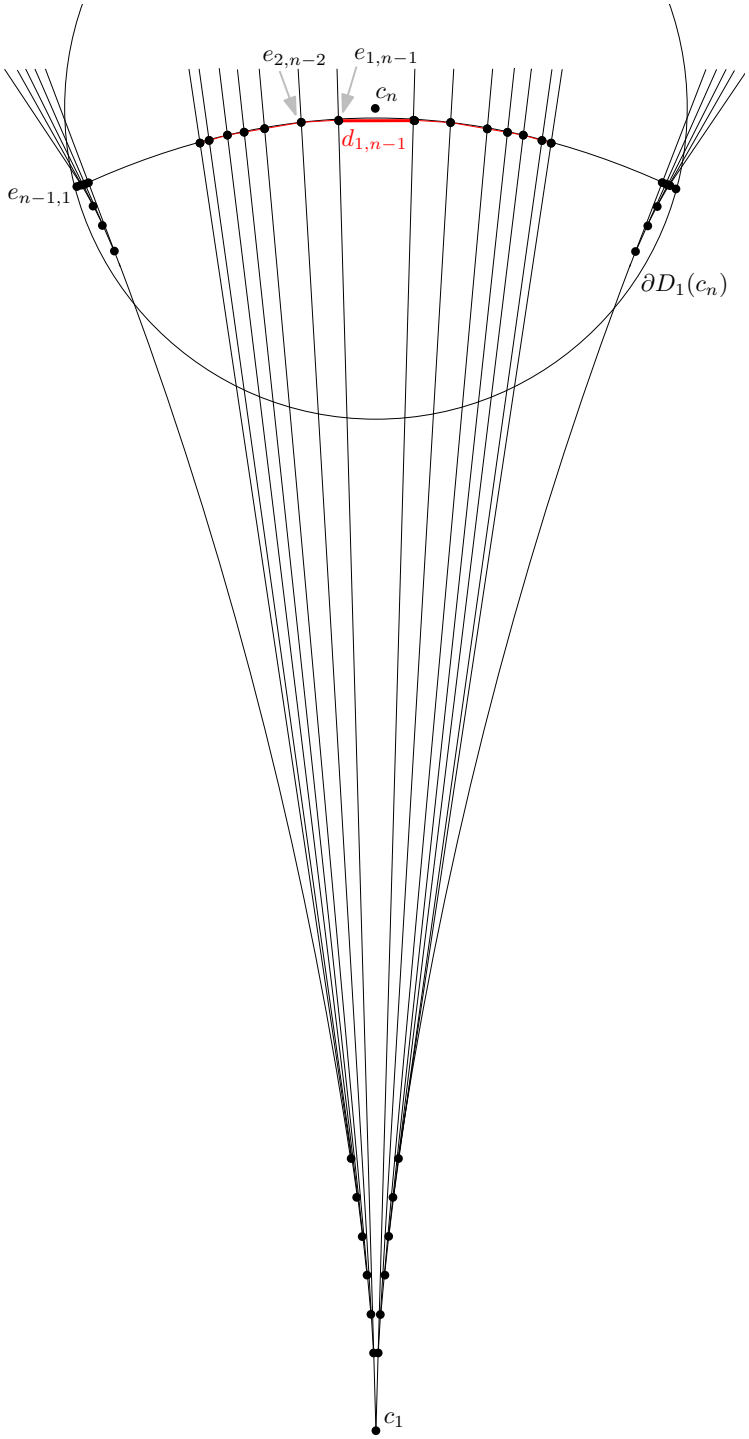

Figure 5.6: The sequence E_n after the last intersection operation in Example 5.7.

The main reason why Algorithm 7 runs in polynomial time for instances with path neighborhoods is that every intersection operation creates no more than two new endpoints. Unfortunately, this is not true if G is a tree that does not equal a path. The runtime of Algorithm 7 strongly depends on the combinatorial complexity of the intermediate admissible regions. One way of computing the combinatorial complexity of such regions is interpreting the involved sequences of circular arcs as arrangements of Jordan arcs:

Lemma 5.8 *Let A and C be convex arc-sets. Also, let $k(A) = a$ and $k(C) = c$ for $a, c \in \mathbb{N}$. Then*

$$k(A \cap C) \leq 4(a + c).$$

Proof. Lemma 5.8 is a special case of the well known Combination Lemma of Sharir et al. in [27]: Let Γ_{red} and Γ_{blue} be two arrangements of Jordan arcs. Also, each pair of arcs from $\Gamma_{\text{red}} \cup \Gamma_{\text{blue}}$ is assumed to intersect in at most s points. In the beginning of the proof of the Combination Lemma, Sharir et al. consider the special case of counting the number of arcs composing the boundary of one of the connected components (which we call E) of the intersection of a single face R of Γ_{red} and a single face B of Γ_{blue}. Let r and b describe the total number of arcs composing the boundary of R and B, respectively, and let u and v be the number of connected components composing the boundary of ∂R and ∂B, respectively. Then, the combinatorial complexity of E is bounded from above by

$$(s + 2)(b + r + 2u + 2v - 4t),$$

where t is the number of connected components of ∂E.
This result can be applied to the sets A and C: Since both sets are convex, they each consist of one connected component, so $u = v = 1$. The intersection of two convex sets is also convex. Therefore $\partial(A \cap C)$ consists of one connected component as well and $t = 1$. Since the Jordan arcs that define the boundary of A and C are circular arcs, each pair of them intersect at most twice, and thus $s = 2$. As a consequence,

$$k(A \cap C) \leq (2 + 2)(a + c + 2 + 2 - 4) = 4(a + c).$$

□

An even better upper bound on the combinatorial complexity of the involved admissible regions can be obtained by using the fact, that the union of two admissible regions is star-shaped, or empty:

Lemma 5.9 *Let A and C be convex arc-sets in the plane and let $A \cap C \neq \emptyset$. Also, let $k(A) = a$ and $k(C) = c$ for $a, c \in \mathbb{N}$. Then*

$$k(A \cap C) \leq 3(a + c).$$

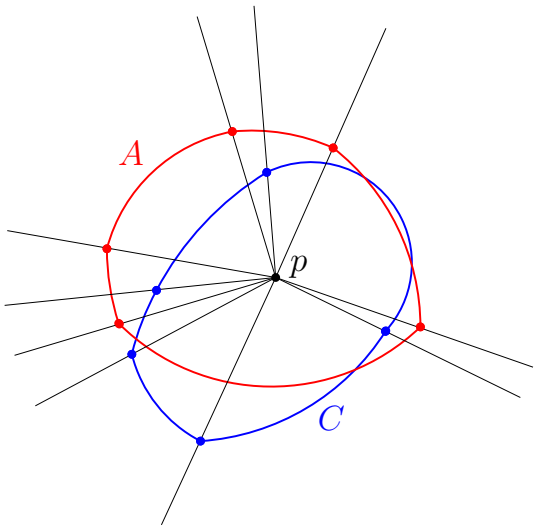

Figure 5.7: The sets A and C and $a + c$ rays with initial point $p \in A \cap C$.

Proof. Since A and C are convex and $A \cap C \neq \emptyset$, the set $A \cup C$ ist star-shaped. In particular, we can consider any point in $p \in A \cap C$ and for every $p' \in \partial A \cup \partial C$ the line segment $\overline{pp'}$ is completely contained in $A \cup C$. Hence, any ray with apex $p \in A \cap C$ intersects exactly once with ∂A and once with ∂C.
Let the endpoints of the circular arcs on ∂A and ∂C in order be named $u_1, \ldots, u_a$ and $v_1 \ldots, v_c$, respectively. For a fixed $p \in A \cap C$, we now divide the plane into $(a + c)$ sections by inserting the $(a + c)$ rays $[pu_i$ and $[pv_j$ for all $1 \leq i \leq a$ and $1 \leq j \leq c$, see Figure 5.7. Each of the resulting regions is framed by two rays and contains exactly two circular arcs: one that originates from ∂A and one that originates from ∂C. Since two circular arcs intersect in at most two points and $A \cup C$ is star shaped, the part of $\partial(A \cap C)$ that lies in the region at hand consists of at most tree circular arcs. Hence,

$$k(A \cap C) \leq 3(a + c).$$

□

This result can be generalized for the intersection of $l \in \mathbb{N}$ convex arc-sets. First, we need the following definition:

Definition 5.10 *Let $h, l, s \in \mathbb{N}$. A sequence $U = (u_1, \ldots, u_h)$ with $u_i \in \mathbb{N}$ for all $1 \leq i \leq h$ is an (l, s)* Davenport-Schinzel sequence *iff*

1. *$1 \leq u_i \leq l$ for all $1 \leq i \leq h$,*
2. *$u_i \neq u_{i+1}$ for all $1 \leq i < h$, and*

3. there are no $s+2$ *indices* $1 \leq i_1 < \ldots < i_{s+2} \leq h$ *with*

$$\begin{aligned} u_{i_1} &= u_{i_j} \quad \text{for all } 1 \leq j \leq s+2 \text{ with } j = 1 \mod 2, \\ u_{i_2} &= u_{i_j} \quad \text{for all } 1 \leq j \leq s+2 \text{ with } j = 0 \mod 2 \end{aligned}$$

and $u_{i_1} \neq u_{i_2}$.

In [27], Sharir et al. give upper bounds on the length of (l, s) Davenport-Schinzel sequences subject to s, including the following:

Lemma 5.11 *The length of any* $(l, 2)$ *Davenport-Schinzel sequence is at most* $2l - 1$.

Proof. The following proof is also included in [27], Theorem 1.9.
Let $h \in \mathbb{N}$ and let $U = (u_1, \ldots, u_h)$ with $u_i \in \mathbb{N}$ for all $1 \leq i \leq h$ be an (l, s) Davenport-Schinzel sequence. The proof proceeds by induction on l:
If $l = 1$, then

$$|U| = h = 1 = (2l - 1).$$

For the induction step $l - 1 \to l$ we suppose that the length of any $(l-1, 2)$ Davenport-Schinzel sequence is at most $2l - 3$. For each $1 \leq a \leq l$ let μ_a denote the smallest index with $u_{\mu_a} = a$. If μ_a is undefined for some a, then U is an $(l-1, 2)$ Davenport-Schinzel sequence and thus

$$h \leq 2l - 3 < 2l - 1$$

by induction hypothesis. Now let μ_a be defined for all $1 \leq a \leq l$. Let b be the symbol for which μ_b is largest. We can assume that U contains only a single occurrence of b (at position μb): Suppose that on the contrary $u_k = b$ for some $k > \mu_b$. Then $c = u_{\mu_b+1} \neq b$ and since U is a $(l, 2)$ Davenport-Schinzel sequence, all appearances of c in U must occur between the positions μ_b and k. Thus $\mu_c = \mu_b + 1$, which is a contradiction to the maximality of μ_b. We now remove the single appearance of b from U and if $u_{\mu_b-1} = u_{\mu_b+1}$, we also remove one of the newly adjacent equal elements from U. We call the resulting sequence U'. U' is an $(l-1, 2)$ Davenport-Schinzel sequence and $h - 2 \leq |U'| \leq h - 1$. Thus by using the induction hypothesis we get

$$h \leq (2l - 3) + 2 = 2l - 1.$$

□

With this, we can now state the following lemma:

Lemma 5.12 *For an $l \in \mathbb{N}$ let $A_1, \ldots, A_l$ be convex arc-sets in the plane. Then*

$$k\left(\bigcap_{i=1}^{l} A_i\right) \leq (2l-1)\sum_{i=1}^{l} k(A_i).$$

Proof. and let $\bigcap_{i=1}^{l} A_i \neq \emptyset$ First, we introduce the following notation to simplify the presentation:

$$A_{\mathrm{I},l} \coloneqq \bigcap_{i=1}^{l} A_i \text{ and}$$
$$A_{\mathrm{U},l} \coloneqq \bigcup_{i=1}^{l} A_i.$$

Obviously,

$$k\left(\bigcap_{i=1}^{l} A_i\right) = 0 \leq (2l-1)\sum_{i=1}^{l} k(A_i),$$

if $\bigcap_{i=1}^{l} A_i = \emptyset$.
So in the following, let $\bigcap_{i=1}^{l} A_i \neq \emptyset$.
The first part of this proof follows the line of argument in the proof of Lemma 5.9: Since the sets $A_1, \ldots, A_l$ are convex and $A_{\mathrm{I},l} \neq \emptyset$, the set $A_{\mathrm{U},l}$ ist star-shaped. Hence, given a point in $p \in A_{\mathrm{I},l}$, for every $p' \in \bigcup_{i=1}^{l} \partial A_i$, the line segment $\overline{pp'}$ is completely contained in $A_{\mathrm{U},l}$. Also, for all $1 \leq i \leq l$, any ray with apex $p \in A_{\mathrm{I},l}$ intersects ∂A_i exactly once.
Let the endpoints of the circular arcs on ∂A_i in order be named $u_1^i, \ldots, u_{k(A_i)}^i$ for all $1 \leq i \leq l$. For a fixed $p \in \bigcap_{i=1}^{l} A_i$, we now divide the plane into $\sum_{i=1}^{l} k(A_i)$ sections by inserting the rays $[pu_j^i$ for all $1 \leq j \leq k(A_i)$ and $1 \leq i \leq l$. Each of the resulting regions, we call them *slices* for now, is bounded by two rays and contains exactly l circular arcs, one originating from each ∂A_i for $1 \leq i \leq l$.
Note that a ray can either be described by two points p (the apex) and a in the plane, which is denoted with $[pa$, or by a point p (the apex) in the plane and a angle $\alpha \in [0, 2\pi[$. For a simpler notation, in the following we will use the latter option. Let the ray with apex p and angle α be denoted with $r(\alpha)$ in the following. Note that $r(\alpha)$ depends on p, but since p is fixed, we refrain from including it in the notation. Let $r(\alpha_1)$ and $r(\alpha_2)$ with $\alpha_1, \alpha_2 \in [0, 2\pi[$ be two of the rays constructed above that frame the same region and let $\alpha_1 < \alpha_2$. We consider the collection $\mathcal{F} = \{f_1, \ldots, f_l\}$ of l functions:

$$\begin{aligned} f_i : \quad & [\alpha_1, \alpha_2] \to \mathbb{R}_0^+ \text{ with} \\ x \quad \mapsto \quad & \|p - (r(x) \cap \partial A_i)\| \end{aligned}$$

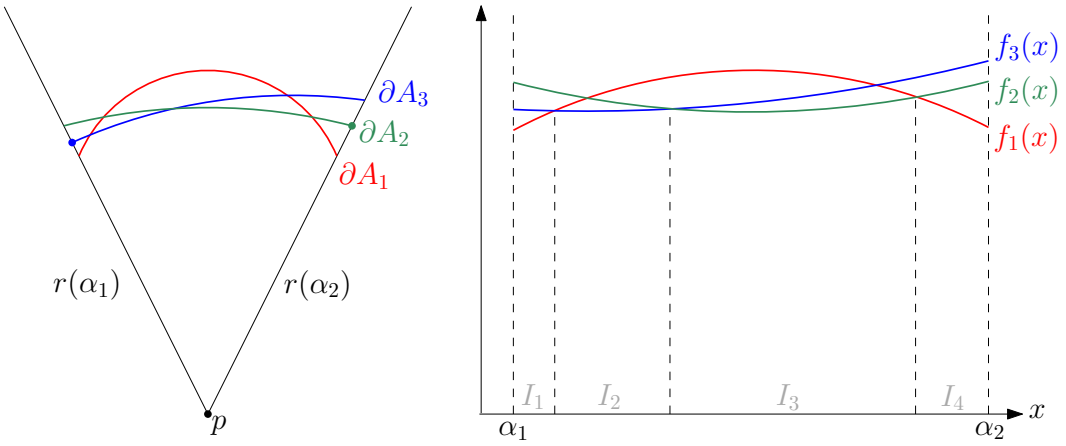

Figure 5.8: Left: The circular arcs of ∂A_i for $1 \leq i \leq 3$ within the region framed by the rays $r(\alpha_1)$ and $r(\alpha_2)$.
Right: The graphs of the corresponding collection $\mathcal{F} = \{f_1, f_2, f_3\}$ of functions together with the subintervals $I_1, \ldots, I_4$ that visualize the change points of $E_{\mathcal{F}}$. Here, $U = (1, 3, 2, 1)$.

for all $1 \leq i \leq l$. The function f_i describes the distance between p and ∂A_i subject to the angle $x \in [\alpha_1, \alpha_2]$. The *lower envelope* of $\mathcal{F}$ is defined as

$$E_{\mathcal{F}}(x) = \min_{1 \leq i \leq n} f_i(x)$$

for $x \in [\alpha_1, \alpha_2]$, and describes the part of $\partial A_{(\mathrm{I},l)}$ that is contained in the respective slice.
Let h be smallest number of subintervals $I_1, \ldots, I_h$ of $[\alpha_1, \alpha_2]$ so that for each $1 \leq s \leq h$ there is an index u_s with $E_{\mathcal{F}}(x) = f_{u_s}(x)$ for all $x \in I_s$. In other words, h is the number of circular arcs that constitute the part of $\partial A_{(\mathrm{I},l)}$ contained in the region at hand, see Figure 5.8.
The sequence of indices of functions $U := (u_1, \ldots, u_h)$ denotes the order in which the functions $f_1, \ldots, f_l$ that describe these circular arcs, appear on $E_{\mathcal{F}}$. Since two circular arcs intersect in at most two points, there are no 4 indices $1 \leq i_1 < i_2 < i_3 < i_4 \leq h$ such that $u_{i_1} = u_{i_3}$ and $u_{i_2} = u_{i_4}$ with $u_{i_1} \neq u_{i_2}$. Hence, U is an $(l, 2)$ Davenport-Schinzel sequence and according to Lemma 5.11

$$h \leq 2l - 1.$$

This means that the part of $\partial A_{\mathrm{I},l}$ contained in the region at hand consists of $2l - 1$ circular arcs at most. Since we divided the plane into $\sum_{i=1}^{l} k(A_i)$ such regions,

$$k\left(\bigcap_{i=1}^{l} A_i\right) \leq (2l - 1) \sum_{i=1}^{l} k(A_i).$$

□

The combinatorial complexity of the intersection of an admissible region with a unit-disk can be bounded as follows:

Lemma 5.13 *Let $r > 0$ and $c \in \mathbb{R}^2$. Let B be an arc-set where the radius of each arc is at least r. Then*

$$k\left(D_r(c) \cap B\right) \leq 2k(B).$$

Proof. Let a be a circular arc of ∂B. There are five possible ways for a to intersect with $\partial D_r(c)$, see Figure 5.9: They

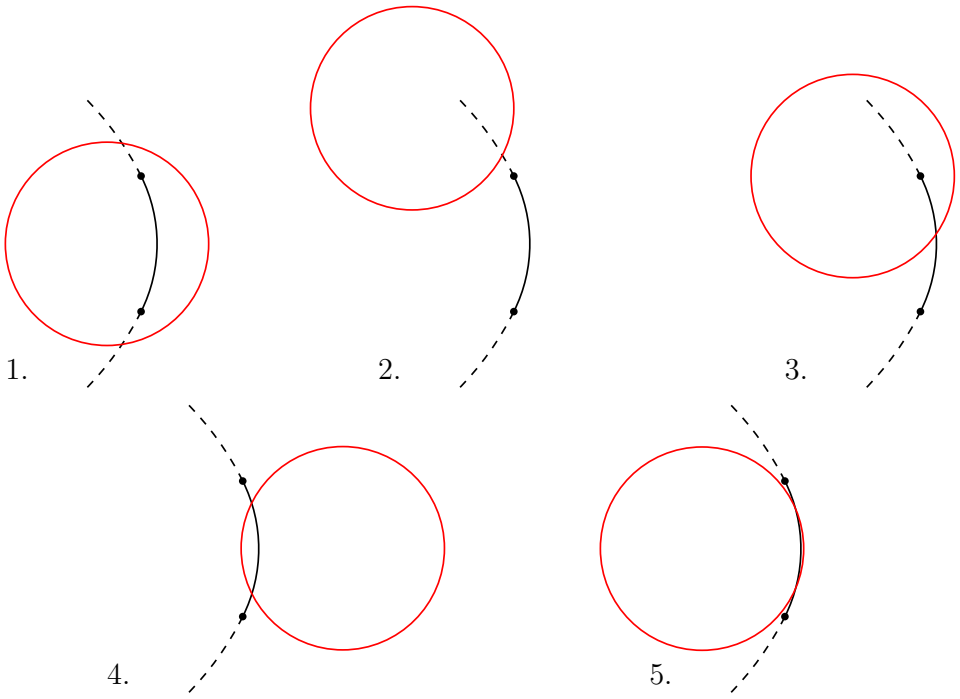

Figure 5.9: Five ways for a (solid circular arc) to interact with $\partial D_r(c)$ (solid circle).

1. do not intersect and a is completely contained in $D_r(c)$. Then, a is one of the circular arcs of $\partial(D_r(c) \cap B)$.
2. do not intersect and a lies completely outside $D_r(c)$. Then, a is not part of $\partial(D_r(c) \cap B)$.
3. intersect in one point. Then, both a part of a and a part of $D_r(c)$ are circular arcs of $\partial(D_r(c) \cap B)$.

4. intersect in two points e_1 and e_2 and the centers of a and $D_r(c)$ are located on different sides of the line $\overline{e_1e_2}$. Then, a part of a appears once on $\partial(D_r(c) \cap B)$ and $k\left(D_r(c) \cap B\right) = 2$.
5. intersect in two points e_1 and e_2 and the centers of a and $D_r(c)$ are both located on the same side of the straight line $\overline{e_1e_2}$. Then, a part of a appears once on $\partial(D_r(c) \cap B)$. Since the radius of a is at least r, a part of $D_r(c)$ appears to either side of the part of a on $\partial(D_r(c) \cap B)$.

Hence, every circular arc of ∂B can (partly) occur on $\partial(D_r(c) \cap B)$ at most once. And since B and $D_r(c) \cap B$ are both convex, the circular arcs composing ∂B appear (partly) on $\partial(D_r(c) \cap B)$ in the same order as they appear on ∂B. In the worst case, part of $\partial D_r(c)$ appear between every two consecutive arcs of ∂B on $\partial(D_r(c) \cap B)$ and thus

$$k\left(D_r(c) \cap B\right) \leq 2k(B).$$

□

Theorem 5.14 *Let $G = T_r$ be a tree rooted in r and let each vertex v have at most m children with $m \geq 2$. Algorithm 7 runs in $O\left((4m(2m-1))^n\, n \log n\right)$ time. If G is also complete, i.e., each vertex has either 0 or m children, the runtime improves to $O(8^{\log_m n} n^3 \log n)$.*

Proof. Let the root r have $n_r \leq m$ children $\mathfrak{c}_1(r), \ldots, \mathfrak{c}_{n_r}(r)$. Recall that $C = (c_1, \ldots, c_n)$ with $c_i = q_i - p_i$ for $1 \leq i \leq n$, see Equation (5.2). Let $c_r \in C$ be the point that corresponds to r and let $h \in \mathbb{N}$ be the height of T_r. Recall, that for any vertex $v \in V$ we denote the updated version of v with v'. Then, the following holds:

$$\begin{aligned}
k(I_{r'}) &= k\left(\left(\bigcap_{i=1}^{n_r} I^{\delta}_{\mathfrak{c}'_i(r)}\right) \cap D_\delta(c_r)\right) \\
&\overset{\text{Lemma 5.13}}{\leq} 2k\left(\bigcap_{i=1}^{n_r} I^{\delta}_{\mathfrak{c}'_i(r)}\right) \\
&\overset{\text{Lemma 5.12}}{\leq} 2(2n_r - 1)\sum_{i=1}^{n_r} k\left(I^{\delta}_{\mathfrak{c}'_i(r)}\right) \\
&\overset{(5.5)}{\leq} 4(2n_r - 1)\sum_{i=1}^{n_r} k\left(I_{\mathfrak{c}'_i(r)}\right) \\
&\leq 4(2m - 1)\sum_{i=1}^{n_r} k\left(I_{\mathfrak{c}'_i(r)}\right) \\
&\leq 4(2m-1)m \cdot \max_{1 \leq i \leq n_r} k\left(I_{\mathfrak{c}'_i(r)}\right).
\end{aligned}$$

Note that this estimation does not use the fact, that r is the root of T_r and thus holds for the admissible region $I_{v'}$ for any $v \in V$. Hence we can find an upper bound on the description complexity of $I_{v'}$ that depends on the description complexity of the admissible regions stored in the updated version of the children of v. So by starting the estimation at I'_r and repeating top-to-bottom through the tree T_r with height h until we reach the leaves, we get

$$k(I_{r'}) \leq 4(2m-1)m \cdot \max_{1 \leq i \leq n_r} k\left(I_{c'_i(r)}\right) \leq (4m(2m-1))^h .$$

Hence, the combinatorial complexity of $I_{r'}$ is $O\left((4m(2m-1))^n\right)$, and since there are $O(n)$ internal vertices in T_r Algorithm 7 runs in $O\left((4m(2m-1))^n\, n \log n\right)$ time.
If T_r is complete, $h = \log_m n$, and therefore

$$\begin{aligned} k(I_{r'}) &\leq (4m(2m-1))^{\log_m n} \\ &\leq 8^{\log_m n} m^{2^{\log_m n}} \\ &= 8^{\log_m n} n^2 \in O(n^5). \end{aligned}$$

Since there are $O(n)$ internal vertices in T_r and the description complexity of the object stored in each vertex is bounded from above by $O(8^{\log_m n} n^2)$, Algorithm 7 runs in $O(8^{\log_m n} n^3 \log n)$ time. □

This estimate of the description complexity of admissible regions that appear during a run of Algorithm 7 holds for EGSM problems under the Euclidean 1-to-1-distance. However, if the correspondence between the pattern and the model is not known, the combinatorial complexity of the respective admissible regions may be even bigger. At present, there are no results known to whether the problem under the directed L_2-Hausdorff distance for tree neighborhoods is solvable in polynomial time. This is one of the major reasons for designing reasonably fast approximation algorithms for the problems of this flavor.

Note that we already considered an approximation algorithm for EGSM problems under the directed L_2-Hausdorff distance for neighborhood graphs that are trees in Chapter 2: Every disk is approximated with a regular polygon with $O(\epsilon^{-1/2})$ vertices, which reduces the combinatorial complexity of the admissible regions, see Algorithm 2. If applied to Problem 8, where the correspondence between the pattern and the model is known, Algorithm 2 gives a $(1+\epsilon)$-approximation to the optimum of the objective function in $O(\epsilon^{-1/2} n(\log n + \log \epsilon^{-1}))$ time. Here, $(1+\epsilon)$-approximation means that if $\delta^{(\epsilon)}$ is the smallest value that permits a YES-instance for a given $\epsilon > 0$, the optimal

value δ^* is bounded by

$$\left(1 - \frac{1}{2}\epsilon\right)\delta^* \leq \delta^{(\epsilon)} \leq \left(1 + \frac{1}{2}\epsilon\right)\delta^*.$$

Another valid strategy to solve the problem approximately is covering every admissible region with a dense-enough grid of Algorithm 11, a $(1+\epsilon)$-approximation algorithm that works on sample-points and runs in $O(\epsilon^{-1}n)$ time and space is given in Appendix A.

On Problem Instances under Unknown Correspondence

So far, we focused on Problem 8, where the correspondence between the points of the pattern and the points of the model is fixed. This is the reason why the admissible regions at hand are convex at any point during Algorithm 7. If we look at the version of the problem, where the correspondence between the points of the pattern and the points of the model is not known, this property does not apply to the admissible regions anymore and estimating the combinatorial complexity of such sets becomes even more complicated:

Problem 9 *Given:*

$$\begin{aligned} P = \{p_1, \ldots, p_n\} \quad & \textit{a point set (the pattern),} \\ Q = \{q_1, \ldots, q_m\} \quad & \textit{a point set (the model), and} \\ G = (V, E) \quad & \textit{a cycle-free graph with } V = \{i \mid 1 \leq i \leq n\} \textit{ and} \\ & E \subseteq \{\{i, j\} |\ i, j \in V\}. \end{aligned}$$

Find: n translation vectors $T = (t_1, \ldots, t_n)$, so that the function

$$\gamma(P, Q, T, G) \coloneqq \max\left(\vec{h}(T(P), Q), \max_{\{i,j\} \in E} \|t_i - t_j\|\right)$$

is minimized.

Problem 9 is similar to Problem 8 apart from the fact that the distance between the transformed pattern and the model is the directed Hausdorff distance. As described in detail in Chapter 2 and Chapter 4, in the beginning of Algorithm 7, the admissible region I_v consists of m disks with radius δ for any vertex $v \in V$ and all such regions are translational copies of each other.
As a consequence, every admissible region I_v (after the initialization step) and every admissible region $I_{v'}$ (after updating v to v' due to an intersection operation) may consists of more than one connected component and some of them may merge during the inflation operation, which is why Observation 5.4

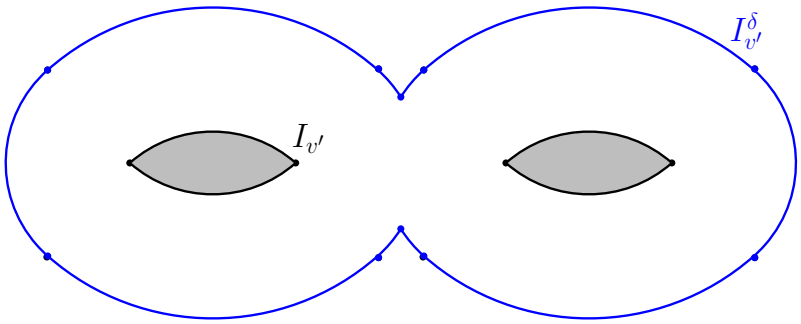

Figure 5.10: The set $I_{v'}$ the set $I^{\delta}_{v'}$ consist of one and two connected components respectively. Here, $k\left(I_{v'}\right) = 4$ and $k\left(I^{\delta}_{v'}\right) = 10$.

does not hold for this setting. A conceptional illustration of a counterexample, where

$$2k(I_v) < k(I^{\delta}_v)$$

is given in Figure 5.10.
Another key feature used in this chapter to estimate the complexity of the admissible regions at hand is displayed in Lemma 5.8: Given two convex sets A and C, so that their boundaries consists of a and c circular arcs, it follows that their intersection $A \cap C$ consists of $4(a + c)$ circular arcs at most. The constraint that A and C are convex implies that each of them consists of one connected component and that their boundaries each consist of one connected component. If the latter property is not given, i.e., A and C contain holes, $k(A \cap C)$ depends on the number of connected components of ∂A and ∂C, as well as the number of connected components of $A \cap C$, which seems hard to estimate.
Furthermore, if we consider two admissible regions that are not necessarily convex and may consist of several connected components, estimating the number of connected components of their intersection becomes a difficult task as well, see Figure 5.11 for a small schematic illustration.
The algorithms that (apporximately) solve EGSM problems of different graph classes and distance measures described in the previous chapters are all designed in order to avoid these difficulties while still computing suitable approximations in reasonable time.

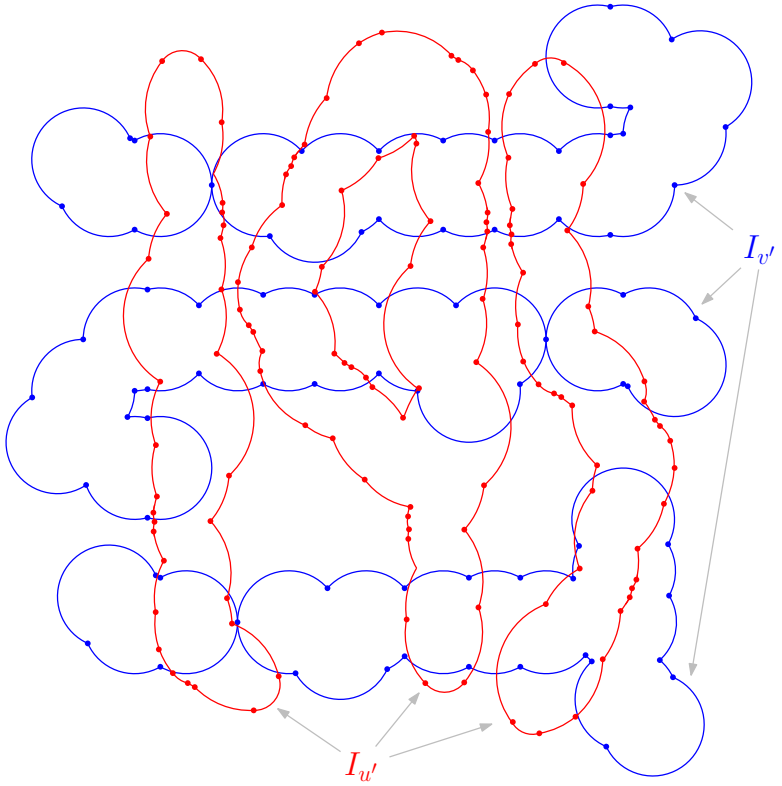

Figure 5.11: A schematic illustration of admissible regions: The set $I^{\delta}_{v'}$ and the set $I^{\delta}_{u'}$ each consist of three connected components, and $k\left(I^{\delta}_{v'} \cap I^{\delta}_{u'}\right) = 14$.

Chapter 6

Variants of the Problem

In the previous chapters, which are self-contained for the most part, we focused on one specific variant of Problem 3, an EGSM problem for point sets under translations in $\mathbb{R}^2$. However, as elaborately described in Section 1.4, there more generally stated Problem 2 offers much more options to choose from.
In this chapter, we consider different variants of Problem 2 that exceed Problem 3 in some aspects and give some insights about possible solution strategies. However, since the content of the sections in this chapter varies a lot, we refrain from describing the EGSM setup and already existing solution strategies in detail again and refer to the previous chapters for more information.
Here and in the following, $\|\cdot\|$ denotes the L_2-norm.

6.1 Weights within the Objective Function

A central topic in discussing EGSM problems is the tradeoff between minimizing the distance between the transformed pattern and the model and maximizing the similarity of neighbored transformations. In this thesis, both goals were seen as equally important and were therefore integrated in a common objective function with the same weight. Depending on the application, it may as well be reasonable to weigh them differently instead.
An example for a problem with interesting characteristics in this context is the following ESGM problem under translations and the injective bottleneck distance:

Problem 10 *Given:*

$$\begin{aligned} P = (p_1, \ldots, p_n) &\quad \textit{a sequence of points (the pattern),} \\ Q = (q_1, \ldots, q_m) &\quad \textit{a sequence of points (the model), and} \\ \delta_o, \delta_t \in \mathbb{R}^+ &\quad \textit{two parameters.} \end{aligned}$$

Find:
A sequence $T = (t_1, \ldots, t_n)$ *of translations with*

$$\mathrm{b}(T(P), Q) \leq \delta_o \text{ and} \tag{6.1}$$

$$\max\left(\max_{1 \leq i < n} \left(\|t_i - t_{i+1}\|\right), \|t_n - t_1\|\right) \leq \delta_t. \tag{6.2}$$

Note that the neighborhood graph is a simple cycle and implicitly given in Constraint (6.2). In Problem 10, the maximum distance between the transformed pattern and the model in object space, δ_o, and the maximum distance between neighbored transformations in transformation space, δ_t, can be chosen independently from each other.
Problem 10 is closely related to the problem of finding Hamiltonian cycles in grid graphs: A grid graph is a finite, node-induced graph whose vertex set consists of points in the plane with integer coordinates, in which two vertices are connected iff the L_2-distance between them is 1.

Problem 11 (Finding Hamiltonian Cycles in Grid Graphs) *Given:*

$$\begin{aligned} G_g = (V_g, E_g) \quad & \textit{a grid graph with} \\ n \quad & \textit{vertices.} \end{aligned}$$

Find: A sequence of n *vertices* $V^* = (v_1^*, \ldots, v_n^*)$ *with*

$$\begin{aligned} v_i^* \in V_g \textit{ for } 1 \leq i \leq n, \\ v_i^* \neq v_j^* \textit{ for } 1 \leq i < j \leq n, \\ \|v_i^* - v_{i+1}^*\| = 1 \textit{ for } 1 \leq i < n \textit{ and} \\ \|v_n^* - v_1^*\| = 1. \end{aligned} \tag{6.3}$$

In other words, the goal is to find a simple cycle within the graph G_g that visits every vertex exactly once. This Problem was proven to be NP-complete in [28].

Theorem 6.1 *Problem 10 is NP-hard.*

Proof. The following proof is a reduction from the NP-complete Problem 11 of finding Hamiltonian cycles in grid graphs.
Let $G_g = (V_g, E_g)$ be a grid graph with n vertices $v_1, \ldots, v_n \in V_g$, i.e., $v_i = (v_i.x, v_i.y) \in \mathbb{Z}^2$ for all $1 \leq i \leq n$. The goal is to decide, whether there is a sequence of n vertices V^* that satisfies Constraints (6.3). From G_g, we now construct an EGSM instance where the pattern P as well as the model Q consist of n points, i.e., $m = n$ in the notation of Problem 10. We set the

points of $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ as well as the parameters δ_o and δ_t as follows:

$$\begin{aligned} \delta_o &:= \frac{1}{\sqrt{2}} - \frac{1}{2} - \epsilon, \\ \delta_t &:= 1, \\ p_i &:= (0,0) \text{ and} \\ q_i &:= -v_i \end{aligned}$$

for some $\epsilon > 0$ and all $1 \leq i \leq n$. In the following, we describe why deciding whether there is a sequence of n vertices V^* that satisfies Constraints (6.3) is the same as deciding whether there is a sequence of translations T that satisfies Contraints (6.1) and (6.2) for the given EGSM instance.
A sequence of translations T that satisfies Contraints (6.1) and (6.2) for the given EGSM instance has the property that every translation in T moves the corresponding point of P at least δ_o-close to some point of Q. For a specific point $p \in P$, the set of such translations is defined as

$$I_p := \bigcup_{q \in Q} \{t \in T \mid \|p + t - q\| \leq \delta_o\} = \bigcup_{q \in Q} D_{\delta_o}(p - q),$$

where $D_{\delta_o}(p - q)$ is the disk with radius δ_o and center $(p - q)$. And since P contains n copies of the same point $(0, 0)$, all I_p together form an arrangement of n^2 disks: For every $v \in V_g$, we have n disks with radius δ_o centered in $q = -v$, and we call every such set of disks a *stack* of disks:
For all $1 \leq i \leq n$ and one fixed $q = -v$, all

$$I_{p_i,q} := \{t \in T \mid \|p_i + t - q\| \leq \delta_o\} = D_{\delta_o}(p_i - q) = D_{\delta_o}(-q) = D_{\delta_o}(v)$$

form one stack of disks.
The centers of the stacks are the vertices of G_g. By construction, there is an edge $e \in E_g$ between two vertices of V_g iff the vertices (respectively the centers of two stacks of disks) have distance 1, see Figure 6.1. On the relation between the instances: As the injective bottleneck distance is used to measure the distance between the pattern and the model, all $p \in P$ are forced to be mapped to pairwise different $q \in Q$. This implies that every translation $t \in T$ needs to be picked from a different stack of disks. The distance between two translations in T that correspond to adjacent vertices in the neighborhood graph is at most $\delta_t = 1$, which is why neighbored translations have to be picked from horizontally or vertically neighbored stacks on the integer grid. This equals adjacent vertices in the grid graph. Note that the distance between

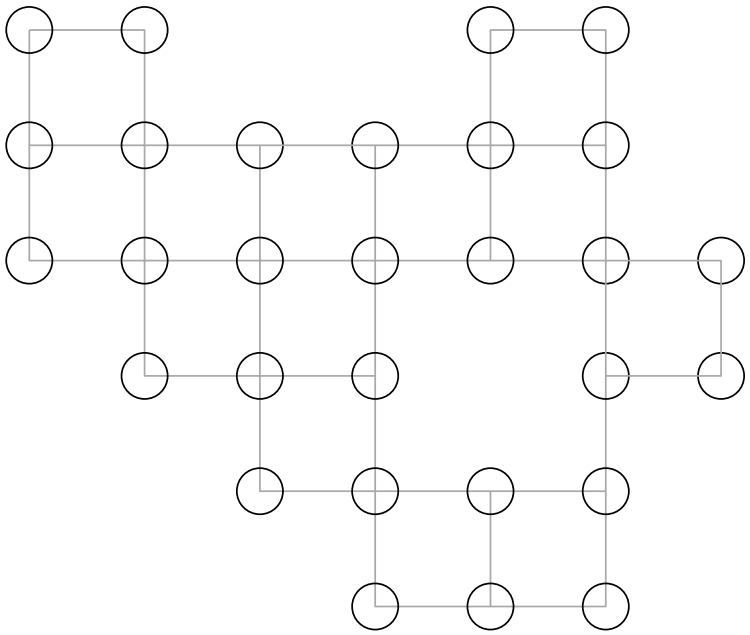

Figure 6.1: Illustration of a grid graph G_g together with stacks of disks with radius δ_o centered in the vertices of G_g.

diagonally neighbored stacks is

$$\sqrt{2} - 2\delta_o = \sqrt{2} - 2\left(\frac{1}{\sqrt{2}} - \frac{1}{2} - \epsilon\right) = 1 + 2\epsilon > 1,$$

see Figure 6.2.
Since the neighborhood graph is a simple cycle, finding a sequence of translations T that solves Problem 10 includes finding a tour through the grid graph where n vertices have to be visited and no vertex is allowed to be visited more than once, i.e., finding a Hamiltonian cycle in the constructed grid graph. □

6.2 Thoughts on Rigid Motions

Recall that $\mathcal{T}$ encodes the class of transformations that match the partitions of P to the model Q. In [16], the authors introduced the first algorithms that solve some EGSM variants, along with an NP-hardness proof for others. In their paper, they mostly considered translations in one direction.
In this thesis, $\mathcal{T}$ was the class of translations in $\mathbb{R}^2$. Measuring the similarity of two translations in translation space according to a suitable norm, e.g., the L_2-norm, the L_∞-norm or some polygonal norm, is a promising approach because the geometric interpretation is clear.

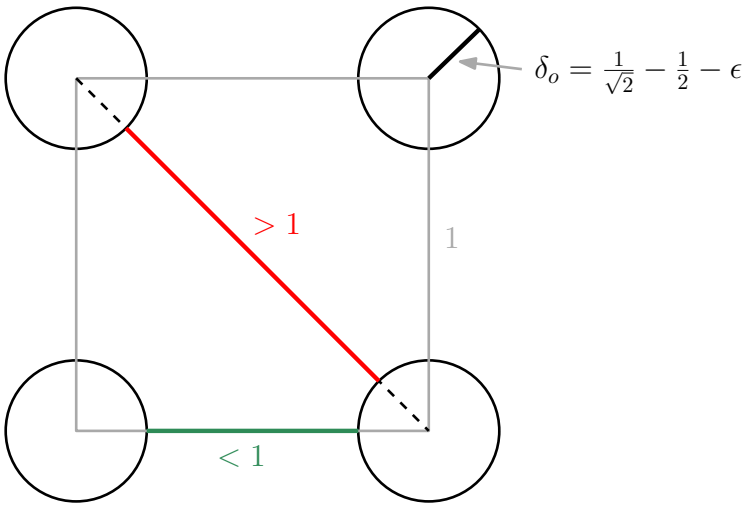

Figure 6.2: Four stacks of disks with radius δ_o on the integer grid.

However, choosing $\mathcal{T}$ to be, e.g., rigid motions makes the choice of a suitable similarity measure more complicated, as the geometric interpretation of the well-established matrix norms, such as the spectral norm is not trivial in this context. A short insight about the difficulty in finding a suitable norm is given in the following.

A Distance Measure for Rigid Motions

In this section, everything is stated in $\mathbb{R}^2$. We represent any point p as vector $p = (p.x, p.y)^{\mathrm{T}}$. A rigid motion $T = (t, \alpha)$ consists of a translation vector t as well as a rotation angle $\alpha \in [0, 2\pi[$ and transforms any point p as follows:

$$T(p) := \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} p + t.$$

Given two rigid motions $T_1 = (t_1, \alpha_1)$ and $T_2 = (t_2, \alpha_2)$, we want to determine their distance in transformation space using a distance measure that allows for a plausible geometric interpretation. Since T_1 and T_2 consist of two parts, the translation vector and the rotation angle, it seems natural to match them separately and then somehow add up the results using suitable weights for each summand. In the following definition, we introduce a possible choice for such a distance measure:

Definition 6.2 *The* rigid motion distance *of two rigid motions* $T_1 = (t_1, \alpha_1)$

and $T_2 = (t_2, \alpha_2)$ *is defined as*

$$\mathrm{d_{RM}}(T_1, T_2) \coloneqq \|t_1 - t_2\| + \frac{4}{\pi} \min\left(|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\right),$$

where $\|\cdot\|$ *denotes the* L_2*-distance.*

Lemma 6.3 *The rigid motion distance is a metric.*

Proof. Let $T_i = (t_i, \alpha_i)$ be a rigid motion for $1 \leq i \leq 3$. The rigid motion distance meets all metric conditions:

1. Non-negativity:

$$\mathrm{d_{RM}}(T_1, T_2) = \|t_1 - t_2\| + \frac{4}{\pi} \min\left(|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\right) \geq 0.$$

2. Identity of indiscernibles:

$$\begin{array}{lll} & \mathrm{d_{RM}}(T_1, T_2) = 0 & \\ \Leftrightarrow & \|t_1 - t_2\| = 0 & \text{and} \quad \min\left(|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\right) = 0 \\ \Leftrightarrow & t_1 = t_2 & \text{and} \quad \alpha_1 = \alpha_2. \end{array}$$

3. Symmetry:

$$\begin{aligned} \mathrm{d_{RM}}(T_1, T_2) &= \|t_1 - t_2\| + \frac{4}{\pi} \min\left(|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\right) \\ &= \mathrm{d_{RM}}(T_2, T_1). \end{aligned}$$

4. Triangle Inequality:
   First, we proof the following statement:

$$\begin{aligned} \min\left(|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\right) + \min\left(|\alpha_2 - \alpha_3|, 2\pi - |\alpha_2 - \alpha_3|\right) \quad (6.4) \\ \geq \min\left(|\alpha_1 - \alpha_3|, 2\pi - |\alpha_1 - \alpha_3|\right). \quad (6.5) \end{aligned}$$

   We proof the upper statement by examining the following cases:

   - $\alpha_1 < \alpha_2 < \alpha_3$ (the case $\alpha_3 < \alpha_2 < \alpha_1$ is equivalent to this case due to symmetry):
     - $|\alpha_1 - \alpha_3| \leq \pi$:
       It follows, that $|\alpha_1 - \alpha_2| \leq \pi$ and $|\alpha_2 - \alpha_3| \leq \pi$ and thus

$$(6.4) \geq (6.5) \Leftrightarrow |\alpha_1 - \alpha_2| + |\alpha_2 - \alpha_3| \geq |\alpha_1 - \alpha_3| \Leftrightarrow 0 \geq 0.$$

- $|\alpha_1 - \alpha_3| > \pi$ and $|\alpha_1 - \alpha_2| > \pi$:
  It follows, that $|\alpha_2 - \alpha_3| \leq \pi$ and thus

$$\begin{aligned}(6.4) \geq (6.5) &\Leftrightarrow 2\pi - |\alpha_1 - \alpha_2| + |\alpha_2 - \alpha_3| \geq 2\pi - |\alpha_1 - \alpha_3| \\ &\Leftrightarrow \alpha_1 - \alpha_2 + \alpha_3 - \alpha_2 \geq \alpha_1 - \alpha_3 \\ &\Leftrightarrow \alpha_3 \geq \alpha_2.\end{aligned}$$

- $|\alpha_1 - \alpha_3| > \pi$, $|\alpha_1 - \alpha_2| \leq \pi$ and $|\alpha_2 - \alpha_3| \leq \pi$:

$$\begin{aligned}(6.4) \geq (6.5) &\Leftrightarrow |\alpha_1 - \alpha_2| + |\alpha_2 - \alpha_3| \geq 2\pi - |\alpha_1 - \alpha_3| \\ &\Leftrightarrow \alpha_2 - \alpha_1 + \alpha_3 - \alpha_2 \geq 2\pi - \alpha_3 + \alpha_1 \\ &\Leftrightarrow \alpha_3 - \alpha_1 \geq \pi.\end{aligned}$$

- $|\alpha_1 - \alpha_3| > \pi$, $|\alpha_1 - \alpha_2| \leq \pi$ and $|\alpha_2 - \alpha_3| > \pi$:

$$\begin{aligned}(6.4) \geq (6.5) &\Leftrightarrow |\alpha_1 - \alpha_2| + 2\pi - |\alpha_2 - \alpha_3| \geq 2\pi - |\alpha_1 - \alpha_3| \\ &\Leftrightarrow \alpha_2 - \alpha_1 - \alpha_3 + \alpha_2 \geq \alpha_1 - \alpha_3 \\ &\Leftrightarrow \alpha_2 \geq \alpha_1.\end{aligned}$$

- $\alpha_1 < \alpha_3 < \alpha_2$ (the case $\alpha_3 < \alpha_1 < \alpha_2$ is equivalent to this case due to symmetry):
  - $|\alpha_1 - \alpha_2| \leq \pi$:
    It follows, that $|\alpha_1 - \alpha_3| \leq \pi$ and $|\alpha_2 - \alpha_3| \leq \pi$ and thus

$$(6.4) \geq (6.5) \Leftrightarrow |\alpha_1 - \alpha_2| + |\alpha_2 - \alpha_3| \geq |\alpha_1 - \alpha_3| \Leftrightarrow 0 \geq 0.$$

  - $|\alpha_1 - \alpha_2| > \pi$ and $|\alpha_1 - \alpha_3| > \pi$:
    It follows, that $|\alpha_2 - \alpha_3| \leq \pi$ and thus

$$\begin{aligned}(6.4) \geq (6.5) &\Leftrightarrow 2\pi - |\alpha_1 - \alpha_2| + |\alpha_2 - \alpha_3| \geq 2\pi - |\alpha_1 - \alpha_3| \\ &\Leftrightarrow \alpha_1 - \alpha_2 + \alpha_2 - \alpha_3 \geq \alpha_1 - \alpha_3 \\ &\Leftrightarrow 0 \geq 0.\end{aligned}$$

  - $|\alpha_1 - \alpha_2| > \pi$, $|\alpha_1 - \alpha_3| \leq \pi$ and $|\alpha_2 - \alpha_3| \leq \pi$:

$$\begin{aligned}(6.4) \geq (6.5) &\Leftrightarrow 2\pi - |\alpha_1 - \alpha_2| + |\alpha_2 - \alpha_3| \geq |\alpha_1 - \alpha_3| \\ &\Leftrightarrow 2\pi - \alpha_2 + \alpha_1 + \alpha_2 - \alpha_3 \geq \alpha_3 - \alpha_1 \\ &\Leftrightarrow \pi \geq \alpha_3 - \alpha_1.\end{aligned}$$

  - $|\alpha_1 - \alpha_2| > \pi$, $|\alpha_1 - \alpha_3| \leq \pi$ and $|\alpha_2 - \alpha_3| > \pi$:

$$\begin{aligned}(6.4) \geq (6.5) &\Leftrightarrow 2\pi - |\alpha_1 - \alpha_2| + 2\pi - |\alpha_2 - \alpha_3| \geq |\alpha_1 - \alpha_3| \\ &\Leftrightarrow 4\pi - \alpha_2 + \alpha_1 - \alpha_2 + \alpha_3 \geq \alpha_3 - \alpha_1 \\ &\Leftrightarrow 2\pi \geq \alpha_2 - \alpha_1.\end{aligned}$$

- $\alpha_2 < \alpha_1 < \alpha_3$ (the case $\alpha_2 < \alpha_3 < \alpha_1$ is equivalent to this case due to symmetry):
  - $|\alpha_2 - \alpha_3| \leq \pi$:
    It follows, that $|\alpha_1 - \alpha_2| \leq \pi$ and $|\alpha_1 - \alpha_3| \leq \pi$ and thus

    $$(6.4) \geq (6.5) \Leftrightarrow |\alpha_1 - \alpha_2| + |\alpha_2 - \alpha_3| \geq |\alpha_1 - \alpha_3| \Leftrightarrow 0 \geq 0.$$

  - $|\alpha_2 - \alpha_3| > \pi$ and $|\alpha_1 - \alpha_3| > \pi$:
    It follows, that $|\alpha_1 - \alpha_2| \leq \pi$ and thus

    $$\begin{aligned} (6.4) \geq (6.5) &\Leftrightarrow |\alpha_1 - \alpha_2| + 2\pi - |\alpha_2 - \alpha_3| \geq 2\pi - |\alpha_1 - \alpha_3| \\ &\Leftrightarrow \alpha_1 - \alpha_2 - \alpha_3 + \alpha_2 \geq \alpha_1 - \alpha_3 \\ &\Leftrightarrow 0 \geq 0. \end{aligned}$$

  - $|\alpha_2 - \alpha_3| > \pi$, $|\alpha_1 - \alpha_3| \leq \pi$ and $|\alpha_1 - \alpha_2| \leq \pi$:

    $$\begin{aligned} (6.4) \geq (6.5) &\Leftrightarrow |\alpha_1 - \alpha_2| + 2\pi - |\alpha_2 - \alpha_3| \geq |\alpha_1 - \alpha_3| \\ &\Leftrightarrow \alpha_1 - \alpha_2 + 2\pi - \alpha_3 + \alpha_2 \geq \alpha_3 - \alpha_1 \\ &\Leftrightarrow \pi \geq \alpha_3 - \alpha_1 \end{aligned}$$

  - $|\alpha_2 - \alpha_3| > \pi$, $|\alpha_1 - \alpha_3| \leq \pi$ and $|\alpha_1 - \alpha_2| > \pi$:

    $$\begin{aligned} (6.4) \geq (6.5) &\Leftrightarrow 2\pi - |\alpha_1 - \alpha_2| + 2\pi - |\alpha_2 - \alpha_3| \geq |\alpha_1 - \alpha_3| \\ &\Leftrightarrow 4\pi - \alpha_1 + \alpha_2 - \alpha_3 + \alpha_2 \geq \alpha_3 - \alpha_1 \\ &\Leftrightarrow 2\pi \geq \alpha_3 - \alpha_2. \end{aligned}$$

With this, the correctness of $(6.4) \geq (6.5)$ is proven and we conclude the following:

$$\begin{aligned} & \mathrm{d}_{\mathrm{RM}}(T_1, T_2) + \mathrm{d}_{\mathrm{RM}}(T_2, T_3) \\ = \quad & \|t_1 - t_2\| + \|t_2 - t_3\| \\ & + \frac{4}{\pi} \min\left(|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\right) \\ & + \frac{4}{\pi} \min\left(|\alpha_2 - \alpha_3|, 2\pi - |\alpha_2 - \alpha_3|\right) \\ \geq \quad & \|t_1 - t_3\| + \frac{4}{\pi} \min\left(|\alpha_1 - \alpha_3|, 2\pi - |\alpha_1 - \alpha_3|\right) \\ = \quad & \mathrm{d}_{\mathrm{RM}}(T_1, T_3). \end{aligned}$$

□

Recall that $D_r(c)$ denotes the disk with center $c \in \mathbb{R}^2$ and radius $r > 0$ and $\partial D_r(c)$ denotes its boundary. The unit disk with center in the origin is denoted with D.
We introduce the following EGSM problem under rigid motions and the rigid motion distance:

Problem 12 *Given:*

$$\begin{aligned} P = \{p_1, \ldots, p_n\} \quad & \textit{a point set (the pattern) with} \\ P_1, \ldots, P_k \quad & \textit{a partition of } P, \\ Q = \{q_1, \ldots, q_m\} \quad & \textit{a point set (the model) with} \\ Q_1, \ldots, Q_k \quad & \textit{a partition of } Q, \textit{ and} \\ G = (V, E) \quad & \textit{an undirected graph with } V = \{i \mid 1 \leq i \leq k\} \textit{ and} \\ & E \subseteq \{\{i, j\} \mid i, j \in V\}. \end{aligned}$$

Find: k rigid motions $\mathsf{T} = \{T_1, \ldots, T_k\}$, so that

$$\gamma(P, Q, \mathsf{T}, G) = \max\left(\max_{\{i,j\} \in E} \left(\mathrm{d_{RM}}(T_i, T_j)\right), \max_{1 \leq i \leq k} \vec{h}(T_i(P_i), Q_i)\right)$$

is minimized.

Note that the correspondence between the partitions of P and the partitions of Q is fixed and the distance between every transformed partition $T_i(P_i)$ of the pattern and the corresponding partition Q_i of the model is measured with the L_2-Hausdorff distance for $1 \leq i \leq k$.
One more difficulty in designing reasonable EGSM problem formulations is the non-trivial tradeoff between minimizing the difference between neighboring rigid motions, i.e., rigid motions encoded as adjacent vertices in the neighborhood graph G in transformation space and minimizing the distance between the transformed pattern $\mathsf{T}(P)$ and the model Q in object space.

Assumptions on the Pattern and the Model

The following ideas are stated under the assumption that the smallest enclosing disks of the point sets P and Q are centered in the origin and the smallest enclosing disk with the greatest diameter is the unit disk in $\mathbb{R}^2$. If these assumptions do not hold, it is possible to scale and translate P and Q so that their transformed versions meet the assumptions made above.

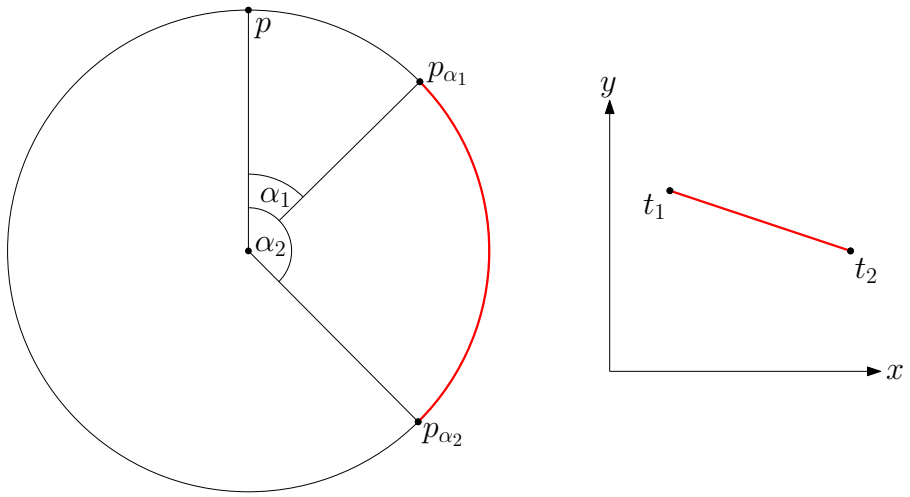

Figure 6.3: The rigid motions $T_1 = (t_1, \alpha_1) = \left(\left(\frac{1}{4}, \frac{3}{4}\right)^{\mathrm{T}}, \frac{\pi}{4}\right)$ and $T_2 = (t_2, \alpha_2) = \left(\left(1, \frac{1}{2}\right)^{\mathrm{T}}, \frac{3\pi}{4}\right)$ and the geometric interpretation of $|\alpha_1 - \alpha_2|$ (red arc on the left) and $\|t_1 - t_2\|$ (red line segment on the right).
Their rigid motion distance is $\mathrm{d}_{\mathrm{RM}}(T_1, T_2) = \|t_1 - t_2\| + \frac{4}{\pi}|\alpha_1 - \alpha_2| = \frac{\sqrt{10}}{4} + 2$.

Rigid Motions and their Effect on the Pattern

There are different ways of measuring the similarity of rigid motions. In particular, one can choose any matrix norm, although the geometric interpretation of most of them is not obvious. As one of the tasks in EGSM is mapping neighboring subpatterns of P similarly in terms of geometry, a distance measure for rigid motions with a suitable geometric interpretation is needed. The rigid motion distance meets this condition well:
Let $T_1 = (t_1, \alpha_1)$ and $T_2 = (t_2, \alpha_2)$ be two rigid motions. The similarity of the translations at hand is measured with the L_2-distance, which is the same distance measure we sucessfully used in most of the EGSM variants under translations. Let p be a point on ∂D and p_{α_1} and p_{α_2} be the images of p rotated by α_1 and α_2 respectively. The points p_{α_1} and p_{α_2} split δD into two arcs, see Figure 6.3. We interpret the difference in the rotation angles as the minimum of the lengths of these arcs on ∂D, which is why we choose $\mathrm{d}_{\mathrm{RM}}(T_1, T_2)$ as

$$\begin{aligned} \mathrm{d}_{\mathrm{RM}}(T_1, T_2) &= \|t_1 - t_2\| + \frac{4}{\pi} \min\left(|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\right) \\ &\leq \|t_1 - t_2\| + 4. \end{aligned}$$

Since P and Q both are point sets within the unit disk, and $P_1, \ldots, P_k$ are to be matched to $Q_1, \ldots, Q_k$, $\|t_1\| \leq 2$ and $\|t_2\| \leq 2$ and thus $\|t_1 - t_2\| \leq 4$. Therefore

the distance between the translations at hand and the distance between the rotation angles at hand contribute a similar share to the rigid motion distance. Of course, the summands of d_{RM} can also be weighted differently if needed.

Although d_{RM} seems suitable for the EGSM framework because of the plausible geometric interpretation, the approach turns out to be insufficient: It is not enough for any two neighbored rigid motions to be somehow similar. It is crucial to also transform neighboring subpatterns of P similarly. However, depending on the size and the location of these subpatterns (with respect to the origin), two rigid motions with a great distance in transformation space may generate two similar pictures of the same subpattern in object space.

Example 6.4 *In the following EGSM instance, the input sets P and Q are divided into subpatterns. Let*

$$\begin{aligned}
P_1 = \{p_1, p_2\} \text{ with} \quad & p_1 = (-1, 0)^{\mathrm{T}}, \\
& p_2 = (1, 0)^{\mathrm{T}}, \\
P_2 = \{p_3, p_4\} \text{ with} \quad & p_3 = (0, 0)^{\mathrm{T}}, \\
& p_4 = (\epsilon, 0)^{\mathrm{T}}, \\
Q_1 = \{q_1, q_2\} \text{ with} \quad & q_1 = (-1, 0)^{\mathrm{T}}, \\
& q_2 = (1, 0)^{\mathrm{T}}, \\
Q_2 = \{q_3, q_4\} \text{ with} \quad & q_3 = (0, 0)^{\mathrm{T}}, \\
& q_4 = (0, \epsilon)^{\mathrm{T}}.
\end{aligned}$$

for some $\epsilon > 0$, see Figure 6.4, and let the neighborhood graph $G = (V, E)$ with $V = \{1, 2\}$ and $E = \{\{1, 2\}\}$, a path with two vertices. In the following, the translation $t = (0, 0)^{\mathrm{T}}$ is denoted with id. *We consider the rigid motions*

$$\begin{aligned}
T_1 &= (\mathrm{id}, 0), \\
T_2 &= \left(\mathrm{id}, \frac{\pi}{2}\right) \text{ and} \\
T_2' &= (\mathrm{id}, 0).
\end{aligned}$$

Let $\mathsf{T} \coloneqq (T_1, T_2)$ and $\mathsf{T}' \coloneqq (T_1, T_2')$.
Note that the sequence T matches the pattern precisely to the model, hence

$$\begin{aligned}
\gamma(P, Q, \mathsf{T}, G) &= \max\left(\mathrm{d}_{\mathrm{RM}}(T_1, T_2), \vec{h}(T_1(P_1), Q_1), \vec{h}(T_2(P_2), Q_2)\right) \\
&= \max(2, 0, 0) = 2.
\end{aligned}$$

Applying T' to the object function results in

$$\begin{aligned}
\gamma(P, Q, \mathsf{T}', G) &= \max\left(\mathrm{d}_{\mathrm{RM}}(T_1, T_2'), \vec{h}(T_1(P_1), Q_1), \vec{h}(T_2'(P_2), Q_2)\right) \\
&= \max\left(0, 0, \sqrt{2}\epsilon\right) = \sqrt{2}\epsilon.
\end{aligned}$$

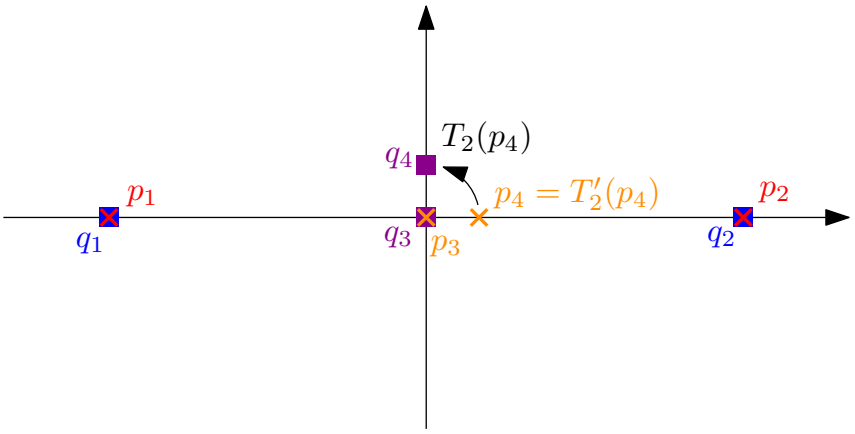

Figure 6.4: The sets $P = T'(P)$, $T_2(p_4)$ and Q.

In Example 6.4, the point p_4 is very close to the origin, hence the impact of any rigid motion $T = (\mathrm{id}, \alpha)$ on the image of p_4 is small in the sense that $T(p_4)$ is close to p_4. The rigid motions T_2 and T_2' have a relatively great distance in transformation space due to the different rotation angles, but the impact of this difference on P_2 is just $O(\epsilon)$. Also, it is not clear if simply requiring approximately equal-sized subpatterns of P and Q is enough in order to avoid problems like displayed in Example 6.4. However, this facts suggest that in order to design a suitable distance measure for rigid motions in the EGSM context it is not enough to use a uniform measure on rigid motions but a distance measure that also somehow takes other features of the problem instance, e.g., the geometric structure of the pattern and the model, into account.
Analyzing the neccessary requirements on P, Q and G, e.g., approximately equal-sized subpatterns of P and Q, so that using $\mathrm{d_{RM}}$ as the distance measure in transformation space is a sufficient choice in the EGSM context is a promising topic of future research.

6.3 On Imprecise Point Sets

Irrespective of the choice of the class $\mathcal{S}$ of geometric objects of the input, the imprecision of the input due to, e.g., inaccuracies in the acquisition, is a prominent issue in theoretical computer science. Suppose that the pattern P and the model Q are point sets in the plane. There is a popular model of imprecise point sets, in which an imprecise point in the plane can be described as a disk $D_r(c)$ with radius $r > 0$ and center $c \in \mathbb{R}^2$. The actual position of the imprecise point can be anywhere in $D_r(c)$ and the probability of all

possible positions depends on a suitable distribution function, e.g., the uniform distribution. Computing distance measures of imprecise point sets seems to be an ambitious task:
In [15], the authors consider different cases of the problem of computing the directed Hausdorff distance of two point sets P and Q if at least one of the sets consists of imprecise points. The authors prove that for some cases of their problem setup the problem is NP-hard and present an algorithm with a polynomial runtime for others.
These ideas can also be incorporated in the EGSM setting, where the input sets P and Q may be imprecise due to the way they were acquired.

6.4 Line Segments, Triangles and Triangulated Surfaces

In the previous chapters, we chose P and Q to be point sets in the plane and all strategies were based to this choice. A possible next step is to consider line segments instead of point sets with the option to generalize the results of the EGSM setting for line segments to triangles or even triangulated surfaces. Intuitively, all strategies considered so far may still work if P and Q are sets of line segments instead of point sets, even though slight adjustments have to be made. For instance, the boundaries of regions of admissible translations in the L_2-setting, that are elaborately discussed in Chapter 5, then consist of circular arcs and line segments, which can be interpreted as circular arcs with infinite radius.

The introduction of new geometric input objects also allows for shifting the focus to other distance measures in object space. A prominent example for measuring the distance between line segments or polygonal curves in general that also provides a plausible geometric interpretation is the *discrete Fréchet distance (DFD)* introduced in [29]. Imagine a dog and its owner taking a walk. Given two finite curves in a metric space, the owner walks on one of the curves and their dog walks on the other one while they are connected by a leash. Both can vary their speed but cannot move backwards. The *Fréchet distance* of the two curves equals the length of the shortest possible leash sufficient for both the owner and their dog to traverse the curves from start to finish. In the discrete case, we are given two polygonal curves defined by the point sequences $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_m)$ for $n, m \in \mathbb{N}$ and the dog and its owner use the points in P and Q as stepping stones from start to finish in the given order, see [30] for a detailed introduction.

Let P and Q be point sequences in the plane. In this section, we denote the L_p-norm of a vector x by $\|x\|_p$ for $p \in \{1, 2, \infty\}$.

Definition 6.5 *A* coupling K *for* $n, m \in \mathbb{N}$ *(i.e., for* P *and* Q*) is a sequence of* $L \in \mathbb{N}$ *ordered pairs of indices* $K = ((\alpha_1, \beta_1), \ldots, (\alpha_L, \beta_L))$ *so that*

- $1 \leq \alpha_i \leq n$ *and* $1 \leq \beta_i \leq m$ *for all* $1 \leq i \leq L$,
- $(\alpha_1, \beta_1) = (1, 1)$ *and* $(\alpha_L, \beta_L) = (n, m)$, *and*
- *for every* $1 \leq k < L$ *the following holds:*

$$(\alpha_{k+1}, \beta_{k+1}) \in \{(\alpha_k + 1, \beta_k), (\alpha_k, \beta_k + 1), (\alpha_k + 1, \beta_k + 1)\}.$$

Let $\mathcal{K}(P, Q)$ *be the set of all couplings of* P *and* Q*. The* discrete Fréchet distance (DFD) *of the point sequences* P *and* Q *is defined as*

$$\mathrm{F}(P, Q) \coloneqq \min_{K \in \mathcal{K}(P,Q)} \max_{(\alpha,\beta) \in K} \|p_\alpha - q_\beta\|_p.$$

A coupling $K^* \in \mathcal{K}(P, Q)$ *is called* optimal *(for* P *and* Q*) if*

$$F(P, Q) = \max_{(\alpha,\beta) \in K^*} \|p_\alpha - q_\beta\|_p.$$

See Figure 6.5 for an illustration.
Note that strictly speaking, the DFD is defined for point sequences. Since we use point sequences to describe polygonal curves, we also use the term *polygonal curves* instead of *point sequences* in the context of the DFD.
In the context of shape matching under translations, we seek to find a translation t so that $\mathrm{F}(t(P), Q) \leq \delta$ for a parameter $\delta \geq 0$. This problem has already been studied along with several variants in, e.g., [30] and [31].
The variant of this problem we consider in the following is the *partial discrete Fréchet matching problem*. In order to formulate this problem, we first need to introduce some notation:

Notation 6.6 *For a point sequence* $Q = (q_1, \ldots, q_m)$ *of* m *points in the plane, let*

$$Q_{s,f} \coloneqq (q_s, \ldots, q_f) \subseteq Q$$

with $1 \leq s \leq f \leq m$ *denote the subsequence of* Q *from* q_s *to* q_f.

Now we consider the following problem:

Problem 13 (Discrete Partial Fréchet Matching under translations) *Given:*

$$\begin{aligned} P = (p_1, \ldots, p_n) &\quad \textit{a sequence of points,} \\ Q = (q_1, \ldots, q_m) &\quad \textit{a sequence of points, and} \\ \delta \geq 0 &\quad \textit{a parameter.} \end{aligned}$$

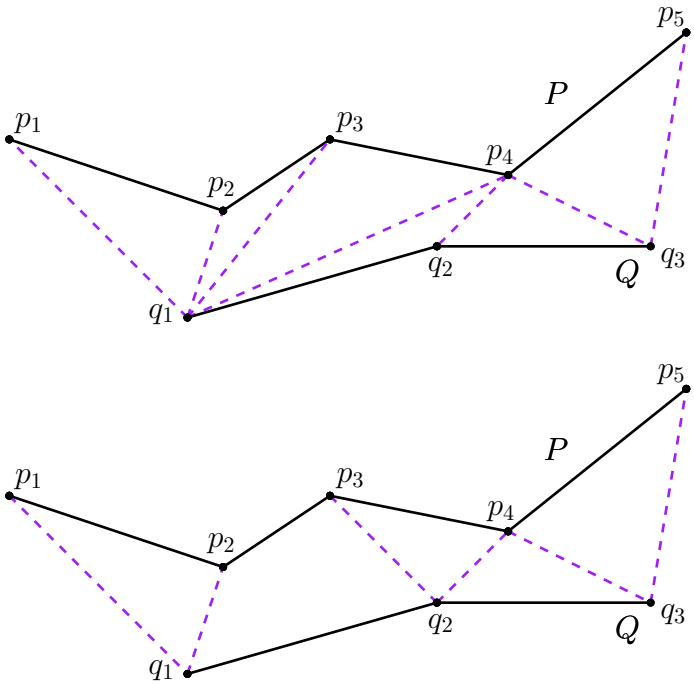

Figure 6.5: Two couplings (dashed lines) of the point sequences P and Q (solid lines). The coupling displayed in the lower illustration is optimal.

Find: a translation t so that there is a subsequence $Q_{s,f} \subseteq Q$ with $1 \leq s \leq f \leq m$ so that

$$\mathrm{F}(t(P), Q_{s,f}) \leq \delta.$$

See Figure 6.6 for an illustration. Note that in Problem 13, it is also possible for P to be matched to a single point of Q.

We can now state the following EGSM problem:

Problem 14 *Given:*

$P = \{P^1, \ldots, P^k\} \subset \mathbb{R}^2$	*a set of k sequences of points (the pattern), with*
$P^i = (p_1^i, \ldots, p_{r_i}^i)$	*a sequence of points of length $r_i \geq 1$*
	for all $1 \leq i \leq k$ and $n = \sum_{i=1}^{k} r_i$,
$Q = (q_1, \ldots, q_m) \subset \mathbb{R}^2$	*a sequence of points (the model), and*
$\delta \geq 0$	*a parameter.*

Find: A sequence of k translations $T = (t_1, \ldots, t_k)$ so that there are k subsequences $Q_{s_i,f_i} \subseteq Q$ for $1 \leq i \leq k$ with $f_i \leq s_{i+1}$ for all $1 \leq i < k$ so

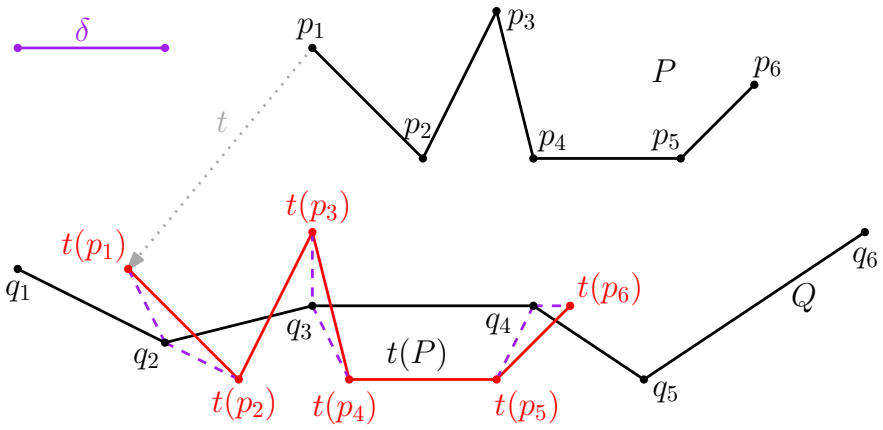

Figure 6.6: The point sequence $t(P)$ and the subsequence $Q_{2,4}$ of the point sequence Q (solid lines) with $\mathrm{F}(t(P), Q_{2,4}) \leq \delta$.

that

$$\max_{1 \leq i \leq k} \mathrm{F}\left(t_i(P^i), Q_{s_i,f_i}\right) \leq \delta.$$

See Figure 6.7 for an illustration.
In this problem formulation, the similarity constraints on the translations are not given as edges in a neighborhood graph. Instead, the given order in which the partitions of P have to be matched to Q provides a (in the context of measuring the DFD under translation) natural kind of similarity constraints on the translations. Later in this section, we will also discuss a variant of Problem 14 where the similarity constraints on the translations are encoded in a given neighborhood graph.

Definition 6.7 *We say that a sequence of translations $T^* = (t_1^*, \ldots, t_k^*)$ solves Problem 14, iff there are k subsequences $Q_{s_i,f_i} \subseteq Q$ for $1 \leq i \leq k$ with $f_i \leq s_{i+1}$ for all $1 \leq i < k$ so that*

$$\max_{1 \leq i \leq k} \mathrm{F}\left(t_i^*(P^i), Q_{s_i,f_i}\right) \leq \delta.$$

Elastic Partial Discrete Fréchet Matching for Polygonal Curves that are Line Segments

For now, we consider instances where $P^l = (p_1^l, p_2^l)$ for all $1 \leq l \leq k$, i.e., the pattern P consists of k polygonal curves that equal line segments. Then, the following holds:

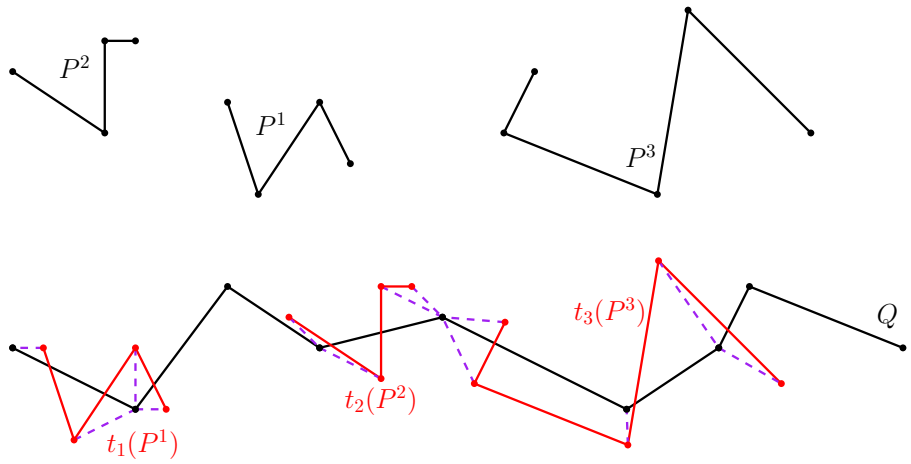

Figure 6.7: The point sequences P^1, P^2 and P^3 are matched to the point sequence Q with the sequence of translations $T = (t_1, t_2, t_3)$ in order.

Lemma 6.8 *Let $P = (p_1, p_2)$ and $Q = (q_1, \ldots, q_m)$ be two sequences of points in the plane. Then, there is an index $1 \leq j \leq m$ so that*

$$F(P, Q) \geq F(P, Q_{j,j})$$

or there is an index $1 \leq j < m$ so that

$$F(P, Q) \geq F(P, Q_{j,j+1}).$$

Proof. Let $K^* \in \mathcal{K}(P, Q)$ be an optimal coupling for P and Q and let the length of $K^* = ((\alpha_1^*, \beta_1^*), \ldots, (\alpha_L^*, \beta_L^*))$ be $L \geq 1$. Then, as illustrated in Figure 6.8, one of the following cases occurs:

- There is an index $1 \leq j \leq m$ so that $(\alpha_l^*, \beta_l^*) = (1, j)$ and $(\alpha_{l+1}^*, \beta_{l+1}^*) = (2, j)$ for an $1 \leq l < L$. Since

  $$F(P, Q_{j,j}) = \max\left(\|p_1 - q_j\|_p, \|p_2 - q_j\|_p\right),$$

  it follows that $F(P, Q) \geq F(P, Q_{j,j})$.
- There is an index $1 \leq j < m$ so that $(\alpha_l^*, \beta_l^*) = (1, j)$ and $(\alpha_{l+1}^*, \beta_{l+1}^*) = (2, j+1)$ for an $1 \leq l < L$. Since

  $$F(P, Q_{j,j+1}) = \max\left(\|p_1 - q_j\|_p, \|p_2 - q_{j+1}\|_p\right),$$

  it follows that $F(P, Q) \geq F(P, Q_{j,j+1})$.

□

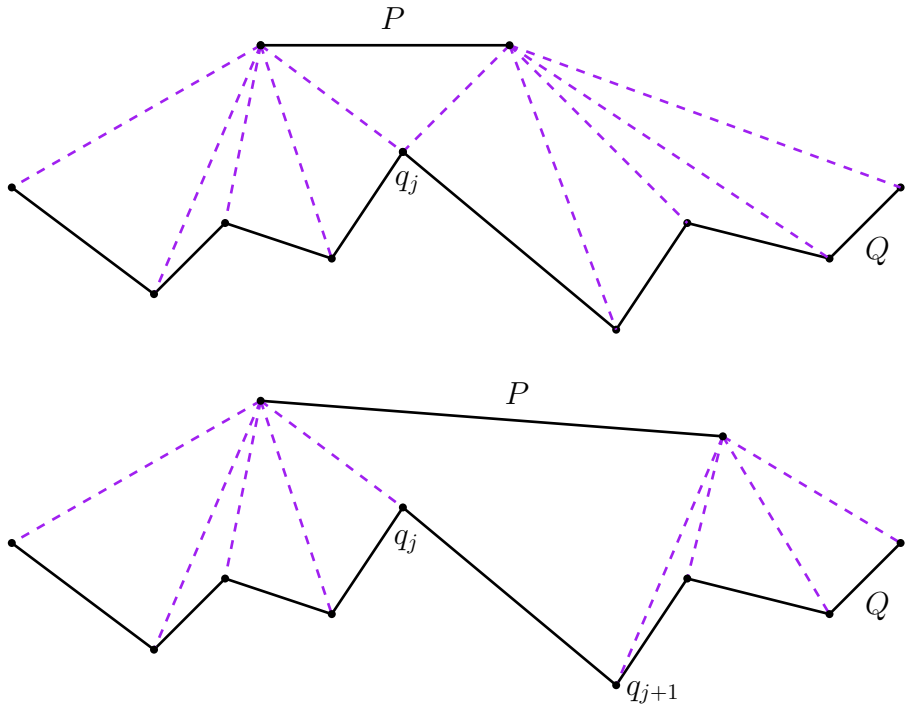

Figure 6.8: First: Two polygonal curves P and Q and an index $1 \leq j \leq m$ so that $F(P,Q) \geq F(P,Q_{j,j})$.
Second: Two polygonal curves P and Q and an index $1 \leq j < m$ so that $F(P,Q) \geq F(P,Q_{j,j+1})$.

As a consequence, if we want to decide if $F(P,Q) \leq \delta$ for a $\delta \geq 0$, one neccessary condition is that there is an index $1 \leq j \leq m$ so that $F(P,Q_{j,j}) \leq \delta$ or an index $1 \leq j < m$ so that $F(P,Q_{j,j+1}) \leq \delta$. The same holds, if we are given $t(P)$ instread of P for a translation t. Hence, finding a translation t so that the partial DFD of $t(P)$ and Q is at most δ is equal to finding a translation t so that the DFD of $t(P)$ and a point or a line segment of the polygonal curve defined by Q is at most δ.

Let $t_1, t_2 \in \mathbb{R}^2$ be the translations that shift p_1 exactly to q_1 and p_2 to q_2 respectively, i.e., $t_1 := (p_1 - q_1)$ and $t_2 := (p_2 - q_2)$. By looking at their distance in translation space, we can determine if there is a translation t that shifts both p_1 and p_2 δ-close to their corresponding point in Q. The same holds if p_1 and p_2 are matched to the same point q_1.

Lemma 6.9 *Let $P = (p_1, p_2)$ and $Q = (q_1, q_2)$ be two point sequences in the plane. Then, there is a translation t so that*

- $\mathrm{F}(t(P), Q) \leq \delta$ *iff*

$$\|(p_1 - q_1) - (p_2 - q_2)\|_p \leq 2\delta.$$

- $\mathrm{F}(t(P), (q_1)) \leq \delta$ *iff*

$$\|p_1 - p_2\|_p \leq 2\delta.$$

Proof. • Let t be a fixed translation. We know that $\mathrm{F}(t(P), Q) = \mathrm{F}((t(p_1), t(p_2)), Q)$. There are three couplings of $t(P)$ and Q:

$$\begin{aligned} K_1 &= ((1,1),(1,2),(2,2)), \\ K_2 &= ((1,1),(2,1),(2,2)) \text{ and} \\ K_3 &= ((1,1),(2,2)), \end{aligned}$$

hence $\mathcal{K}(t(P), Q) = \{K_1, K_2, K_3\}$. Since

$$\begin{aligned} &\max(\|t(p_1) - q_1\|_p, \|t(p_2) - q_2\|_p) \\ &\leq \max(\|t(p_1) - q_1\|_p, \|t(p_1) - q_2\|_p, \|t(p_2) - q_2\|_p) \text{ and} \\ &\max(\|t(p_1) - q_1\|_p, \|t(p_2) - q_2\|_p) \\ &\leq \max(\|t(p_1) - q_1\|_p, \|t(p_2) - q_1\|_p, \|t(p_2) - q_2\|_p), \end{aligned}$$

K_3 is an optimal coupling for $t(P)$ and Q, hence

$$F(t(P), Q) = \max_{(\alpha,\beta)\in K_3} \|t(p_\alpha) - q_\beta\|_p = \max(\|t(p_1) - q_1\|_p, \|t(p_2) - q_2\|_p).$$

With this, we get

$$\begin{aligned} F(t(P), Q) \leq \delta &\Leftrightarrow \|t(p_1) - q_1\|_p \leq \delta \text{ and } \|t(p_2) - q_2\|_p \leq \delta \\ &\Leftrightarrow \|t + (p_1 - q_1)\|_p \leq \delta \text{ and } \|t + (p_2 - q_2)\|_p \leq \delta \\ &\Leftrightarrow t \in D_\delta(p_1 - q_1) \cap D_\delta(p_2 - q_2), \end{aligned}$$

where $D_r(c)$ denotes the L_p-disk with radius $r > 0$ and center $c \in \mathbb{R}^2$. Hence there is a translation t so that $F(t(P), Q) \leq \delta$ iff

$$\begin{aligned} & D_\delta(p_1 - q_1) \cap D_\delta(p_2 - q_2) \neq \emptyset \\ \Leftrightarrow \quad & \|(p_1 - q_1) - (p_2 - q_2)\|_p \leq 2\delta. \end{aligned}$$

- Let t be a fixed translation. We know that $\mathrm{F}(t(P), Q) = \mathrm{F}((t(p_1), t(p_2)), (q_1))$. There is one coupling of $t(P)$ and Q:

$$K = ((1,1),(2,1)),$$

hence $\mathcal{K}(t(P), Q) = \{K\}$ and

$$\begin{aligned} F(t(P), Q) \leq \delta &\Leftrightarrow \max(\|t(p_1) - q_1\|_p, \|t(p_2) - q_1\|_p) \leq \delta \\ &\Leftrightarrow \|t(p_1) - q_1\|_p \leq \delta \text{ and } \|t(p_2) - q_1\|_p \leq \delta \\ &\Leftrightarrow \|t + (p_1 - q_1)\|_p \leq \delta \text{ and } \|t + (p_2 - q_1)\|_p \leq \delta \\ &\Leftrightarrow t \in D_\delta(p_1 - q_1) \cap D_\delta(p_2 - q_1), \end{aligned}$$

Hence there is a translation t so that $F(t(P), (q_1)) \leq \delta$ iff

$$\begin{aligned} & D_\delta(p_1 - q_1) \cap D_\delta(p_2 - q_1) \neq \emptyset \\ \Leftrightarrow \quad & \|(p_1 - q_1) - (p_2 - q_1)\|_p \leq 2\delta \\ \Leftrightarrow \quad & \|(p_1 - p_2)\|_p \leq 2\delta. \end{aligned}$$

□

The problem of deciding whether there is a translation that brings a line segment P^l δ-close to Q in terms of the partial DFD can be converted into the problem of finding a path with minimum edge weight in a directed weighted graph:

Definition 6.10 *Let $P_1, \ldots, P_k$ be k point sequences of size 2 and let Q be a sequence of m points. For every $1 \leq l \leq k$, we define the directed weighted graph $G_{P^l,Q}$ as follows:*

$$\begin{aligned} G_{P^l,Q} &= (V_{P^l,Q}, E_{P^l,Q}) \quad \textit{with} \\ V_{P^l,Q} &\coloneqq \{v^l_{i,j} \mid 1 \leq i \leq 2, 1 \leq j \leq m\} \textit{ and} \\ E_{P^l,Q} &\coloneqq E^h_{P^l,Q} \cup E^v_{P^l,Q} \cup E^d_{P^l,Q} \end{aligned}$$

with

$$\begin{aligned} E^d_{P^l,Q} &\coloneqq \{(v^l_{1,j}, v^l_{2,j+1}) \mid 1 \leq j \leq m-1\}, \\ E^v_{P^l,Q} &\coloneqq \{(v^l_{1,j}, v^l_{2,j}) \mid 1 \leq j \leq m\}, \textit{ and} \\ E^h_{P^l,Q} &\coloneqq \{(v^l_{2,j}, v^l_{2,j+1}) \mid 1 \leq j \leq m-1\}. \end{aligned}$$

Let

$$w : E_{P^l,Q} \to \mathbb{R}^+_0$$

be the weight function on the edges of $E_{P^l,Q}$. We define

$$w\left((v^l_{i,j}, v^l_{i',j'})\right) \coloneqq \begin{cases} \|(p^l_i - q_j) - (p^l_{i'} - q_{j'})\|_p & \textit{if } (v^l_{i,j}, v^l_{i',j'}) \in \left(E^d_{P^l,Q} \cup E^v_{P^l,Q}\right) \\ 0 & \textit{if } (v^l_{i,j}, v^l_{i',j'}) \in E^h_{P^l,Q} \end{cases}$$

for $1 \leq i, i' \leq 2$ and $1 \leq j, j' \leq m$.

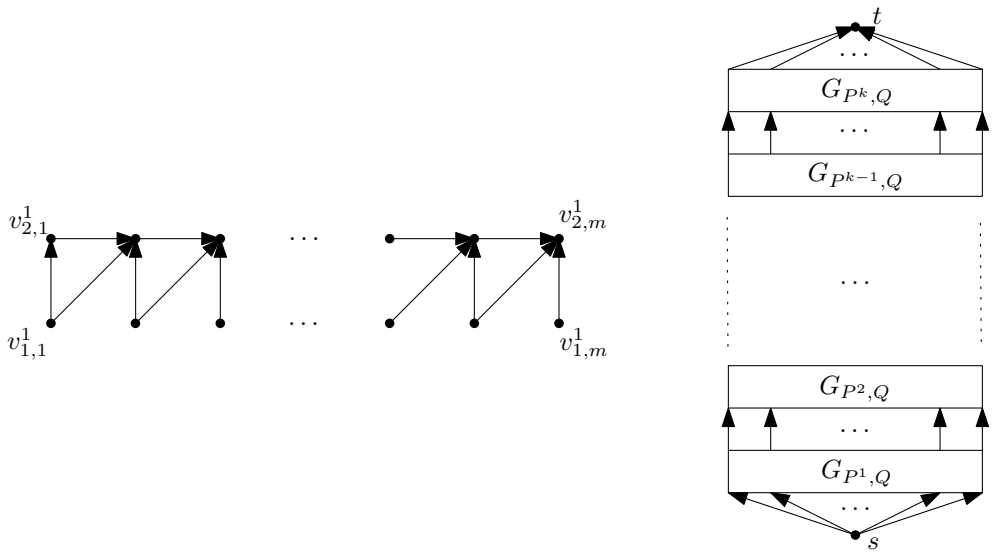

Figure 6.9: Left: An illustration of the directed graph $G_{P^1,Q}$.
Right: An illustration of the directed graph $G_{P,Q}$.

The left part of Figure 6.9 provides a schematical illustration.

For every $1 \leq l \leq k$, the graph $G_{P^l,Q}$ consists of $2m$ vertices where every vertex $v^l_{i,j}$ corresponds to a pair of points $p^l_i \in P^l$ and $q_j \in Q$ with $1 \leq i \leq 2$ and $1 \leq j \leq m$. All possible matchings are encoded in the edges of $G_{P^l,Q}$. If p^l_1 is matched to q_j, p^l_2 has to be matched to either q_j or q_{j+1}, see Lemma 6.8. Hence there is an edges from $v^l_{1,j}$ to $v^l_{2,j}$ for all $1 \leq j \leq m$ and to $v^l_{2,j+1}$ for all $1 \leq j < m$. Also there are edges from $v^l_{2,j}$ to $v^l_{2,j+1}$ for $1 \leq j < m$, which have no significance now but will be needed later.
The vertex $v^l_{i,j}$ corresponds to the pair of points p^l_i and q_j. However, it can also be associated with a point $(p^l_i - q_j)$ in translation space, i.e., the translation that shifts p^l_i to q_j. An edge $e \in (E^d_{P^l,Q} \cup E^v_{P^l,Q})$ can be interpreted as the L_p-difference between two such translations. Therefore, this difference is assigned as the weight on the corresponding edges.

Definition 6.11 *Let $\pi \coloneqq (v_1, \ldots, v_r)$ be a path in an edge-weighted graph $G = (V, E)$ with $v_i \in V$ for $1 \leq i \leq r$. The* weight *of a path $\pi = (v_1, \ldots, v_r)$ is defined as*

$$\mathrm{mW}(\pi) \coloneqq \max_{1 \leq i < r} w\left((v_i, v_{i+1})\right),$$

i.e., the weight of the heaviest edge of π.

Corollary 6.12 *There is a translation t and a subsequence $Q_{j,j'}$ of Q with*

$1 \leq j \leq j' \leq m$ *and* $j \leq j' \leq j+1$ *so that*

$$\mathrm{F}(t(P^l), Q_{j,j'}) \leq \delta,$$

iff there is a path π *in* $G_{P^l,Q}$ *from* $v^l_{1,j}$ *to* $v^l_{2,j'}$ *so that*

$$\mathrm{mW}(\pi) \leq 2\delta.$$

Proof. The correctness of the statement follows directly from Lemma 6.8 and Lemma 6.9. □

Corollary 6.13 *There is a sequence of translations that solves Problem 14 iff there are* k *paths* π_l *for* $1 \leq l \leq k$ *with the following property:* π_l *is a path in* $G_{P^l,Q}$ *from* v^l_{1,j_l} *to* v^l_{2,j'_l} *with* $1 \leq j_l \leq j'_l \leq m$ *and* $j_l \leq j'_l \leq j_l + 1$ *so that*

$$\mathrm{mW}(\pi_l) \leq 2\delta$$

and $j'_l \leq j_{l+1}$ *for all* $1 \leq l < k$.

Proof. The correctness of the statement follows directly from Lemma 6.8 and Corollary 6.12. □

Now we appropriately combine the graphs $G_{P^l,Q}$ in order to decide Problem 14 by including a source s and a sink t to the set of vertices and by adding edges that link consecutive subgraphs $G(P^l, Q)$:

Definition 6.14 *Let* $G_{P,Q} = (V_{P,Q}, E_{P,Q})$ *be a directed weighted graph with*

$$\begin{aligned} V_{P,Q} &\coloneqq \{s\} \cup \{t\} \cup \left(\bigcup_{l=1}^{k} V_{P^l,Q} \right) \text{ and} \\ E_{P,Q} &\coloneqq \left(\bigcup_{l=1}^{k} E_{P^l,Q} \right) \cup E^s_{P,Q} \cup E^t_{P,Q} \cup E^c_{P,Q} \end{aligned}$$

with

$$\begin{aligned} E^s_{P,Q} &\coloneqq \{(s, v^1_{1,j}) \mid 1 \leq j \leq m\}, \\ E^t_{P,Q} &\coloneqq \{(v^k_{2,j}, t) \mid 1 \leq j \leq m\}, \\ E^c_{P,Q} &\coloneqq \{(v^l_{2,j}, v^{l+1}_{1,j}) \mid 1 \leq l \leq k-1, 1 \leq j \leq m\}, \end{aligned}$$

and $w(e) \coloneqq 0$ *for all edges* $e \in \left(E^s_{P,Q} \cup E^t_{P,Q} \cup E^c_{P,Q}\right)$.

See the right side of Figure 6.9 for a schematical illustration of $G_{P,Q}$.

Theorem 6.15 *There is a sequence of translations that solves Problem 14, iff there is a (directed) path π in $G_{P,Q}$ from s to t so that* $\mathrm{mW}(\pi) \leq 2\delta$.

Proof. Concerning $G_{P,Q}$ we make the following observations:

- Every edge $(u, v) \in E_{P,Q}$ is directed. Let

$$u \in V_{P,Q} \setminus \left\{ \{s\} \cup \{t\} \cup \{v_{2,j}^k \mid 1 \leq j \leq m\} \right\}.$$

  Then $u = v_{i,j}^l$ for an $1 \leq l \leq k$, $1 \leq i \leq 2$ and $1 \leq j \leq m$ and $v = v_{i',j'}^{l'}$ with $l' \geq l$, $i' \geq i$ and $j' \geq j$. All other edges are of the form $(s, v_{1,j}^1)$ or $(v_{2,j}^k, t)$ for $1 \leq j \leq m$. As a consequence, there is no path from s to t that visits a vertex twice, i.e., the graph is cycle-free.
- Every path π in G from s to t contains exactly one edge from $E_{P^l,Q}^d \cup E_{P^l,Q}^v$ for each $1 \leq l \leq k$ and they appear in increasing order with l along π, since the subgraphs $G_{P^l,Q}$ are combined consecutively by the edges of $E_{P,Q}^c$. All other edges of the path have weight 0.
- There is a path from $v_{2,j}^l \in V_{P^l,Q}$ to $v_{1,j'}^{l+1} \in V_{P^{l+1},Q}$ for all $1 \leq l \leq k-1$ and $1 \leq j \leq j' \leq m$.

Recall that the edges in $E_{P^l,Q}^d \cup E_{P^l,Q}^v$ encode all possible assignments between P^l and Q. As a consequence, the set of paths from s to t encodes the set of all possible valid couplings of P and Q.
According to Corollary 6.13, there is a sequence of translations that solves Problem 14 iff there are k paths π_l for $1 \leq l \leq k$ so that π_l is a path in $G_{P^l,Q}$ from v_{1,j_l}^l to $v_{2,j_l'}^l$ with $1 \leq j_l \leq j_l' \leq m$ and $j_l \leq j_l' \leq j_l + 1$ with $\mathrm{mW}(\pi_l) \leq 2\delta$ and $j_l' \leq j_{l+1}$ for all $1 \leq l < k$. Suppose, $\pi_1, \ldots, \pi_k$ with this property exist. Since the set of paths from s to t encodes the set of all possible valid couplings of P and Q, there is a path from s to t that contains the paths $\pi_1, \ldots, \pi_k$ as subpaths. It follows that

$$\mathrm{mW}(\pi) = \max_{1 \leq l \leq k} \mathrm{mW}(\pi_l),$$

since all edges in $E_{P,Q} \setminus (E_{P^l,Q}^d \cup E_{P^l,Q}^v)$ have weight 0. On the other hand, if there is an index $1 \leq l \leq k$ so that $\mathrm{mW}(\pi_l') > 2\delta$ for all paths π_l' in $G_{P^l,Q}$ it follows that $\mathrm{mW}(\pi) > 2\delta$ for all paths π from s to t. □

We now state the following algorithm:

Algorithm 8 *We are given a set of k sequences of points $P = \{P^1, \ldots, P^k\}$ with $P^i = (p_1^i, p_2^i)$ for all $1 \leq i \leq k$, a sequence of points $Q = (q_1, \ldots, q_m)$ and a parameter $\delta \geq 0$.*

First, we construct the directed weighted graph $G_{P,Q} = (V_{P,Q}, E_{P,Q})$ from P and Q as elaborately described above. Then, by using the depth-first-search variant for finding shortest paths in DAGs, see [32], we test, if there is a path for $G_{P,Q}$ from s to t.
One of the following cases occurs:

1. *A path from s to t is found, and* YES *is returned as the answer to Problem 14 for a pattern that consists of line segments. A witness T can be computed from the information encoded in the path at hand.*
2. *There is no path from s to t and* NO *is returned as the answer to Problem 14 for a pattern that consists of line segments.*

Theorem 6.16 *Problem 14 can be decided in $O(nm)$ time if the pattern consists of line segments, also when returning a witness $T = (t_1, \ldots, t_k)$.*

Proof. We use Algorithm 8 to decide Problem 14 for a pattern that consists of line segments.
The construction of $G_{P,Q}$ takes $O(nm)$ time, since $|V_{P,Q}| = 2km$ and $n = 2k$ and

$$\begin{aligned}
|E_{P,Q}| &= \left| \left(\bigcup_{l=1}^{k} E_{P^l,Q} \right) \cup E^s_{P,Q} \cup E^t_{P,Q} \cup E^m_{P,Q} \right| \\
&= \left(\sum_{l=1}^{k} |E^h_{P^l,Q}| + |E^v_{P^l,Q}| + |E^d_{P^l,Q}| \right) + |E^s_{P,Q}| + |E^t_{P,Q}| + |E^m_{P,Q}| \\
&= k((m-1) + m + (m-1)) + m + m + (k-1)m \\
&= (4k+1)m - 2k \\
&= (2n+1)m - n \in O(nm)
\end{aligned}$$

and all operations during the construction of the graph can be carried out in constant time.
Let π be a path in $G_{P,Q}$ from s to t. Obviously, π contains exactly one edge from $E^d_{P^l,Q} \cup E^v_{P^l,Q}$ for each $1 \leq l \leq k$, at most $m-1$ edges from $\bigcup_{l=1}^{k} E^h_{P^l,Q}$, exactly $k-1$ edges from $E^c_{P,Q}$ and exactly one edge from $E^s_{P,Q}$ and $E^t_{P,Q}$ each. Hence, π contains at most $2k + m = n + m$ edges in total.
The question if there is a path π from s to t with $\mathrm{mW}(\pi) \leq 2\delta$ can be answered by using the depth-first-search variant for finding shortest paths in DAGs in $O(nm)$ time, see [32]. Let π' be a path in $G_{P,Q}$ from s to t with $\mathrm{mW}(\pi') \leq 2\delta$. Moreover, let $(v'^l_{1,j}, v'^l_{2,j'})$ with $1 \leq j \leq j' \leq m$ and $j \leq j' \leq j+1$ be the edge from $E^d_{P^l,Q} \cup E^v_{P^l,Q}$ in π' for all $1 \leq l \leq k$. We define

$$t_l = \|(p^l_1 - q_j) - (p^l_2 - q_{j'})\|_p$$

for all $1 \leq l \leq k$. Then $T = (t_1, \ldots, t_k)$ can be returned as a witness without increasing the runtime of Algorithm 8. □

In the optimization version of Problem 14, the goal is to find the optimal (smallest) value δ^*, so that there is a sequence of translations that meets the constraints of Problem 14 for this specific $\delta = \delta^*$.

Theorem 6.17 *The optimization version of Problem 14 can be solved in $O(nm)$ time if the pattern consists of line segments, also when returning a witness $T = (t_1, \ldots, t_k)$.*

Proof. We adjust Algorithm 8 slightly: The depth-first-search variant for finding shortest paths in DAGs, see [32], can be applied on $G_{P,Q}$ in order to find the path π^* with minimal weight in $O(nm)$ time, since $G_{P,Q}$ is a DAG. Then, $\mathrm{mW}(\pi^*) = \delta^*$ can be returned as the answer to the optimization version of Problem 14. □

Note that the strategy introduced above does not depend on the L_p-norm at hand for $p \in \{1, 2, \infty\}$. In fact, the strategy works for other norms as well.

Elastic Partial Discrete Fréchet Matching for Polygonal Curves

In this section, the subpatterns $P^1, \ldots, P^k$ encode polygonal curves that may consist of more than one line segment each. In this setting, the strategy introduced in the previous section (transforming the problem into a problem of finding a path of minimal weight in a single DAG) does not work:
For now, we consider Problem 13 for the sub-pattern $P^1 = (p_1^1, \ldots, p_{r_1}^1)$, the model $Q = (q_1, \ldots, q_m)$ and the parameter $\delta \geq 0$, i.e., the question is whether there is a subsequence $Q_{s,f} \subseteq Q$ with $1 \leq s \leq f \leq m$ and a translation t so that

$$\mathrm{F}(t(P^1), Q_{s,f}) \leq \delta.$$

The answer to Problem 13 is YES, iff there is a subsequence $Q_{s,f} \subseteq Q$ so that there is a translation t and a coupling K^δ for P^1 and $Q_{s,f}$ with

$$\mathrm{F}(t(P^1), Q_{s,f}) = \max_{(\alpha,\beta)\in K^\delta} \|t(p_\alpha^1) - q_\beta\|_p \leq \delta. \tag{6.6}$$

Now suppose the indices s and f as well as a coupling K^δ are fixed and the task is to decide whether there is a translation t that meets Equation (6.6).

Let $|K^\delta| = L > 0$. Then there is a translation that meets Equation (6.6), iff

$$\left(\bigcap_{(\alpha,\beta)\in K^\delta} D_\delta(p^1_\alpha - q_\beta) \right) \neq \emptyset, \tag{6.7}$$

where each $D_\delta(p^1_\alpha - q_\beta)$ is an L_p-disk with radius δ centered in $(p^1_\alpha - q_\beta)$. Clearly, the question whether there is a coupling K^δ that meets Equation (6.7) cannot simply be transformed into a problem of finding a path in a single directed graph in the same way as we did in the previous section, because in this case, the validility of a coupling K^δ depends on the combinatorial structure of the intersection the disks $D_\delta(p^1_\alpha - q_\beta)$ for all $(\alpha, \beta) \in K^\delta$ and not just on the fact whether, e.g., consecutive disks each intersect in at least one point. In the following, we therefore present another approach in order to decide Problem 13 for the sub-pattern P^1, if it consists of more than two points. And with this as a basis we will then present a strategy to decide Problem 14.

In [33], the authors describe an algorithm that considers the classical DFD problem under translation, i.e., given P^1, Q and $\delta \geq 0$, it decides whether there is a translation t so that $\mathrm{F}(t(P^1), Q) \leq \delta$, in $O(r_1^3 m^2(1 + \log(m r_1^{-1})))$ time, if $m \geq r_1$ and in $O(r_1^2 m^3(1 + \log(r_1 m^{-1})))$ time, if $m < r_1$. To simplify the presentation, we assume $r_l < m$ for all $1 \leq l \leq m$ in the following. Since their algorithm can easily be adjusted to decide the partial DFD problem under translation (Problem 13), it will be the basis for our algorithm, see Algorithm 10 below. Note that the following strategy works for all L_p-norms with $p \in \{1, 2, \infty\}$. However, since the authors of [33] focus on the L_2-distance and it is the distance measure that seems the most complicated to analyze, we focus on the L_2-distance in the following.

Given the point sequences P^1 and Q as well as a parameter $\delta \geq 0$, the authors consider the arrangement

$$\mathcal{D}_1 \coloneqq \{D_\delta(p^1 - q) \mid p^1 \in P^1, q \in Q\}$$

of $r_1 m$ disks with radius δ. $\mathcal{D}_1$ consists of $O(r_1^2 m^2)$ 0-, 1- and 2-dimensional faces. Let f be a face of this arrangement. Then there is a translation $t \in f$ so that $\mathrm{F}(t(P^1), Q) \leq \delta$ iff there is a coupling K^δ for P^1 and Q with $(f \cap D_\delta(p^1_\alpha, q_\beta)) \neq \emptyset$ for all $(\alpha, \beta) \in K^\delta$. In order to test, whether there is such a coupling K^δ, a 0-1-matrix M^1_f with r_1 rows and m columns is generated for every face f of the arrangement. We set

$$M^1_f[i, j] = \begin{cases} 1, & \text{if } (f \cap D_\delta(p^1_i, q_j)) \neq \emptyset, \\ 0, & \text{else,} \end{cases}$$

see the left part of Figure 6.10.

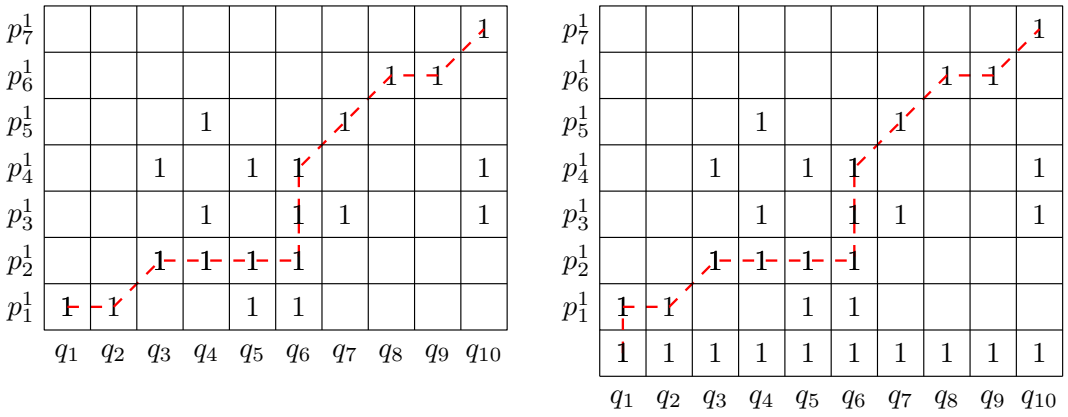

Figure 6.10: Left: Matrix M_f^1 as introduced in [33] for $P^1 = (p_1^1, \ldots, p_6^1)$ and $Q = (q_1, \ldots, q_{10})$ along with a path from $M_f^1[1,1]$ to $M_f^1[7,10]$.
Right: Matrix M_f^1 with an additional row of 1-entries at the bottom along with a path from $M_f^1[0,1]$ to $M_f^1[7,10]$.

Definition 6.18 *We call a* 1*-entry* $M_f^1[a',b']$ *for* $1 \leq a' \leq r_1$ *and* $1 \leq b' \leq m$ reachable *from* $M_f^1[a,b]$ *for* $1 \leq a \leq a'$ *and* $1 \leq b \leq b'$*, iff there is a (weakly) row- and column-monotone sequence of* 1*-entries from* $M_f^1[a,b]$ *to* $M_f^1[a',b']$ *where all consecutive* 1*-entries* $M_f^1[i,j]$ *(with* $a \leq i \leq a'$ *and* $b \leq j \leq b'$*) and* $M_f^1[i',j']$ *of the sequence have the property that* $(i',j') \in \{(i,j+1),(i+1,j),(i+1,j+1)\}$.

Note that in order to determine the 1-entries of M_f^1 that are reachable from a fixed 1-entry of M_f^1, in every step one can move from the current entry of M_f^1 to the entry above, to the right or to the entry to the right of the one above (diagonal). There is a translation $t \in f$ so that $\mathrm{F}(t(P^1), Q) \leq \delta$, if $M_f^1[r_1, m]$ is reachable from $M_f^1[1,1]$. This procedure is repeated for every face of the arrangement. Finally, we can conclude that there is a translation t so that $\mathrm{F}(t(P^1), Q) \leq \delta$ iff there is at least one face f in the arrangement so that $M_f^1[r_1, m]$ is reachable from $M_f^1[1,1]$. Actually, solving this decision problem by finding a sequence of 1-entries in M_f^1 is similar to finding a path in a suitable directed graph that depends on the face f at hand.
The algorithm described in [33], which we use as a black box in the following, uses a specific data structure to partition M_f^1 into blocks which then are analyzed individually in order to decide the DFD problem under translation at hand. Their algorithm returns additional information about the cells in M_f^1:

- for each 1-entry of $M_f^1[r_1, j']$ with $1 \leq j' \leq m$, the list of 1-entries $M_f^1[1,j]$ with $1 \leq j \leq j'$ so that $M_f^1[r_1, j']$ is reachable from $M_f^1[1,j]$ and
- for each 1-entry of $M_f^1[1,j]$ with $1 \leq j \leq m$, the list of 1-entries $M_f^1[r_1, j']$ with $j \leq j' \leq m$ so that $M_f^1[r_1, j']$ is reachable from $M_f^1[1,j]$.

This allows us to adjust the algorithm in order to decide the partial setting. In order to test, if there is a translation t so that the partial DFD of $t(P^1)$ and Q restricted to a single face f of the arrangement is at most δ, it is enough to check whether there is a 1-entry $M_f^1[r_1, j']$, with $1 \leq j' \leq m$ with a list, as described above, that contains at least one element.
By using the following adjustment, we can simplify the procedure so that we only have to check the corresponding list of one cell of M_f^1 without changing the runtime: We slightly alter the matrix M_f^1 by inserting a row of 1-entries at the bottom, hence there are m additional 1-entries $M_f^1[0, j]$ with $1 \leq j \leq m$, see the right part of Figure 6.10. We observe the following:

Observation 6.19 *There is an index $1 \leq j \leq m$ so that $M_f^1[r_1, j']$ with $j \leq j' \leq m$ is reachable from $M_f^1[1, j]$, iff $M_f^1[r_1, j']$ is reachable from $M_f^1[0, 1]$.*

Also, we can find the smallest index $\bar{j}$ so that $M_f^1[r_1, \bar{j}]$ is reachable from $M_f^1[0, 1]$ without increasing the runtime of the algorithm by checking the corresponding list of $M_f^1[0, 1]$. Let f_1 be the minimum of all such indices. We repeat the same strategy for all faces of the arrangement and compute the face f^* so that $M_{f^*}^1[r_1, f_1]$ is reachable from $M_f^1[0, 1]$ with $1 \leq f_1 \leq m$. Hence there is an index $1 \leq s_1 \leq f_1$ and a translation t with

$$\mathrm{F}(t(P^1), Q_{s_1,f_1}) \leq \delta.$$

Of course, this strategy works for every point sequence $P^l \in P$ with $1 \leq l \leq k$.

We formally state the following algorithm, called A9 in short:

Algorithm 9 *We are given two point sequences P^l with $1 \leq l \leq k$ and $Q = (q_1, \ldots, q_m)$ as well as a parameter $\delta \geq 0$. We compute an answer to the question of there is a translation t and indices s_l and f_l' so that*

$$\mathrm{F}(t(P^l), Q_{s_l,f_l'}) \leq \delta \tag{6.8}$$

by applying the following the strategy:
First, we compute the arrangement

$$\mathcal{D}_l \coloneqq \{D_\delta(p^l - q) \mid p^l \in P^l, q \in Q\}.$$

For every face f of the arrangement, we construct the 0-1-matrix M_f^l with $r_l + 1$ rows and m columns as elaborately described above. Then, we use the algorithm described in [33] to determine the smallest index $1 \leq j_f \leq m$ so that $M_f^l[r_l, j_f]$ is a 1-entry that is reachable from $M_f^l[0, 1]$, or if there is no such index at all. If a test was positive for at least one of the faces of $\mathcal{D}_l$, we set $f_l \coloneqq \min_f j_f$.
We return

- NO, *if there is no such translation* t*, i.e., none of the tests was positive, and*
- (YES, f_l)*, if there is a translation that satisfies Inequality* (6.8)*, where* f_l *is the smallest index* $1 \leq f_l \leq j'$ *so that there is an index* $1 \leq s_l \leq f_l$ *with* $\mathrm{F}(t(P^l), Q_{s_l,f_l}) \leq \delta$*.*

We state the following theorem:

Theorem 6.20 *Given two point sequences* $P^l = (p_1^l, \ldots, p_{r_l}^l)$ *and* $Q = (q_1, \ldots, q_m)$ *with* $r_l < m$*, and a parameter* $\delta \geq 0$*, Algorithm 9 decides in* $O(r_l^3 m^2 (1 + \log \frac{m}{r_l}))$ *time, whether there is an index* $1 \leq f_l' \leq m$ *and a translation* t *so that there is an index* s_l *with*

$$\mathrm{F}(t(P^l), Q_{s_l,f_l'}) \leq \delta$$

and, in case of a YES*-instance, also returns the smallest index* $1 \leq f_l \leq m$ *so that there is an index* $1 \leq s_l \leq f_l$ *with*

$$\mathrm{F}(t(P^l), Q_{s_l,f_l}) \leq \delta.$$

Proof. In [33], the authors proved the correctness of their algorithm that decides the DFD problem under translation for two given point sequences P^l and Q. Let f be the face at hand. We made altered their algorithm in two ways:

1. We checked whether there is a 1-entry $M_f^l[r_l, j']$, with $1 \leq j' \leq m$ with a list, that contains at least one element instead of just verifying this for $M_f^l[r_l, m]$.
2. We added one row of 1-entries at the bottom of every matrix M_f^l.

On Alteration 2: We observe, that there are two indices $1 \leq s_l \leq f_l \leq m$ and a translation t restricted to face f with

$$\mathrm{F}(t(P^1), Q_{s_l,f_l}) \leq \delta,$$

iff there are two indices $1 \leq s_l \leq f_l \leq m$ so that $M_f^l[r_l, f_l]$ is reachable from $M_f^l[1, s_l]$. This is true, iff there is an index $1 \leq f_l \leq m$ so that $M_f^l[r_l, f_l]$ is reachable from $M_f^l[0, 1]$, see Observation 6.19. Since the algorithm given in [33] verifies this correctly, Algorithm 9 also verifies this correctly by making use of the information that is computed and stored in the lists attached to the entries of M_f^l as described in 1.
The correctness of Algorithm 9 follows.
Since M_f^l has $r_l + 1$ rows and m columns and $r_l < m$, the algorithm introduced in [33] takes $O((r_l + 1)^3 m^2 (1 + \log \frac{m}{(r_l+1)})) = O(r_l^3 m^2 (1 + \log \frac{m}{r_l}))$ time. Since this algorithm and Algorithm 9 take approximately the same time, the stated upper bound on the runtime of Algorithm 9 follows. □

This strategy can be applied on the point sequences P^l for $1 \leq l \leq k$ consecutively in order to decide Problem 14:

Algorithm 10 *We are given a set of k sequences of points $P = \{P^1, \ldots, P^k\}$ where $P^i = (p_1^i, \ldots, p_{r_i}^i)$ is a sequence of points of length $r_i \geq 1$ for all $1 \leq i \leq k$ and $n = \sum_{i=1}^{k} r_i$, a sequence of points $Q = (q_1, \ldots, q_m)$ and a parameter $\delta \geq 0$. We set $s_1 = 1$. For $l = 1$ to k, we proceed as follows:*
We call $\mathrm{A9}(P^l, Q_{s_l,m})$.

- *If* $\mathrm{A9}(P^l, Q_{s_l,m}, \delta) =$NO*, we stop and return* NO *as the answer to Problem 14.*
- *If* $\mathrm{A9}(P^l, Q_{s_l,m}, \delta) = (\textsc{Yes}, f_l)$*, we set $s_{l+1} = f_l$ and proceed with $l = l + 1$.*

After the last step of the algorithm, one of the following cases occurs:

- $\mathrm{A9}(P^k, Q_{s_k,m}, \delta) =$NO *and we return* NO *as the answer to Problem 14.*
- $\mathrm{A9}(P^k, Q_{s_k,m}, \delta) = (\textsc{Yes}, f_k)$ *with $f_k \leq m$. Then,* YES *is returned as the answer to Problem 14.*

Note that by tracking the faces, e.g., storing them in a list, that admit the Yes-instances of the calls of $\mathrm{A9}(P^l, Q_{s_l,m})$ in every step, a sequence of admissible translations $(t_1, \ldots, t_k)$ can also be returned as a witness without increasing the runtime of the algorithm.

Theorem 6.21 *Algorithm 10 is correct.*

Proof. Suppose, Algorithm 10 returns YES. Since Algorithm 9 is correct and by construction $f_l \leq s_{l+1}$ for all $1 \leq l < k$, P^l is matched to Q correctly for every $1 \leq l \leq k$ and thus Algorithm 10 returns the correct answer to Problem 14.
Now suppose Algorithm 10 returns NO and suppose, this is false. Then there are translations $(t_1, \ldots, t_k)$ and indices s'_l and f'_l for $1 \leq l \leq k$ with $s'_l < f'_{l+1}$ for all $1 \leq l < k$ so that

$$\mathrm{F}(t_l(P^l), Q_{s'_l,f'_l}) \leq \delta$$

for all $1 \leq l \leq k$. Let $1 \leq j \leq k$ be the greatest index so that $\mathrm{A9}(P^j, Q_{s_j,m}, \delta) = (\textit{Yes}, f_j)$ during a run of Algorithm 10, i.e., $\mathrm{A9}(P^{j+1}, Q_{s_{j+1},m}, \delta) =$ NO. Then, $f'_j \geq f_j$ since A9 returns the smallest index f_l that permits a valid partial DFD matching under translation of P^l to Q (that is, P^l is matched to the subsequence Q_{s_l,f_l}) for a given index s_l fot all $1 \leq l \leq k$. Since all possible matchings of $P^{j+1}, \ldots, P^k$ to Q starting at index f'_j are a subset of all possible matchings starting at index f_j, and Algorithm 9 is correct. This means that there is no valid matching of P^{j+1} to $Q_{f_j,m} = Q_{s_{j+1},m}$, hence the answer to problem 14 is NO, which is a contradiction. □

Theorem 6.22 *Problem 14 can be decided in*

$$O\left(\sum_{l=1}^{k} r_l^3 m^2 \left(1 + \log \frac{m}{r_l}\right)\right)$$

time, if $r_l < m$ *for all* $1 \leq l \leq k$.

Proof. Algorithm 9 takes $O(r_l^3 m^2(1 + \log(m r_l^{-1})))$ time per call. Adding up the runtime of all calls for $1 \leq l \leq k$ results in the runtime claimed above. □

Algorithm 10 can be altered in order to decide the optimization version of Problem 14, where the goal is to find the optimal (smallest) value δ^*, so that there is a sequence of translations that meets the constraints of Problem 14 for this specific $\delta = \delta^*$: Consider a fixed P^l and the question whether there is a translation t so that the partial DFD of $t(P^l)$ and Q is at most δ. Recall that part of the algorithm described above is testing for all faces of the arrangement $\mathcal{D}_l$, if the translations they describe permit a Yes-instance of the subproblem at hand. As elaborately described in [33], there are $O(r_l^3 m^3)$ values of δ that lead to a combinatorial change of this arrangement as the disks involved expand or shirk by varying δ accordingly. Hence the following holds:

Theorem 6.23 *The optimization version of Problem 14 can be decided in*

$$O\left(\left(\sum_{l=1}^{k} r_l^3 m^2 \left(1 + \log \frac{m}{r_l}\right)\right)\left(\sum_{l=1}^{k} r_l^3 m^3\right)\right)$$

time, also when returning a witness $T = (t_1, \ldots, t_k)$.

In [33], the authors use a parametric search technique and a parallel implementation of their decision algorithm that finds δ^* after $O(\log(m + r_l))$ parallel steps on $O(m^2 r_l^2)$ processors. A slightly adjusted version of this strategy could probably also applied in our setting. However, since we have k point sequences $P^1, \ldots P^k$ that are to be matched to Q, we have $O(\sum_{l=1}^{k} r_l^3 m^3)$ critical values in total, which leads to a total of $O(\log(m + \sum_{l=1}^{k} r_l)) = O(m + n)$ parallel steps. Hence, the runtime stated in Theorem 6.23 could be reduced to

$$O\left(\left(\sum_{l=1}^{k} r_l^3 m^2 \left(1 + \log \frac{m}{r_l}\right)\right) \log\left(m + n\right)\right).$$

On Instances with Given Neighborhood Graph

Instead of designing the similarity of the involved translations as the order in which the polygonal curves of the pattern are to be matched along the model,

we can also choose to use a neighborhood graph with k vertices instead or even combine both strategies.
However, including a neighborhood graph also means taking up all difficulties of this approach, e.g., the difficulty of estimating the combinatorial complexity of admissible regions after various inflations that were discussed in the previous chapters. This means that, e.g., there are no known polynomial time algorithms for EGSM instances under the partial DFD for graphs that contain cycles or graphs that are incomplete trees.

6.5 Extension to Higher Dimensions

So far, we considered EGSM problems for point sets in the plane. However, in some applications, strategies to match geometrical objects in 3D-space well are needed. Problem 3 (EGSM under translations for point sets in the plane) can easily be adapted to EGSM for point sets in $\mathbb{R}^3$ and does not differ much from the original problem, so that slightly adapted versions of the solution strategies given in Chapters 2 to 5 can be used to solve it. However, since the description complexity of the involved geometrical objectso increases with increasing number of dimensions of the input, the runtime of the corresponding exact and approximate algorithms also increases to a certain degree.

In this section, everything is stated in $\mathbb{R}^3$ and $\|\cdot\|$ denotes the L_2-norm. For a point $p = (p.x, p.y, p.z) \in \mathbb{R}^3$ and a translation vector $t = (t.x, t.y, t.z) \in \mathbb{R}^3$,

$$t(p) \coloneqq (p.x + t.x, p.y + t.y, p.z + t.z).$$

We consider the following EGSM problem:

Problem 15 *Given:*

$$\begin{aligned} P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^3 \quad & \textit{a point set (the pattern),} \\ Q = \{q_1, \ldots, q_m\} \subset \mathbb{R}^3 \quad & \textit{a point set (the model),} \\ G = (V, E) \quad & \textit{an undirected graph with } V = \{i \mid 1 \leq i \leq n\} \textit{ and} \\ & E \subseteq \{\{i, j\} \mid i, j \in V\}, \textit{ and} \\ \delta \geq 0 \quad & \textit{a parameter.} \end{aligned}$$

Find: A sequence of n translation vectors $T = (t_1, \ldots, t_n)$ so that

$$\max\left(\mathrm{d}(T(P), Q), \max_{\{i,j\} \in E} \|t_i - t_j\|\right) \leq \delta,$$

where d *is a suitable distance measure.*

On Exact Algorithms for Problem Instances with Tree-Neighborhoods

For now, let $G = T$ be a tree.

In Chapter 5, we considered a variant of Problem 15 for points in the plane for tree neighborhoods where d equals the Euclidean 1-to-1-distance. We gave insights about the geometric structure of the so-called admissible regions computed by Algorithm 7 as intermediate results as well as lower and upper bounds on their description complexity. In the following, we use the same notation as in Chapter 5.

The strategy of Algorithm 7 and the variants of it used in this thesis to solve EGSM problems for tree-neighborhoods work for different norms and in different dimensions. Recall that for EGSM for points in the plane under the Euclidean distance, the algorithm essentially used intersection and inflation operations on admissible regions and propagated them bottom-to-top through T. These admissible regions are L_2-disks and sets that originated from intersecting and inflating L_2-disks, so-called arc-sets. One major reason why there are no polynomial upper bounds on the description complexity of these sets if the corresponding neighborhood graph is an arbitrary tree or d equals the Euclidean Hausdorff distance is that inflating an arc-set (computing the Minkowski sum of an arc-set with an L_2-disk) creates a new arc-set with a boundary that consists of at most twice as much arcs as the original arc-set.

In $\mathbb{R}^3$, this issue becomes even more important:

Notation 6.24 *For an $r > 0$ and $c \in \mathbb{R}^3$, $S_r(c)$ denotes the sphere with radius r centered in c and S_r denotes the sphere with radius r and center in the origin.*

Example 6.25 *Consider the three spheres $S_{r_i}(c_i)$ with $r_i > 0$ and $c_i \in \mathbb{R}^3$ for $1 \leq i \leq 3$ and the sphere S_{r_4} for $r_4 > 0$. Suppose $\bigcap_{i=1}^{3} S_{r_i}(c_i) \neq \emptyset$ and suppose the intersection $\bigcap_{i=1}^{3} S_{r_i}(c_i)$ has three faces (which originated from different spheres), three circular arcs and two vertices, see Figure 6.11. The Minkowski sum of this object with S_{r_4} has eight faces, twelve arcs and six vertices, see Figure 6.11.*

The inflation of an admissible region that originated from many intersection- and inflation operations may result in an even more complex admissible region and up to now we do not know of any way to estimate the description complexity of such objects. However, this is different if the L_1- the L_∞- or a polygonal norm is used instead of the L_2-norm.

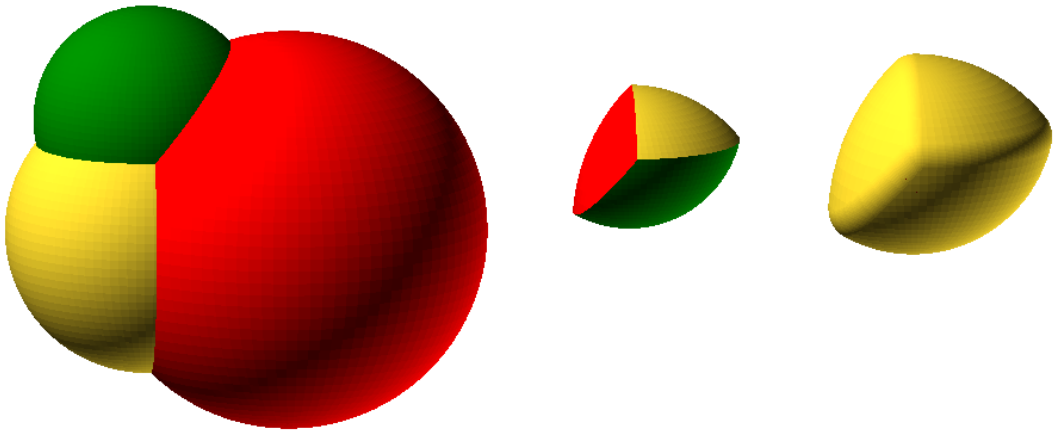

Figure 6.11: Left: a collection of three spheres $S_{r_i}(c_i)$ for $1 \leq i \leq 3$.
Middle: the intersection of the three spheres $\bigcap_{i=1}^{3} S_{r_i}(c_i)$.
Right: the Minkowski sum of $\bigcap_{i=1}^{3} S_{r_i}(c_i)$ and S_{r_4}.

On Approximation Algorithms for Problem Instances with Tree-Neighborhoods

In this section, let $G = T$ be a tree.
Considering Problem 15 under the L_1- the L_∞- or under a polygonal norm avoids the problems that arise during a run of algorithm B under the L_2-norm: In Chapter 2, we considered this variant of Problem 15 for points in the plane. Under the L_1-norm, at the start, the admissible regions consist of unions of axis-parallel squares, which are then propagated bottom-to-top through T by intersecting and inflating them and the resulting admissible regions. In $\mathbb{R}^3$ the squares become axis-parallel cubes, but the strategy to compute a solution stays the same. Since a cube with diagonal 2δ is contained in a sphere with radius δ, Algorithm 1, if adapted to $3D$-input, can be used to approximate the Euclidean setting:

Corollary 6.26 *Algorithm 1, adapted to 3D-input, with tree-neighborhoods gives a $\sqrt{3}$-approximation for Problem 15 with the same settings under the L_2-norm.*

Chapter 2 also contains an extension of this strategy to polygonal norms, which then allows for a $(1+\epsilon)$-approximation for the Euclidean setting. This strategy can also be adapted to Problem 15. The main idea is to approximate all spheres with convex polyhedra that are translational copies of each other. In doing so, the unions of polygons that constitute the admissible regions at the start of Algorithm 2 for the problem in the plane become unions of convex polyhedra

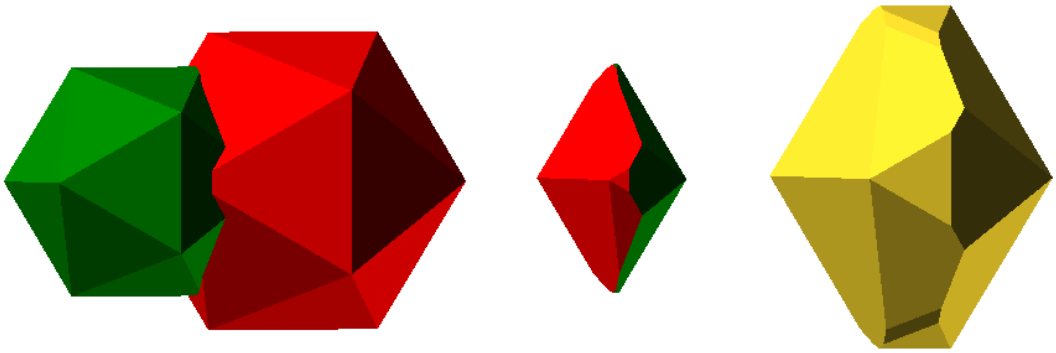

Figure 6.12: Left: two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$.
Middle: the intersection of the polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$.
Right: the Minkowski sum of $\mathcal{P}_1 \cap \mathcal{P}_2$ and a scaled version of $\mathcal{P}_1$.

in $\mathbb{R}^3$.
Note that in [34], the author proves that every sphere $S_1(c)$ with $c \in \mathbb{R}^3$ can be approximated with some convex polyhedron $\mathcal{P}$ with $O(\epsilon^{-1})$ vertices so that $\vec{h}(S_r c, \mathcal{P}) \leq \epsilon$. By computing one of these polyhedra and using it for the $\mathbb{R}^3$-version of Algorithm 2, we get the following observation:

Observation 6.27 *For any $\epsilon > 0$, Algorithm 2, adapted to 3D-input for convex polyhedra, computes a $(1+\epsilon)$-approximation for Problem 15 under the L_2-norm.*

Theorem 6.28 *For any $b \in \mathbb{N}$, Algorithm 2, adapted to 3D-input for convex polyhedra, gives a $(1+\epsilon)$-approximation for Problem 15 under the L_2-norm in $O(bn^3m^2(\log b + \log n + \log m))$ time.*

Proof. At the start of the algorithm, every vertex of G corresponds to a set of admissible translations that is approximated by a collection of m convex polyhedra with $O(b)$ vertices. The union of m convex polyhedra that are translational copies of each other form a union of pseudo spheres, and computing the union of m such pseudo disks in $\mathbb{R}^3$ takes $O(m^2 b(\log m + \log b))$ time, see [22]. Pairwise intersecting nm convex objects creates at most n^2m^2 convex objects. At the beginning there are $O(nm)$ polyhedra with description complexity $O(b)$ each. Let $f > 0$ be the number of faces of the the polyhedron at hand, thus $f \in O(b)$. The following line of argument follows the line of argument in Lemma 2.12, where convex polygons in $\mathbb{R}^2$ is considered: Since all polyhedra are translational copies of each other, the number of possible orientations for each face of every polyhedron is f. The intersection as well as the Minkowski sum of two such polyhedra is convex, see Figure 6.12. Hence, every face of the resulting polyhedron has a different orientation. Since there are f different

orientations in total, the resulting polyhedron again has f faces at most. Since there are no more than $O(n^2m^2)$ polyhedra in total, the description complexity of every admissible region at any iteration of the algorithm is bounded by $O(bn^2m^2)$. There are n sets of admissible translations, which is why $O(bn^3m^2)$ space is required in total.
In [35], the authors give a strategy for computing the intersection of two convex polyhedra in $O(k \log k)$ time, where k is the number of vertices of both polyhedra. This strategy can be used to calculate the intersection of admissible regions. Since G is a tree with n vertices, admissible regions will be intersected n times at most. Hence, the algorithm described above runs in $O(bn^3m^2(\log b + \log n + \log m))$ time. □

Chapter 7

Discussion and Outlook

In this thesis, we studied the very flexible and versatile topic of elastic geometric shape matching, which is of significant relevance in many applications such as computer assisted surgery. Computing a most accurate transformation is crucial in this setting, and modelling the needed registrations as EGSM problem instances offers the opportunity to describe and handle local and global distortions of the pattern, such as the deformation of the organ due to the operation process itself, as well as the thorax movement due to breathing, or the influence of magnetic fields on the tracking device etc., in a very accurate way.
Many applications benefit from the flexibility of the EGSM framework that allows for a both globally consistent and locally precise mapping in cases, where, e.g., local deformations may occur, and thus leads to more accurate results than other approaches such as the conventional GSM approach. However, after EGSM was first introduced in 2011 in the dissertation of Stehn [14] and a paper by Knauer, Kriegel and Stehn [15], it soon became clear that computing a solution for a given EGSM problem can be challenging. The number of variants of this problem is vast, depending on how the different options are chosen to match the application at hand, and only minor changes in the problem setup result in the need of completely different strategies to compute a solution, which is why there are no efficient approaches for most of the EGSM variants.

7.1 Contribution

The goal of this research was getting a deeper understanding of the EGSM framework in order to design efficient exact solutions for different EGSM variants as well as efficient approximation algorithms for EGSM variants where finding efficient exact solutions is not possible due to, e.g., the hardness of the

problems at hand.
All considered problem instances share that the pattern and the model are point sets in the plane and each point of the pattern forms an individual subshape of the pattern. Also, the distance between the translated pattern and the model is measured with the directed Hausdorff distance under different norms or the 1-to-1-distance under the Euclidean norm and the transformation class at hand is the class of translations. This thesis contains the designs of the following solution strategies:

- We presented an efficient exact algorithm for the decision variant of an EGSM problem for neighborhood graphs that are trees where the norm at hand is the L_1-norm or a polygonal norm and gave an FPTAS for the same problem under the Euclidean norm.
- An FPTAS for an EGSM problem under the 1-to-1-distance and the Euclidean norm where the neighborhood graph is a simple cycle was designed.
- We gave an algorithm that, for an $\epsilon > 0$, gives a $(1 + \epsilon)$-approximation to the optimum of the objective function for an EGSM problem variant under the directed Euclidean Hausdorff distance with a given feedback vertex set and an algorithm that gives a $(1 + \epsilon)$-approximation to the optimum of the objective function for the problem with a given path or tree decomposition.
- An algorithm from [16] was adapted to an EGSM problem for tree neighborhoods under the Euclidean 1-to-1-distance that runs polynomial time and space, if the neighborhood graph is a path and gave non-polynomial upper bounds on the time and the space required by the algorithm to solve the problem if the neighborhood graph is a tree. This can be improved to polynomial if the tree is complete. We also gave a rough description on how the difficulties in estimating the combinatorial complexity advance if the correspondence between the points of the pattern and the points of the model is not fixed.
- We gave insights about EGSM under rigid motions and discussed how existing strategies for problem variants for point sets can be modified to solve EGSM problems for line segments, triangles and triangulated surfaces. We also considered EGSM in higher dimensions, with differently weighted objectives and for imprecise input sets.

Overall, we provided efficient strategies for many different EGSM problem variants under translations, which can be used in different applications. With this we built a solid basis for further research in different aspects of the framework such as EGSM in higher dimensions or under rigid motions.

7.2 Future Work

Due to the flexibility of the EGSM framework, there is a great number of options to choose from, depending on the application at hand. This results in many different problem variants and the need for a wide range of solution strategies. In this thesis, we were able to discuss but a fraction of these variants and the strategies to solve them and further research on both improving existing strategies as well as considering completely new problem variants seems very promising.

On Linear Time Propagation. In a student project [36] and a master thesis [37], among other things Problem 8 from Chapter 5 was considered, where the distance measure at hand is the 1-to-1-distance under the Euclidean, the L_1- or a polygonal norm and the neighborhood graph is a path or a tree. An implementation of the algorithm given in Section 5.2 was presented that seems to handle any intersection and inflation operation in linear time subject to the combinatorial complexity of the geometric shapes involved.
If this was proven to work, the upper bound on the runtime all of the algorithms presented in Chapters 2, 3 and5 could be improved by a logarithmic factor.

The EGSM Variants of Chapter 6. In Chapter 6, we briefly discussed different EGSM variants that are not covered by Problem 3, the EGSM variant that is considered in the previous chapters. Each of these variations seems to be a promising topic for further research:
Considering imprecise input points and an extension to the 3D-space along with the possibility to match objects such as triangulated surfaces are of great interest in many applications. While EGSM for line segments and EGSM in 3D seem to be solvable by just customizing existing strategies, considering other topics such as EGSM under rigid motions require new strategies in order to solve the EGSM problem at hand.

Including Time Dependency. In all EGSM variants we considered so far, including the most general version Problem 2, we are given two sets P and Q and the goal is to find a set of transformations T so that $\mathrm{d}(T(P), Q)$ is minimized. However, in many applications such as computer-aided medicine, the pattern may change over time so that many EGSM instances of the same problem, but with a changing pattern, have to be solved. In this case, the transformations that are assigned to the same part of the pattern at different times should be somehow similar as well. In the following, we consider two ways to design discrete and continuous time dependent EGSM problem formulations.

A Discrete Setting: We are given a model Q from a class of geometric shapes $\mathcal{S}$, a class of transformations $\mathcal{T}$ and a neighborhood graph $G = (V, E)$. Now suppose, we have $t \in \mathbb{N}$ different points in time and for every point in time $1 \leq l \leq t$, we are given a different pattern $P^l \in \mathcal{S}$ together with a partition $\{P_1^l, \ldots, P_k^l\}$ of P^l and the task is to compute a transformation ensemble T^l from class $\mathcal{T}$ for every $1 \leq l \leq t$ so that the distance between $T(P^l)$ and Q is minimized (according to a suitable distance measure) and the similarity of the transformations that are adjacent in the corresponding neighborood graph is maximized (according to a suitable similarity measure). Additionally, the similarity of transformations that act on the same subpattern at succeeding points in time, should be maximized.
The discrete time dependent EGSM problem can be formulated as follows:

Problem 16 *Given:*

$$\begin{aligned}
\mathcal{S} &\quad \textit{a class of geometric shapes,}\\
\mathcal{T} &\quad \textit{a class of transformations,}\\
P^1, \ldots, P^t \in \mathcal{S} &\quad t \textit{ patterns with}\\
\{P_1^l, \ldots, P_k^l\} &\quad \textit{a partition of } P^l \textit{ for } 1 \leq l \leq t,\\
Q \in \mathcal{S} &\quad \textit{the model, and}\\
G = (V, E) &\quad \textit{an undirected graph with } V = \{i \mid 1 \leq i \leq k\} \textit{ and}\\
&\quad E \subseteq \{\{i, j\} |\ i, j \in V\}.
\end{aligned}$$

Find: t transformation ensembles $T^l = (t_1^l, \ldots, t_k^l) \in \mathcal{T}$ for $1 \leq l \leq t$, so that

$$\min_{1 \leq l \leq t} \mathrm{d}(T^l(P^l), Q)$$

with $\mathrm{d} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$ is minimized and

$$\max\left(\max_{1 \leq l \leq t}\left(\mathrm{sim}_G\, T^l\right), \mathrm{sim}\{(t_i^l, t_i^{l+1}) \mid 1 \leq l \leq t-1 \textit{ and } 1 \leq i \leq k\}\right)$$

with $\mathrm{sim}_G : \mathcal{T} \to \mathbb{R}_0^+$ is maximized.

On the Continuous Setting: In a continuous setting for a time interval $[0, t]$ with $t \in \mathbb{R}^+$, the subpatterns and transformations become functions, i.e.:

$$\begin{aligned}
P_i : [0, t] &\to \mathcal{S} \text{ and}\\
t_i : [0, t] &\to \mathcal{S},
\end{aligned}$$

for $1 \leq i \leq k$. In order to estimate, how fast the transformation function changes over time, the derivative of the transformation function can be analyzed.

In most cases, a discrete problem formulation is sufficient, since the input of any application is discrete. However, time dependent EGSM is an interesting topic for future research because there are many applications that benefit from a time dependent setting and up to now there are no results on this topic.
The most challenging part of this setup seems to be the computation of the similarity constraints on the transformations. In the previous chapters we saw that EGSM problems seem to get more complex with increasing number of cycles in the neighborhood graph. In the discrete setting, including the similarity constraints of transformations that act on the same subpattern at succeeding points in time essentially equals creating a neighborhood graph that consists of t copies of G (one copy for every point in time) and linking them with edges that encode these similarity constraints. In doing so, a graph with kt vertices and many cycles is created.

Appendix A

An Algorithm on Samples

In the following, we use the notation and the results introduced in Chapter 5 and take up the end of Section 5.3. Let the neighborhood graph $G = T_r$, a tree rooted in r.
We already considered Algorithm 2, an approximation algorithm for EGSM problems under the directed Euclidean Hausdorff distance for neighborhood graphs that are trees in Chapter 2: Every disk is approximated with a regular polygon with $O(\epsilon^{-1/2})$ vertices, which reduces the description complexity of the admissible regions. For Problem 8, where the correspondence between the pattern and the model is known, Algorithm 2 gives a $(1+\epsilon)$-approximation to the optimum of the objective function in $O(\epsilon^{-1/2} n(\log n + \log \epsilon^{-1}))$ time. Here, $(1+\epsilon)$-approximation means that if $\delta^{(\epsilon)}$ is the approximation to δ^* computed by Algorithm 2 for a given $\epsilon > 0$, the optimal value δ^* is bounded by

$$\left(1 - \frac{1}{2}\epsilon\right)\delta^* \leq \delta^{(\epsilon)} \leq \left(1 + \frac{1}{2}\epsilon\right)\delta^*.$$

However, there is an even simpler way to compute a $(1+\epsilon)$-approximation for the EGSM variant discussed in Chapter 5. In the following, we present an algorithm that computes an $(1+\epsilon)$-approximation to the optimum of the objective function of Problem 8 in $O(\epsilon^{-1} n)$ time and space for a given $\epsilon > 0$. The main idea is based on sampling each t_i from a dense enough ϵ_{grid}-grid that covers I_{v_i}, where the sample-rate ϵ_{grid} is chosen depending linearly on ϵ and a 3-approximation to δ^*. The approximation algorithm uses the same strategy as the exact algorithm from Section 5.2, but instead of propagating the concrete admissible regions I_{v_i}, the convex hull S_{v_i} of the points on the ϵ_{grid}-grid that lie in the admissible region at hand are propagated, see Figure A.1. The inflation and intersection operations are also done solely on these objects. Since the admissible regions are convex at any point of the algorithm, the convex hulls

each have complexity $O(\epsilon^{-1})$. Any of the convex hulls can be computed in $O(\epsilon^{-1})$ time and inflating them also takes $O(\epsilon^{-1})$ time. Since all of these objects are part of the same grid, it takes $O(k\epsilon^{-1})$ time to compute the intersection of k such objects. This leads to a total runtime of $O(n\epsilon^{-1})$. The details of the algorithm will be discussed in the remainder.

Notation A.1 *We define*

$$\delta^{(3)} \coloneqq \gamma(C, C, G).$$

Then, $T^{(3)}$ is a 3-approximation to δ^*, see Lemma 3.5 for details.
In other words, if $\delta \geq \delta^{(3)}$ (or $\delta \leq 3^{-1}\delta^{(3)}$) we instantly know that (δ, C, G) is a YES-instance (or NO-instance). If $3^{-1}\delta^{(3)} < \delta < \delta^{(3)}$, the area of any admissible region that is sampled is bounded by a disk of radius $\delta^{(3)}$ from above and the area of any inflated admissible region is bounded by a disk of radius $2\delta^{(3)}$ from above. Let ϵ_{grid} be the sample-rate of the grid of the admissible regions.

A.1 The Algorithm

At the start, we need to shift every point c_i of the input sequence C to the closest grid-point, where $\bar{C} \coloneqq (\bar{c}_1, \ldots, \bar{c}_n)$ denotes the shifted input sequence. As a consequence, the center of any admissible region $I_{v_i} = D_\delta(\bar{c}_i)$ is part of the grid.

Notation A.2 *We define*

$$\begin{aligned} S &\coloneqq \{(l\epsilon_{\text{grid}}, m\epsilon_{\text{grid}}) \in \mathbb{R}^2 \mid l, m \in \mathbb{Z} \wedge \|(l\epsilon_{\text{grid}}, m\epsilon_{\text{grid}})\| \leq \delta\} \text{ and} \\ S_{v_i} &\coloneqq \{(l\epsilon_{\text{grid}}, m\epsilon_{\text{grid}}) \in \mathbb{R}^2 \mid l, m \in \mathbb{Z} \wedge \|(l\epsilon_{\text{grid}}, m\epsilon_{\text{grid}}) - \bar{c}_i\| \leq \delta\} \end{aligned}$$

for $1 \leq i \leq n$ *and denote the convex hull of any point set* R *with* $\mathfrak{h}(R)$.

Note that the point sequences $\mathfrak{h}(S)$ and $\mathfrak{h}(S_{v_i})$ can each be computed in $O(\epsilon^{-1})$ time in a straightforward manner, since D_δ and $D_\delta(\bar{c}_i)$ are convex and the convex hulls $\mathfrak{h}(S)$ and $\mathfrak{h}(S_{v_i})$ each consist of $O(\epsilon^{-1})$ points, see Figure A.1.

Then, we can follow the same strategy as Algorithm 2 described in Section 5.2: We pick a vertex $r \in V$ and henceforth consider T_r, the tree rooted in r. In each step of the algorithm, an unmarked vertex v of T_r is selected with the property that all children of v are marked/updated. Let $\mathfrak{c}_1(v), \ldots, \mathfrak{c}_{n_v}(v)$ be the n_v children of v. Then, the admissible region of v and those of the children of v are merged into a new admissible region that is stored in the new vertex

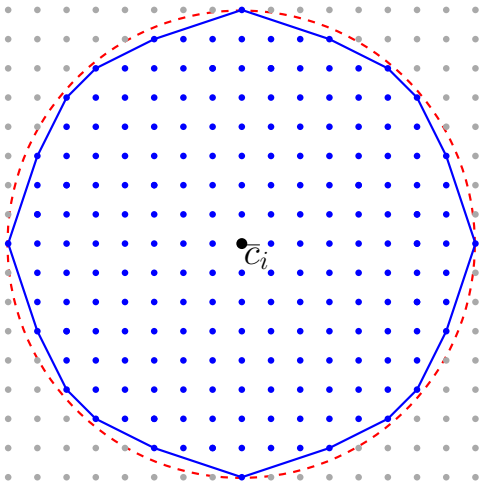

Figure A.1: The set S (the solid line indicates the order of the points of $\mathfrak{h}(S)$) is applied to $D_\delta(\bar{c}_i)$ (dashed circle).

v', the updated and marked version of v. Here, merging means inflating the admissible regions of $\mathfrak{c}_1(v), \ldots, \mathfrak{c}_{n_v}(v)$ and then computing the intersection of them and the admissible region stored in v.

At any point of the algorithm, the admissible regions are approximated by the convex hull of the set of points on the ϵ_{grid}-grid they cover. The key operations, inflating and intersecting such regions, can be done as follows.
The convex hull $\mathfrak{h}(S_v^\delta)$ is formally given by

$$\begin{aligned} \mathfrak{h}(S_v^\delta) &= \mathfrak{h}(p \in \mathbb{R}^2 \mid p \in S_v \oplus S) \\ &= \mathfrak{h}(p \in \mathbb{R}^2 \mid p \in \mathfrak{h}(S_v) \oplus \mathfrak{h}(S)). \end{aligned}$$

However, the sequence $\mathfrak{h}(S_v^\delta)$ can be computed from $\mathfrak{h}(S_v)$ and $\mathfrak{h}(S)$ in a straightforward manner without explicitly computing $\mathfrak{h}(S_v) \oplus \mathfrak{h}(S)$. The sequence $\mathfrak{h}(S_{v'})$ is given by

$$\mathfrak{h}(S_{v'}) \coloneqq \mathfrak{h}\left(S_v \bigcap_{i=1}^{n_v} S^\delta_{\mathfrak{c}_i(v)} \right),$$

see Figure A.2.
We can now state the following algorithm:

Algorithm 11 *We are given a sequence of points $C = (c_1, \ldots, c_n)$, a directed graph $G = (V, E)$ and a parameter $\delta \geq 0$.*
First, very point c_i of the input sequence C is shifted to the closest grid-point, where $\bar{C} \coloneqq (\bar{c}_1, \ldots, \bar{c}_n)$ denotes the shifted input sequence. Also, we compute $\mathfrak{h}(S_v)$ for every vertex $v \in V$ and store it in v.

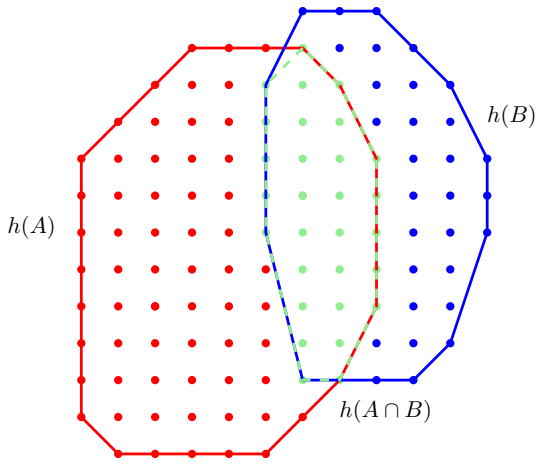

Figure A.2: Point sets A and B with $\mathfrak{h}(A)$ and $\mathfrak{h}(B)$, as well as $A \cap B$ and $\mathfrak{h}(A \cap B)$ (dashed lines).

Then, we pick a vertex $r \in V$ and henceforth consider T_r, the tree rooted in r. In each step of the algorithm, an unmarked vertex v of T_r is selected with the property that all children of v are marked/updated. Let $\mathfrak{c}_1(v), \ldots, \mathfrak{c}_{n_v}(v)$ be the n_v children of v. We update v to v' by inflating $\mathfrak{h}(S_v)$ by δ as follows

$$\mathfrak{h}(S_v^\delta) = \mathfrak{h}(p \in \mathbb{R}^2 \mid p \in \mathfrak{h}(S_v) \oplus \mathfrak{h}(S)),$$

and then storing the updated set of admissible translations

$$\mathfrak{h}(S_{v'}) = \mathfrak{h}\left(S_v \bigcap_{i=1}^{n_v} S^\delta_{\mathfrak{c}_i(v)}\right),$$

in v'.
This process is repeated until:

1. *There is a vertex v with $I_v = \varnothing$ (after a contraction):*
   The process stops and NO *is returned as the answer to Problem 8.*
2. *The root r is updated and $I_{r'} \neq \varnothing$:*
   The algorithm terminates and returns YES *as the answer to Problem 8.*

A.2 Correctness and Complexity.

Theorem A.3 *Let*

$$\epsilon_{\text{grid}} := \frac{\epsilon\delta}{3\left(2-\sqrt{2}\right)}$$

for a given $\epsilon > 0$. The optimization version of Algorithm 11 gives a $(1+\epsilon)$-approximation for the optimization version of Problem 8 under the L_2-norm under the 1-to-1-distance.

Proof. There are two sources of relative error that have to be evaluated to guarantee the quality of the approximation:

- At start, the points of the input sequence C are shifted to the nearest grid-point, in order to ensure that, if $\delta^* = 0$, the optimal sequence $(t_1 = \bar{c}_1 \ldots = t_n = \bar{c}_n)$ is actually admissible, resulting in the set $\bar{C}$. We define
$$e_{\text{shift}} \coloneqq \max_{1 \leq i \leq n} \|c_i - \bar{c}_i\| \leq \frac{\sqrt{2}\epsilon_{\text{grid}}}{2}$$
for any $1 \leq i \leq n$.
- The distance between any point within $D_\delta(\bar{c})$ and the nearest grid-point within $D_\delta(\bar{c})$, which is obviously bounded from above by $e_{\text{grid}} \coloneqq \epsilon_{\text{grid}}$.

Let $\delta \leq (1 - \epsilon 2^{-1})\delta^*$. Since
$$e_{\text{shift}} = \frac{\sqrt{2}\epsilon_{\text{grid}}}{2} = \frac{\sqrt{2}\epsilon\delta}{6\left(2-\sqrt{2}\right)} < \frac{\epsilon\delta}{2},$$
every disk $D_\delta(\bar{c}_i)$ is completely contained in the interior of $D_{\delta^*}(c_i)$. Suppose, δ permits a YES-instance. Then, there is a sequence of translations $T = (t_1, \ldots, t_n)$ with $t_i \in D_\delta(\bar{c}_i)$ for every $1 \leq i \leq n$ and as a consequence, $\|t_i - c_i\| < \delta^*$ for every $1 \leq i \leq n$. Hence, the value
$$\max_{1 \leq i \leq n} \|t_i - c_i\| < \delta^*$$
permits a YES-instance, which is a contradiction.
Now let $\delta \geq (1 + \epsilon 2^{-1})\delta^*$. Since
$$e_{\text{shift}} + e_{\text{grid}} = \frac{\sqrt{2}\epsilon_{\text{grid}}}{2} + \epsilon_{\text{grid}} = \frac{\epsilon\delta}{6} \leq \frac{\epsilon\delta^*}{2},$$
there is at least one grid-point within $D_\delta(\bar{c}_i)$ that is a witness for a YES-instance. □

Theorem A.4 *A $(1+\epsilon)$-approximation to Problem 8 under the L_2-norm under the 1-to-1-distance can be computed in $O(n\epsilon^{-1})$ time and a $(1+\epsilon)$-approximation to the optimization version of this problem can be computed in $O((n\epsilon^{-1}) \log \epsilon^{-1})$ time.*

Proof. Since $\mathfrak{h}(S_{v_i}^{\delta})$ is convex, only contains points of the ϵ_{grid}-grid and can be covered by a disk with radius 2δ, the length of $\mathfrak{h}(S_v^{\delta})$ as well as the time that is needed to compute it is bounded by $4 \cdot 2\delta\epsilon \in O(\epsilon^{-1})$. Hence, computing the convex hulls of all admissible regions in the beginning as well as the inflation process for all $n-1$ regions takes $O(n\epsilon^{-1})$ time.
The sequence $\mathfrak{h}(S_{v'})$ can also be computed from $\mathfrak{h}(S_v)$ and $\mathfrak{h}(S_{\mathfrak{c}_i(v)}^{\delta})$ for $1 \leq i \leq n_v$ in a straightforward manner in $O(n_v\epsilon^{-1})$ time. Since there are n vertices in total, the total time that is needed to perform all intersection operations is also bounded by $O(n\epsilon^{-1})$. Therefore, the total runtime of Algorithm 11 is $O(n\epsilon^{-1})$.
Note that Algorithm 11 considers the decision version of an EGSM problem. It can easily be adapted to solve the optimization version of the same problem approximately by including it in a binary search on the interval $[3^{-1}\delta^{(3)}, \delta^{(3)}]$, which needs $O(\log \epsilon^{-1})$ time. As a consequence, the optimization version of Algorithm 11 runs in $O((n\epsilon^{-1}) \log \epsilon^{-1})$ time. □

List of Figures

Bibliography

[1] H. Alt, L. Guibas, Discrete Geometric Shapes: Matching, Interpolation, and Approximation, in: Handbook of Computational Geometry, Elsevier B.V., 2000, pp. 121–153.

[2] M. d. Berg, O. Cheong, M. v. Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, Vol. 3, Springer-Verlag TELOS, Santa Clara, CA, USA, 2008.

[3] S. Xu, Robust traffic sign shape recognition using geometric matching, Intelligent Transport Systems, IET 3 (2009) 10 – 18.

[4] R. H. Davis, J. Lyall, Recognition of Handwritten Characters – A Review, in: Image and Vision Computing, Vol. 4, Butterworth-Heinemann, Newton, MA, USA, 1986, pp. 208–218.

[5] H. Alt, L. Scharf, S. Scholz, Probabilistic Matching and Resemblance Evaluation of Shapes in Trademark Images, in: Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR), Amsterdam, The Netherlands, 2007, pp. 533–540.

[6] T. Yoshikawa, M. Koeda, H. Fujimoto, Shape Recognition and Grasping by Robotic Hands with Soft Fingers and Omnidirectional Camera, in: IEEE International Conference on Robotics and Automation (ICRA), 2008, pp. 299–304.

[7] J. B. A. Maintz, M. A. Viergever, A Survey of Medical Image Registration, in: Medical Image Analysis, Vol. 2, 1998, pp. 1–36.

[8] S. Venkatasubramanian, Geometric Shape Matching and Drug Design, Ph.D. thesis, Department of Computer Science, Stanford University (1999).

[9] R. C. Veltkamp, M. Hagedoorn, State of the Art in Shape Matching, in: Principles of Visual Information Retrieval. Advances in Pattern Recognition, Springer, London, 2001, pp. 87–119.

[10] S. Rusinkiewicz, M. Levoy, Efficient Variants of the ICP Algorithm, in: Proceedings Third International Conference on 3-D Digital Imaging and Modeling, 2001, pp. 145–152.

[11] L. G. Brown, A Survey of Image Registration Techniques, ACM Comput. Surv. 24 (4) (1992) 325–376.

[12] M. Peterhans, T. Oliveira, V. Banz, D. Candinas, S. Weber, Computer-Assisted Liver Surgery: Clinical Applications and Technological Trends, Critical Reviews&Trade; in Biomedical Engineering 40 (3) (2012) 199–220.

[13] V. M. Banz, P. C. Müller, P. Tinguely, D. Inderbitzin, D. Ribes, M. Peterhans, D. Candinas, S. Weber, Intraoperative Image-Guided Navigation System: Development and Applicability in 65 Patients Undergoing Liver Surgery, Langenbeck's Archives of Surgery 401 (4) (2016) 495–502.

[14] F. Stehn, Geometric Hybrid Registration, Ph.D. thesis, Institut für Informatik, Freie Universität Berlin (2011).

[15] C. Knauer, M. Löffler, M. Scherfenberg, T. Wolle, The Directed Hausdorff Distance between Imprecise Point Sets, Theoretical Computer Science 412 (2011) 4173–4186.

[16] C. Knauer, F. Stehn, Elastic Geometric Shape Matching for Point Sets under Translations, in: Proceedings of Algorithms and Data Structures - 14th International Symposium (WADS), 2015, pp. 578–592.

[17] A. M. Bazen, S. H. Gerez, Fingerprint Matching by Thin-Plate Spline Modelling of Elastic Deformations, in: Pattern Recognition, Vol. 36, 2003, pp. 1859–1867.

[18] K. Rohr, H. Stiehl, R. Sprengel, T. Buzug, J. Weese, M. Kuhn, Landmark-Based Elastic Registration using Approximating Thin-Plate Splines, in: Medical Imaging, IEEE Transactions on, Vol. 20, 2001, pp. 526–534.

[19] A. Myronenko, X. Song, Point Set Registration: Coherent Point Drift, in: Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 32, 2010, pp. 2262–2275.

[20] H. Abdelmunim, A. Farag, Elastic Shape Registration Using an Incremental Free Form Deformation Approach with the ICP Algorithm, in: Canadian Conference on Computer and Robot Vision (CRV), 2011, pp. 212–218.

[21] S. Rusinkiewicz, M. Levoy, Efficient Variants of the ICP Algorithm, in: Proceedings of the Third International Conference on 3D Digital Imaging and Modeling, 2001, pp. 145–152.

[22] P. K. Agarwal, J. Pach, M. Sharir, State of the Union (of Geometric Objects): A Review, in: Discrete & Computational Geometry, 2007, pp. 9–48.

[23] A. Barvinok, D. S. Johnson, G. J. Woeginger, R. Woodroofe, Finding Maximum Length Tours Under Polyhedral Norms, in: Proceedings of IPCO VI, Lecture Notes in Computer Science 1412, Springer, 1998, pp. 195–201.

[24] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, 1st Edition, Springer Publishing Company, Incorporated, 2015.

[25] R. Fleischer, X. Xu, Computing Minimum Diameter Color-Spanning Sets, in: Proceedings of the 4th international conference on Frontiers in algorithmics, FAW'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 285–292.

[26] Y. Wang, Y. Xu, An Approximation Algorithm for the Smallest Color-Spanning Circle Problem , in: D. Xu, D. Du, D. Du (Eds.), Computing and Combinatorics, Springer International Publishing, Cham, 2015, pp. 171–182.

[27] M. Sharir, P. K. Agarwal, Davenport-Schinzel Sequences and their Geometric Applications, Cambridge University Press, New York, NY, USA, 1996.

[28] A. Itai, C. Papadimitriou, J. Szwarcfiter, Hamilton Paths in Grid Graphs, SIAM J. Comput. 11 (1982) 676–686.

[29] T. Eiter, H. Mannila, Computing Discrete Frechet Distance, Technical Report CD- TR 94/64 (05 1994).

[30] H.-K. Ahn, C. Knauer, M. Scherfenberg, L. Schlipf, A. Vigneron, Computing the Discrete Fréchet Distance with Imprecise Input, in: O. Cheong, K.-Y. Chwa, K. Park (Eds.), Algorithms and Computation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 422–433.

[31] A. Mosig, M. Clausen, Approximately Matching Polygonal Curves with Respect to the Fréchet Distance, Computational Geometry 30 (2) (2005)

113 – 127, special Issue on the 19th European Workshop on Computational Geometry.

[32] J. Kleinberg, E. Tardos, Algorithm Design, Pearson Education, Addison Wesley, 2006.

[33] R. B. Avraham, H. Kaplan, M. Sharir, A faster Algorithm for the Discrete Fréchet Distance under Translation (2015).

[34] P. M. Gruber, Chapter 1.10 - Aspects of Approximation of Convex Bodies, in: P. Gruber, J. Wills (Eds.), Handbook of Convex Geometry, North-Holland, Amsterdam, 1993, pp. 319 – 345.

[35] D. Muller, F. Preparata, Finding the intersection of two convex polyhedra, Theoretical Computer Science 7 (2) (1978) 217 – 236.

[36] S. Lützow, Implementation of ESM Problems under the 1-to-1– and Hausdorff Distance, M.Sc. Project, Universität Bayreuth, Germany (2016).

[37] S. Lützow, Deciding ESM Problem Instances under Various Distance Measures and Variable Underlying Metrics, M.Sc. Thesis, Universität Bayreuth, Germany (2017).

Own Publications

[A38] Christian Knauer, Luise Sommer, and Fabian Stehn. Elastic Geometric Shape Matching for Translations under Manhattan Norm. In Proceedings of the 31th European Workshop on Computational Geometry (EuroCG), 2015.

[A39] Christian Knauer, Luise Sommer, and Fabian Stehn. An FPTAS for an Elastic Shape Matching Problem with Cyclic Neighborhoods. In Computational Science and Its Applications – ICCSA 2018, pages 425–443, 2018.

[A40] Christian Knauer, Luise Sommer, and Fabian Stehn. Elastic Geometric Shape Matching for Translations under the Manhattan Norm. In Computational Geometry: Theory and Applications, 2018.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet habe.

Weiterhin erkläre ich, dass ich die Hilfe von gewerblichen Promotionsberatern bzw. -vermittlern oder ähnlichen Dienstleistern weder in Anspruch genommen habe, noch künftig in Anspruch nehmen werde.

Zusätzlich erkläre ich hiermit, dass ich keinerlei frühere Promotionsversuche unternommen habe.

Bayreuth, den 11. Juli 2022,