Efficient Algorithms for Shape and Pattern Matching

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Axel Mosig

aus

Düsseldorf

Bonn, Oktober 2003

Efficient Algorithms for Shape and Pattern Matching

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Axel Mosig

aus

Düsseldorf

Bonn, Oktober 2003

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Referent: Prof. Dr. Michael Clausen
2. Referent: Prof. Dr. Rolf Klein
Tag der Promotion:

Danksagung

Die vorliegende Arbeit ist entstanden im Rahmen der Arbeitsgruppe *Kommunikationssysteme und Algorithmen* am Institut für Informatik III der Universität Bonn. An dieser Stelle möchte ich all jenen Personen herzlich danken, die direkt oder indirekt zum Entstehen und Gelingen dieser Arbeit beigetragen haben.

Mein ganz besonderer Dank gilt Herrn Prof. Dr. Michael Clausen, dessen Denkanstöße und stete Diskussionsbereitschaft das Entstehen dieser Arbeit erst ermöglicht haben und der als Leiter der Arbeitsgruppe nicht nur in wissenschaftlicher Hinsicht eine große Unterstützung war.

Den Mitgliedern der Arbeitsgruppe, insbesondere Vlora Arifi, Rolf Bardeli, Roland Engelbrecht, Frank Kurth, Meinard Müller und Tido Röder ebenso wie Andreas Ribbrock und Michaela Schmitz, danke ich für zahlreiche hilfreiche Anmerkungen und Korrekturen sowie für das angenehme Arbeitsklima und sonstige Dinge, die dazu beitrugen, die Fertigstellung dieser Arbeit voranzutreiben.

Herrn Prof. Dr. Rolf Klein danke ich für die Übernahme des Korreferats.

Der Systemgruppe Altbau danke ich für die stets zuverlässige Verwaltung der Rechnersysteme, auf denen diese Arbeit größtenteils entstanden ist.

Weiter danke ich meinen Eltern, meinen Geschwistern sowie meinem Freundeskreis dafür, dass ich stets ein offenes Ohr finden konnte sowie für die unersetzliche menschliche Unterstützung während der vergangenen Jahre. Herrn Michael Meyer und den Herren Mikio Braun, Tobias Ebel, Daniel Hanisch und Tilman Lange danke ich darüber hinaus für geduldiges Korrekturlesen und einige wichtige Anmerkungen.

Schließlich Danke ich Herrn Suresh Venkatasubramanian für die Mühe, die damit verbunden war, die Implementierung des GRID-Algorithmus bereitzustellen, sowie für einige hilfreiche Anmerkungen zu diesem Algorithmus.

Contents

List of Figures

Chapter 1

Introduction

The problem of *shape matching* (sometimes also referred to as *pose estimation* or *pattern matching*) is a problem that raises naturally in many applications: suppose we have a reference pattern P and a query pattern Q, can we transform Q so that the transformed version of Q is similar to P? This general problem usually entails further questions:

- How can we model objects?
- What is a suitable class of admissible transformations?
- How can we define similarity between objects?

Finding answers to these questions usually depends on the application scenario as well as the availability of appropriate algorithms.
In this thesis, a general approach to certain pose estimation tasks is proposed. The focus is set on *geometric* patterns, i.e., the patterns are objects contained in a Euclidean vector space.

Modeling Objects. A straightforward way of modeling objects is to view an object as a set of points. As an example from chemistry, a molecule can be viewed as a set of atoms, where each atom has a coordinate in three-dimensional space. Another example comes from image processing: image data can be viewed as two-dimensional point sets by extracting a set of characteristic points (possibly with certain features containing color information attached to the points) from an image.
Another natural way of modeling objects is to describe objects as (polygonal) curves. From image data, for example, one can often extract the boundary of a shape resulting in an object described by a polygonal curve. In molecular biology, the *backbone* of a protein, i.e., the spatial arrangement of the amino acid sequence, can also be considered as a polygonal curve. Polygonal curves are one dimensional objects. The natural generalization of polygonal curves to higher dimensions are *simplical complexes*. Correspondingly, surfaces in three dimensions (the surfaces of molecules, for instance) are often modeled as two dimensional simplicial complexes that can be thought of as surfaces patched together from triangles.
The objects considered in this thesis are point sets and polygonal curves in a finite dimensional Euclidean vector space.

Classes of Transformations. The most simple class of transformations are *translations*, i.e., objects can be shifted in space, but not rotated, scaled or reflected. Rotations, scalings and reflections are other typical classes of transformations. Particularly important are *rigid*

motions that we obtain by composing translations and rotations. If we admit scalings in addition, we get the group of *similarity motions*. Another common class of transformations is the composition of translations and scalings, which is also known as *homothetic transformations*. All aforementioned classes of transformations have in common that they are *groups* in an algebraic sense: each transformation has an inverse transformation, there is a neutral transformation, and combining transformations is associative. Structural properties of these groups take an important place in different parts of this thesis. As a consequence, algebraic considerations play an important role.

Measuring Similarity between Objects. Instead of measuring similarity between two objects, we consider *distance measures* between objects, i.e., functions that rather measure a dissimilarity. Some of these distance measures satisfy the properties of a *metric*, others are closely related to distance measures that are metrics. Pose estimation (or pattern matching) based on distance measures has mostly been studied in the area of *computational geometry*. Many distance measures that have been considered in this area share common properties and are identified as one certain class of distance measures in this thesis. Furthermore, a new distance measure is introduced and its relations to formerly known ones examined.

Problems of pose estimation (or geometric pattern matching in general) have been studied in many different areas of computer science and its applications. One important field dealing with geometric pattern matching problems is *Computer Vision*. Classical methods for solving pose estimation problems that emerged from Computer Vision include the (Generalized) Hough Transform [14], Pose Clustering [63], the Alignment Method [44] or Geometric Hashing [48], that are sometimes also referred to as *voting schemes*.
As already mentioned above, pattern matching based on distance measures between objects has mostly been studied in the area of computational geometry. The first studies of such issues are due to Alt et al. [10], for a survey see [7]. This thesis mainly deals with problems typically studied in computational geometry. Whenever possible, we refer to related work that has been studied in other areas.

1.1 Overview and Results of this Thesis

Chapter 2. As a starting point for this thesis, Chapter 2 introduces basic concepts and notation that are needed in other chapters.

Chapter 3. This chapter uses the notion of *G-inverted lists* developed in [23, 22] and sets up relations between G-inverted lists and pattern matching with respect to the directed Hausdorff distance. This does not immediately lead to efficient algorithms. However, the notion of *transporter sets* that occurs in this context is a major foundation for the results and matching algorithms developed in Chapters 4 through 7.

Chapter 4. Based on the notion of transporter sets from Chapter 3, we study *intersections* of transporter sets. These intersections occur in the pattern matching tasks dealt with in Chapter 5 and can hence be seen as a preperation for the practical and theoretical results of Chapter 5. From a computational point of view, we deal with the problem of deciding whether the intersection of finitely many transporter sets is empty or not.
Major attention is put on characterizing the stuctures occuring in this context from an algebraic and algebraic geometry point of view. In particular, the observation that the groups

that are typically considered as transformation groups are *linear algebraic groups* allows to specify natural generalizations of some results known for particular groups to almost arbitrary (i.e., linear algebraic) groups in arbitrarily high dimensional spaces.

Chapter 5. The algorithmic results from Chapter 4 — deciding the emptiness of the intersection of transporter sets — are not of practical relevance. However, these intersections occur when matching with respect to distance measures belonging to the set of *relational distance measures*. Examples of relational distance measures are the Hausdorff- and the bottleneck distance. As the major result of Chapter 5, generic matching algorithms are developed that work for matching with respect to arbitrary relational distance measures under an arbitrary linear algebraic group.

Furthermore, we discuss certain properties of relational distance measures that allow us to state faster matching algorithms, sometimes at the cost of obtaining only an approximate solution of the problem. These improved algorithms generalize a number of ideas that were used to speed up matching algorithms for special distance measures and special transformation groups.

In a second part of Chapter 5, we provide an algorithm for matching with respect to the *root-mean-square Haudorff distance* (which does not belong to the class of relational distance measures) under arbitrary linear algebraic groups. To the best of the author's knowledge, these are the first results for larger groups than pure translations or rigid motions in the plane (which have been studied in [1] and [46], repsectively). Furthermore, these results generalize to a whole class of non-relational distance measures that evolve from relational distance measures by exchanging a certain norm that is part of many relational distance measures.

Chapter 6. The practical value of the algorithms from Chapter 5 is limited for large transformation groups of high dimension by the fact that it relies on techniques from real algebraic geometry. These are generally hard to implement for polynomials involving more than one variable. In fact, many of these techniques have not yet been implemented successfully, as discussed in Chapter 2. For the group of rigid motions in three dimensions, however, one can apply ideas known for matching with respect to the directed Hausdorff distance and generalize these for matching with respect to arbitrary relational distance measures. Again, one can make use of certain properties of relational distance measures for speeding up the matching algorithms.

Chapter 7. The topic of Chapter 7 is the Fréchet distance which is a well-known distance measure for (polygonal) curves. First, a discrete version of the Fréchet distance is introduced. After providing some combinatorial structures underlying this distance measure, we show that the discrete Fréchet distance is a pseudo-metric.

Furthermore, it is shown that this discrete version is upper bounded and lower bounded by the continuous version of the Fréchet distance arbitrarily tight by considering *oversampled* versions of polygonal curves. Due to these bounds, any algorithm for matching with respect to the discrete Fréchet distance yields an approximation algorithm for matching with respect to the continuous Fréchet distance.

Matching algorithms for the discrete Fréchet distance result from Chapter 5, since we are dealing with a relational distance measure. Furthermore, we show that the discrete Fréchet distance is related to the *Dynamic Time Warping Distance* (studied in the context of computer vision and handwriting recognition in [54]) in the same way as the root-mean-square

Hausdorff distance is related to the Hausdorff distance. Hence, we can use the ideas from Chapter 6 in order to design algorithms for matching with respect to the Dynamic Time Warping distance. This problem has been addressed before only under the group of translations, and the (rather theoretically relevant) results obtained in this thesis are the first ones for matching with respect to larger groups than pure translations or rigid motions in the plane.

Chapter 8. Many of the algorithms described in Chapters 3 through 7 can be implemented and have been tested in practice. A description of these implementations as well as some of the running times obtained in practice are presented in this final chapter.

Chapter 2

Basic Concepts and Notation

Many aspects of the problems arising in geometric pattern matching can be characterized algebraically. Therefore, this chapter summarizes some basic algebraic concepts. In particular, concepts from group theory and algebraic geometry will be introduced.

2.1 General Notation

Let $[x, y]$ denote the compact real interval between the two reals x and y; moreover, for integers a and b, let $[a : b]$ denote the set $\{a, a+1, \ldots, b\}$ of all integers between a and b. We also use $(a : b)$, $[a : b)$ and $(a : b]$ in order to denote open and "half open" integer intervals. Given two sets X and Y, Y^X denotes the set of all mappings from X to Y; for $f \in Y^X$ and $I \subseteq X$, we denote $f[I] := \{f(x) \mid x \in I\}$. Analogously, we denote $f^{-1}[J] := \{x \in X \mid f(x) \in J\}$ for $J \subseteq Y$. Since $x \in X^{[a:b]}$ is completely described by a sequence of $b - a + 1$ values in X, we also write $x = \langle x_a, \ldots, x_b \rangle \in X^{[a:b]}$. Let $V = \mathbb{R}^k$ denote the k-dimensional Euclidean vector space with the Euclidean norm $\|.\| := \|.\|_2$. For $P \in V^X$ for some set X, we define the norm $\|P\|_\infty := \sup_{x \in X} \|P(x)\|$.

2.2 Group Theory

Let G be a group and M an arbitrary set. A map $\circ\colon G \times M \to M$ is called a *group action* if $1_G \circ m = m$ for all $m \in M$ (where 1_G denotes the neutral element of G) and $(gh) \circ m = g \circ (h \circ m)$ for all $g, h \in G$ and $m \in M$. As a shorthand notation, we also write gm instead of $g \circ m$, and sometimes we will call a group G acting on a set a *transformation group* and the set M a *G-set*. The set $Gm := \{gm \mid g \in G\}$ is called the *G-orbit of m*. The G-orbits are the equivalence classes of the equivalence relation $m \sim m'$ iff $m' = gm$, for some $g \in G$. A set $R \subseteq M$ is called a *system of representatives* if the union of all G-orbits is M, i.e., if $M = \sqcup_{r \in R} Gr$, where $\sqcup$ denotes the disjoint union. Given a system of representatives R, there is a unique $r_m \in R$ for each $m \in M$ so that $m = g_m r_m$ for some $g_m \in G$. Note that in general, g_m is not unique – the so-called *stabilizer subgroup* $G_m := \{g \in G \mid gm = m\}$ may contain more than one element. If all elements of M are contained in the same G-orbit, i.e., $M = Gm$ for any $m \in M$, we say that G acts *transitively* on M.
As it will turn out to be useful for the pattern matching task considered in Chapters 4, 5 and 7, we study the structure of transformation groups. We write $U \leq G$ if U is a subgroup of G and $U < G$ if, in addition, $U \neq G$. A subgroup $N \leq G$ is called a *normal subgroup* if for all

$n \in N$ and $g \in G$, we have $gng^{-1} \in N$. If N is a normal subgroup of G, we write $N \trianglelefteq G$. If we have a group G with $N \trianglelefteq G$, $H < G$, $H \cap N = \{1\}$ and $G = \{nh \mid n \in N, h \in H\}$, we call G the *semidirect product* of N and H, denoted by $G = N \rtimes H$ for short. If $G = N \rtimes H$, each $g \in G$ can be decomposed uniquely into $g = nh$ with $n \in N$ and $h \in H$. Furthermore, we have $(nh)(n'h') = (nhn'h^{-1})(hh')$ (where $nhn'h^{-1} \in N$) and $(nh)^{-1} = (h^{-1}n^{-1}h)h^{-1}$ (where $h^{-1}nh \in N$).

Often, the G-set M is the k-dimensional Euclidean vector space $V := \mathbb{R}^k$ with the group $T(k, \mathbb{R})$ of translations acting on V. We are particularly interested in subgroups G of the affine group $\mathrm{AGL}(k, \mathbb{R})$ which is the semidirect product of the abelian normal subgroup $T(k, \mathbb{R})$ and the general linear group $\mathrm{GL}(k, \mathbb{R})$, i.e., $\mathrm{AGL}(k, \mathbb{R}) = T(k, \mathbb{R}) \rtimes \mathrm{GL}(k, \mathbb{R})$. We are usually interested in the case $G = T(k, \mathbb{R}) \rtimes H$ with $H \leq \mathrm{GL}(k, \mathbb{R})$. In typical applications, $V = \mathbb{R}^k$, and H is the group of rotations $\mathrm{SO}(k, \mathbb{R})$ or the group $\mathrm{SC}(k, \mathbb{R}) := \mathbb{R}_{>0} \cdot \mathrm{id}_k$ of uniform scalings (where id_k denotes the $k \times k$ unit matrix) or the product of these two groups

$$\mathrm{RS}(k, \mathbb{R}) := \{gh \mid g \in \mathrm{SO}(k, \mathbb{R}), h \in \mathrm{SC}(k, \mathbb{R})\}.$$

We denote $\mathrm{SE}(k, \mathbb{R}) := T(k, \mathbb{R}) \rtimes \mathrm{SO}(k, \mathbb{R})$ for the group of *rigid motions* (sometimes also referred to as the *special Euclidean group*), $\mathrm{HT}(k, \mathbb{R}) := T(k, \mathbb{R}) \rtimes \mathrm{SC}(k, \mathbb{R})$ for the group of *homothetic motions* and $\mathrm{SM}(k, \mathbb{R}) := T(k, \mathbb{R}) \rtimes \mathrm{RS}(k, \mathbb{R})$ for the group of *similarity motions*. Whenever we do not specify the ground field of these groups (for instance by writing $T(k)$), we implicitely assume that the ground field is $\mathbb{R}$ (i.e., $T(k) = T(k, \mathbb{R})$).

Typical pattern matching tasks involve a vector space V equipped with a metric d and a class of transformations, whose elements allow modifications of those objects; in most cases, we have $V = \mathbb{R}^k$ and the transformation group contains the group $T(k)$ of all translations in V as a subgroup. More formally, we have a group G (with neutral element 1) acting on V. A metric space (M, d) with a group G acting on M will be called a *metric G-space*. Sometimes, we require the transformation group G to be *invariant* with respect to the metric d, i.e., $d(gv, gw) = d(v, w)$, for all $g \in G$ and all $v, w \in V$.

Finally, for a set M, let $\mathcal{P}(M)$ denote the set of all subsets of M, and let $\mathcal{P}_f(M)$ denote the set of all *finite* subsets of M. If M is a G-set, both $\mathcal{P}(M)$ and $\mathcal{P}_f(M)$ become G-sets by defining $gA := \{ga \mid a \in A\}$ for $g \in G$ and $A \in \mathcal{P}(M)$ or $A \in \mathcal{P}_f(M)$, respectively.

2.3 Algebraic Geometry

We now introduce some concepts from algebraic geometry, and in particular concepts from real algebraic geometry, needed in Chapters 4 and 5. For comprehensive studies of algebraic geometry and real algebraic geometry, we refer to [26] and [17]. For the introduction of some basic concepts from Algebraic Geometry in the following, we mostly pursue the approaches from Humphreys [41].

2.3.1 Basic Concepts

Let $\mathbb{C}[T] := \mathbb{C}[T_1, \ldots, T_n]$ denote the algebra of n-variate complex polynomials. An *affine variety* in $\mathbb{C}^n$ is the set

$$\mathbf{V}(F) := \{x \in \mathbb{C}^n \mid \forall f \in F\colon f(x) = 0\}.$$

of common zeros of a subset F of $\mathbb{C}[T]$, so that $\mathbf{V}(F) \subseteq \mathbb{C}^n$. In turn, if X is a subset of $\mathbb{C}^n$, then

$$\mathbf{I}(X) := \{f \in \mathbb{C}[T] \mid \forall x \in X\colon f(x) = 0\}$$

is an ideal in $\mathbb{C}[T]$, the *vanishing ideal* of X. An ideal I of $\mathbb{C}[T]$ is called *radical* iff

$$I = \sqrt{I} := \{f \mid \exists r \geq 0\colon f^r \in I\}.$$

Theorem 2.3.1 (Hilbert's Nullstellensatz) *The operators $\mathbf{V}, \mathbf{I}$ set up a 1–1 inclusion-reversing correspondence between the collection of all radical ideals in $\mathbb{C}[T]$ and the collection of all affine varieties in $\mathbb{C}^n$.* □

As $\mathbf{V}(0) = \mathbb{C}^n$, $\mathbf{V}(\mathbb{C}[T]) = \emptyset$, $\mathbf{V}(I \cap J) = \mathbf{V}(I) \cup \mathbf{V}(J)$, and $\mathbf{V}(\sum_{\alpha \in A} I_\alpha) = \bigcap_{\alpha \in A} V(I_\alpha)$ (for ideals I, J, I_α of $\mathbb{C}[T]$), the affine varieties in $\mathbb{C}^n$ are the closed subsets of a topology on $\mathbb{C}^n$, the so-called *Zariski topology*. By Hilbert's Basis Theorem, this topology is *Noetherian*, i.e., every descending chain of closed sets becomes finally stationary. Thus every affine variety X has only finitely many maximal irreducible subsets $X_1, \ldots, X_n$ with $X = X_1 \cup \cdots \cup X_n$. (Recall that X_i is irreducible iff X_i cannot be written as the union of two proper, nonempty closed subsets.) It turns out that in this topology, a subset X of $\mathbb{C}^n$ is irreducible iff the vanishing ideal $\mathbf{I}(X)$ is prime. In this case, the algebra $\mathbb{C}[X] := \mathbb{C}[T]/\mathbf{I}(X)$, which can be viewed as the *algebra of polynomial functions on* X, is an integral domain. Its field of fractions, denoted $\mathbb{C}(X)$, is called the *field of rational functions* on X. The transcendence degree of the field extension $\mathbb{C}(X) \supset \mathbb{C}$ is called the *dimension* $\dim X$ of X. Roughly speaking, $\dim X$ is the minimal number of parameters to describe X.
A *morphism* of affine varieties $X \subseteq \mathbb{C}^n$ and $Y \subseteq \mathbb{C}^m$ is a map $\phi\colon X \to Y$ that is described by m polynomial functions $\psi_1, \ldots, \psi_m \in \mathbb{C}[X_1, \ldots, X_n]$, i.e., $\phi(x) = (\psi_1(x), \ldots, \psi_m(x))$, for all $x \in X$. If $X \subseteq \mathbb{C}^n$ and $Y \subseteq \mathbb{C}^m$ are affine varieties, then $X \times Y$ is Zariski-closed in $\mathbb{C}^{n+m}$ and its induced topology defines the *Zariski-product topology* on $X \times Y$. Let G be both a group and an affine variety. G is called a *linear algebraic group* iff the multiplication map $\mu\colon G \times G \to G$, $\mu(x, y) := x \cdot y$, and the inversion map $i\colon G \to G$, $i(g) := g^{-1}$, are both morphisms of varieties.
Within the scope of this thesis, the groups we are mainly interested in are linear algebraic groups. As an example (for the case that the ground field is $\mathbb{R}$), consider the group $\mathrm{SO}(2, \mathbb{R})$ of rotations in $\mathbb{R}^2$: Since every rotation can be identified with a point on the unit circle (and vice versa), we have a group structure on the set $S^1 = \{(s, c) \in \mathbb{R}^2 \mid s^2 + c^2 - 1 = 0\}$. Note that this parameter space is the set of roots of one polynomial. Before we get to subgroups of $\mathrm{GL}(k, \mathbb{R})$, we study subgroups of $\mathrm{GL}(k, \mathbb{C})$. As a group with a complex algebraic parameter space, consider the group $\mathrm{GL}(n, \mathbb{C})$, which is a principal open subset of $\mathbb{C}^{n \times n} = \mathbb{C}^{n^2}$ defined by the non-vanishing of the determinant:

$$\mathrm{GL}(n, \mathbb{C}) = \{A \in \mathbb{C}^{n \times n} \mid \det A \neq 0\}.$$

This group can be viewed as an affine variety in $\mathbb{C}^{n^2+1}$ defined by

$$\mathbf{V}(Y \cdot \det - 1) \equiv \mathrm{GL}(n, \mathbb{C}),$$

where Y is a new indeterminate. A linear algebraic group G is a Zariski-closed subgroup of some $\mathrm{GL}(n, \mathbb{C})$. Further examples of linear algebraic groups are the following:

1. *The group $T(k, \mathbb{C})$ of translations in $\mathbb{C}^k$.* For any $k > 0$, the underlying variety is $\mathbb{C}^k$ itself.

2. The groups $O(k,\mathbb{C})$ and $\mathrm{SO}(k,\mathbb{C})$. In case of $O(k,\mathbb{C})$, the underlying variety is a subset of $\mathbb{C}^{k^2}$ (i.e., a subset of all $k \times k$-matrices). The polynomials describing the variety result from the equality $M^\top M = \mathrm{id}_k$, where M is a $k \times k$-matrix. For $\mathrm{SO}(k,\mathbb{C})$, the variety is described by the polynomial equalities $M^\top M = \mathrm{id}_k$ and $\det M = 1$.

3. Semidirect products of linear algebraic groups. If $G = N \rtimes H$, where N and H are linear algebraic groups and H acts morphically on N, then G is a linear algebraic group (see [41], Section 8.4). As a result, the affine general linear group $\mathrm{AGL}(k,\mathbb{C}) = T(k,\mathbb{C}) \rtimes \mathrm{GL}(k,\mathbb{C})$ becomes a linear algebraic group.

4. Uniform scalings in $\mathbb{C}^k$. For the group of uniform scalings, we can identify the set $Z := \{\lambda \in \mathbb{C} \mid \lambda \neq 0\}$ with the group $\mathrm{GL}(1,\mathbb{C})$. An algebraic representation of Z into $\mathrm{GL}(k,\mathbb{C})$ for any $k > 0$ is given by $\rho((\lambda)) := \lambda \mathrm{id}_k$.

5. Rigid motions. Taking the semidirect product of $T(k,\mathbb{C})$ and $\mathrm{SO}(k,\mathbb{C})$, we obtain the group of rigid motions (sometimes also referred to as the special Euclidean group) in $\mathbb{C}^k$ denoted by $\mathrm{SE}(k,\mathbb{C})$.

6. Similarity motions. Let $n \in \mathrm{SO}(k,\mathbb{C})$ and $g \in \mathrm{RS}(k,\mathbb{C})$. Since $\det(gng^{-1}) = 1$, the group $\mathrm{RS}(k,\mathbb{C})$ is the semidirect product of $\mathrm{SC}(k,\mathbb{C})$ and $\mathrm{SO}(k,\mathbb{C})$. By Example 3, $\mathrm{RS}(k,\mathbb{C})$ is a linear algebraic group, too. We have $\mathrm{SM}(k,\mathbb{C}) = T(k,\mathbb{C}) \rtimes \mathrm{RS}(k,\mathbb{C})$, so that $\mathrm{SM}(k,\mathbb{C})$ is a linear algebraic group also.

So far, all considerations took place over the algebraically closed ground field $\mathbb{C}$. In the pattern matching scenario, the field of real numbers is more appropriate. So it is our next concern to transfer the above notions to the real field.

Let k be a subfield of $\mathbb{C}$. (We are mainly interested in the case $k = \mathbb{R}$.) A closed subset X of $\mathbb{C}^n$ is said to be *k-closed* if $X = \mathbf{V}(F)$ for some $F \subseteq k[T]$. The k-closed sets are the closed sets of a topology on $\mathbb{C}^n$, the *Zariski-k-topology*. X is said to be *defined* over k iff $\mathbf{I}(X)$ is generated by polynomials in $k[T]$. In characteristic 0, X is k-closed iff X is defined over k.

If $X \subseteq \mathbb{C}^n$ is k-closed, then $X(k) := X \cap k^n$ denotes the (possibly empty) set of its k-rational points. In the following, we will encounter numerous groups that are $\mathbb{R}$-rational points of certain linear algebraic groups.

For the group $\mathrm{SO}(3,\mathbb{R})$, there is a more convenient way to desribe the underlying variety than applying the aforementioned general algebraic description of $\mathrm{SO}(k,\mathbb{C})$ for $k = 3$. We have already seen that $\mathrm{SO}(2,\mathbb{R})$ can be parameterized algebraically by the unit circle. For $\mathrm{SO}(3,\mathbb{R})$, we use *unit quaternions* to specify an algebraic parameterization, as proposed in [49] and [61]. The set of all unit quaternions can be described by the variety $S^3 = \{u \in \mathbb{R}^4 \mid \|u\| = 1\}$.

A unit quaternion $u = (u_0, u_1, u_2, u_3)$ defines a rotation matrix as follows:

$$\rho(u_0,u_1,u_2,u_3) := \begin{pmatrix} 2(u_0^2+u_1^2)-1 & 2(u_1u_2-u_0u_4) & 2(u_1u_3+u_0u_2) \\ 2(u_1u_2+u_0u_3) & 2(u_0^2+u_2^2)-1 & 2(u_2u_3-u_0u_1) \\ 2(u_1u_3-u_0u_2) & 2(u_2u_3+u_0u_1) & 2(u_0^2+u_3^2)-1 \end{pmatrix}.$$

From now on, we only consider subgroups of $\mathrm{AGL}(d,\mathbb{R})$, and whenever the ground field is $\mathbb{R}$, we write $\mathrm{AGL}(d) := \mathrm{AGL}(d,\mathbb{R})$. Analogously, we ommit the ground field for all subgroups of $\mathrm{AGL}(k)$ defined above.

2.3.2 Real Algebraic Geometry

By $\mathbb{R}[X]$, we denote the ring of univariate polynomials over the reals, and by $\mathbb{R}[X_1, \ldots, X_k]$, we denote the set of all polynomials in the real variables $X_1, \ldots, X_k$; analogously, $\mathbb{R}(X)$ and $\mathbb{R}(X_1 \ldots, X_k)$ denote the field of all rational functions in variable X and variables $X_1, \ldots, X_k$ over the reals, respectively. For $x \in \mathbb{R}$, let $\operatorname{sign}(x) = 0$ if $x = 0$, $\operatorname{sign}(x) = 1$ if $x > 0$ and $\operatorname{sign}(x) = -1$ if $x < 0$. As in the complex case, one associates to every subset F of $\mathbb{R}[X_1, \ldots, X_k]$ the set of common zeroes $\mathbf{V}(F) = \{x \in \mathbb{R}^n \mid \forall f \in F \colon f(x) = 0\}$. In turn, every subset S of $\mathbb{R}^k$ defines its vanishing ideal

$$\mathbf{I}(S) := \{f \in \mathbb{R}[X_1, \ldots, X_k] \mid \forall x \in S \colon f(x) = 0\}.$$

In contrast to the complex case, for every affine variety $S \subseteq \mathbb{R}^k$ we have $S = \mathbf{V}(f)$: Since $\mathbf{I}(S)$ is finitely generated by Hilbert's Basis Theorem, we have $\langle f_1, \ldots, f_r \rangle = \mathbf{I}(S)$. Furthermore, defining $f := f_1^2 + \cdots + f_r^2$, we have $S = \mathbf{V}(\langle f_1, \ldots, f_r \rangle) = \mathbf{V}(f)$.
An *atomic polynomial expression* over $\mathbb{R}^k$ is a mapping $A \colon \mathbb{R}^k \to \{0, 1\}$ of the form $A(x) = 1 \Leftrightarrow P(x) \bowtie 0$ for all $x \in \mathbb{R}^k$, where $\bowtie \in \{=, <, >, \neq, \geq, \leq\}$ and $P \in \mathbb{R}[X_1, \ldots, X_k]$. A *polynomial expression* is a mapping $A \colon \mathbb{R}^k \to \{0, 1\}$ of the form $A(x) = A_1(x) \wedge \cdots \wedge A_\ell(x)$, where each A_i is an atomic polynomial expression. A set $S \subseteq \mathbb{R}^k$ is *semialgebraic* if, for some polynomial expression A, it can be written as

$$S = \{x \in \mathbb{R}^k \mid A(x) = 1\}.$$

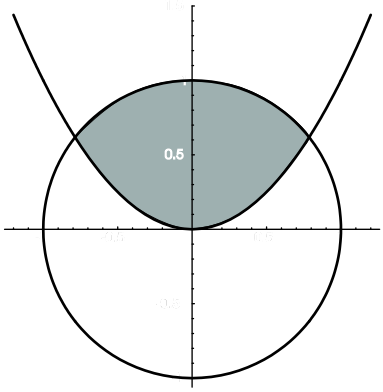


Figure 2.1: Example of a semialgebraic set $S = \{(x, y) \mid x^2 + y^2 - 1 \leq 0, y - x^2 \geq 0\}$.

Let $A \subseteq \mathbb{R}^k$ and $B \subseteq \mathbb{R}^m$ be semialgebraic. A mapping $f \colon A \to B$ is called a *semialgebraic function* if the set

$$\operatorname{graph}(f) := \{(x, y) \in A \times B \mid y = f(x)\}$$

is semialgebraic. We can also associate a dimension to every semialgebraic set, based on the following fact:

Theorem 2.3.2 *Every semialgebraic set $A \subseteq \mathbb{R}^k$ can be written in the form*

$$A = A_1 \sqcup \cdots \sqcup A_r,$$

where each A_i is a point or semialgebraically homoemorphic to $(0, 1)^{d_i}$.

Definition 2.3.3 *Using the notation from Theorem 2.3.2 above,*

$$d := \max_{i \in [1:r]} d_i$$

is called the (real) dimension *of* A.

Since real varieties are semialgebraic sets as well, the above definition assigns a dimension to each real variety as well. Note that for algebraic sets, this notion of dimension is equal to the Krull-dimension, which is defined as the maximum length of a chain of prime ideals

$$\mathbf{I}(A) \subsetneq p_1 \subsetneq \cdots \subsetneq p_r \subsetneq \mathbb{R}[X_1, \ldots, X_k]$$

Remark 2.3.4 *Semialgebraic sets and semialgebraic functions have the following properties:*

- *Semialgebraic sets are closed under (finite) intersections and unions.*
- *Let* $S \subseteq \mathbb{R}^k$ *be semialgebraic. Defining* $\mathrm{dist}_S \colon \mathbb{R}^k \to \mathbb{R}$, $x \mapsto \inf_{y \in S} \|x - y\|$ *for* $x \in \mathbb{R}^k$, *we get a semialgebraic function, since*

  $$\mathrm{graph}(\mathrm{dist}_S) \quad = \quad \{(x,t) \mid \forall y \in \mathbb{R}^k \colon y \in S \Longrightarrow \|x - y\| \geq t \text{ and } t \text{ minimal}\}$$

  is a set described by a first order formula (see Section 2.3.3).
- *Given a semialgebraic function* $f \colon \mathbb{R}^k \to \mathbb{R}$ *and a semialgebraic set* $S \subseteq \mathbb{R}^k$ *as well as some* $\varepsilon > 0$, *the set*

  $$\{x \in S \mid f(S) \leq \varepsilon\}$$

  is semialgebraic.
- *Given a semialgebraic function* $g \colon \mathbb{R}^k \to \mathbb{R}^\ell$ *and a semialgebraic set* $S \subseteq \mathbb{R}^{k-1}$, *the set*

  $$g(S, y) := \{g(x, y) \mid x \in S\}$$

  is a semialgebraic set.
- *Given a semialgebraic set* S *and a semialgebraic function* $f : S \to \mathbb{R}^k$, *the set*

  $$f[S] := \{f(s) \mid s \in S\}$$

  is a semialgebraic set.

2.3.3 Quantifier Elimination and Cell Enumeration

Certainly the most important result in conjunction with semialgebraic sets is the *Tarski-Seidenberg principle*, which states that quantified formulas over the reals (composed by $+, *, =$ and $>$) allow quantifier elimination, i.e., each such quantified formula defines a semialgebraic set. Algorithms performing the task of quantifier elimination are usually based on *cell enumeration* procedures that we focus on next.
Let $\sim$ be an equivalence relation defined over some set M. Then $\sim$ partitions M into equivalence classes. We call a set $X \subset M$ a $\sim$-*transversal* of M iff X contains precisely one element from each equivalence class and a $\sim$-*suptransversal* of M iff X contains at least one element from each equivalence class.

If we have a family of k-variate polynomials $\mathbf{P} = \langle P_i \rangle_{i \in I}$ for some index set I, we define the *sign vector of* $\mathbf{P}$ *at* $x \in \mathbb{R}^d$ as $\operatorname{sign}\mathbf{P}(x) := \langle \operatorname{sign}(P_i(x)) \rangle_{i \in I}$. We obtain an equivalence relation $\sim_{\mathbf{P}}$ (induced by $\mathbf{P}$) on $\mathbb{R}^k$ by defining $x \sim_{\mathbf{P}} x'$ iff x and x' lie in the same connected component of the set $\{y \in \mathbb{R}^k \mid \operatorname{sign}\mathbf{P}(y) = \operatorname{sign}\mathbf{P}(x')\}$. Such an equivalence class will also be referred to as a $\mathbf{P}$-*cell*. The set of all $\mathbf{P}$-cells partitions $\mathbb{R}^k$ into disjoint regions. By $\Sigma(\mathbf{P})$, we denote the set of all sign vectors whose $\mathbf{P}$-cells are non-empty, i.e.,

$$\Sigma(\mathbf{P}) = \{\sigma \in [-1:1]^I \mid \exists x \in \mathbb{R}^k : \sigma = \langle \operatorname{sign}(P_i(x)) \rangle_{i \in I}\}.$$

If $\mathbf{V}$ is a variety in $\mathbb{R}^k$, each family of polynomials $\mathbf{P}$ partitions $\mathbf{V}$ into equivalence classes. We refer to these equivalence classes as $(\mathbf{P}, \mathbf{V})$-cells. A set $X \subset \mathbf{V}$ will be called a $(\mathbf{P}, \mathbf{V})$-suptransversal if X contains at least one point from each $(\mathbf{P}, \mathbf{V})$-cell.
The following two problems related to semialgebraic sets and $\mathbf{P}$-cells will play an important role in Chapters 4 and 5:

1. *Emptyness of semialgebraic sets:* Given a semialgebraic set $S = \{x \in \mathbb{R}^k \mid A(x) = 1\}$ for some polynomial expression A, decide whether S is empty or not.

2. *Cell enumeration:* Given a family of k-variate real polynomials $\mathbf{P}$ and a real variety $\mathbf{V} \subseteq \mathbb{R}^k$, enumerate all $(\mathbf{P}, \mathbf{V})$-cells.

Obviously, every cell enumeration algorithm can be used to solve the emptyness problem. Numerous algorithms for solving these two problems have been proposed. The first implementable method for solving the cell enumeration problem was *Cylindrical Algebraic Decomposition*, which was introduced by Collins [24].

2.3.4 Cylindrical Algebraic Decomposition and Related Techniques

A Cylindrical Algebraic Decomposition is a particular way of partitioning $\mathbb{R}^k$ into connected semialgebraic sets. In the following definition, we write $f < g$ for $f, g \colon D \to \mathbb{R}$, iff $f(x) < g(x)$ for all $x \in D$. For $f, g \colon D \to \mathbb{R}$, we denote

$$\operatorname{band}(f, g) := \{(x, y) \in D \times \mathbb{R} \mid f(x) < y < g(x)\}.$$

Furthermore, note that each semi-algebraic subset of $\mathbb{R}^1$ can be decomposed as the union of finitely many points and open intervals.

Definition 2.3.5 *A* Cylindrical Algebraic Decomposition *(c.a.d. for short) is a sequence $C_1, \ldots, C_k$, where for each $i \in [1:k]$, C_i is a finite partition of $\mathbb{R}^i$ into semialgebraic subsets (called* cells*), satisfying the following three conditions:*

1. *each cell $c \in C_1$ is either a point or an open interval.*

2. *For every $i \in [1 : k-1]$ and for every $c \in C_i$, there are finitely many continuous semialgebraic functions*

$$-\infty =: f_{c,0} < \cdots < f_{c,\ell_c+1} := \infty, \quad f_{c,i} \colon c \to \mathbb{R},$$

with the property that the cylinder $c \times \mathbb{R}$ can be decomposed into the disjoint union

$$c \times \mathbb{R} = \bigsqcup_{j \in [1:\ell_c]} \operatorname{graph}(f_{c,j}) \;\sqcup\; \bigsqcup_{j \in [0:\ell_c]} \operatorname{band}(f_{c,j}, f_{c,j+1}).$$

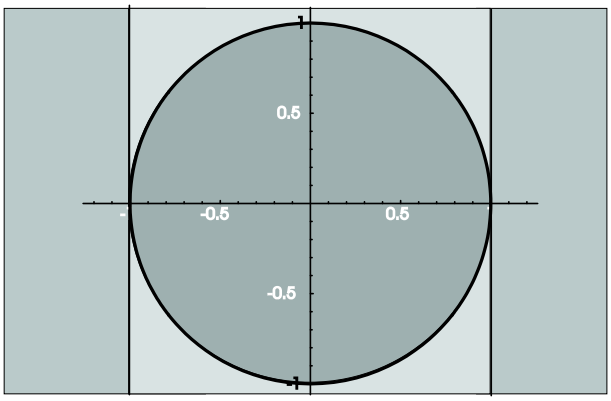


Figure 2.2: Example of a cylindrical algebraic decomposition adapted to $\mathbf{P} = \langle x^2 + y^2 - 1 \rangle$. We have $C_1 = \langle c_{1,1} := (-\infty, -1), c_{1,2} := \{-1\}, c_{1,3} := (-1, 1), c_{1,4} := \{1\}, c_{1,5} := (1, \infty) \rangle$ as well as $\ell_{c_{1,1}} = \ell_{c_{1,5}} = 0$, $\ell_{c_{1,2}} = \ell_{c_{1,4}} = 1$ and $f_{c_{1,2},1} = f_{c_{1,4},1} = 0$. Finally, we have $\ell_{c_{1,3}} = 2$, $f_{c_{1,3},1} = -\sqrt{1 - x^2}$ and $f_{c_{1,3},2} = \sqrt{1 - x^2}$. Altogether, this c.a.d. partitions $\mathbb{R}^2$ into 14 cells.

3. *C_{i+1} consists of all those* $\text{graph}(f_{c,j})$ *and all* $\text{band}(f_{c,j}, f_{c,j+1})$*, for* $c \in C_i$.

The concept of c.a.d. is particularly interesting if each cell of a c.a.d. is a subset of some $\mathbf{P}$-cell, where $\mathbf{P}$ is a family of polynomials. We call such a c.a.d. *adapted* to $\mathbf{P}$. In the sequel, the notion of Cylindrical Algebraic Decomposition will usually refer to a c.a.d. adapted to some $\mathbf{P}$. The time complexity for computing an adapted c.a.d. is *doubly exponential* in the number of variables, i. e., computing an adapted c.a.d. essentially requires $O(\ell^{2^d})$ time, ℓ denoting the number of polynomials and d the number of variables involved. Generally, c.a.d. is considered impractical for more than three variables. The time complexity of c.a.d. is due to the fact that the number of c.a.d.-cells is doubly exponential in the number of variables. So far, we know a bound for the number of cells of a c.a.d. adapted to $\mathbf{P}$. Now, a single $\mathbf{P}$-cell may be decomposed into *several* c.a.d.-cells, and hence the number of cells of a c.a.d. adapted to $\mathbf{P}$ can be significantly larger than the number of $\mathbf{P}$-cells. The question arises whether the bound for the (doubly exponential) number of c.a.d.-cells is also a tight bound for the actual number of all $\mathbf{P}$-cells, in other words: is Cylindrical Algebraic Decomposition an optimal description for the set of all $\mathbf{P}$-cells? As results on non-c.a.d.-based algorithms for quantifier elimination by Canny [18] and Renegar [57] as well as Basu, Pollack and Roy [16] suggest, the answer to this question is no — c.a.d. is not an optimal description for the set of all $\mathbf{P}$-cells. Basu, Pollack and Roy [16] obtain algorithms that are single exponential in the number of variables. Just as well, they (theoretically) provide algorithms for computing cell enumerations that are single exponential in the number of variables. Their results can be summarized as follows:

Theorem 2.3.6 *Let $d > 0$, $W \in \mathbb{R}[X_1, \ldots, X_d]$ and let d' denote the real dimension of the variety $\mathbf{V} = \{x \in \mathbb{R}^d \mid W(x) = 0\}$. Furthermore, let $U \in \mathbb{R}[X_1, \ldots, X_d]^I$ be a family of polynomials with $|I| =: \ell < \infty$. Define an equivalence relation on $\mathbf{V}$ by $x \sim_{U,\mathbf{V}} y$ iff for all $u \in U$ $\text{sign}(u(x)) = \text{sign}(u(y))$. If all $u \in U$ have degree at most D, then a $(U, \mathbf{V})$-suptransversal can be computed in $O(\ell^{d'+1} D^{O(d)})$ time. Furthermore, the number of cells is bounded by $O(\ell^{d'} O(D)^d)$.*

In Chapters 4 and 5, we will mostly be interested in semialgebraic sets that partition a *subvariety* of $\mathbb{R}^d$ rather than $\mathbb{R}^d$ itself into cells. The subvariety of $\mathbb{R}^d$ usually is a linear algebraic group.
An important issue that needs to be considered is that all non-c.a.d.-based methods for computing cell enumerations did not prove to be of practical value — in fact, currently available implementations of quantifier elimination such as the QEPCAD [25] and the c.a.d. implemented in Mathematica [64] rely on improvements of Collin's method [25] and not on the — at least in theory — asymptotically faster algorithms by Canny, Renegar or Basu et al. [18, 57, 16]. The running times for some of the matching algorithms in Chapters 4 and 5 based on these results should hence be considered as theoretical upper bounds only. Finding *practical* algorithms that match these time bounds remains an interesting challenge in real algebraic geometry and its applications.

2.3.5 Ordered Cell Enumeration

Sometimes it is useful and possible to enumerate the cells of a system of polynomials in a more systematic way. Given $\mathbf{P} = \langle P_i \rangle_{i \in I} \in \mathbb{R}[X_1, \ldots, X_k]^I$, we define a neighborhood relation on the set of all sign vectors (and hence on the set of all $\mathbf{P}$-cells) by defining two sign vectors to be neighbored iff the union of the corresponding $\mathbf{P}$-cells is connected and they differ in at most one sign condition. Taking each (non-empty) $\mathbf{P}$-cell as a vertex and the set of all pairs of neighbored cells as edges, we obtain a graph that we call the *cell graph of* $\mathbf{P}$, denoted by $\Gamma(\mathbf{P})$.

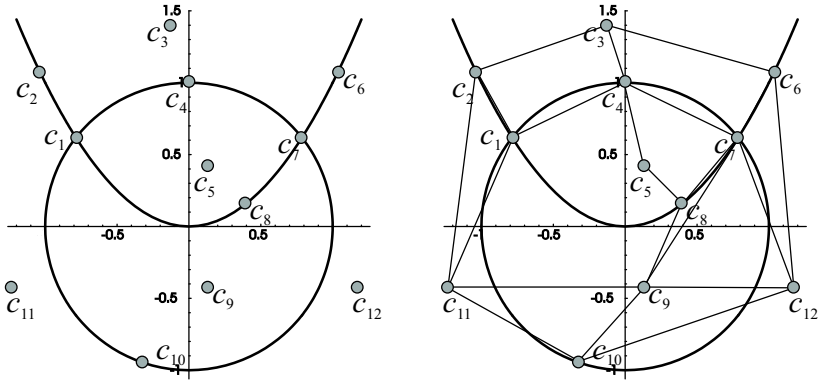


Figure 2.3: Example of a $\mathbf{P}$-suptransversal (left) and the corresponding cell graph $\Gamma(\mathbf{P})$ (right).

Now, let $\{\sigma_1, \ldots, \sigma_L\}$ be the set of all sign vectors whose $\mathbf{P}$-cells are non-empty. The output of a cell enumeration algorithm is a sequence $\langle \sigma_{\pi(1)}, \ldots, \sigma_{\pi(L)} \rangle$ for some permutation π. For solving some problems more efficiently, it is helpful if the sign vectors $\sigma_{\pi(i)}$ and $\sigma_{\pi(i+1)}$ differ in precisely one position. We formalize this type of cell enumeration as follows:

Definition 2.3.7 *Let I denote a finite index set and $\mathbf{P} \in \mathbb{R}[X_1, \ldots, X_d]^I$ a family of polynomials. Furthermore, let $\mathbf{V}$ be a subvariety of $\mathbb{R}^d$ and $L \in \mathbb{N}$. An* ordered $(\mathbf{P}, \mathbf{V})$-cell enumeration *(or* ordered $(\mathbf{P}, \mathbf{V})$-suptransversal*) is a pair*

$$(x_0, \langle (x_1, i_1), \ldots, (x_L, i_L) \rangle) \in \mathbb{R}^d \times (\mathbb{R}^d \times I)^{[1:L]},$$

such that

(1) *the set* $\{x_0, \ldots, x_L\}$ *is a* $(\mathbf{P}, \mathbf{V})$*-suptransversal and*

(2) *for all* $j \in [0 : L-1]$*, the sign vectors* $\operatorname{sign} \mathbf{P}(x_j)$ *and* $\operatorname{sign} \mathbf{P}(x_{j+1})$ *differ only in position* i_{j+1}*.*

Trivially, one can take the output of an arbitrary (non-ordered) cell enumeration algorithm, set up $\Gamma(\mathbf{P})$ and then produce an ordered sequence of sign vectors using depth-first-search or breadth-first-search. However, setting up $\Gamma(\mathbf{P})$ is prohibitively expensive, since it takes $O(L^2)$ time, L denoting the cardinality of the suptransversal. There are two special classes of polynomials where ordered cell enumeration can be done more efficiently:

- If the polynomials in $\mathbf{P}$ are univariate, the computed cell representatives can be sorted in $O(L \log L)$ time. Enumerating these cells in the order obtained by sorting yields an ordered sequence of sign conditions.
- If the polynomials in $\mathbf{P}$ are linear (i.e., the polynomials' degree is 1), the cells of $\mathbf{P}$ are polytopes. For this special case, other methods for cell enumeration than c.a.d. and its relatives are known; *reverse search* from [62] is an ordered cell enumeration algorithm for polytopes.

Ordered cell enumeration will be useful for providing more efficient matching algorithms in Chapters 4 and 5.

Chapter 3

Pattern Matching using G-Inverted Lists

The notion of G-inverted lists in [22, 23] provides a group-theoretical approach to pattern matching. The concept of G-inverted lists generalizes inverted files [45] that are used in full-text retrieval. G-inverted lists are based on group theoretical concepts and have been applied in various applications and prototypes. The applications range from music- and audio retrieval ([59]) to image databases [58] and indexing three dimensional surfaces [52]. Thus, G-inverted lists build a bridge between full-text retrieval and a large class of pattern matching problems. The results presented in this chapter extend the concept of G-inverted lists to problems that are typically considered in *geometric* pattern matching.
We present a connection between G-inverted lists and the directed Hausdorff distance which is a state-of-the-art distance measure used in geometric pattern matching (see [42, 21] and [7] for a survey on related problems). Although the results presented in this chapter do not immediately yield practical and efficient algorithms, they form the basis for the results obtained in all later chapters – all pattern matching methods presented in this work share the property that in some way subsets of transformation groups are intersected.
We first give a brief survey on G-matches and G-inverted lists (that are used for efficiently computing G-matches), as presented in [23, 22]. The notion of G-matches will then be generalized to (G, ε)-matches and (G, ε)-inverted lists, which are immediately related to the directed Hausdorff distance.

3.1 G-Matches and G-Inverted Lists

As a general scenario for pattern matching, we view a pattern P as well as a query Q as a finite subset of some set M, i.e., $P \in \mathcal{P}_f(M)$, where $\mathcal{P}_f(M)$ denotes the set of all finite subsets of M.
Let G be a group and M a G-set. Since G also acts on $\mathcal{P}_f(M)$, it also acts on a pattern P and a query Q.
In [23, 22], the scenario is generalized to a database of N patterns $P_1, \ldots, P_N$. In the following, we restrict ourselves to the situation where only a single pattern P is given, which simplifies notation significantly. We also omit preprocessing steps for the document P which can formally be treated as G-morphisms, as demonstrated in [23, 22].

We now define the set of all *G-matches of Q with respect to P* as the set of all group elements g that move Q so that the transformed version gQ is a subset of P, or, more formally,

$$G_P(Q) := \{g \in G \mid gQ \subseteq P\}. \tag{3.1}$$

For $m \in M$, we define the *G-inverted list of m* as

$$G_P(m) := G_P(\{m\}) = \{g \in G \mid gm \in P\}, \tag{3.2}$$

and it can be shown that for any $h \in G$

$$G_P(hm) = G_P(\{m\})h^{-1} = \{gh^{-1} \in G \mid gm \in P\}. \tag{3.3}$$

Given a system R of representatives of the G-orbits of M, the G-inverted list of m can now be written as $G_P(r_m)g_m^{-1}$, where $m = g_m r_m$ and $r_m \in R$ denotes the representative of the G-orbit of m. Thus, G-inverted lists only need to be computed for representatives $r \in R$. From these lists, G-inverted lists for arbitrary $m \in M$ can be computed.
Observe that the set of G-matches of a query Q can be written as

$$G_P(Q) = \bigcap_{q \in Q} G_P(q), \tag{3.4}$$

so that using Eq. (3.3) the set of G-matches can be computed using the intersection

$$G_P(Q) = \bigcap_{q \in Q} G_P(r_q)g_q^{-1}. \tag{3.5}$$

In terms of constructing an index (i.e., the set of all G-inverted lists for a pattern P), this means that we only need to compute G-inverted lists for each representative $r \in R$; to be more precise, an inverted list is constructed for each $r \in \{s \in R \mid \exists g \in G\colon gs \in P\}$. For computing G-matches, the intersection in Eq. (3.5) can be computed efficiently using binary search based on an ordering of G; in case that the operation of the group preserves this order (i.e., $g_1 < g_2 \iff g_1 h < g_2 h$ for all $g_1, g_2, h \in G$), the intersection can be performed even more efficiently; also, some results about adaptive set intersections can be applied, see [27, 15]. In [23, 22], some extensions of the general framework of G-matches and G-inverted lists are described for allowing different types of fault tolerance:

- The notion of G-matches can be extended to (G, k)-matches so that $|gQ \setminus P| \leq k$. Answering this type of query is also referred to as *k-mismatch searching*.
- Fuzzy queries can be specified. A *fuzzy query* is a family $(F_i)_{i \in I}$ of subsets of M; a G-match $g \in G$ of a fuzzy query transforms at least one element of each fuzzy set F_i into an element of P: $gF_i \cap P \neq \emptyset$, for all $i \in I$.
- Further fault tolerance can be achieved by assigning a probability distribution over each fuzzy set contained in a fuzzy query.

Many methods for solving geometric pattern matching problems use certain *distance measures* such as the Hausdorff or the bottleneck distance (see [65, 7] for detailed surveys) for specifying the fault tolerance. Thus, it is our next concern to examine in which ways G-matches are related to pattern-matching with respect to these distance measures.

3.1.1 Pattern Matching in G-Sets

We are now prepared to set up some general notion for the pattern matching tasks considered in this work. To this end, let G be a group, M a G-set and $\mathbf{d}$ a mapping $\mathbf{d} : M \times M \to \mathbb{R}_{\geq 0}$. M will usually be some object space, like the set of all finite point sets in the plane or the set of all polygonal curves in some Euclidean vector space. The mapping $\mathbf{d}$ serves as a distance function, measuring the "dissimilarity" of two objects in M; $\mathbf{d}$ is not necessarily a metric.
In this scenario, we are usually given two objects $P, Q \in M$ as well as a fault tolerance parameter $\varepsilon \geq 0$ and want to gain some information about the set of all $(G, \varepsilon, \mathbf{d})$-*matches of* Q *with respect to* P defined as

$$G(P, Q, \varepsilon, \mathbf{d}) := \{g \in G \mid \mathbf{d}(gQ, P) \leq \varepsilon\}. \tag{3.6}$$

Usually we want to decide whether $G(P, Q, \varepsilon, \mathbf{d})$ is empty or not and, if not empty, we want to find some $g \in G(P, Q, \varepsilon, \mathbf{d})$.

3.1.2 (G, ε)-Matches and (G, ε)-Inverted Lists

The most immediate connection from G-matches to distance measures used in geometric pattern matching is the relation to the *directed Hausdorff distance* d_H, which is defined for two compact (in particular finite) subsets P, Q of a metric space (M, d) as

$$d_{\mathrm{H}}(Q, P) := \max_{q \in Q} \min_{p \in P} d(q, p). \tag{3.7}$$

We also say that d_{H} is the *Hausdorff distance with respect to* d, since in some cases there might be more than one metric defined on M. As opposed to the undirected Hausdorff distance, which is defined as

$$\mathbf{d}_{\mathrm{H}}(P, Q) := \max\{d_{\mathrm{H}}(P, Q), d_{\mathrm{H}}(Q, P)\}, \tag{3.8}$$

the directed Hausdorff distance does *not* define a metric, since it does not satisfy the triangle inequality. However, it is the directed version that is usually studied in pattern matching, since Q can be thought of as an approximate subconstellation of P if $d_H(Q, P)$ is small. In particular, we have $Q \subseteq P$ iff $d_H(Q, P) = 0$. This motivates the following definition of (G, ε)-*matches of* Q *with respect to* P:

$$G_P^\varepsilon(Q) := \{g \in G \mid d_H(gQ, P) \leq \varepsilon\}. \tag{3.9}$$

An easy computation shows that $G_P^0(Q) = G_P(Q)$; thus (G, ε)-matches generalize the notion of a G-match. At the same time, (G, ε)-matches are a special type of fuzzy queries and are thus also a special case of G-matches.
There are basic differences between G-matches and (G, ε)-matches. If the stabilizer of each $m \in M$ is finite, then $G_P(Q)$ is a finite set as well. We will see, however, that $G_P^\varepsilon(Q)$ consists of infinitely (and even uncountably infinitely) many group elements in many cases even if the stabilizers of all $m \in M$ are finite. In these cases, it does not make sense to compute the set of all (G, ε)-matches. Instead we focus on the *decision problem* by asking whether $G_P^\varepsilon(Q)$ is empty or not; in case $G_P^\varepsilon(Q) \neq \emptyset$ we are also interested in computing at least one single $g \in G_P^\varepsilon(Q)$ as a "witness" for the non-emptiness of the set of all (G, ε)-matches.
The *optimization problem* asking for the smallest ε so that $G_P^\varepsilon(Q) \neq \emptyset$ will not be considered further; it has, however, been studied for example in [36]. There is also work dealing with

the decision problem; comparing the results from [36] and other related work with the ones obtained in this thesis is left to Chapter 5, where we develop different matching algorithms with respect to the directed Hausdorff distance.
It seems reasonable to solve our decision problem by techniques similar to those developed for computing G-matches. This motivates us to define the (G,ε)-*inverteted list of* $m \in M$ as

$$G_P^\varepsilon(m) := G_P^\varepsilon(\{m\}) = \{g \in G \mid d_H(\{gm\}, P) \leq \varepsilon\}, \tag{3.10}$$

which is equivalent to writing

$$G_P^\varepsilon(m) = \{g \in G \mid \exists p \in P\colon d(gm,p) \leq \varepsilon\},$$

since $d_H(\{gm\},P) \leq \varepsilon \iff \exists p \in P\colon d(gm,p) \leq \varepsilon$. Thus, the equivalence

$$g \in G_P^\varepsilon(m) \quad \iff \quad \exists p \in P\colon d(gm,p) \leq \varepsilon \tag{3.11}$$

holds, which we will need in the following. Again, it can easily be seen that $G_P^0(m) = G_P(m)$ for all $m \in M$.
Let us now study the properties of (G,ε)-inverted lists and their use for algorithms for the decision problem stated above. As we will see, properties analogous to those of G-inverted lists hold for (G,ε)-inverted lists.

Lemma 3.1.1 *For $Q \in \mathcal{P}_f(M)$, we have*

$$G_P^\varepsilon(Q) = \bigcap_{q \in Q} G_P^\varepsilon(q). \tag{3.12}$$

Proof. We have

$$\begin{aligned}
g \in G_P^\varepsilon(Q) &\iff d_H(gQ,P) \leq \varepsilon \\
&\iff \max_{q \in gQ} \min_{p \in P} d(q,p) \leq \varepsilon \\
&\iff \max_{q \in Q} \min_{p \in P} d(gq,p) \leq \varepsilon \\
&\iff \forall\, q \in Q \,:\, \min_{p \in P} d(gq,p) \leq \varepsilon \\
&\iff \forall\, q \in Q \,:\, d_H(\{gq\},P) \leq \varepsilon \\
&\iff \forall\, q \in Q \,:\, g \in G_P^\varepsilon(q),
\end{aligned}$$

in other words: $G_P^\varepsilon(Q) = \bigcap_{q \in Q} G_P^\varepsilon(q)$. □
This property is a generalization of Eq. (3.4). We will now see that Eq. (3.5) can also be generalized to (G,ε)-inverted lists in a straightforward way:

Lemma 3.1.2 *For $g \in G$ and $m \in M$, we have*

$$G_P^\varepsilon(gm) = G_P^\varepsilon(m)g^{-1}.$$

Proof.

$$\begin{aligned}
x \in G_P^\varepsilon(gm) &\iff d_H(\{xgm\},P) \leq \varepsilon \\
&\iff xg \in G_P^\varepsilon(m) \\
&\iff \exists\, h \in G_P^\varepsilon(m) \,:\, h = xg \\
&\iff \exists\, h \in G_P^\varepsilon(m) \,:\, x = hg^{-1} \\
&\iff x \in G_P^\varepsilon(m)g^{-1}.
\end{aligned}$$

□

Thus, the set of all (G, ε)-matches of Q with respect to P can be written as

$$G_P^\varepsilon(Q) = \bigcap_{q \in Q} G_P^\varepsilon(r_q) g_q^{-1}, \tag{3.13}$$

where for a system of representatives $R \subset M$ of the G-orbits of M, we write $q = g_q r_q$ with $r_q \in R$ for each $q \in Q$.

3.1.3 Construction of (G, ε)-Inverted Lists

So far, we have seen how (G, ε)-matches can be described as the intersection of (G, ε)-inverted lists. However, we have not said anything yet about the construction of the inverted lists involved in the intersection. Therefore, we present a number of characterizations of (G, ε)-inverted lists that are of some use in constructing these lists and, later on, in actually computing the intersection from Eq. (3.13) with the help of techniques developed in Chapter 4. In order to see that this task requires some effort consider the example shown in Figure 3.1: let $G = \mathrm{SO}(2, \mathbb{R})$ act on $M = \mathbb{R}^2$ by matrix-vector multiplication. Then, $R := \mathbb{R}_{\geq 0} \times \{0\}$ is a system of representatives. Given a pattern $P := \{p_1\}$, $p_1 := (0, 2)$, a query $Q := \{q_1\}$, $q_1 := (-\frac{3}{2}, 0)$ and $\varepsilon := 1$, we observe that $G_P^\varepsilon(q_1) \neq \emptyset$, although the points p_1 and q_1 lie in different G-orbits. The results provided in this section will help to understand how the (G, ε)-inverted lists are constructed both in this example and in general.

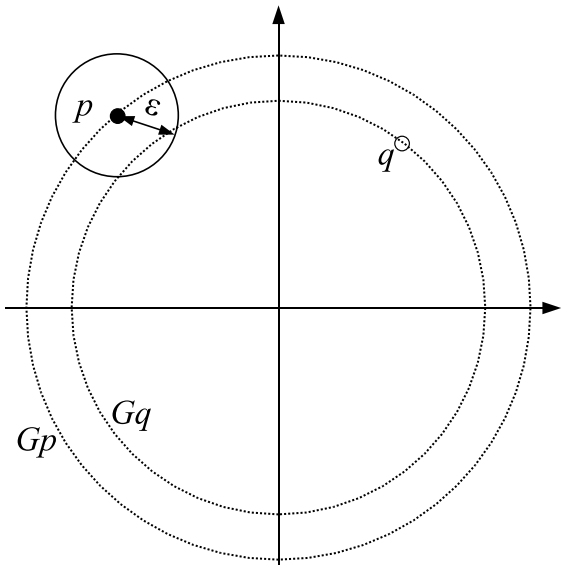


Figure 3.1: Elements from different G-orbits of $M = \mathbb{R}^2$ match in case $G = \mathrm{SO}(2)$.

Before characterizing $G_P^\varepsilon(r_q)$ formally, we define the set

$$R_P := \{r \in R \mid \exists\, p \in P, g \in G : p = gr\} \subseteq R$$

for a point set $P \in \mathcal{P}_f(M)$, i.e., the set of all representatives whose G-orbits contain elements of P.

A major role in the characterization of (G, ε)-inverted lists will be taken by *transporter subsets* of the transformation group G. Given two subsets A and B of the G-set M, we define the *G-transporter from A to B* as

$$\mathrm{Tran}_G(A, B) := \{g \in G \mid gA \subseteq B\}; \tag{3.14}$$

analogously, we define for $a, b \in M$

$$\begin{aligned} \mathrm{Tran}_G(a, B) &:= \{g \in G \mid ga \in B\} \text{ and} \\ \mathrm{Tran}_G(a, b) &:= \{g \in G \mid ga = b\}. \end{aligned} \tag{3.15}$$

As it turns out, the lists $G_P^\varepsilon(s)$ to be constructed consist of building blocks

$$\tau_{x,y}^{G,\varepsilon} := \{h \in G \mid d(hx, y) \leq \varepsilon\} \tag{3.16}$$

$$= \mathrm{Tran}_G(x, U_\varepsilon(y)) \tag{3.17}$$

that we will refer to as the *(G, ε)-transporter from x to y*. Thus $\tau_{x,y}^{G,\varepsilon}$ denotes the set of all group elements $h \in G$ that may act on $x \in M$, so that the transformed element hx is contained in the ε-neighborhood of the second object y.
If d is *G-invariant*, i.e., if $d(x, y) = d(gx, gy)$ for all $x, y \in M$ and $g \in G$, these transporter subsets can be characterized by representatives of x and y; therefore, let $r, s \in R$ and $g, h \in G$, so that $x = gr$ und $y = hs$. Then we get:

$$\begin{aligned} \tau_{x,y}^{G,\varepsilon} &= \{a \in G \mid d(agr, hs) \leq \varepsilon\} \\ &\overset{d\ G\text{-invariant}}{=} \{a \in G \mid d(h^{-1}agr, s) \leq \varepsilon\} \\ &\overset{b:=h^{-1}ag}{=} \{hbg^{-1} \in G \mid d(br, s) \leq \varepsilon\} \\ &= h\{b \in G \mid d(br, s) \leq \varepsilon\}g^{-1} \\ &= h\tau_{r,s}^{G,\varepsilon}g^{-1}. \end{aligned} \tag{3.18}$$

Furthermore, the *complex product* of two subsets of a group will simplify the characterization of (G, ε)-inverted lists. The complex product $X \cdot Y$ of $X, Y \subseteq G$ is defined as

$$X \cdot Y := \{xy \mid x \in X, y \in Y\}$$

and, as can easily be shown, has the property

$$X \cdot Y = \bigcup_{x \in X} xY. \tag{3.19}$$

These definitions finally allow us to characterize $G_P^\varepsilon(s)$ for any $s \in M$.

Lemma 3.1.3 *Let M be a G-set and let $d\colon M \times M \to \mathbb{R}_{\geq 0}$ be G-invariant. Then,*

$$G_P^\varepsilon(s) = \bigcup_{r \in R_P} G_P(r) \cdot \tau_{s,r}^{G,\varepsilon}. \tag{3.20}$$

Proof. First we note that for arbitrary $g, h \in G$ and $r, s \in R$, we have

$$d(hs, gr) \leq \varepsilon \iff h \in g\tau_{s,r}^{G,\varepsilon}, \tag{3.21}$$

as the following equivalences show:

$$\begin{aligned} d(hs, gr) \leq \varepsilon &\overset{d\ G\text{-invariant}}{\iff} d(g^{-1}hs, r) \leq \varepsilon \\ &\overset{(3.16)}{\iff} g^{-1}h \in \tau_{s,r}^{G,\varepsilon} \\ &\iff h \in g\tau_{s,r}^{G,\varepsilon}. \end{aligned}$$

Furthermore we note that for arbitrary $p, p' \in M$ with $p' = hp$ for some $h \in G$, we have

$$\{p\} = \bigcup_{g \in \mathrm{Tran}_G(p', p)} \{gp'\}. \tag{3.22}$$

Now we get

$$\begin{aligned} G_P^{\varepsilon}(s) &= \{h \in G \mid d_{\mathrm{H}}(\{hs\}, P) \leq \varepsilon\} \\ &= \{h \in G \mid \exists p \in P \colon d(hs, p) \leq \varepsilon\} \\ &= \bigcup_{p \in P} \{h \in G \mid d(hs, p) \leq \varepsilon\} \\ &\overset{(3.22)}{=} \bigcup_{p \in P} \bigcup_{g_p \in \mathrm{Tran}_G(r_p, p)} \{h \in G \mid d(hs, g_p r_p) \leq \varepsilon\} \\ &\overset{(3.21)}{=} \bigcup_{p \in P} \bigcup_{g_p \in \mathrm{Tran}_G(r_p, p)} \{h \in G \mid h \in g_p \tau_{s,r_p}^{G,\varepsilon}\} \\ &= \bigcup_{p \in P} \bigcup_{g_p \in \mathrm{Tran}_G(r_p, p)} g_p \tau_{s,r_p}^{G,\varepsilon} \\ &= \bigcup_{r \in R_P} \bigcup_{g \in G_P(r)} g\tau_{s,r}^{G,\varepsilon} \\ &\overset{(3.19)}{=} \bigcup_{r \in R_P} G_P(r) \cdot \tau_{s,r}^{G,\varepsilon}. \end{aligned}$$

□

Remark 3.1.4 *The union ranging over all representatives R_P stated in the preceding lemma can be further simplified; it suffices to let the union range over representatives in the ε-*neighborhood *of s. This requires a suitable distance between representatives. In fact, it makes sense to use the Hausdorff distance between the orbits of $r, s \in R$, i.e., $d_O(r, s) := d_H(Gr, Gs)$. (Note that this requires the orbits to be compact.) Defining the closed neighborhood $U_O(\varepsilon, r) := \{s \in R \mid d_O(r, s) \leq \varepsilon\}$, one can show that it suffices to let the union range over $R_P \cap U_O(\varepsilon, s)$ instead of the whole set of representatives R_P.*

3.2 Examples

We will now consider simple examples of (G, ε)-inverted lists for two different G-sets M. For more complex groups than the group of translations in $\mathbb{R}^d$ or the group of rotations in the plane – the group of rigid motions, for example – the construction of G-inverted lists becomes

more complex and leads to difficult intersection problems; for these more complex groups, it does not make sense to compute (G, ε)-inverted lists in a precomputed index. However, the intersections to be computed still yield efficient algorithms that will be studied in Chapters 4 and 5.
From the results of the last section, we know that, given a G-set M and a G-invariant distance function, we have to follow three steps for characterizing (G, ε)-inverted lists:

1. Identify a system of representatives R.

2. Characterize $\tau_{r,s}^{G,\varepsilon}$ for $r, s \in R$.

3. Characterize the effect of left-multiplication by some $g \in G$ and right-multiplication by some $h \in G$, i.e., characterize $g\tau_{r,s}^{G,\varepsilon}h$.

Complete (G, ε)-inverted lists are then characterized as the union of several building blocks of the type $g\tau_{r,s}^{G,\varepsilon}$. For computing (G, ε)-matches, a right multiplication is performed on the (G, ε)-inverted lists involved, and thus building blocks of the type $g\tau_{r,s}^{G,\varepsilon}h$ need to be intersected.

3.2.1 Translations in $\mathbb{R}^2$

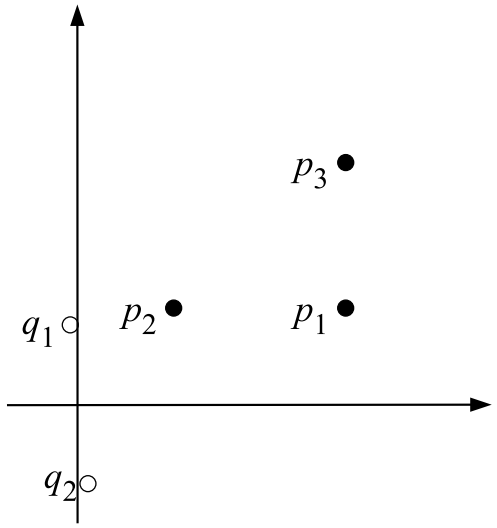
Let $M = \mathbb{R}^2$ and let $G := T(2)$ (where $T(k)$ denotes the group of all translations in $\mathbb{R}^k$, as introduced in Chapter 2). Furthermore, as the metric underlying the Hausdorff distance we use $d := d_\infty$, where d_∞ denotes the metric induced by the norm $\|.\|_\infty$. The metric d_∞ is $T(2)$-invariant, so we can follow the three steps stated in the beginning of this section:

1. Since $T(2)$ acts transitively on $\mathbb{R}^2$, there is only one orbit. Hence, we choose $R := \{(0,0)\}$ as a straight-forward system of representatives for the single $T(2)$-orbit of $\mathbb{R}^2$.

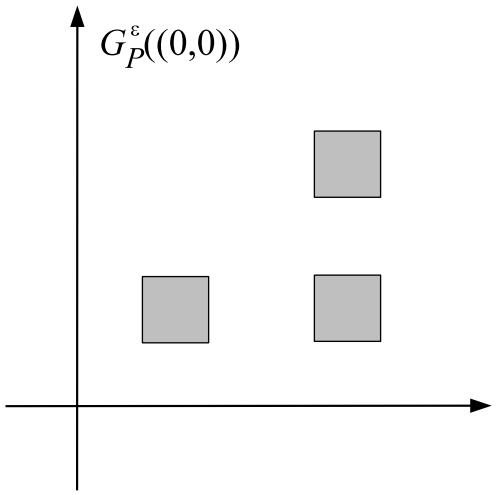
2. We only have one orbit, so we only need to characterize $\tau_{(0,0),(0,0)}^{T(2),\varepsilon}$. The set of all translations $T(2)$ can be identified with $\mathbb{R}^2$, and $\tau_{(0,0),(0,0)}^{T(2),\varepsilon}$ can easily be identified with the square
$$\tau_{(0,0),(0,0)}^{T(2),\varepsilon} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq \varepsilon\}$$
which is centered at the origin.

3. Multiplying such square with some $g = (g_1, g_2) \in T(2)$ simply moves the square by g_1 in the x component and g_2 in the y-component – no matter if the multiplication is from the left or from the right, since $T(2)$ is abelian.

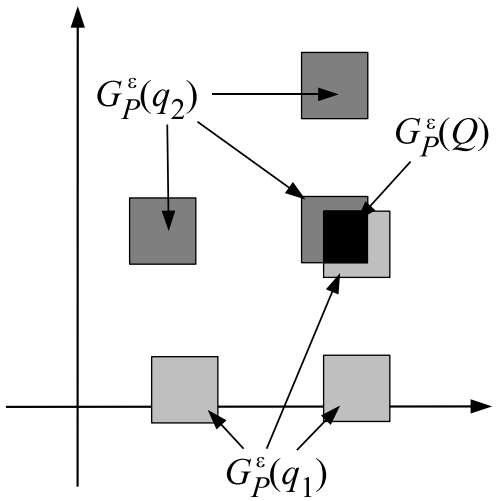
Due to these three properties, computing the set of (G, ε)-matches of $Q = \{q_1, \ldots, q_n\}$ with respect to $P = \{p_1, \ldots, p_m\}$ only requires a family of n sets of rectangles, each of which contains m equal-sized squares, to be intersected. A complete example is shown in Figure 3.2. For solving this rectangle intersection problem algorithmically, methods based on orthogonal range searching exist, see [29].
Note that the methods described here in two dimensions also work in higher dimensions, i.e., for $M = \mathbb{R}^k$ and $G = T(k)$.



A pattern $P = \{p_1, p_2, p_3\} \subset \mathbb{R}^2$ and a query $Q = \{q_1, q_2\} \subset \mathbb{R}^2$.



(G, ε)-inverted list $G_P^\varepsilon((0,0))$. Since translations act transitively on $\mathbb{R}^2$, it suffices to construct one (G, ε)-inverted list for the point $(0, 0)$.



Construction of the (G, ε)-matches $G_P^\varepsilon(Q) = G_P^\varepsilon(q_1) \cap G_P^\varepsilon(q_2)$. Note that the lists $G_P^\varepsilon(q_i)$ are shifted versions of $G_P^\varepsilon((0,0))$.

Figure 3.2: Construction of (G, ε)-inverted lists and (G, ε)-matches for $G = T(2)$ and $M = \mathbb{R}^2$

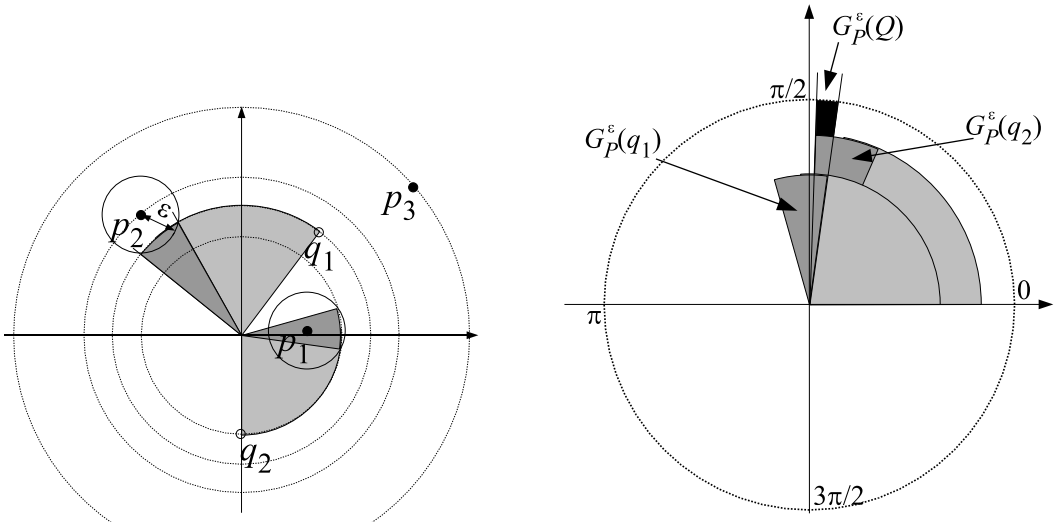
3.2.2 Rotations in $\mathbb{R}^2$

In our next example, we consider $M := \mathbb{R}^2$ and $G := \mathrm{SO}(2)$ acting on $\mathbb{R}^2$ by matrix-vector multiplication as rotations around the origin. As our metric d, we consider $d := d_2$, i.e., the Euclidean distance which is SO(2)-invariant.

1. In the example shown in Figure 3.1, we have already seen that the set $R := \mathbb{R}_{\geq 0} \times \{0\}$ is a set of representatives of the SO(2)-orbits. This set of representatives has the special property that the distance d_O between the G-orbits as defined in Remark 3.1.4 equals the Euclidean distance between the two orbits' representatives.

2. A transporter subset $\tau_{r,s}^{G,\varepsilon}$ can be characterized as a circular arc on the unit circle.

3. Since SO(2) is an abelian group, we have $g\tau_{r,s}^{G,\varepsilon}h = gh\tau_{r,s}^{G,\varepsilon}$, so it suffices to consider left-multiplication. If g represents the rotation by an angle ϕ around the origin, then $g\tau_{r,s}^{G,\varepsilon}$ corresponds to the circular arc $\tau_{r,s}^{G,\varepsilon}$ rotated by ϕ.



A pattern $P = \{p_1, p_2, p_3\}$ and a query $Q = \{q_1, q_2\}$

Construction of $G_P^\varepsilon(Q) = G_P^\varepsilon(q_1) \cap G_P^\varepsilon(q_2)$

Figure 3.3: Construction of (G, ε)-inverted lists and computation of (G, ε)-matches for $G = \mathrm{SO}(2)$.

Thus, for constructing $G_P^\varepsilon(s)$ for some $s \in R$, it suffices to take the union in Eq. (3.20) over all r with $d(r, s) \leq \varepsilon$.
For computing $(\mathrm{SO}(2), \varepsilon)$-inverted matches of $Q = \{q_1, \ldots, q_n\}$ w.r.t. $P = \{p_1, \ldots, p_m\}$ we need to compute the intersection of n sets of circular arcs, each of these consisting of m circular arcs. This can be done by unrolling the circular arcs to the interval $[0, 2\pi)$, which will be studied more detailed in Chapter 4.
Very similar results as for $G = \mathrm{SO}(2)$ hold for the group of uniform scalings SC(2) acting on $\mathbb{R}^2$. The unit circle S^1 can be used as a system of representatives, and the transporter sets $\tau_{r,s}^{G,\varepsilon}$ can be identified with closed real intervals; SC(2) is abelian, and multiplying a transporter set $\tau_{x,y}^{G,\varepsilon}$ by $g \in \mathrm{SC}(2)$ corresponds to shifting the closed interval along the real line.

Note that constructing (G, ε)-inverted lists and intersecting them for the cases $G = \mathrm{SO}(2)$ and $G = \mathrm{SC}(2)$ can be done using elementary geometry only. The reason for this is that these groups only have dimension one — $\mathrm{SO}(2)$ is parametrized by the unit circle, and $\mathrm{SC}(2)$ is parametrized by the real line; the resulting characterization of the transporter sets $\tau_{r,s}^{G,\varepsilon}$ as circular arcs or closed intervals leads to easily solvable computational problems. For the case of translations in $\mathbb{R}^k$, the construction and intersection of (G, ε)-inverted lists is easy because the transporter sets can be identified with orthogonal closed intervals in $\mathbb{R}^k$, and for intersecting these intervals, well-studied data structures and algorithms can be applied.
For more complex groups such as rigid motions or the group $\mathrm{SO}(3)$ of rotations in $\mathbb{R}^3$, there are no simple data structures to represent the transporter subsets to be intersected. In fact, intersecting these groups' transporter subsets requires some effort, which is left to the next chapter.

Chapter 4

Intersecting Transporter Subsets

As we have seen in the previous chapter, (G, ε)-inverted lists allow us to match patterns with respect to the (directed) Hausdorff distance – as long as we are able to handle the intersection of the (G, ε)-transporters involved. In this chapter, we focus on the problem of intersecting (G, ε)-transporters. To be more precise, we focus on the problem of deciding whether the intersection of a given finite family of (G, ε)-transporters is empty or not, where G always is a subgroup of the affine group $\mathrm{AGL}(k)$ acting on $V = \mathbb{R}^k$. This problem is equivalent to the following one studied in the work by Alt, Mehlhorn, Wagener and Welzl [10]: Let $\langle p_1, \ldots, p_N \rangle$ and $\langle q_1, \ldots, q_N \rangle$ be two sequences of points of equal length. Determine whether there exists a transformation g that maps each q_i into the ε-neighborhood of p_i.
The algorithm we propose allows to decide whether a family of (G, ε)-transporters is non-empty. In case G contains translations as a subgroup, we propose an *approximate* version of this algorithm based on eliminating the translation components of the transporter sets; these methods will be studied in detail for different transformation groups. The problem of intersecting transporter sets is not of immediate interest in pattern matching applications. However, gaining insights into intersections of transporter sets will be helpful in the following chapter, since these intersections are related to algorithms for matching with respect to distance measures such as the Hausdorff- and the bottleneck distance. The algorithms presented in the sequel are approximate algorithms that yield smaller time complexities than algorithms that solve the problem exactly. Furthermore, the results apply to a large class of transformation groups.

4.1 Notation and Basic Concepts

Throughout this chapter, V denotes the Euclidean vector space $\mathbb{R}^k$, where $\|x - y\|$ denotes the Euclidean distance between x and y, and G denotes a subgroup of the affine group $\mathrm{AGL}(k)$. Furthermore, G contains the group of all translations in V as a subgroup, i.e., $T(k) \leq G \leq \mathrm{AGL}(k)$.
Since they play a crucial role in this chapter, we repeat the definition of the (G, ε)-*transporter from* $q \in V$ *to* $p \in V$ from Chapter 3, which we had defined as

$$\tau_{q,p}^{G,\varepsilon} \quad := \quad \{g \in G \mid \|gq - p\| \leq \varepsilon\}.$$

We now consider a metric space that is induced by $(V, \|.\|)$: for fixed $K \in \mathbb{N}_{\geq 0}$, consider the set $V^{[0:K-1]}$ of all sequences of K vectors in V. Then, G acts on $\langle p_0, \ldots, p_{K-1} \rangle \in V^{[0:K-1]}$ in

a canonical way via $g\langle p_0, \ldots, p_{K-1}\rangle := \langle gp_0, \ldots, gp_{K-1}\rangle$. Furthermore, defining $d^K(P,Q) := \max_i \|p_i - q_i\|$, we obtain a metric space $(V^{[0:K-1]}, d^K)$.
To verify that d^K indeed defines a metric, one can easily see that $d^K(P,Q) = 0 \Leftrightarrow P = Q$ as well as $d^K(P,Q) = d^K(P,Q)$ are immediately inherited from the corresponding properties of the Euclidean distance between two points. In order to show that d^K satisfies the triangle inequality, let $P, Q, R \in V^{[0:K-1]}$. Obviously, there is an $i \in [0:K-1]$ so that $\|p_i - r_i\| = d^K(P,R)$. Due to $\|p_i - q_i\| \leq d^K(P,Q)$, $\|q_i - r_i\| \leq d^K(Q,R)$ and the triangle inequality holding for the Euclidean distance, it follows that $d^K(P,R) \leq \|p_i - r_i\| \leq \|p_i - q_i\| + \|q_i - r_i\| \leq d^K(P,Q) + d^K(Q,R)$.
For sequences $P = \langle p_0, \ldots, p_{K-1}\rangle$ and $Q = \langle q_0, \ldots, q_{K-1}\rangle$ in $V^{[0:K-1]}$ the set of all (G, d^K, ε) matches of Q and P is the intersection of (G,ε)-transporters corresponding to the components of P and Q:

$$G(P, Q, \varepsilon, d^K) = \{g \in G \mid d^K(P, gQ) \leq \varepsilon\} = \bigcap_{0 \leq i < k} \tau^{G,\varepsilon}_{q_i,p_i}. \tag{4.1}$$

We study the problem of matching point sequences, which essentially consists of determining whether the intersection of a finite number of (G,ε)-transporters is empty or not.
The problem we focus on is the following: given $\varepsilon \geq 0$ and $P, Q \in V^{[0:K-1]}$, decide whether $G(P, Q, \varepsilon, d^K)$ is empty. By Eq. (4.1) this transporter set is the intersection of the individual transporters $\tau^{G,\varepsilon}_{q_i,p_i}$. In general, there are no obvious algorithms and data structures for handling (G,ε)-transporters and their intersections for arbitrary groups $G \leq \mathrm{AGL}(k)$. In order to simplify the intersection problem, we develop an approximate algorithm for the group of translations in the following section. This idea can be used to eliminate translation components of transporter sets, which simplifies the intersection significantly. Finally, applying techniques from real algebraic geometry, we generalize the idea underlying this algorithm to larger transformation groups in a later section.

4.1.1 Approximately Intersecting Translation Transporter Sets

For Euclidean vector spaces $V = \mathbb{R}^k$, the group $T(k)$ of all translations in V has the pleasing property that $T(k)$ is the additive group $(V, +)$ of the vector space, and we can identify each $p \in V$ with a translation and vice versa. As a result, a transporter set $\tau^{G,\varepsilon}_{q,p}$ can be written as an ε-neighbourhood in V, namely $U_\varepsilon(q - p)$, and can hence be identified with a closed hypersphere in V (e.g., a disk for $V = \mathbb{R}^2$ and a sphere for $V = \mathbb{R}^3$). For the intersection task to be solved, this means that we only have to decide whether the intersection of a family of equal sized hyperspheres is non-empty.
Suppose we are given a radius ε and a family of K hyperspheres with centers $\langle t_i\rangle_{i \in [0:K-1]}$. We can state a simple method for *approximately* deciding whether the intersection of the K hyperspheres of radius ε is non empty as follows: if $\|t_0 - t_i\| \leq 2\varepsilon$ for all $i \in [1:K-1]$, let the algorithm answer *non-empty*, and *empty* otherwise. Apparently, the algorithm answers *non-empty* if $\cap_i U_\varepsilon(t_i) \neq \emptyset$. As shown in Figure 4.1, the reverse statement does not hold. However, one can easily see that $\cap_i U_{2\varepsilon}(t_i) = \emptyset$ implies that the algorithm answers *empty*. In other words, our simple algorithm has an indecision interval of ε and hence yields a 2-approximate solution to the decision problem whether $\cap_i U_\varepsilon(t_i)$ is empty or not. Considering a vector space V of fixed dimension, the running time of the algorithm is $O(K)$, as a distance $\|t_0 - t_i\|$ is computed K times.

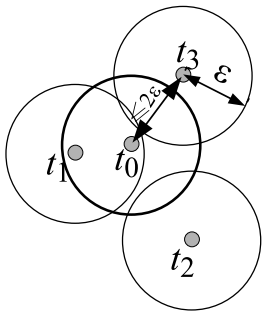


Figure 4.1: An arragement of translations and translation transporters

Note that t_0 plays a special role as a "reference translation". The choice of t_0 is arbitrary; we could also use t_i for any other $i \in [0 : K-1]$.
The idea behind this simple algorithm will now be generalized to larger transformation groups.

4.1.2 Eliminating Translation Components

For larger transformation groups than pure translations, we consider a transformation group G that is the semidirect product of the group of translations $T(k)$ and some other group $H \leq \mathrm{GL}(k)$, i.e., $G = T(k) \rtimes H$. In order to simplify the intersection problem, we apply the projection η of G onto H with kernel $T(k)$, i.e., $\eta(th) := h$, for $t \in T(k)$ and $h \in H$. Instead of sets $A \subseteq G$, we work with the η-image of such a set A:

$$\eta[A] := \{h \in H \mid \exists t \in T(k) \colon th \in A\}.$$

This projection is well defined since for each $g \in G$, there is a uniquely defined $t \in T(k)$ and $h \in H$ so that $g = th$, which is due to the fact that G is the semidirect product of $T(k)$ and H.
Looking at Eq. (4.1), an analogous statement does not hold for the intersection of η-images of individual (G, ε)-transporters: for $p, q \in V$ we always have $\eta[\tau_{q,p}^{G,\varepsilon}] = H$, so that $\cap_i \eta[\tau_{q_i,p_i}^{G,\varepsilon}] = H$. Thus to get non-trivial transporters we use projected transporter sets

$$\eta[\tau_{q,p}^{G,\varepsilon} \cap \tau_{q',p'}^{G,\varepsilon}] = \{h \in H \mid \exists t \in T(k) \colon th \in \tau_{q,p}^{G,\varepsilon}, th \in \tau_{q',p'}^{G,\varepsilon}\} \tag{4.2}$$

as our building blocks.
A natural question that is raised by Eq. (4.2) is how we obtain *pairs* of transporters in order to get projected transporters that are non trivial. Recall that the method for intersecting translation transporters from Section 4.1.1 uses a reference translation t_0 for an approximate algorithm. Generalizing this idea, we now intersect each transporter $\tau_{q_i,p_i}^{G,2\varepsilon}$ with the same reference transporter $\tau_{q_0,p_0}^{G,2\varepsilon}$. This motivates us to define

$$H_i := \eta[\tau_{q_0,p_0}^{G,\varepsilon} \cap \tau_{q_i,p_i}^{G,\varepsilon}]. \tag{4.3}$$

Now, consider the case that $\cap_i H_i \neq \emptyset$, so that there is some $h \in \cap_i H_i$. We define translations $t_i := p_i - hq_j$ and use $t_0 = p_0 - hq_0$ as our reference translation. Intuitively speaking, for each $h \in \cap_i H_i$, we get one instance of intersections of K hyperspheres. As it turns out, with

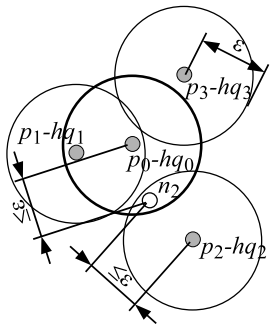


Figure 4.2: An arrangement of translations and translation transporters

$t_i = p_i - hq_i$, the simple algorithm from the last section always computes output *non-empty* when deciding whether $\cap_i U_\varepsilon(t_i) \neq \emptyset$. For proving this by the considerations from the previous section, it suffices to prove that $\|t_i - t_0\| \leq 2\varepsilon$ for all $i \in [1:K]$, which can be shown as follows: Since $h \in \eta[\tau^{H,\varepsilon}_{p_0,q_0} \cap \tau^{H,\varepsilon}_{p_i,q_i}]$, there is a translation n_i for each $i \in [1:K-1]$ such that $\|p_i - n_i hq_i\| \leq \varepsilon$ and $\|p_0 - n_i hq_0\| \leq \varepsilon$. As can be formally shown (see Figure 4.2 for an illustration), we have $\|n_i - t_i\| \leq \varepsilon$ as well as $\|n_i - t_0\| \leq \varepsilon$, and hence we obtain $\|t_i - t_0\| \leq 2\varepsilon$. (Instead of a formal proof, we refer to Theroem 4.1.2, where the idea sketched above is elaborated and completely formalized.)
Altogether, the above considerations yield that $t_0 h \in \cap_i \tau^{G,2\varepsilon}_{p_i,q_i}$, and, in particular, $\cap_i \tau^{G,2\varepsilon}_{p_i,q_i} \neq \emptyset$. Note that we obtained this conclusion about the intersection of the *complete* transporters from just looking at the *projected* transporters H_i – we did not have to deal explicitly with the translation part of the transformation group. Again, for an elaboration of the ideas sketeched so far, we refer to Theroem 4.1.2.
We have just seen that examining intersections of projected transporters may be used to draw conclusions about the intersection of the complete transporters. Our next concern is to formalize this idea and to obtain provable results about approximate transporter intersections. After that, it remains to characterize the projected transporter sets $\eta[\tau^{G,\varepsilon}_{q_0,p_0} \cap \tau^{G,\varepsilon}_{q_i,p_i}]$.
A major key for formalizing approximate transporter intersections are interesting structural properties of the group of translations in a vector space: $T(k)$ is abelian and acts transitively on V. Furthermore, the Euclidean distance is $T(k)$-invariant. Hence, the following Lemma, which will be of some use later on, holds:

Lemma 4.1.1 *Let N be an abelian group acting transitively on V, i.e., for each pair $(v,w) \in V^{[0:1]}$ there exists an $n \in N$ with $nv = w$. Furthermore, let d be an N-invariant metric on V. Then $d(nx, n'x) = d(ny, n'y)$, for arbitrary $n, n' \in N$ and $x, y \in V$.*

Proof. Since N acts transitively on V, we can write $x = ty$ for some $t \in N$. As N is abelian and d is N-invariant, we get $d(nx, n'x) = d(nty, n'ty) = d(tny, tn'y) = d(ny, n'y)$. □

As shown in the following Theorem, this property allows us to ignore translations and focus on merely intersecting projected transporter subsets.

Theorem 4.1.2 *Let $V, G, H, K, \varepsilon \geq 0$ be defined as above. For $P, Q \in V^{[0:K-1]}$ and every $i \in [0:K-1]$ define*

$$G_{i,\varepsilon} := \tau^{G,\varepsilon}_{q_i,p_i} \quad \text{and} \quad H_{i,\varepsilon} := \eta[G_{0,\varepsilon} \cap G_{i,\varepsilon}].$$

The following implications hold:

(1) $\cap_{i\in[1:K-1]} H_{i,\varepsilon} = \emptyset \implies \cap_{i\in[0:K-1]} G_{i,\varepsilon} = \emptyset.$

(2) $\cap_{i\in[1:K-1]} H_{i,\varepsilon} \neq \emptyset \implies \cap_{i\in[0:K-1]} G_{i,2\varepsilon} \neq \emptyset.$

Proof. (1) It suffices to show that $\cap_{i\in[0:K-1]} G_{i,\varepsilon} \neq \emptyset \Rightarrow \cap_{i\in[1:K-1]} H_{i,\varepsilon} \neq \emptyset$. Thus, let $g \in \cap_{i\in[0:K-1]} G_{i,\varepsilon}$. Then $g \in G_{0,\varepsilon} \cap G_{i,\varepsilon}$ for arbitrary $i \in [0 : K-1]$. Writing $g = nh$ for uniquely defined $n \in T(k)$ and $h \in H$, this implies $h \in \eta[G_{0,\varepsilon} \cap G_{i,\varepsilon}]$ for any i. Thus, we have $h \in \cap_i H_{i,\varepsilon}$, in other words, $\cap_i H_{i,\varepsilon} \neq \emptyset$.

(2) To begin with, let $h \in \cap_i H_{i,\varepsilon}$. We claim that the group element $g_0 := t_0 h$, with the translation $t_0 := p_0 - hq_0$, belongs to $\cap_i G_{i,2\varepsilon}$.
First of all, we observe that $g_0 q_0 = p_0$. As $h \in H_{i,\varepsilon} = \eta[G_{0,\varepsilon} \cap G_{i,\varepsilon}]$ for any $i \in [0 : K-1]$, there is an $n_i \in T(k)$ such that

$$g_i' := n_i h \in G_{0,\varepsilon} \cap G_{i,\varepsilon} = \tau^{G,\varepsilon}_{q_0,p_0} \cap \tau^{G,\varepsilon}_{q_i,p_i}.$$

Using this fact, the triangle inequality and Lemma 4.1.1 applied to hq_i and hq_0, we obtain for arbitrary $i \in [1 : K-1]$:

$$\begin{aligned} \|g_0 q_i - p_i\| &\leq \|g_0 q_i - g_i' q_i\| + \|g_i' q_i - p_i\| \\ &\leq \|t_0 h q_i - n_i h q_i\| + \varepsilon \\ &= \|t_0 h q_0 - n_i h q_0\| + \varepsilon \\ &= \|p_0 - g_i' q_0\| + \varepsilon \leq 2\varepsilon. \end{aligned}$$

This proves the second claim. □
Expressed in simple terms, the preceding Theorem states that deciding whether the intersection $\cap_i H_{i,\varepsilon}$ is non-empty yields an approximate solution to the decision problem asking if the intersection $\cap_i \tau^{G,\varepsilon}_{q_i,p_i}$ is non-empty.
We now take a closer look at the projected (G,ε)-transporters $\eta[\tau^{G,\varepsilon}_{q_0,p_0} \cap \tau^{G,\varepsilon}_{q_i,p_i}]$. To this end, we observe that for any $G \leq \mathrm{AGL}(k)$ it suffices to compute $\eta^{G,\varepsilon}_{P,Q}$ for *centered line segments* P and Q, i.e., line segments whose center is located at the origin. Obviously, centered line segments are point symmetric with respect to the origin. This fact in combination with further nice properties of centered line segments allows us to state the following lemma.

Lemma 4.1.3 *Let $V = \mathbb{R}^k$ for some $k > 0$ and $P, Q \in V^{[0:1]}$ be two line segements in V, and let $\tilde{P} = \langle -\tilde{p}, \tilde{p}\rangle, \tilde{Q} = \langle -\tilde{q}, \tilde{q}\rangle \in V^{[0,1]}$ denote the centered versions of P and Q, respectively, so that $\tilde{p} = \frac{1}{2}(p_1 - p_0)$ and $\tilde{q} = \frac{1}{2}(q_1 - q_0)$. Moreover, let $G = T(k) \rtimes H$ for some $H \leq \mathrm{GL}(k)$. Then, the following holds:*

(1) $\|\tilde{P} - \tilde{Q}\|_\infty \leq \|\tilde{P} - n\tilde{Q}\|_\infty$ for any $n \in T(k)$.

(2) $\eta[\tau^{G,\varepsilon}_{q_0,p_0} \cap \tau^{G,\varepsilon}_{q_1,p_1}] = \tau^{H,\varepsilon}_{-\tilde{q},-\tilde{p}} = \tau^{H,\varepsilon}_{\tilde{q},\tilde{p}}$.

Proof. (1) We have $\|\tilde{P} - \tilde{Q}\|_\infty = \max\{\|-\tilde{p} - (-\tilde{q})\|, \|\tilde{p} - \tilde{q}\|\} = \|\tilde{p} - \tilde{q}\|$ and $\|\tilde{P} - (\tilde{Q} + \langle n, n\rangle)\|_\infty = \max\{\|\tilde{p} - \tilde{q} + n)\|, \|\tilde{p} - \tilde{q} - n\|\}$. As for any $a \in V$,

$$\|a\| = \|\frac{1}{2}(a+n) + \frac{1}{2}(a-n)\| \leq \frac{1}{2}(\|a+n\| + \|a-n\|) \leq \frac{1}{2} \cdot 2\max\{\|a+n\|, \|a-n\|\},$$

our claim follows with $a = \tilde{p} - \tilde{q}$.
(2) We start with the second equality. Since for any $h \in \mathrm{GL}(k)$, we have $-h\tilde{q} = h(-\tilde{q})$, the equality follows from $\|\tilde{p} - h\tilde{q}\| = \|(-\tilde{p}) - h(-\tilde{q})\|$, for all $h \in H$.
We get to the proof of the first equality. Note that $\eta[\tau^{G,\varepsilon}_{-\tilde{q},-\tilde{p}} \cap \tau^{G,\varepsilon}_{\tilde{q},\tilde{p}}] = \eta[\tau^{G,\varepsilon}_{q_0,p_0} \cap \tau^{G,\varepsilon}_{q_1,p_1}]$, since $\tilde{P}$ and $\tilde{Q}$ are translated versions of P and Q, respectively. Now, it suffices to prove $\eta[\tau^{G,\varepsilon}_{-\tilde{q},-\tilde{p}} \cap \tau^{G,\varepsilon}_{\tilde{q},\tilde{p}}] = \tau^{H,\varepsilon}_{\tilde{q},\tilde{p}}$.
$\tau^{H,\varepsilon}_{\tilde{q},\tilde{p}} \subseteq \eta[\tau^{G,\varepsilon}_{-\tilde{q},-\tilde{p}} \cap \tau^{G,\varepsilon}_{\tilde{q},\tilde{p}}]$ follows immediately since $h\tilde{Q}$ is centered, too, and it remains to show the reverse inclusion $\tau^{H,\varepsilon}_{\tilde{q},\tilde{p}} \supseteq \eta[\tau^{G,\varepsilon}_{-\tilde{q},-\tilde{p}} \cap \tau^{G,\varepsilon}_{\tilde{q},\tilde{p}}]$. To this end, let $h \in \eta[\tau^{G,\varepsilon}_{-\tilde{q},-\tilde{p}} \cap \tau^{G,\varepsilon}_{\tilde{q},\tilde{p}}]$. By definition of η, there is a translation t such that $th \in \tau^{G,\varepsilon}_{-\tilde{q},-\tilde{p}} \cap \tau^{G,\varepsilon}_{\tilde{q},\tilde{p}}$, in other words, $\|\tilde{P} - th\tilde{Q}\|_\infty \leq \varepsilon$. From (1), we get $\|\tilde{P} - h\tilde{Q}\|_\infty \leq \|\tilde{P} - th\tilde{Q}\|_\infty \leq \varepsilon$, so that $h \in \tau^{H,\varepsilon}_{-\tilde{q},-\tilde{p}} \cap \tau^{H,\varepsilon}_{\tilde{q},\tilde{p}}$, and in particular, $h \in \tau^{H,\varepsilon}_{\tilde{q},\tilde{p}}$. □
Let $P, Q \in V^{[0:1]}$ be arbitrary point pairs, and let t_P and t_Q denote the translations that move the center of P and Q, respectively, to the origin, so that $\tilde{P} := t_P P$ and $\tilde{Q} := t_Q Q$ are centered. Since P and $\tilde{P}$ as well as Q and $\tilde{Q}$ only differ by a translation, we have

$$\eta[\tau^{G,\varepsilon}_{q_0,p_0} \cap \tau^{G,\varepsilon}_{q_1,p_1}] = \eta[\tau^{G,\varepsilon}_{\tilde{q}_0,\tilde{p}_0} \cap \tau^{G,\varepsilon}_{\tilde{q}_1,\tilde{p}_1}].$$

Finally, by Lemma 4.1.3, we get

$$\eta[\tau^{G,\varepsilon}_{q_0,p_0} \cap \tau^{G,\varepsilon}_{q_1,p_1}] = \tau^{H,\varepsilon}_{\tilde{q}_1,\tilde{p}_1}. \tag{4.4}$$

This allows us to state Theorem 4.1.2 in a more convenient way, showing that it suffices to intersect (H,ε)-transporters instead of (G,ε)-transporters:

Theorem 4.1.4 *Let $V, G, H, K, \varepsilon \geq 0$ as above. For $P, Q \in V^{[0:K-1]}$ and every $i \in [0:K-1]$, let $\tilde{p}_i := \frac{1}{2}(p_0 - p_i)$ and $\tilde{q}_i := \frac{1}{2}(q_0 - q_i)$. The following implications hold:*

(1) $\quad \cap_{i\in[1:K-1]} \tau^{H,\varepsilon}_{\tilde{q}_i,\tilde{p}_i} = \emptyset \implies \cap_{i\in[0:K-1]} \tau^{G,\varepsilon}_{q_i,p_i} = \emptyset.$

(2) $\quad \cap_{i\in[1:K-1]} \tau^{H,\varepsilon}_{\tilde{q}_i,\tilde{p}_i} \neq \emptyset \implies \cap_{i\in[1:K-1]} \tau^{G,2\varepsilon}_{q_i,p_i} \neq \emptyset.$

Proof. The claim follows immediately from Theorem 4.1.2, Lemma 4.1.3 and Eq. (4.4). □

4.1.3 Rotations and Scalings in 2-Space

We now study two transformation groups for point sequences in $\mathbb{R}^2$, namely rigid motions and homothetic motions, applying the results of the last section to provide efficient algorithms for handling transporter intersections.
The first goal of this section is to show the following:

- We may characterize each $\tau^{\mathrm{SO}(2),\varepsilon}_{q,p}$ as a closed interval on the unit circle S^1.
- $\tau^{\mathrm{SC}(2),\varepsilon}_{q,p}$ can be identified as a closed interval on the real line.

Circular arcs and real intervals are geometric objects whose intersection can be handled easily, and by Theorem 4.1.4, dealing with these simple intersections suffices for handling the intersections of $(\mathrm{SE}(2),\varepsilon)$-transporters and $(\mathrm{HT}(2),\varepsilon)$-transporters, respectively.
The constructions shown in Figures 4.3 and 4.4 yield intervals on S^1 and over $\mathbb{R}$, respectively, for $\tau^{H,\varepsilon}_{q,p}$. The idea behind these constructions is to compute $Hq \cap U_\varepsilon(p)$, i.e., the intersection

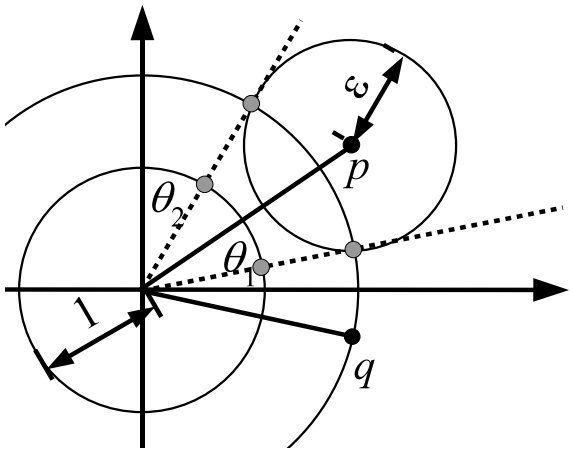


Figure 4.3: Construction of $\tau_{q,p}^{H,\varepsilon} = [\theta_1, \theta_2]$ for $H = \mathrm{SO}(2)$. Elementary geometric computations yield the angles θ_1 and θ_2.

of the H-orbit of q and $U_\varepsilon(p)$. Now, the intersection of a circle with a disc is a circular arc for $H = \mathrm{SO}(2)$, namely the circular arc enclosed by the angles θ_0 and θ_1, as depicted in Figure 4.3. For $H = \mathrm{SC}(2)$, Hq is a straight line. The intersection of a straight line with a disc is a line segment, and thus $\tau_{q,p}^{\mathrm{SC}(2),\varepsilon}$ can be identified with the closed interval specified in Figure 4.4.

We now combine the results of Eq. (4.1), Theorem 4.1.4 and the construction shown in Figure 4.4 in order to obtain an efficient algorithm for deciding (approximately) whether $\cap_i \tau_{q_i,p_i}^{G,\varepsilon} \neq \emptyset$, starting with $G = \mathrm{HT}(2)$. The projected (G,ε)-transporters to be intersected according to Theorem 4.1.4 are compact real intervals. In order to approximately decide whether the (G,ε)-transporter of two point sequences is non-empty, these results suggest that in case $H = \mathrm{SC}(2)$, we only need to decide whether the intersection of a certain family of compact real intervals is non-empty.

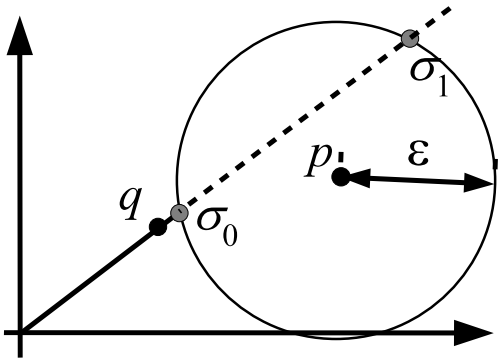


Figure 4.4: Construction of $\tau_{q,p}^{H,\varepsilon} = \{\lambda \mid \|\sigma_0\|/\|q\| \leq \lambda \leq \|\sigma_1\|/\|q\|\}\mathrm{id}_V$ for $H = \mathrm{SC}(2)$ acting on $V = \mathbb{R}^2$.

Remark 4.1.5 *Let $[x_1, y_1], \ldots, [x_N, y_N]$ be compact real intervals. Then $\cap_{i\in[1:N]}[x_i, y_i] = [\max_{i\in[1:N]} x_i, \min_{j\in[1:N]} y_j]$. In particular, this intersection is non-empty iff $\max_{i\in[1:N]} x_i \leq \min_{j\in[1:N]} y_j$.*

Since maximum and minimum of a family of K reals can be computed in $O(K)$ time, we can give an approximate solution to the decision problem whether $G(P, Q, \varepsilon, d^K) \neq \emptyset$ for $G = \mathrm{SC}(2)$ in $O(K)$ time.
Now, let us consider the case $H = \mathrm{SO}(2)$. As shown in Figure 4.3, the projected (G, ε)-transporters to be intersected are circular arcs. We view circular arcs as intervals on the unit circle. Intervals on the unit circle differ in some respects from intervals over $\mathbb{R}$; for example, the intersection of two intervals on S^1 may consist of up to two disjoint circular arcs.
As a first step for resolving such issues, we define an order over the points on the unit circle as follows: the unit circle can be parameterized by the real interval $[0, 2\pi)$. We define $x \preceq y$ iff the parameter angle of x is at most the parameter angle of y. Note that computing this order as well as all other computations described so far can be performed algebraically, i.e., involving addition, subtraction, multiplication, division and computing roots of real numbers and without involving trigonometric functions.
However, there is no property corresponding to Remark 4.1.5, and hence we cannot decide whether a family of circular arcs is non-empty by computing the maximum of all left interval borders and the minimum of right interval borders. What we can do, however, is to sort the left and right borders of the unrolled real intervals and then sweep over this arrangement of intervals as follows:

1. Compute the $O(K)$ borders of all K transporters and sort them in order to obtain an ordered cell enumeration.

2. Start with the first cell and mark the polynomials whose sign condition is satisfied. If all K polynomials are marked, we have found a cell that is contained in the intersection of the K transporters and we are done.

3. Traverse the cells in the given order. Each time a transition to a new cell is performed, one transporter needs to be marked, or the mark of one transporter needs to be removed. If the number of marked transporters is K, return the current cell representative.

4. If at no time, K transporters were marked, return 0.

Obviously, the first step requires $O(K \log K)$ time. The second step requires $O(K)$ time, since the first interval border can be contained in up to K transporters. Finally, updating one transporter mark for a single border element in Step 3 takes time $O(1)$, so that Step 3 requires $O(K)$ time in total. Hence, the overall running time obtained is $O(K \log K)$.
The time complexity for the first cell is $O(K)$, while $O(1)$ time is needed for each of the other $O(K)$ cells. Hence, the total time complexity is dominated by the $O(K \log K)$ time it takes to sort the interval borders.
We summarize these results in the following

Theorem 4.1.6 *Let $\varepsilon \geq 0$, $V = \mathbb{R}^2$ and $P, Q \in V^{[0:K-1]}$. There is an algorithm that computes output*

$$\begin{cases} 1 & \text{if } G(P, Q, \varepsilon, d^K) \neq \emptyset, \\ 0 & \text{if } G(P, Q, 2\varepsilon, d^K) = \emptyset, \\ 0 \text{ or } 1 & \text{otherwise,} \end{cases}$$

in $O(K)$ time for $G = \mathrm{HT}(2)$ and in $O(K \log K)$ time for $G = \mathrm{SE}(2)$.

Studying this approach from an algebraic geometry point of view, one obtains semialgebraic sets and arrangements defined by semialgebraic sets, which we now study in more detail.

4.2 Intersecting Arbitrary AGL(k)-Transporters

Generalizing the decision algorithm from the last section to larger transformation groups requires some effort, using methods from computational real algebraic geometry. There is some related work that also deals with applications of techniques from real algebraic geometry to geometric pattern matching; the first work using such techniques is the method proposed by Ambühl, Chakraborty and Gärtner [11] for finding largest common point sets with respect to the bottleneck distance under different transformation groups. Their algorithm can be modified in a straightforward way for solving other matching problems, in particular for deciding exactly (i.e., without an indecision interval) whether $G(P, Q, \varepsilon, d^K) = \emptyset$. Another work using real algebraic geometry for pattern matching has been proposed by Wenk [66] for matching polygonal curves under the Fréchet distance.
The time complexity of the method stated in Theorem 2.3.6 depends exponentially on the dimension of the variety that is partitioned into cells by a family of polynomials. In pattern matching applications, the underlying variety usually is a linear algebraic transformation group. Since the projection technique from Theorem 4.1.4 allows to ignore translation parts (at the cost of an indecision interval), our algorithms will be generally dealing with groups of smaller dimension and will hence yield smaller time complexities.
Note that due to Theorem 4.1.2, it suffices to focus on groups $H \leq \mathrm{GL}(k)$ rather than subgroups of AGL(k). However, since AGL(k) can be embedded in GL($k+1$), the results stated below generalize in a straightforward way to arbitrary subgroups of AGL(k).

4.2.1 Transporter sets as semialgebraic sets

Throughout this section, let H be a linear algebraic group acting rationally on $\mathbb{R}^k$. The group variety H is considered to be an algebraic subset of $\mathbb{R}^d$ for some $d > 0$. As a first step towards describing transporter sets as semialgebraic sets, we consider a vector $v := hp$ with $h \in H$ and $p \in \mathbb{R}^k$. Since the group action of H on $\mathbb{R}^k$ is a rational group action, the i-th coordinate of v, denoted by v_i, is a rational function in the d coordinates of the group element h. Hence, we can write $v_i = e_i/d_i$ for some $e_i, d_i \in \mathbb{R}[X_1, \ldots, X_d]$. Defining $D := \prod_i d_i$, we obtain a polynomial condition

$$D^2 \sum_i v_i^2 - D^2\varepsilon^2 \leq 0$$

that is equivalent to $\|p_i - q_i\|^2 \leq \varepsilon^2$. We summarize this in the following Lemma.

Lemma 4.2.1 *Let $V = \mathbb{R}^k$ and $H \leq \mathrm{GL}(k)$ a linear algebraic group acting rationally on V. Then, for any $p, q \in V$, $\tau_{q,p}^{H,\varepsilon}$ is a semialgebraic set. In particular, the intersection of finitely many transporter sets is a semialgebraic set.*

For deciding the emptiness of $\cap_i \tau_{q_i,p_i}^{G,\varepsilon}$, one can use cell enumeration procedures such as c.a.d. or the related technique of Theorem 2.3.6. Using the result stated in Theorem 2.3.6, we obtain the following result:

Lemma 4.2.2 *Let V, K and H be defined as in Lemma 4.2.1, and let d' denote the real dimension of H. Then, given $\langle x_0, \ldots, x_{K-1} \rangle \in V^{[0:K-1]}$ and $\langle y_0, \ldots, y_{K-1} \rangle \in V^{[0:K-1]}$, $K > 0$, one can decide in $O(K^{d'+1})$ time whether $\cap_{i \in [0:K-1]} \tau^{H,\varepsilon}_{y_i,x_i}$ is empty or not.*

Chapter 5

Pattern Matching with Respect to Relational Distance Measures

The results on intersections of transporter subsets of transformation groups by themselves only allow matching point sequences in V^K of the same length $K \in \mathbb{N}_{>0}$ with respect to d^K. In typical applications, however, the objects P and Q to be matched are not point sequences of the same length. Often, P and Q are point sets that usually are of different cardinality, and it is not known which points from Q match which points from P, so that d^K is not a relevant distance measure. For point sets, the distance measures that have been studied intensively are the Hausdorff distance and the bottleneck distance as well as some relatives of these two. In this chapter, we show how the Hausdorff and the bottleneck distance are related to d^K, which finally leads to matching algorithms with respect to these two popular distance measures, partially improving known upper bounds for such matching algorithms.
In the subsequent chapter, we will study the Fréchet distance that is defined between polygonal curves. A discrete version of the Fréchet distance will again be very closely related to the metric space (V^K, d^K) from the last chapter.
A major goal of this chapter and of Chapter 7 is to show that the Hausdorff distance, the bottleneck distance and the discrete version of the Fréchet distance (as introduced in the next chapter) belong to one certain class of distance measures, so that matching algorithms for these three (as well as a number of other) distance measures can use the same techniques.

Convention 5.0.3 *Throughout this chapter, we use the following notational conventions:*

- *k denotes the dimension of the Euclidean vector space $V := \mathbb{R}^k$.*
- *$P = \langle p_1, \ldots, p_m \rangle \in V^{[1:m]}$ and $Q = \langle q_1, \ldots, q_n \rangle \in V^{[1:n]}$ denote sequences of m and n points in V, respectively.*
- *$G = T(k) \rtimes H$ denotes a transformation group acting on V, where $H \leq \mathrm{GL}(k)$.*
- *By $V^+ := \cup_{k \geq 1} V^{[1:k]}$, we denote the set of all finite sequences of points in V.*

5.1 Related Work

One of the first works that considered distance measures for geometric pattern matching between point sets is the work by Alt et al. [10]. In this work, the notion of congruence

between point sets is generalized to *approximate congruence*, which we refer to as the *bottleneck distance* in the sequel. In order to measure the bottleneck distance between two point sets of equal cardinality, one determines an optimal bijection between the two point sets. A bijection assignment is optimal if the maximum of all distances between points assigned to each other is minimal. In [10], algorithms for matching under several transformation groups are considered. Subsequent works by Schirra [39] as well as Indyk et al. [46] consider different approximate and randomized techniques in order to obtain faster matching algorithms. Finally, in [11] and [20] algorithms for finding *largest common point sets* with respect to the bottleneck distance as well as applications in molecular biology are considered.
Besides the bottleneck distance, the *Hausdorff distance* has been considered as a natural distance measure between point sets. Instead of bijections, the Hausdorff distance minimizes one-to-many-assignments and hence is also suitable for point sets of different cardinality. The Hausdorff distance can be computed faster than the bottleneck distance. Thus, the numerous algorithms developed for matching (and, in particular, approximate matching) under different transformation groups (see, e.g., [12, 36, 42]) usually have asymptotically smaller running times than algorithms for matching with respect to the bottleneck distance. Recent work by Cardoze and Schulman [19] as well as Indyk et al. [46] reduce the geometric problem of matching with respect to the Hausdorff distance to a combinatorial pattern matching problem, obtaining the asymptotically fastest of all known matching algorithms. However, the results from [33] suggest that these theoretical improvements do not yield better running times in real world applications.
In the rest of this chapter, we systematically develop a class of matching algorithms that is capable of solving most of the problems addressed in the above mentioned literature. We compare these methods that are based on the technique of eliminating translation components and cell enumeration to known results from the works cited above.

5.2 Relational Distance Measures

In this section, we describe a class of distance measures between families of points that are typically used for geometric pattern matching. Later on, we provide matching algorithms for distance measures belonging to this class. Besides the directed and the undirected Hausdorff distance, both of which we already encountered in Chapter 3 in the context of (G, ε)-matches and (G, ε)-inverted lists, we will study the *bottleneck distance* $\mathbf{d}_{\mathrm{B}}$ (sometimes also referred to as *approximate congruence*) that was studied for the first time in the context of pattern matching by Alt et al. [10].

Definition 5.2.1 *Let $P, Q \in V^{[1:n]}$. The* bottleneck distance $\mathbf{d}_{\mathrm{B}}$ *between the two point sequences is defined as*

$$\mathbf{d}_{\mathrm{B}}(P, Q) = \min_{\pi \in S_n} \max_{i \in [1:n]} \|p_{\pi(i)} - q_i\|,$$

where S_n denotes the set of all permutations of $[1:n]$.

We now characterize distance measures such as the Hausdorff and the bottleneck distance by means of relations. Recall that we work with *sequences of points* $P \in V^{[1:m]}$ and $Q \in V^{[1:n]}$ rather than with sets of points. Then, for $\mathbf{d} \in \{\mathbf{d}_{\mathrm{B}}, \mathbf{d}_{\mathrm{H}}, d_{\mathrm{H}}\}$, we can decide whether $\mathbf{d}(P, Q) \leq \varepsilon$ by considering the relation

$$R(P, Q, \varepsilon) := \{(i, j) \mid \|p_i - q_j\| \leq \varepsilon\} \subseteq [1:m] \times [1:n].$$

In case of the bottleneck distance, we have $\mathbf{d}_{\mathrm{B}}(P,Q) \leq \varepsilon$ iff there is a complete matching in the bipartite graph described by $R(P,Q,\varepsilon)$, viewed as a set of edges. In case of the directed Hausdorff distance, we have $d_{\mathrm{H}}(P,Q) \leq \varepsilon$ iff for all $j \in [1:n]$, there is an $i \in [1:m]$ such that $(i,j) \in R(P,Q,\varepsilon)$; for the undirected Hausdorff distance, the same criterion additionally needs to be satisfied for all $i \in [1:m]$ vice versa.

Definition 5.2.2 *We say that a distance measure* $\mathbf{d}$ *is* relational *iff for all* $m,n > 0$ *there is a set of relations* $\mathbf{R}(\mathbf{d},m,n) \subseteq 2^{[1:m]\times[1:n]}$ *so that*

$$\mathbf{d}(P,Q) \leq \varepsilon \iff R(P,Q,\varepsilon) \in \mathbf{R}(\mathbf{d},m,n)$$

for all $P \in V^{[1:m]}$ *and* $Q \in V^{[1:n]}$.

An essential factor for the time complexity of matching algorithms is the time $T(\mathbf{d},m,n)$ required to decide whether $R(P,Q,\varepsilon) \in \mathbf{R}(\mathbf{d},m,n)$. For the Hausdorff and the bottleneck distance, we obtain the following time bounds:

- *Bottleneck distance:* We need to check whether the bipartite graph defined by the edges given by $R(P,Q,\varepsilon)$ contains a complete matching. (In the sequel, we refer to this bipartite graph given by $R(P,Q,\varepsilon)$ as the *bipartite distance graph of* P *and* Q *for distance* ε.) Finding a maximum cardinality matching (and hence deciding whether there is a complete matching) can be done using the method by Hopcroft and Karp [40], requiring $O(|E|\sqrt{|V|})$ time for a graph with edges E and vertices V, yielding a time bound of $O(n^{2.5})$ in our case where n denotes the cardinality of P and Q. Better time bounds can be achieved by using the algorithm by Efrat and Itai from [30] , who obtain a time complexity of $O(n^{1.5}\log n)$ for the following problem: Given two sequences of points in the plane, P and Q, define a bipartite graph $\Gamma(P,Q)$ with vertices $P \cup Q$ and an edge between each pair (p_i,q_j) weighted by the Euclidean distance between these two points. In $O(n^{1.5}\log n)$ time (where $n = |P| = |Q|$), the algorithm stated in [30] determines a minimum value r so that there is a matching in $\Gamma(P,Q)$ whose edges all have a weight of at most r. Obviously, this algorithm may be used for our decision problem as well. Note that the running time for computing the maximum cardinality matching is dominated by the time $O(n^2)$ required for computing $R(P,Q,\varepsilon)$.

- *Hausdorff Distance:* For the directed Hausdorff distance, we only need to test in $O(n)$ time whether all vertices corresponding to points in Q have an edge adjacent to this vertex. For the undirected Hausdorff distance, we need to do the same additionally for all vertices corresponding to points in P, thus requiring $O(m+n)$ time. In both cases, the time complexity is dominated by the time of $O(mn)$ for computing $R(P,Q,\varepsilon)$.

5.2.1 Examples and Counterexamples

Besides the Hausdorff and the bottleneck distance, there are several other known distance measures that turn out to be relational distance measures. First of all, Indyk and Venkatasubramanian [47] propose the *generalized bottleneck distance*. The bottleneck distance is generalized in the sense that the restriction of bijective correspondences between the two point sets is relaxed to one-to-$\leq p$-correspondences for some fixed positive integer p. This relaxation is motivated by the fact that the Hausdorff distance can be computed faster than the

bottleneck distance, so that it is a reasonable idea to identify a class of distance measures that includes both distance measures as well as "intermediates" between the two distance measures; the time complexities are expected to lie between the time complexities for computing $\mathbf{d}_{\mathrm{H}}$ and $\mathbf{d}_{\mathrm{B}}$.
The distance function $\mathbf{d}_{\mathrm{B}}^p$ one obtains by this relaxation is formally defined by the set of relations

$$\mathbf{R}(\mathbf{d}_{\mathrm{B}}^p, m, n) := \{A \subseteq [1:m] \times [1:n] \mid \forall b \in [1:n] \colon 1 \leq |A \cap ([1:m] \times \{b\})| \leq p\},$$

If we let ξ_A denote the indicator matrix of A, we have $A \in \mathbf{R}(\mathbf{d}_{\mathrm{B}}^p, m, n)$ if and only if all column sums of ξ_A are in $[1:p]$. Note that $\mathbf{d}_{\mathrm{B}}^1 = \mathbf{d}_{\mathrm{B}}$ and $\mathbf{d}_{\mathrm{B}}^\infty = \mathbf{d}_{\mathrm{H}}$.
In [47], algorithms are proposed for matching with respect to $\mathbf{d}_{\mathrm{B}}^p$ under translations and rigid motions in $\mathbb{R}^2$. These algorithms are (in several repsects) approximate as well as randomized algorithms through the use of Hall's Matching Theorem [37] for finding matchings in a bipartite graph. The ideas underlying the matching algorithm can also be used for merely computing $\mathbf{d}_{\mathrm{B}}^p(P,Q)$. The time bounds for computing $\mathbf{d}_{\mathrm{B}}^p(P,Q)$ obtained for $p > 1$ are smaller than the time bounds for computing $\mathbf{d}_{\mathrm{B}}(P,Q)$.

Another class of distance measures that is a subset of relational distance measures are the variants of the Hausdorff distance proposed by Duboisson and Jain [28], who propose 24 variants of the Hausdorff distance (one of which is $\mathbf{d}_{\mathrm{H}}$ itself). Ten of these distance measures result from *ranked* versions of the directed Hausdorff distance:

$$d_{\mathrm{H}}^K(P,Q) \quad := \quad \max{}^K\{\min_{p \in P} \|q - p\| \mid q \in Q\}, \tag{5.1}$$

where $\max^K X$ denotes the K-th largest value contained in a finite set $X \subset \mathbb{R}$. The number K corresponds to a number of mismatches, i.e., a number of points from Q that do not need to match with any point in P; obviously, we have $d_{\mathrm{H}} = d_{\mathrm{H}}^0$. $\mathbf{R}(d_{\mathrm{H}}^K, m, n)$ is the set of all relations R where for at most K of all $j \in [1:n]$, there is no i such that $(i,j) \in R$.
In order to demonstrate that there are numerous distance measures that are *not* relational distance measures as well, we examine another distance measure described by Duboisson and Jain. We obtain this modified version of the Hausdorff distance (also referred to as the *mean-value Hausdorff distance* in the following) by summing up the distances to each nearest neighbor of $q \in Q$ (instead of taking the maximum of all these distances):

$$\mathbf{D}_{\mathrm{H}}(P,Q) := \frac{1}{n} \sum_{q \in Q} \min_{p \in P} \|p - q\|.$$

This distance measure is *not* a relational one, as shown in Figure 5.1. A distance measure closely related to $\mathbf{D}_{\mathrm{H}}$ that is not a relational distance measure either is the *mean-square Hausdorff distance* studied in [1], which is defined as

$$\mathbf{D}_{\mathrm{H}}^2(P,Q) := \frac{1}{n} \sum_{q \in Q} \min_{p \in P} \|p - q\|^2.$$

Sometimes, the *root-mean-square* Hausdorff distance $\sqrt{\mathbf{D}_{\mathrm{H}}^2(P,Q)}$ is considered. Matching algorithms for $\mathbf{D}_{\mathrm{H}}$ and $\mathbf{D}_{\mathrm{H}}^2$ are introduced in Section 5.6.
As a rule of thumb, one can say that distance measures involving summation of distances such as $\mathbf{D}_{\mathrm{H}}$ (or the *dynamic time warping distance* to be introduced in Chapter 7) are not relational distance measures.

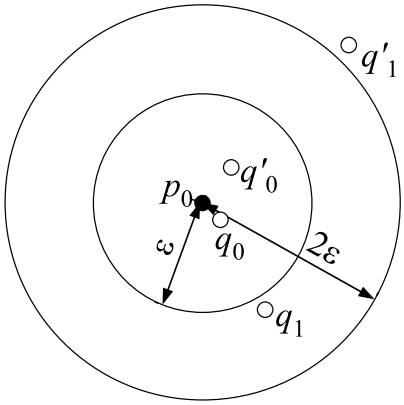


Figure 5.1: $\mathbf{D}_{\mathrm{H}}$ is not a relational distance measure: Let $P = \langle p_1 \rangle$, $Q = \langle q_1, q_2 \rangle$ and $Q' = \langle q'_1, q'_2 \rangle$. We have $\mathbf{D}_{\mathrm{H}}(P, Q) \leq \varepsilon$ and $\mathbf{D}_{\mathrm{H}}(P, Q') > \varepsilon$, but $R(P, Q, \varepsilon) = R(P, Q', \varepsilon)$.

5.2.2 Basic Matching Algorithms

There is a close relation between relational distance measures and intersections of transporter sets that were investigated in the last chapter. The next remark shows the close relation between the $(G, \varepsilon, \mathbf{d})$-matches of a relational distance measure $\mathbf{d}$ and the intersections of transporter sets from the preceding chapter.

Lemma 5.2.3 *Let $P \in V^{[1:m]}$ and $Q \in V^{[1:n]}$ and let $\mathbf{d}$ be a relational distance measure. Furthermore, let G be a group acting on V. Then,*

$$G(P, Q, \varepsilon, \mathbf{d}) = \bigcup_{S \in \mathbf{R}(\mathbf{d}, m, n)} \bigcap_{(i,j) \in S} \tau^{G,\varepsilon}_{q_j, p_i}.$$

In particular, $G(P, Q, \varepsilon, \mathbf{d}) \neq \emptyset$ iff $\bigcap_{(i,j) \in S} \tau^{G,\varepsilon}_{q_j, p_i} \neq \emptyset$ for some $S \in \mathbf{R}(\mathbf{d}, m, n)$.

Proof. We have

$$\begin{aligned}
g \in G(P, Q, \varepsilon, \mathbf{d}) &\iff \mathbf{d}(P, gQ) \leq \varepsilon \\
&\iff R(P, gQ, \varepsilon) \in \mathbf{R}(\mathbf{d}, m, n) \\
&\iff \exists S \in \mathbf{R}(\mathbf{d}, m, n) \colon \forall (i,j) \in S \colon \|p_i - gq_j\| \leq \varepsilon \\
&\iff \exists S \in \mathbf{R}(\mathbf{d}, m, n) \colon g \in \cap_{(i,j) \in S} \tau^{G,\varepsilon}_{q_j, p_i} \\
&\iff g \in \cup_{S \in \mathbf{R}(\mathbf{d}, m, n)} \cap_{(i,j) \in S} \tau^{G,\varepsilon}_{q_j, p_i},
\end{aligned}$$

which proves our claim. □

For $P \in V^{[1:m]}$ and $Q \in V^{[1:n]}$, we consider the family $(\tau^{G,\varepsilon}_{q_j, p_i})_{i \in [1:m], j \in [1:n]}$ of mn transporter sets. According to the above remark, we want to decide whether at least one of the intersections $\bigcap_{(i,j) \in S} \tau^{G,\varepsilon}_{q_j, p_i}$ is non-empty. To this end, we define an equivalence relation on G (depending on P, Q, G and ε) such that *every* intersection of transporters is a union of equivalence classes. If we compute a subset C of G containing at least one element from each equivalence class, then $G(P, Q, \varepsilon, \mathbf{d}) \neq \emptyset$ iff $C \cap G(P, Q, \varepsilon, \mathbf{d}) \neq \emptyset$. Such a C will be called a (P, Q, G, ε)-*suptransversal*. In order to decide whether $G(P, Q, \varepsilon, \mathbf{d}) \neq \emptyset$, it suffices to test each $g \in C$ for membership in $G(P, Q, \varepsilon, \mathbf{d})$.

We have an equivalence relation on G defined by $g \sim_{P,Q,G,\varepsilon} g'$ if and only if for all i, j we have $g \in \tau^{G,\varepsilon}_{q_j,p_i} \Leftrightarrow g' \in \tau^{G,\varepsilon}_{q_j,p_i}$, i.e., g is contained in exactly the same transporter sets as g'. The $\sim_{P,Q,G,\varepsilon}$-equivalence class that some $g \in G$ belongs to will be called the (P, Q, G, ε)-*cell of* g.

Algorithm 5.2.4

Input: P, Q, G, ε *as in Convention 5.0.3; relational distance measure* $\mathbf{d}$

Output:

$$\begin{cases} g \in G(P, Q, \varepsilon, \mathbf{d}) & \text{if } G(P, Q, \varepsilon, \mathbf{d}) \neq \emptyset \\ \textit{false} & \textit{otherwise.} \end{cases}$$

```
Match(P, Q, G, d, ε)
    compute a (P, Q, G, ε)-suptransversal C ⊆ G;
    for each g ∈ C do
       if d(P, gQ) ≤ ε then return g;
    end;
    return false;
end.
```

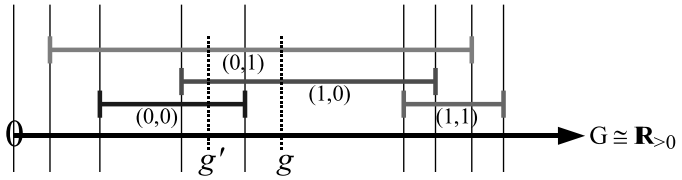


Figure 5.2: An arrangement of four SC(2)-transporter sets and two group elements g, g' from adjacent cells.

In Chapter 2, more efficient algorithms were introduced for the case that a cell representation of transporter sets could be enumerated in an ordered way according to Definition 2.3.7. Deciding the intersection problem under the group SO(2) could be sped up to an $O(n \log n)$ algorithm instead of an $O(n^2)$ algorithm. This can be seen as a generalization of the following observation made in [10] for speeding up matching under the bottleneck distance: If g and g' are two group elements from adjacent cells (see Figure 5.2), it is not necessary to completely compute $\mathbf{d}_{\mathrm{B}}(P, gQ)$ and $\mathbf{d}_{\mathrm{B}}(P, g'Q)$, because the bipartite distance graph defined by $R(P, gQ, \varepsilon)$ and the one defined by $R(P, g'Q, \varepsilon)$ differ in *at most one edge*. In terms of an algorithm that computes a maximum bipartite matching, this means that only one augmenting path needs to be examined for updating the maximum cardinality matching in $R(P, g'Q, \varepsilon)$. Alt et al. [10] argue that modifying the algorithm by Hopcroft and Karp [40] requires $O(mn)$ time for updating the maximum cardinality matching if only one edge is added or removed, since only one augmenting path (instead of $O(\sqrt{m+n})$ many) needs to be computed. The time bound $O(mn)$ compares to $O(mn\sqrt{m+n})$ time required for completely computing a matching "from scratch". In case of matching with respect to the undirected Hausdorff distance, the algorithmic advantage that can be drawn from ordered cell enumeration is even larger.

As we have seen in the example above, maintaining a *dynamic data structure* for computing the distance function in case of ordered cell enumeration can be very useful for developing faster algorithms. In order to generalize this idea, we give a formal definition of what operations a *dynamic data structure for a relational distance measure* $\mathbf{d}$ needs to support:

Definition 5.2.5 *Let* $\mathbf{d}$ *denote a relational distance measure. Using* $X \triangle Y$ *as the notation for the symmetric difference between two sets* X *and* Y*, we say that a family of sets* $(\mathbf{S}_{m,n})_{m,n\in\mathbb{N}}$ *is a* dynamic data structure for $\mathbf{d}$ *if for each* $m, n \in \mathbb{N}$*, there mappings*

$$\begin{array}{rcrcl} \text{init}_{m,n} & : & 2^{[1:m]\times[1:n]} & \to & \mathbf{S}_{m,n} \\ \text{state}_{m,n} & : & \mathbf{S}_{m,n} & \to & 2^{[1:m]\times[1:n]} \\ \text{update}_{m,n} & : & \mathbf{S}_{m,n} \times [1:m] \times [1:n] & \to & \mathbf{S}_{m,n} \\ \text{test}_{m,n} & : & \mathbf{S}_{m,n} & \to & \{\mathbf{true}, \mathbf{false}\} \end{array}$$

with the following properties:

(1) $\text{state}_{m,n}(\text{init}_{m,n}(R)) = R$ *for all* $R \in 2^{[1:m]\times[1:n]}$

(2) $\text{state}_{m,n}(\text{update}_{m,n}(R, i, j)) = R \triangle \{(i, j)\}$ *for all* $R \in 2^{[1:m]\times[1:n]}$*,* $i \in [1:m]$ *and* $j \in [1:n]$.

(3) $\text{test}_{m,n}(S) = \mathbf{true} \iff \text{state}_{m,n}(S) \in \mathbf{R}(\mathbf{d}, m, n)$ *for all* $S \in \mathbf{S}_{m,n}$.

We have already seen one example, namely the dynamic data structure for speeding up matching under the bottleneck distance proposed in [10]. In this case, init(R) simply computes one maximum bipartite matching (taking $T_{\text{init}}(n) = O(n^2\sqrt{n})$ time); test(R) yields *true* if the cardinality of the bipartite matching is n and *false* otherwise (taking $T_{\text{test}}(n) = O(1)$ time). The function update(R, i, j) toggles the edge between the vertices corresponding to p_i and q_j and then updates the maximum cardinality matching (taking $T_{\text{update}}(n) = O(n^2)$ time). The mapping state$\colon \mathbf{S} \to 2^{[1:m]\times[1:n]}$ can be thought of as the mapping that yields the bipartite graph corresponding to the current configuration; it is only needed for defining the semantics of the other three functions.
As another important example, we now provide a dynamic data structure for d_{H}. In this case, it is reasonable to define $\mathbf{S}$ as all pairs of an integer vector and an integer counter $\mathbf{S} = [1:m]^n \times [1:n]$. For $S = ((s_1, \ldots, s_n), N) \in \mathbf{S}$, the value s_j at position j counts how many points in P the point q_j is mapped to in the currently active configuration. N counts how many values s_j are non-zero. Obviously, the initialization step for this data structure for some $R \in 2^{[1:m]\times[1:n]}$ can be done in $T_{\text{init}}(m, n) = O(mn)$ time. The test-step simply consists of returning *true* iff $N = n$, hence $T_{\text{test}}(m, n) = O(1)$. Performing update$(R, i, j)$ consists of either increasing or decreasing the counter s_j. In addition, if s_j is increased to one or decreased to zero, N needs to be increased or decreased correspondingly. This yields $T_{\text{update}}(m, n) = O(1)$.
Corresponding to Definition 2.3.7, we also use the term *ordered* (P, Q, G, ε)*-suptransversal* for (P, Q, G, ε)-suptransversals with the property that the state of at most one transporter changes from one cell to another.

Algorithm 5.2.6
Input: P, Q, G, ε *as in Convention 5.0.3; relational distance measure* $\mathbf{d}$ *with dynamic data*

structure **S**.
Output:

$$\begin{cases} g \in G(P,Q,\varepsilon,\mathbf{d}) & \text{if } G(P,Q,\varepsilon,\mathbf{d}) \neq \emptyset \\ \text{false} & \text{otherwise.} \end{cases}$$

```
Ordered-Match(P, Q, G, ε)
    compute an ordered (P, Q, G, d, ε)-suptransversal
          (x_0, ⟨(x_1, (i_1, j_1)), ..., (x_L, (i_L, j_L))⟩);
    S := init(R(P, x_0 Q, ε));
    if test(S)==true then return x_0;
    for k = 1...L do
       update(S, (i_k, , j_k));
       if test(S)==true then return x_k;
    end;
    return false;
end.
```

For the asymptotical running time, we distinguish two cases: non-ordered cell enumeration and ordered cell enumeration. The time complexity in the first case mainly depends on the cardinality of the suptransversal and the time to compute such a transversal as well as the time to compute $\mathbf{d}(P,Q)$. We consider the case of a linear algebraic group acting rationally on $\mathbb{R}^k$. By Lemma 4.2.1, each transporter set is a semialgebraic set described by one polynomial. Hence, we can apply the result stated in Theorem 2.3.6 in order to obtain the (theoretical) running times stated in Table 5.1: let d denote the real dimension of the transformation group G. There are $O((mn)^d)$ cells whose enumeration takes $O((mn)^{d+1})$ time, the total time complexity of Algorithm 5.2.4 is $O((mn)^d(mn + T(\mathbf{d},m,n)))$, where $T(\mathbf{d},m,n)$ denotes the time to compute $\mathbf{d}(P,Q)$.
For the case of ordered cell enumeration, the times $T_{\text{init}}(m,n)$, $T_{\text{update}}(m,n)$ and $T_{\text{test}}(m,n)$ need to be incorporated into the analysis of the running time. Obviously, init(R) is called once, whereas update(S,i) and test(S) are called for each cell. Hence, we obtain an overall time complexity of $O(T_{\text{init}}(m,n) + T_{\text{enumerate}}(m,n,G) + (T_{\text{update}} + T_{\text{test}})L(m,n,G))$, where $T_{\text{enumerate}}(m,n,G)$ denotes the time to compute an ordered cell enumeration and $L(m,n,G)$ denotes the cardinality of the computed suptransversal. For example, we have

$$T_{\text{enumerate}}(m,n,\text{SO}(2)) = T_{\text{enumerate}}(m,n,\text{SC}(2)) = O(mn\log(mn))$$

and $L(m,n,\text{SO}(2)) = L(m,n,\text{SC}(2)) = O(mn)$. Altogether, we obtain the running times stated in Table 5.1.

Transformation group	$d_{\text{H}}, \mathbf{d}_{\text{H}}$	$\mathbf{d}_{\text{B}}$
$T(k)$	$O((mn)^{k+1})$	$O((mn)^{k+1}\sqrt{m+n})$
SO(2), SC(2)	$O((mn)\log(mn))$	$O((mn)^2)$
SE(2), HT(2)	$O((mn)^4)$	$O((mn)^4\sqrt{m+n})$
SM(2)	$O((mn)^5)$	$O((mn)^5\sqrt{m+n})$
SE(3)	$O((mn)^7)$	$O((mn)^7\sqrt{m+n})$

Table 5.1: Time complexities for solving matching tasks exactly.

5.3 Matching Point Sets with Reference Points

Using algorithms similar to those used for intersecting transporter sets in Chapter 4, we now provide matching algorithms for relational distance measures. Again, we make use of transporter sets, their projections and their characterizations as semialgebraic sets. In Chapter 4, we used the starting points of two point sequences to be matched as *reference points*. In the sequel, we generalize this idea for matching under relational distance measures that also have such reference points. The concept of reference points was first studied systematically by Alt, Aichholzer and Rote [3] for the Hausdorff distance.
We define reference points for relational distance functions as follows.

Definition 5.3.1 *Let V and G be defined as in Convention 5.0.3. Furthermore, let $\mathbf{d}$ denote a relational distance function on V^+. For $1 \leq c \in \mathbb{R}$, a mapping $r : V^+ \to V$ is called a* $(V, G, \mathbf{d}, c)$-reference point *if*

1. *r is a G-morphism, i.e., $r(gP) = gr(P)$ for all $g \in G$ and $P \in V^+$.*
2. *r is Lipschitz continuous in the sense that for all families of points $P, Q \in V^+$, we have $\|r(P) - r(Q)\| \leq c \cdot \mathbf{d}(P, Q)$.*

Note that if r is a $(V, G, \mathbf{d}, c)$-reference point, then it is also a $(V, H, \mathbf{d}, c)$-reference point for any subgroup H of G.

5.3.1 Reference Points for Hausdorff and Bottleneck Distance

Reference points for the Hausdorff distance introduced in [3] are the centroid of the convex hull and the so-called *Steiner point*. We summarize the main result from [3] and refer to this work for further details.

Theorem 5.3.2 ([3]) *There is a $(\mathbb{R}^2, \mathrm{SM}(2), \mathbf{d}_{\mathrm{H}}, 4/\pi)$-reference point that can be computed in $O(n \log n + m \log m)$ time.* □

Heffernan and Schirra [39] implicitly use the centroid of a point set as a reference point for the bottleneck distance under rigid motions in the plane. We state a more general result, showing that the idea from [39] also fits into the framework of reference points for relational distance measures, even under the group $\mathrm{AGL}(k)$ for arbitrary $V = \mathbb{R}^k$.

Lemma 5.3.3 *Let $V = \mathbb{R}^k$. Then, the centroid of a sequence $P \in V^{[1:m]}$, denoted by $r(P) := \frac{1}{m} \sum_{i \in [1:m]} p_i$, is a $(V, \mathrm{AGL}(k), \mathbf{d}_{\mathrm{B}}, 1)$-reference point.*

Proof. Let $g \in \mathrm{AGL}(k)$. Then, we have $gp = hp + t$ for uniquely defined $h \in \mathrm{GL}(k)$ and $t \in V$. Now,

$$\begin{aligned} r(gP) &= \frac{1}{m} \sum_i (hp_i + t) \\ &= \frac{1}{m} \left(\sum_i hp_i \right) + t \\ &\overset{h \text{ linear}}{=} h \left(\frac{1}{m} \sum_i p_i \right) + t \\ &= gr(P). \end{aligned}$$

It remains to show that r is Lipschitz-continuous with Lipschitz-factor 1. We have

$$\|r(P) - r(Q)\| \quad = \quad \|\frac{1}{m}\sum_i (p_i - q_i)\|.$$

For an arbitrary permutation $\pi \in S_m$, we get

$$\begin{aligned}\|r(P) - r(Q)\| &= \frac{1}{m}\|\sum_i (p_i - q_{\pi(i)})\| \\ &\leq \frac{1}{m} \cdot m \cdot \max_i \|p_i - q_{\pi(i)}\| \\ &= \max_i \|p_i - q_{\pi(i)}\|.\end{aligned}$$

Since this holds for arbitrary $\pi \in S_m$, we get in particular

$$\|r(P) - r(Q)\| \leq \min_{\pi \in S_m} \|\max_i (p_i - q_{\pi(i)})\| = \mathbf{d}_{\mathrm{B}}(P, Q),$$

which finally proves the claim. □

The discrete Fréchet distance introduced in Chapter 7 will provide another example of a relational distance function with a reference point.

5.3.2 Projecting Transporters Using Reference Points

The basic idea of matching patterns using a reference point is that the reference points of P and Q yield the translation part of a match, and the matching task is reduced to matching under the group $H := G/T(k)$. Using reference points for projecting transporter subsets as in Theorem 4.1.4, we obtain the following result:

Theorem 5.3.4 *Let V, G and H be defined as in Convention 5.0.3, and let $\mathbf{d}$ be a relational distance measure on V^+ with a $(V, G, \mathbf{d}, c)$-reference point r. Given $P \in V^{[1:m]}$ and $Q \in V^{[1:n]}$, we define*

$$\begin{aligned}\tilde{p}_i &:= \frac{1}{2}(p_i - r(P)) \text{ for } i \in [1:m] \text{ and} \\ \tilde{q}_j &:= \frac{1}{2}(q_j - r(Q)) \text{ for } j \in [1:n].\end{aligned}$$

For $E \in \mathbf{R}(\mathbf{d}, m, n)$, we have

(1) $\bigcap_{(i,j)\in E} \tau^{H,c\varepsilon}_{\tilde{q}_j,\tilde{p}_i} = \emptyset \implies \bigcap_{(i,j)\in E} \tau^{G,\varepsilon}_{q_j,p_i} = \emptyset.$

(2) $\bigcap_{(i,j)\in E} \tau^{H,c\varepsilon}_{\tilde{q}_j,\tilde{p}_i} \neq \emptyset \implies \bigcap_{(i,j)\in E} \tau^{G,2c\varepsilon}_{q_j,p_i} \neq \emptyset.$

Proof. We start with the proof of (1). It suffices to show that

$$\bigcap_{(i,j)\in E} \tau^{G,\varepsilon}_{q_j,p_i} \neq \emptyset \implies \bigcap_{(i,j)\in E} \tau^{H,c\varepsilon}_{\tilde{q}_j,\tilde{p}_i} \neq \emptyset.$$

To this end, let $g \in \cap_{(i,j)\in E} \tau^{G,\varepsilon}_{q_j,p_i}$. As $E \in \mathbf{R}(\mathbf{d}, m, n)$, this implies $g \in G(P, Q, \varepsilon, \mathbf{d})$. Since r is a reference point, we get

$$\begin{aligned} & \|r(P) - r(gQ)\| \leq c\varepsilon \\ \iff \quad & \|r(P) - gr(Q)\| \leq c\varepsilon \\ \iff \quad & g \in \tau^{G,c\varepsilon}_{r(Q),r(P)}. \end{aligned}$$

As $c \geq 1$, $\tau^{G,\varepsilon}_{q_j,p_i} \subseteq \tau^{G,c\varepsilon}_{q_j,p_i}$, and we get

$$g \in \cap_{(i,j)\in E}(\tau^{G,c\varepsilon}_{q_j,p_i} \cap \tau^{G,c\varepsilon}_{r(Q),r(P)}).$$

Since $G = T(k) \rtimes H$, we can write $g = th$ for uniquely defined $t \in T(k)$ and $h \in H$, yielding for all $(i,j) \in E$

$$h \in \eta[\tau^{G,c\varepsilon}_{q_j,p_i} \cap \tau^{G,c\varepsilon}_{r(Q),r(P)}] = \tau^{H,c\varepsilon}_{\tilde{q}_j,\tilde{p}_i},$$

where the last equality follows from Lemma 4.1.3. This proves (1).

For the proof of (2), let $h \in \cap_{(i,j)\in E} \tau^{H,c\varepsilon}_{\tilde{q}_j,\tilde{p}_i}$. From Lemma 4.1.3 and the definition of $\tilde{p}_i$ and $\tilde{q}_j$, we know that for all $(i,j) \in E$

$$h \in \tau^{H,c\varepsilon}_{\tilde{q}_j,\tilde{p}_i} = \eta[\tau^{G,c\varepsilon}_{r(Q),r(P)} \cap \tau^{G,c\varepsilon}_{q_j,p_i}]. \tag{5.2}$$

We claim that the group element $g_r := t_r h$ with $t_r := r(P) - r(hQ) = r(P) - hr(Q)$ is contained in $\cap_{(i,j)\in E} \tau^{G,2c\varepsilon}_{q_j,p_i}$.
First, we observe that $g_r r(Q) = hr(Q) + t_r = r(P)$. Furthermore, due to Eq. (5.2), we get for all $(i,j) \in E$:

$$\exists n_{i,j} \in T(k) \colon g'_{i,j} := n_{i,j} h \in \tau^{G,c\varepsilon}_{r(Q),r(P)} \cap \tau^{G,c\varepsilon}_{q_j,p_i}. \tag{5.3}$$

Using the triangle inequality, Eq. (5.3) and Lemma 4.1.1 (with $x := hq_j$, $y := hr(Q)$ and the Euclidean distance), we get

$$\begin{aligned} \|p_i - g_r q_j\| &\leq \|p_i - g'_{i,j} q_j\| + \|g_r q_j - g'_{i,j} q_j\| \\ &\leq c\varepsilon + \|t_r h q_j - n_{i,j} h q_j\| \\ &= c\varepsilon + \|t_r h r(Q) - n_{i,j} h r(Q)\| \\ &= c\varepsilon + \|r(P) - g'_{i,j} r(Q)\| \\ &\leq c\varepsilon + c\varepsilon = 2c\varepsilon. \end{aligned}$$

□

Let us come back to Theorem 4.1.2. Since it can easily be seen that p_1 is a reference point for d^K, the preceding theorem can be viewed as a further generalization of Theorem 4.1.4. Just as Theorem 4.1.4 yielded an approximate matching algorithm for the distance measure d^K, Theorem 5.3.4 yields an approximate matching algorithm for arbitrary relational distance measures $\mathbf{d}$.

5.3.3 Matching Algorithms

Theorem 5.3.4 suggests a straightforward method for matching patterns using reference points under a transformation group $G = T(k) \rtimes H$:

Algorithm 5.3.5
Input: *G, P, Q, ε as in Convention 5.0.3; relational distance measure $\mathbf{d}$; reference points $r_P = r(P), r_Q = r(Q) \in V$ for some $(V, G, \mathbf{d}, c)$-reference mapping r*
Output: *According to Theorem 5.3.6*

```
Reference-Match(P, Q, d, ε, r_P, r_Q)
    Compute P̃ and Q̃ as defined in Theorem 5.3.4;
    return Relational-Distance-Match(P̃, Q̃, H, cε);
end.
```

The obvious advantage is that one has to deal with the group H, whose dimension is smaller than that of G (since the k parameters for the translational part can be neglected). As a further improvement for some groups, Algorithm 5.3.5 can make use of Algorithm `Ordered-Match` instead of Algorithm `Relational-Distance-Match` whenever one knows how to compute ordered suptransversals for the group H. As has been argued before, this is possible in the case of rigid motions and homothetic motions in $\mathbb{R}^2$.

Theorem 5.3.6 *Let P and Q be families of m and n points in the plane, respectively. Furthermore, let $G \in \{G_r, G_s\}$. Given input as specified, the output of Algorithm 5.3.5 is as follows:*

$$\begin{cases} g \in G(P, Q, c\varepsilon, \mathbf{d}) & \text{if } G(P, Q, \varepsilon, \mathbf{d}) \neq \emptyset, \\ false & \text{if } G(P, Q, 2c\varepsilon, \mathbf{d}) = \emptyset, \\ \textit{false or } g \in G(P, Q, c\varepsilon, \mathbf{d}) & \textit{otherwise.} \end{cases}$$

Furthermore, the algorithm takes $O((mn)^d(mn + T(\mathbf{d}, m, n)) + R(m) + R(n))$ time, where computing $r(P)$ and $r(Q)$ takes $R(m)$ and $R(n)$ time, respectively.

The time complexities resulting from this theorem are stated in the table below, which also compares the time bounds achieved to known results for these specific problems.
The quality of approximation can be further improved by running algorithm 5.3.5 not only for the reference points themselves, but also for a certain number of sample points from the $c\varepsilon$-neighborhood of the reference point. This idea has first been applied by Schirra for the bottleneck distance. For details, we refer to Schirra's work [39].

5.4 Matching without Reference Points

We have just seen how reference points can be used for working with projected transporter sets, yielding efficient approximate matching algorithms. However, for many distance measures such as the directed Hausdorff distance, no reference points are known; for some distance measures, such as the directed Hausdorff distance, it can even be shown that there are no reference points at all. If we want to obtain faster algorithms based on eliminating translation

Group	$\mathbf{d}_{\mathrm{H}}$		$\mathbf{d}_{\mathrm{B}}$	
	Our result	Known result	Our result	Known result
$T(k)$		$O((m+n)\log(m+n))$[3]	$O(n^{2.5})$	$O(n^{1.5})$[39]
SE(2), HT(2)	$O(mn\log(mn))$	$O(mn\log(mn)\cdot\log^*(mn))$ [3]	$O(n^{4.5})$	$O(n^{2.5})$[39]
SM(2)	$O((mn)^3)$	see SE(2) [3]	$O(n^{6.5})$	
SE(3)	$O((mn)^4)$	$O((mn)^3)$ [21]	$O(n^{8.5})$	$O(n^{8.5})$ [20]

Table 5.2: Time complexities for solving matching tasks using reference points. Note that the running times of known methods referred to are much more specialized for the given matching task and hence yield better asymptotical running times than the algorithms introduced here, which result from a generic framework.

components in these cases, we have to find alternative ways to provide matching algorithms. The basic idea in this situation is to try whether some point p_i (or some point q_j) can serve as a reference point. As a first step, we assume that the distance measure $\mathbf{d}$ has the additional property that if $\mathbf{d}(P,Q) \leq \varepsilon$, each point in Q is identified with some point in P. We formalize this as follows:

Definition 5.4.1 *Let* $\mathbf{d}$ *be a relational distance measure. We say that* $\mathbf{d}$ is right-complete *iff* $E \in \mathbf{R}(\mathbf{d}, m, n)$ *implies that for each* $j \in [1:n]$ *there is an* $i \in [1:m]$ *such that* $(i,j) \in E$.

Obviously, the directed Hausdorff distance is right-complete. The algorithmic advantage that can be drawn from right-complete distance measures is that $g \in G(P,Q,\varepsilon,\mathbf{d})$ implies the existence of some $i \in [1:m]$ such that $\|p_i - gq_1\| \leq \varepsilon$. We use the point pair (p_i, q_1) as our substitute for the pair of reference points $(r(P), r(Q))$. Our strategy for designing a matching algorithm is to try for each $i \in [1:m]$ if p_i is a suitable "reference point". This allows us to state the following matching algorithm:

Algorithm 5.4.2
Input: $P \in V^{[1:m]}$ *and* $Q \in V^{[1:n]}$; $\varepsilon \geq 0$, $G = T(k) \rtimes H \leq \mathrm{AGL}(k)$
Output: *According to Theorem 5.4.3*

```
Right-Complete-Match(P, Q, ε)
    for (i ∈ [1:m])
       M:=Reference-Match(d, P, Q, ε, p_i, q_1)
       if (M ∈ G) then return M;
    return false;
end.
```

We summarize our results in the following

Theorem 5.4.3 *Given input as specified, the output of Algorithm 5.4.2, denoted by* M*, is as follows:*

1. $G(P,Q,\varepsilon,\mathbf{d}) \neq \emptyset \Rightarrow M \in G(P,Q,\mathbf{d},2\varepsilon)$
2. $G(P,Q,2\varepsilon,\mathbf{d}) = \emptyset \Rightarrow M = false.$

Furthermore, the algorithm takes $O((mn)^d(m^2n + T(\mathbf{d}, m, n, \varepsilon)))$ *time.*

In case $m = O(n)$, the asymptotic time bound is the same as for the algorithm proposed in [21] which, however, allows to (approximately) determine the minimum value of ε so that $G_P^\varepsilon(Q) \neq \emptyset$ for matching under the group of rigid motions; Algorithm 5.4.2 only allows to solve the decision problem. However, it works for both the group of rigid motions and the group of homothetic motions in the plane.
Based on the method from [21], Indyk, Motwani and Venkatasubramanian [46] proposed an approximate algorithm for the decision problem under the group of rigid motions with respect to the Hausdorff distance. The analysis of their matching algorithm incorporates combinatorial distance bounds of point sets in $\mathbb{R}^2$, leading to asymptotically smaller time complexities if the specified distance ε for the decision algorithm is significantly smaller than the diameter of the point set Q. Similar results on combinatorial distance bounds can be applied to Algorithm 5.4.2.
Finally, it should be mentioned that the reductions to combinatorial pattern matching problems (see [46, 19]) yield asymptotically smaller time bounds than the time complexities stated in Table 5.3. Concerning these results, we refer to the remarks made in Section 5.1.

Group	d_{H}	
	Our result	Known result
SE(2)	$O(m^2 n \log(mn))$	$O(m^2 n \log m)$
SM(2)	$O(m^3 n^4 \log m)$	
SE(3)	$O(m^4 n^5 \log m)$	$O(m^3 n \log m)$

Table 5.3: Some time complexities for solving matching tasks approximately involving right complete distance measures (and, whenever possible, ordered cell enumeration).

The above algorithm can be extended in a straightforward way to work for distance measures that are not right-complete. One simply introduces an extra `for` loop so that instead of testing all m pairs (p_i, q_1) as a reference point, one tests all mn pairs (p_i, q_j). The resulting running time is the time bound stated in Theorem 5.4.3 with an extra factor of n.

5.5 Finding Largest Common Point Sets

The algorithms from the last sections enable us to find out whether some query object Q can be transformed so that it becomes similiar to P or an (approximate) part of a pattern P. In some applications, one is interested in the largest subset Q' of Q that has a small distance $\mathbf{d}$ to some (preferably large) subset P' of P. In this section, we provide the concept of *largest common point sets* as a generic approach to tackle such problems under arbitrary relational distance measures.
We consider weight functions that, given two point sequences $P, Q \in V^+$ and a fault tolerance parameter ε, assign a weight (or a degree of correspondence) of P and Q with distance ε, i.e., W is a mapping

$$W\colon V^+ \times V^+ \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}.$$

As an example for such a weight function, let $W_{\mathrm{B}}(P, Q, \varepsilon)$ denote the maximum cardinality matching in the bipartite graph defined by $R(P, Q, \varepsilon)$, see Figure 5.3 for an instance where

$W_{\mathrm{B}}(P, Q, \varepsilon) = 5$. The maximum cardinality matching yields maximal sets $Q' \subseteq Q$ and $P' \subseteq P$ of equal cardinality such that $\mathbf{d}_{\mathrm{B}}(P', Q') \leq \varepsilon$. In order to apply the technique of cell enumeration, we usually want the weight function W to be *invariant with respect to* $\sim_{P,Q,G,\varepsilon}$*-cells* in the sense that for any two $g, g' \in G$, we have

$$g \sim_{P,Q,G,\varepsilon} g' \quad \Longrightarrow \quad W(P, gQ, \varepsilon) = W(P, g'Q, \varepsilon). \tag{5.4}$$

One way to achieve this is to provide a weight function that takes $R(P, Q, \varepsilon)$ as a parameter rather than P, Q and ε themselves: suppose we have a mapping $w\colon 2^{[1:m]\times[1:n]} \to \mathbb{R}_{\geq 0}$, then we obtain a weight function $W\colon V^+ \times V^+ \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ by defining $W_w(P, Q, \varepsilon) := w(R(P, Q, \varepsilon))$. Obviously, W_w satisfies the condition from Eq. (5.4). As an example, consider W_{B}: the maximum cardinality matching can be computed merely by considering the bipartite graph defined by $R(P, Q, \varepsilon)$.

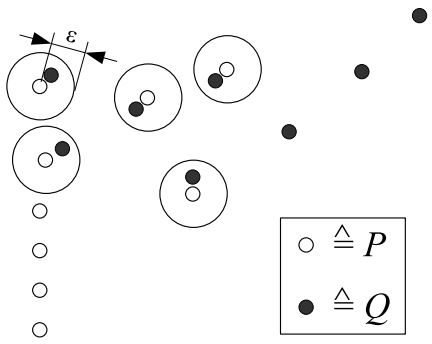


Figure 5.3: Instance of P, Q and ε with $W_{\mathrm{B}}(P, Q, \varepsilon) = 5$ (since the maximum cardinality matching in the bipartite graph defined by $R(P, Q, \varepsilon)$ contains 5 edges).

Our goal is to find a transformation $g \in G$ that maximizes $W(P, gQ, \varepsilon)$. For suitable weight functions W, maximizing $W(P, gQ, \varepsilon)$ solves the problem of determining a *largest common point set* with respect to the Hausdorff and the bottleneck distance, as studied in [11, 20]. Finding largest common point sets will also be a suitable concept for finding *largest common subcurves* with respect to the (discrete) Fréchet distance in Chapter 7. This motivates the following definition.

Definition 5.5.1 *Let $V = \mathbb{R}^k$ be a Euclidean vector space, and let $P, Q \in V^+$. Furthermore, let $G \leq \mathrm{AGL}(k)$ be a group acting on V and let $W\colon V^+ \times V^+ \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denote a weight function. The* largest common point set of P and Q with distance ε and with respect to W under G *is defined as*

$$\mathrm{LCP}\,(P, Q, G, W, \varepsilon) := \max_{g \in G} W(P, gQ, \varepsilon).$$

In case that W is invariant with respect to $\sim_{P,Q,G,\varepsilon}$-cells, $\mathrm{LCP}\,(P, Q, G, W, \varepsilon)$ can be computed using a variant of Algorithm 5.2.4 based on cell enumeration. If $G = T(k) \rtimes H$, $H \leq \mathrm{GL}(k)$, we can make use of projecting transporter sets through Theorem 5.3.4, as shown by Algorithm 5.5.2. This algorithm incorporates the idea behind Algorithm 5.4.2 in the sense that some pair (p_i, q_j) serves as a substitute for a pair of reference points. In the case of right-complete distance measures, only each p_i needed to be tested for all $i \in [1 : m]$, since q_1 was guaranteed

to match some point p_i. In case of computing LCP $(P, Q, W, G, \varepsilon)$, this is *not* guaranteed. The only fact that can be used is that *some* q_j is matched with *some* p_i. Hence, an extra `for`-loop running over all q_j, $j \in [1:n]$, is introduced to the following algorithm.

Algorithm 5.5.2
Input: $P \in V^{[1:m]}, Q \in V^{[1:n]}; \varepsilon \geq 0$
Output: *According to Lemma 5.5.3*

```
Relational-LCP(P, Q, ε)
    w := −∞;
    for (i ∈ [1 : m])
       for (j ∈ [1 : n])
          Determine P := {τ^{G,ε}_{q_j,p_i} | i ∈ [1 : m], j ∈ [1 : n]};
          Compute a set C ⊂ R^d of cell representatives of P;
          for g ∈ C
             if W(R(P, gQ, ε)) > w
                w := W(R(P, gQ, ε));
    return w;
end.
```

Note that the above algorithm can also make use of ordered cell enumerations. We require a dynamic data structure for the function W instead of a dynamic data structure for the distance measure $\mathbf{d}$. The generalization of Definition 5.2.5 is straightforward and hence omitted.

Lemma 5.5.3 *Let ℓ be the output of Algorithm 5.5.2. Then,* LCP $(G, W, \varepsilon, P, Q) \leq \ell \leq$ LCP $(G, W, 2\varepsilon, P, Q)$. *Furthermore, the algorithm takes $O((mn)^d(m^2n^2 + T(W, m, n)))$ time, where $T(W, m, n)$ denotes the time for computing $W(R(P, gQ, \varepsilon))$.*

Remark 5.5.4 *Computing the largest common point set is very closely related to computing G-matches allowing a certain number of mismatches (in case of the Hausdorff distance, this problem is also known as the* percentile based Hausdorff distance *[43]): Defining*

$$G(P, Q, \varepsilon, \mathbf{d}, k) := \{g \in G \mid \exists Q' \subseteq Q \colon |Q'| \geq |Q| - k \wedge \mathbf{d}(P, Q') \leq \varepsilon\},$$

we get

$$G(P, Q, \varepsilon, \mathbf{d}) = G(P, Q, \varepsilon, \mathbf{d}, 0)$$

and

$$\text{LCP}\,(G, W, \mathbf{d}, \varepsilon, P, Q) = |Q| - \max\{k \in \mathbb{N} \mid G(P, Q, \varepsilon, \mathbf{d}, k) \neq \emptyset\}.$$

The algorithms proposed in this chapter can be adapted to compute G-matches allowing mismatches with little effort. However, we do not elaborate on this any further.

5.6 Non-Relational Distance Measures by Exchanging Norms

As demonstrated by the example in Figure 5.1, it is easy to find examples of distance measures that are not relational. In may cases, non-relational distance measures are closely related to a corresponding relational distance measure. For example, the mean-value Hausdorff distance

$\mathbf{D}_{\mathrm{H}}$ is very closely related the Hausdorff distance. The difference between d_{H} and $\mathbf{D}_{\mathrm{H}}$ is that an ℓ_∞-norm (taking the maximum of all nearest neighbors) is exchanged by an ℓ_1-norm (summing up all nearest-neighbors).
In this section, we show that for some distance measures resulting from such a change of a norm, one can also find (approximate) matching algorithms by computing suptransversals of arrangements defined by transporter sets. To begin with, we consider the distance measure $\mathbf{D}_{\mathrm{H}}$ defined in Section 5.2.1. $\mathbf{D}_{\mathrm{H}}(P,Q)$ can be computed in $O(n \log m)$ time in the same manner as $d_{\mathrm{H}}(P,Q)$. To the best of the author's knowledge, the only matching algorithm for matching with respect to $\mathbf{D}_{\mathrm{H}}$ is the one proposed by Agarwal, Har-Peled, Sharir and Wang [1]. Given two sets P and Q of m and n points, respectively, in $V = \mathbb{R}^k$, they provide a randomized and approximate $O(mn \log(mn))$-time algorithm for finding a translation $t \in T(k)$ that minimizes $\mathbf{D}_{\mathrm{H}}(P,Q)$. The algorithms proposed in this section apply to matching under arbitrary linear algebraic groups acting rationally on $V = \mathbb{R}^k$.
Instead of considering all (G,ε)-transporters from some q_j to some p_i, we take a look at a larger set of transporters: for each $\ell \in [1:n]$, we consider all $(G,\ell\varepsilon)$-transporters from some q_j to some p_i. Hence, we need to deal with the arrangement defined by the family of transporters

$$\langle \tau_{q_j,p_i}^{G,\ell\varepsilon} \rangle_{i\in[1:m],j,\ell\in[1:n]}, \tag{5.5}$$

which is not defined by mn many transporter sets but by mn^2 many. The matching algorithm then proceeds in the same way as Algorithm 5.2.4 — for each representative g from the suptransversal of the arrangement, we compute $\mathbf{D}_{\mathrm{H}}(P,gQ)$. If for some representative g, $\mathbf{D}_{\mathrm{H}}(P,gQ) \leq \varepsilon$, then g is returned as output. If for all representatives g, $\mathbf{D}_{\mathrm{H}}(P,gQ) > \varepsilon$, *false* is returned as output.
It remains to be shown that this procedure yields an approximate solution to the problem of matching with respect to $\mathbf{D}_{\mathrm{H}}$. To be more precise, we claim that the procedure described above yields the following output:

$$\begin{cases} g \in G(P,Q,2\varepsilon,\mathbf{D}_{\mathrm{H}}) & \text{if } G(P,Q,\varepsilon,\mathbf{D}_{\mathrm{H}}) \neq \emptyset, \\ \mathit{false} & \text{if } G(P,Q,2\varepsilon,\mathbf{D}_{\mathrm{H}}) = \emptyset, \\ \mathit{false} \text{ or } g \in G(P,Q,\varepsilon,\mathbf{D}_{\mathrm{H}}) & \text{otherwise.} \end{cases} \tag{5.6}$$

We first show that $G(P,Q,\varepsilon,\mathbf{D}_{\mathrm{H}}) \neq \emptyset$ implies output $g \in G(P,Q,2\varepsilon,\mathbf{D}_{\mathrm{H}})$. To this end, observe that $G(P,Q,\varepsilon,\mathbf{D}_{\mathrm{H}}) \neq \emptyset$ implies that there is some g' such that $\mathbf{D}_{\mathrm{H}}(P,g'Q) \leq \varepsilon$. In particular, this implies

$$\sum_{j\in[1:n]} \|p_{\nu_j} - g'q_j\| \leq n\varepsilon, \tag{5.7}$$

where p_{ν_j} denotes the nearest neighbor of $g'q_j$ in P. Eq. (5.7) implies that for all $j \in [1:n]$, we have $\|p_{\nu_j} - g'q_j\| \leq n\varepsilon$, in other words, for all $j \in [1:n]$, there is a uniquely defined $\ell_j \in [1:n]$ such that $(\ell_j - 1)\varepsilon < \|p_{\nu_j} - g'q_j\| \leq \ell_j\varepsilon$. Hence,

$$\sum_{j\in[1:n]} \ell_j \leq 2n. \tag{5.8}$$

In terms of transporter sets, this is equivalent to writing

$$g' \in \bigcap_{j\in[1:n]} \tau_{q_j,p_{\nu_j}}^{G,\ell_j\varepsilon} \setminus \tau_{q_j,p_{\nu_j}}^{G,(\ell_j-1)\varepsilon}. \tag{5.9}$$

We now take a closer look at the intersection from Eq. (5.9). Using the triangle inequality in combination with Eq. (5.8), we obtain for each g contained in this intersection

$$\sum_{j\in[1:n]} \|p_{\nu_j} - gq_j\| \leq \sum_{j\in[1:n]} \ell_j\varepsilon \leq 2n\varepsilon$$

Now, computing a cell enumeration of the arrangement defined by the mn^2 transporter sets, such g will be computed at some point during the execution of the algorithm, producing the desired output $g \in G(P, Q, 2\varepsilon, \mathbf{D}_{\mathrm{H}})$.
In order to prove Eq. (5.6), it remains to be shown that $G(P, Q, 2\varepsilon, \mathbf{D}_{\mathrm{H}}) = \emptyset$ implies output *false*. We show the reverse implication: if the output is not *false*, we have $G(P, Q, 2\varepsilon, \mathbf{D}_{\mathrm{H}}) \neq \emptyset$. This implication obviously holds, since any output g of the algorithm is by construction contained in $G(P, Q, 2\varepsilon, \mathbf{D}_{\mathrm{H}})$.

The algorithm described in this section can be modified in a straightforward way to be used for matching with respect to the mean square Hausdorff distance $\mathbf{D}_{\mathrm{H}}^2$ that was introduced in Section 5.2.1 as well. Instead of dealing with the arrangement defined by the family of transporters defined in Eq. (5.5), we work with the arrangement defined by the family of transporters

$$\langle \tau_{q_j,p_i}^{G,\ell\varepsilon^2} \rangle_{i\in[1:m],j,\ell\in[1:n]},$$

which is defined by mn^2 many transporter sets as well. The rest of the algorithm works the same way, yielding the same quality of approximation.

As a closing remark for this chapter, it should be mentioned that in Chapter 7, we show that a change from an ℓ_∞-norm to an ℓ_2-norm leads from the discrete Fréchet distance to the dynamic time warping distance.
Finally, the ideas described in the last section can be applied to (approximately) finding largest common point sets with respect to $\mathbf{D}_{\mathrm{H}}$ by means of the ideas from Section 5.5.

Chapter 6

Pattern Matching Using Candidate Sets

In the last chapter, we have seen a generic approach for matching families of points with respect to relational distance measures. The results from Chapter 5, however, leave some issues unresolved — the cell enumeration routines that we refer to for matching under larger groups than rigid or homothetic motions in the plane cannot be considered to be implementable. Hence, the question arises whether one can state alternative approaches that are easier to implement.
We provide (at least partially) an answer in this chapter: we first give a practical, generic method for matching under rigid motions in the plane with respect to relational distance measures. In the second part of this chapter, we show how this method generalizes to matching under rigid motions in three dimensions.

6.1 Introduction and Related Work

The algorithm for matching with respect to a relational distance measure $\mathbf{d}$ under the group SE(3) resulting from Algorithm 5.3.5 cannot be considered to be practical, because it requires cell enumeration techniques for polynomials in three variables (for a discussion of the practical relevance of these techniques, we refer to the discussion in Chapter 2). For the special case of $V = \mathbb{R}^3$ and $G = \mathrm{SE}(3)$, however, we can state algorithms that can be implemented easily. In addition to being easier to implement, these algorithms yield approximate solutions to the *minimization* problem rather than the decision problem of determining whether $G(P, Q, \varepsilon, \mathbf{d})$ is non-empty.
Throughout this section, we always have $V = \mathbb{R}^2$ or $V = \mathbb{R}^3$, and $P \in V^{[1:m]}$ as well as $Q \in V^{[1:n]}$. Furthermore, $\mathbf{d}$ denotes a relational distance measure, and the transformation group we consider either is the group $G = \mathrm{SE}(2)$ of rigid motions in the plane or the group of rigid motions in three-space, $G = \mathrm{SE}(3)$. We develop an algorithm that gives a ξ-approximate solution to the decision problem whether $G(P, Q, \varepsilon, \mathbf{d})$ is empty or not, for $\xi \geq 6$ in the two dimensional case and $\xi \geq 16$ in the three dimensional case. The method relies on ideas stated by Chakraborty et al. [20] for finding approximate largest common point sets with respect to the bottleneck distance. Their algorithm, in turn, is based on ideas by Goodrich et al. [36] as well as some work by Akutsu [2]. Like the algorithms based on cell enumeration, this method computes a finite set C of transformations and, for each $g \in C$, computes $\mathbf{d}(P, gQ)$.

The algorithms presented in this section use certain sets of *candidate transformations* (or *candidate sets*, for short) instead of suptransversals of arrangements.
We start with the simplest and most general case of matching under SE(2) with respect to an arbitrary relational distance measure **d**. After that, we show how reference points or right-completeness can be used to obtain faster algorithms than the algorithm for arbitrary relational distance measures. The asymptotically best time bounds can be achieved in the case that the distance measure **d** has both properties, i.e, **d** has a reference point and is right-complete.
After providing algorithms for matching under SE(2) in Section 6.2, we generalize these methods (in the spirit of Goodrich et al. [36]) to matching point sequences in $\mathbb{R}^3$ under the group SE(3) in Section 6.3. Again, we can state faster algorithms for distance measures with reference points as well as for right-complete distance measures and distance measures having both, right-completeness and a reference point.
Note that the same structural properties — reference points and right-completeness — are used in designing algorithms based on candidate sets as well as in designing algorithms based on cell enumeration. This suggests that these concepts are quite natural properties of relational distance measures. In fact, we encounter further distance measures satisfying these properties in Chapter 7, where different discrete versions of the Fréchet distance are introduced.
Before we develop algorithms, we introduce some notation. For $x \in \mathbb{R}^2$, we write $\mathrm{SO}_x(2)$ for the group of all rotations about the point x. For $x \in \mathbb{R}^3$ and $y \in \mathbb{R}^3$, $x \neq y$, we denote the ray starting at x and going through y by $[x;y] := \{x + \lambda(y-x) \mid \lambda \geq 0\}$. Furthermore, we write $\mathrm{SO}_x(3)$ for the group of all rotations about a point $x \in \mathbb{R}^3$. By $\mathrm{SO}_{x,y}(3)$, we denote the group of all rotations around the axis $[x;y]$. Note that $\mathrm{SO}_{x,y}(3) \simeq \mathrm{SO}(2)$. In analogy to the ray $[x;y]$, $[x;y;z] := \{x + \kappa(y-x) + \lambda(z-x) \mid \kappa \in \mathbb{R}, \lambda > 0\}$ denotes the half-plane defined by the three non-collinear points x, y and z.

6.2 Matching under SE(2) Using Candidate Sets

Throughout this section, let V denote the Euclidean two-space. The notion of candidate sets is motivated by a simple observation: Suppose we are given two pairs of points in the plane, (a_1, a_2) and (b_1, b_2). Then, there is a uniquely defined rigid motion that transports b_1 onto a_1 and makes the two straight lines defined by the two point pairs collinear, see Figure 6.1 for an example. This motivates us to formally define a candidate transformation as follows.

Definition 6.2.1 *Let $V = \mathbb{R}^2$ and let $A, B \in V^2$, where $A = (a_1, a_2)$ and $B = (b_1, b_2)$. We say that $g \in \mathrm{SE}(2)$ is an* (A, B)-candidate transformation *iff*

(C1) $a_1 = gb_1$ *and*

(C2) $[a_1; a_2] = [gb_1; gb_2]$.

Note that actually, (C2) implies (C1). For constructing an (A, B)-candidate transformation, however, it turns out to useful to establish first (C1) and then (C2). Furthermore, note that if $a_1 \neq a_2$ and $b_1 \neq b_2$, the candidate transformation is uniquely defined.

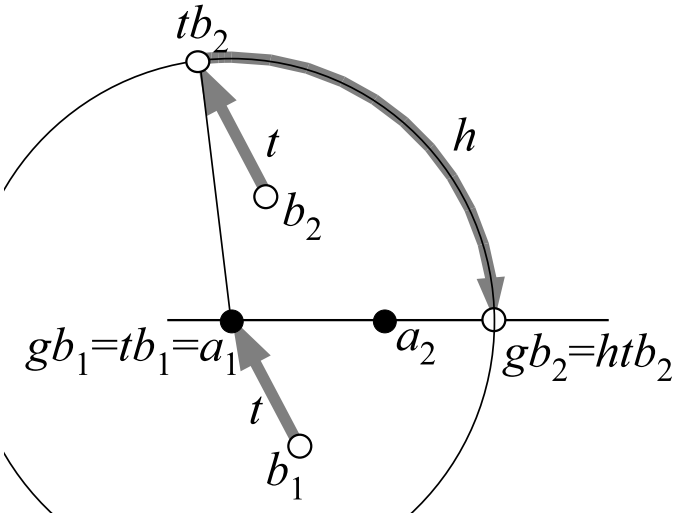


Figure 6.1: Construction of an (A, B)-candidate transformation g composed of a translation t and a rotation $h \in \mathrm{SO}_{a_1}(2)$, such that $g = th$.

6.2.1 Arbitrary Relational Distance Measures

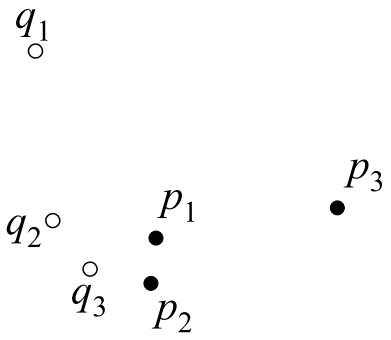
In a pattern matching scenario, we are given two sequences of points $P = \langle p_1, \ldots, p_m \rangle$ and $Q = \langle q_1, \ldots, q_n \rangle$. The idea underlying our basic pattern matching algorithm based on candidate transformations is to generate all point pairs (p_{i_1}, p_{i_2}) contained in P as well as all point pairs (q_{j_1}, q_{j_2}) contained in Q. Then, for all $A = (p_{i_1}, p_{i_2})$ and for all $B = (q_{j_1}, q_{j_2})$, we compute an (A, B)-candidate transformation g. We also refer to the set of all candidate transformations as a *candidate set*. For each element g of such a candidate set, we compute $\mathbf{d}(P, gQ)$ and determine the candidate transformation h that yields the smallest distance. We now state the complete algorithm and refer to Figure 6.2 for an illustration.

Algorithm 6.2.2
Input: $P \in V^{[1:m]}$ *and* $Q \in V^{[1:n]}$*; relational distance measure* $\mathbf{d}$.
Output: *According to Lemma 6.2.3.*

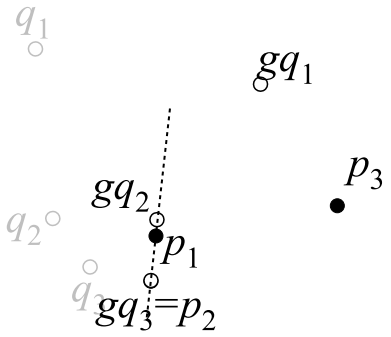
```
Candidate-Match(P, Q, d)
    D := ∞;
    for (i1, i2) ∈ {(μ1, μ2) ∈ [1 : m] × [1 : m] | μ1 ≠ μ2}
       for (j1, j2) ∈ {(ν1, ν2) ∈ [1 : n] × [1 : n] | ν1 ≠ ν2}
          A := (p_{i1}, p_{i2});
          B := (q_{j1}, q_{j2});
          Compute an (A, B)-candidate transformation g;
          d := d(P, gQ);
          if (d < D) then D := d; h := g;
    return h;
end.
```

It remains to be shown in what way this algorithm solves the problem of matching with respect to $\mathbf{d}$ under SE(2).

(a) Two point sets $P = \{p_1, p_2, p_3\}$ and $Q = \{q_1, q_2, q_3\}$ to be matched under SE(2) *with respect to the undirected Hausdorff distance.*



(b) Altogether, 36 *candidate transformations are computed. The figure on the right demonstrates an (A, B)-candidate transformation g for $A = (p_2, p_1)$ and $B = (q_3, q_2)$. However, $\mathbf{d}_{\mathrm{H}}(P, gQ)$ is rather large.*



(c) The (A, B)-candidate transformation that yields the smallest undirected Hausdorff distance is the candidate transformation h for $A = (p_2, p_3)$ and $B = (q_3, q_1)$. Hence, the algorithm returns h.

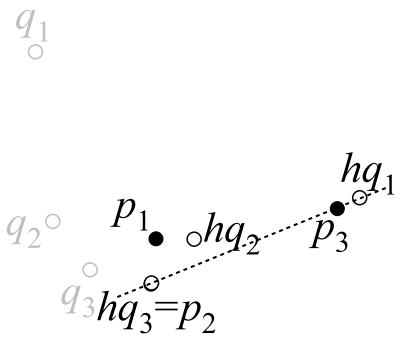


Figure 6.2: Illustration of Algorithm 6.2.2 for $\mathbf{d} = \mathbf{d}_{\mathrm{H}}$.

Lemma 6.2.3 *Let $G = \mathrm{SE}(2)$. Given input as specified, Algorithm 6.2.2 computes a transformation $h \in \mathrm{SE}(2)$ such that*

$$\mathbf{d}(P, hQ) \leq 6\varepsilon,$$

for all $\varepsilon > \inf_{h \in G} \mathbf{d}(P, hQ)$.

The proof of this lemma is based on the following observation that is also illustrated in Figure 6.4.

Remark 6.2.4 *Let $a_1, a_2, a_3 \in \mathbb{R}^2$ such that $\|a_1 - a_2\| \leq \|a_1 - a_3\|$. Then, $h \in \mathrm{SO}_{a_1}(2)$ implies*

$$\|a_2 - ha_2\| \leq \|a_3 - ha_3\|.$$

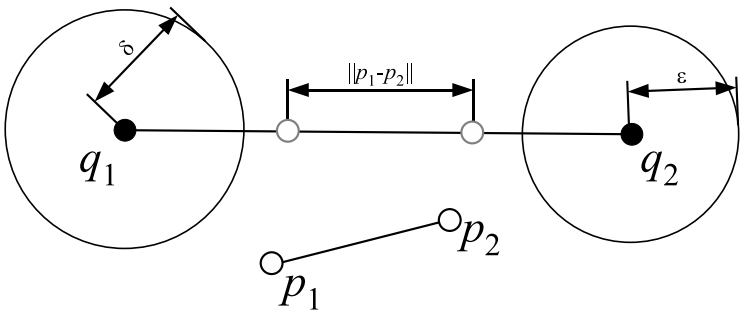


Figure 6.3: *Sketch for an indirect proof of Remark 6.2.5.* Suppose that $|\|q_1 - q_2\| - \|p_1 - p_2\|| > \varepsilon + \delta$ and, w.l.o.g, assume $\|q_1 - q_2\| > \|p_1 - p_2\|$. This results in the situation shown in the figure. Clearly, there is no point pair (p_1, p_2) such that $\|p_1 - q_1\| \leq \delta$ as well as $\|p_2 - q_2\| \leq \delta$.

Another bound based on some elementary geometric observations that will be useful later on is shown in Figure 6.3 and is stated in the following remark.

Remark 6.2.5 *Let $V = \mathbb{R}^k$ for some $k > 0$ and let $(p_1, p_2) \in V^2$ as well as $(q_1, q_2) \in V^2$. If $\|p_1 - q_1\| \leq \delta$ and $\|p_2 - q_2\| \leq \varepsilon$, then*

$$|\,\|q_1 - q_2\| - \|p_1 - p_2\|\,| \;\leq\; \varepsilon + \delta.$$

Proof of Lemma 6.2.3. Let $\varepsilon > \inf_{h \in G} \mathbf{d}(P, hQ)$. Then, there is a transformation $g \in G$ satisfying $\mathbf{d}(P, gQ) = \varepsilon$. Starting with g, we succesively construct group elements g_1 and g_2 such that g_1 satisfies property (C1), and g_2 satisfies (C1) as well as (C2) for some point pairs $A = (p_a, p_b)$ and $B = (q_c, q_d)$.

Constructing g_1. Let $Q' := \{q \in Q \mid \exists i \in [1:m] \colon \|p_i - gq\| \leq \varepsilon\}$, i.e., the set of all points in Q that are matched with some point in P. Furthermore, let $c, d \in [1:n]$ denote indices such that the pair (q_c, q_d) is a diameter pair of Q'. Since $q_c \in Q'$ and $\mathbf{d}$ is a relational distance measure, there is an index $a \in [1:m]$ such that $\|p_a - gq_c\| \leq \varepsilon$ as well as there is an index $b \in [1:m]$ such that $\|p_b - gq_d\| \leq \varepsilon$.
We now construct g_1 by defining t as the translation that maps gq_c to p_a. Then, for all $(i, j) \in R(P, gQ, \varepsilon)$ we have

$$\|gq_j - tgq_j\| = \|gq_c - tgq_c\| = \|gq_c - p_a\| \leq \varepsilon.$$

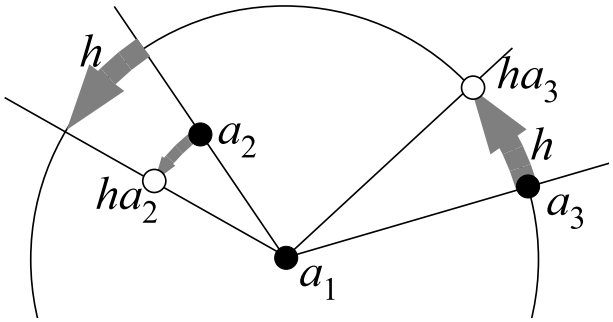


Figure 6.4: Illustration of Remark 6.2.4: Since $\|a_1 - a_2\| \leq \|a_1 - a_3\|$, we also have $\|a_2 - ha_2\| \leq \|a_3 - ha_3\|$ if h denotes a rotation about a_1.

Hence, for all $(i, j) \in R(P, gQ, \varepsilon)$, we obtain

$$\|p_i - tgq_j\| \leq \mathbf{d}(P, gQ) + \mathbf{d}(P, gQ) = 2\varepsilon, \tag{6.1}$$

for all $(i, j) \in R(P, Q, \mathbf{d}(P, gQ))$. Note that we have $tgq_c = p_a$, i.e., $g_1 := tg$ satisfies property (C1) of an (A, B)-candidate transformation for $A := (p_a, p_b)$ and $B := (q_c, q_d)$ for any index $b \in [1 : m]$.

Constructing g_2. We construct g_2 by providing a suitable index $b \in [1 : m]$. To this end, observe that since $q_d \in Q'$, and, since $\mathbf{d}$ is relational, there is an index $b \in [1 : m]$ such that $\|p_b - gq_d\| \leq \varepsilon$.
Using Eq. (6.1), we get $\|p_b - g_1 q_d\| \leq 2\varepsilon$. We now consider the uniquely defined rotation $h \in \mathrm{SO}_{p_a}(2)$ satisfying $[p_a; p_b] = [p_a; hg_1 q_d]$. As $hg_1 q_c = p_a$, the element $g_2 := hg_1$ becomes an (A, B)-candidate transformation.
As an elementary geometric argument shows (see Figure 6.5), we have

$$\|g_1 q_d - hg_1 q_d\| \leq 4\varepsilon. \tag{6.2}$$

By Remark 6.2.4 and the fact that q_d is the point in Q' farthest away from q_c, this implies $\|g_1 q - h_1 g_1 q\| \leq 4\varepsilon$ for all $q \in Q'$. Using the triangle inequality and Eq. (6.1), we obtain

$$\|p_i - g_2 q_j\| \leq 6\varepsilon \tag{6.3}$$

for all $(i, j) \in R(P, gQ, \varepsilon)$, and hence $\mathbf{d}(P, g_2 Q) \leq 6\mathbf{d}(P, gQ)$. Since $h \in \mathrm{SO}_{p_a}(2)$ leaves p_a unchanged (i.e., $hp_a = p_a$), $g_2 := hg_1$ satisfies (C1) as well as (C2) of an (A, B)-candidate transformation for $A = (p_a, p_b)$ and $B = (q_c, q_d)$.
Now we know that for some point pairs A and B, there exists an (A, B)-candidate transformation h satisfying $\mathbf{d}(P, hQ) \leq 6\varepsilon$. Since the algorithm checks all possible (A, B)-candidate transformations for any point pairs A and B from P and Q, respectively, our claim follows. □
Goodrich et al. [36] state an approximation factor of 4 rather than 6, as presented in the above proof. Their proof, however, rather suggests to use the factor 6 for their result as well. In terms of Figure 6.5, they use the inequality $\|tgq_d - htq_d\| \leq 2\varepsilon$ instead of $\|tgq_d - htq_d\| \leq 4\varepsilon$. In fact, Figure 6.5 shows that $\|tgq_d - htq_d\| \leq 2\varepsilon$ does *not* hold. It is, however, not clear whether the approximation factor 4 for the algorithm holds anyway.

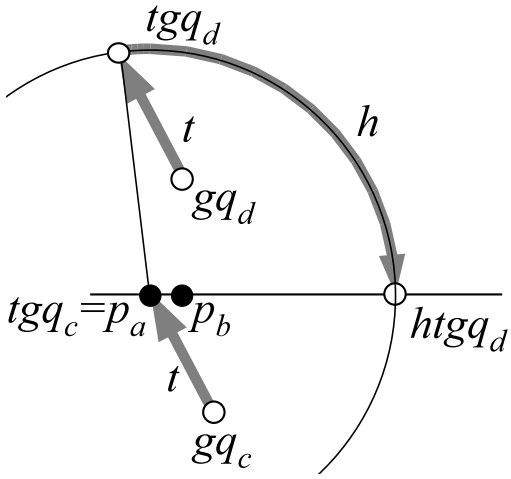


Figure 6.5: *Sketch for the bound claimed in Eq.* (6.2). By construction, we have $\|p_b - tgq_d\| \leq 2\varepsilon$, where $\varepsilon := \mathbf{d}(P, gQ)$. Furthermore, we have $\|p_b - htq_d\| = \|p_a - p_b\| - \|htgq_c - htgq_d\| = \|gq_c - gq_d\| \leq 2\varepsilon$ (where the last inequality follows from $\|p_a - gq_d\| \leq \varepsilon$ and $\|p_b - gq_d\| \leq \varepsilon$ using Remark 6.2.5 with $p_1 := p_a, p_2 := p_b, q_1 := gq_c, q_2 := gq_d$ and $\delta := \varepsilon$). Using the triangle inequality, we get $\|tgq_d - htq_d\| \leq 4\varepsilon$.

The time complexity of Algorithm 6.2.2 obviously is $O(m^2n^2T_{\mathbf{d}}(m,n))$ (where $T_{\mathbf{d}}(m,n)$ denotes the time needed for computing $\mathbf{d}(P,Q)$), since the two nested `for`-loops are passed through $O(m^2)$ and $O(n^2)$ times, respectively. In general, the running times of algorithms depending on candidate sets are determined mainly by the cardinality of the candidate set.

6.2.2 Right-Complete Distance Measures

In some cases, further knowledge about the distance measure $\mathbf{d}$ can be used for decreasing the cardinality of the candidate set that is needed for solving the matching problem. As a first class of such distance measures, we study right-complete distance measures. Recall that a distance measure $\mathbf{d}$ is right-complete iff for each relation $R \in \mathbf{R}(\mathbf{d}, m, n)$ and for all $j \in [1:n]$, there is an $i \in [1:m]$ such that $(i,j) \in R$ — in other words, whenever $\mathbf{d}(P,Q) \leq \varepsilon$ for some $P \in V^{[1:m]}$ and $Q \in V^{[1:n]}$, every point in Q is involved in the matching.
Looking at the proof of Lemma 6.2.3, we observe that from the point set Q', which is the set of all points from Q that are involved in a match, a diameter pair is used to find a suitable candidate transformation. If $\mathbf{d}$ is a right complete distance-measure, *all* points of Q are involved in a match. In terms of the proof of Lemma 6.3.3, this means that we always have $Q' = Q$. For designing a matching algorithm for right-complete distance measures, we can draw an essential advantage out of this: we can compute a diameter pair (q_c, q_d) in advance instead of trying *all* pairs (q_{j_1}, q_{j_2}).
For an algorithm matching with respect to a right-complete distance measure, the observation made above suggests the following algorithm: In Algorithm 6.2.6, we remove the inner `for`-loop and in place of this loop, we compute a diameter pair of Q at the beginning of the algorithm. In fact, this is the idea underlying the work [36] (which only considers matching with respect to the directed Hausdorff distance). Algorithm 6.2.2 is a generalization of this

approach to arbitrary right-complete distance measures.

Algorithm 6.2.6
Input: $P \in V^{[1:m]}$ *and* $Q \in V^{[1:n]}$*; relational and right-complete distance measure* $\mathbf{d}$.
Output: *Analogous to Lemma 6.2.3.*

```
Right-Complete-Candidate-Match(P, Q, d)
    Compute a diameter pair (q_c, q_d) of Q.
    B := (q_c, q_d);
    D := ∞;
    for (i_1, i_2) ∈ {(μ_1, μ_2) ∈ [1 : m] × [1 : m] | μ_1 ≠ μ_2}
        A := (p_{i_1}, p_{i_2});
        Compute an (A, B)-candidate transformation g;
        x := d(P, gQ);
        if (x < D) then D := x; h := g;
    return h;
end.
```

The output of the algorithm in fact satisfies the same approximation properties as Algorithm 6.2.2, and for a proof of these bounds, one only needs to modify the proof of Lemma 6.2.3 in a straightforward way. Hence, this proof is omitted.
For an analysis of the running time, note that the diameter pair of Q can be computed in $O(n \log n)$ time based on the convex hull of Q. Then, the remaining `for` loop runs through $O(m^2)$ cycles, while in each cycle, $\mathbf{d}(P, gQ)$ is computed exactly once. Hence, we obtain an overall running time of $O(n \log n + m^2 T_{\mathbf{d}}(m, n))$, where $T_{\mathbf{d}}(m, n)$ denotes the time needed for computing $\mathbf{d}(P, gQ)$.

6.2.3 Distance Measures with a Reference Point

Another helpful feature of a distance measure that can be used to make the time complexity of Algorithm 6.2.2 smaller are reference points. Let $\mathbf{d}$ denote a relational distance measure with a $(V, G, \mathbf{d}, c)$-reference mapping r for $G = \mathrm{SE}(2)$. Just as reference mappings helped to eliminate translations when matching by means of cell enumeration, reference points can be used to find the translation part of a candidate transformation. Recall that the proof of Lemma 6.2.3 starts with defining a translation from a point gq_c (for some fixed transformation $g \in \mathrm{SE}(2)$) to a point p_a. If we have a distance measure $\mathbf{d}$ with a reference mapping r, we can replace this step by using the translation from $gr(Q)$ to $r(P)$. As a result, the inner and the outer `for`-loop in Algorithm 6.2.2 only need to run through m and n cycles, respectively, instead of over $O(m^2)$ and $O(n^2)$ cycles.

Algorithm 6.2.7
Input: $P \in V^{[1:m]}, Q \in V^{[1:n]}$ *for* $V = \mathbb{R}^2$*; relational distance measure* $\mathbf{d}$*; reference points* $r_P = r(P), r_Q = r(Q) \in V$ *for some* $(V, G, \mathbf{d}, c)$*-reference mapping* r.
Output: *According to Lemma 6.2.8.*

```
Reference-Candidate-Match(P, Q, r_P, r_Q, d)
    D := ∞;
```

```
    for (i ∈ [1 : m])
       for (j ∈ [1 : n])
          A := (r_P, p_i);
          B := (r_Q, q_j);
          Compute a candidate transformation g for A and B;
          d := d(P, gQ);
          if (d < D) then D := d; h := g;
    return h;
end.
```

Lemma 6.2.8 *Let $G = \mathrm{SE}(2)$. Given input as specified, Algorithm 6.2.7 computes a transformation $h \in \mathrm{SE}(2)$ such that*

$$\mathbf{d}(P, hQ) \leq 2(c+1)\varepsilon,$$

for any $\varepsilon > \inf_{g \in G} \mathbf{d}(P, gQ)$.

Proof. The proof works similar to the proof of Lemma 6.2.3.
Let $\varepsilon > \inf_{g \in G} \mathbf{d}(P, gQ)$ and let $g \in G$ satisfy $\mathbf{d}(P, gQ) = \varepsilon$. Starting with g, we succesively construct group elements g_1 and g_2 such that g_1 satisfies property (C1), and g_2 satisfies (C1) as well as (C2) for some point pairs $A = (r_P, p_a)$ and $B = (r_Q, q_b)$.

Constructing g_1. We construct g_1 by defining t as the translation that transports gr_Q onto r_P. Since $r_P = r(P)$ and $r_Q = r(Q)$ and r is a reference mapping, we have $\|gr(Q) - r(P)\| = \|r(gQ) - r(P)\| \leq c\mathbf{d}(P, gQ) = c\varepsilon$. Hence, the length of the translation vector t is at most $c\varepsilon$, so that for any point q translated by t, we have $\|q - tq\| \leq c\varepsilon$. Using this in combination with the triangle inequality, we obtain

$$\begin{aligned} \|p_i - tgq_j\| &\leq \|p_i - gq_j\| + \|gq_j - tgq_j\| \\ &\leq \varepsilon + c\varepsilon = (c+1)\varepsilon, \end{aligned} \tag{6.4}$$

for all $(i, j) \in R(P, Q, \varepsilon)$. Note that we have $tgr_Q = r_P$, i.e., $g_1 := tg$ satisfies property (C1) of an (A, B)-candidate transformation for $A := (r_P, p_b)$ and $B := (r_Q, q_d)$ for any index $b \in [1 : m]$.

Constructing g_2. As in the proof of Lemma 6.2.3, let Q' denote the set of all points in Q that are matched with some point in P. Furthermore, let $d \in [1 : n]$ denote the index of a point in Q' that is farthest away from r_Q. Since $q_d \in Q'$ and $\mathbf{d}$ is a relational distance measure, there is an index $b \in [1 : m]$ such that $\|p_b - gq_d\| \leq \varepsilon$.
Using Eq. (6.4), we get $\|p_b - g_1 q_d\| \leq (c+1)\varepsilon$. We now consider the unique rotation $h \in \mathrm{SO}_{r_P}(2)$ satisfying $[r_P; p_b] = [hg_1 r_Q; hg_1 q_d]$. (Note that $hg_1 r_Q = r_P$.)
Then, using the triangle inequality in combination with Remark 6.2.5 (with $\delta := c$), we get

$$\begin{aligned} \|g_1 q_d - hg_1 q_d\| &\leq \|g_1 q_d - p_b\| + \|p_b - hg_1 q_d\| && (6.5) \\ &\leq (c+1)\varepsilon + (c+1)\varepsilon && (6.6) \\ &= 2(c+1)\varepsilon. && (6.7) \end{aligned}$$

By Remark 6.2.4 and the fact that q_d is the point in Q' farthest away from r_Q, this implies $\|g_1 q - h_1 g_1 q\| \leq 2(c+1)\varepsilon$ for all $q \in Q'$. Using the triangle inequality and Eq. (6.4), we obtain

$$\|p_i - g_2 q_j\| \leq 3(c+1)\varepsilon$$

for all $(i,j) \in R(P, gQ, \varepsilon)$. Since $h \in \mathrm{SO}_{r_P}(2)$ leaves $r_Q = g_1 r_Q$ unchanged, $g_2 := hg_1$ satisfies (C1) as well as (C2) and hence is an (A,B)-candidate transformation for $A = (r_P, p_b)$ and $B = (r_Q, q_d)$.
Since the algorithm checks all (A,B)-candidate transformations for any possible point pairs A and B, our claim follows. □

6.2.4 Right-Complete Distance Measures with a Reference Point

Even faster versions of Algorithm `Candidate-Matching` can be stated if the distance measure $\mathbf{d}$ involved has *both* a reference point and the property of right-completeness. Observe that in the proof of Lemma 6.2.8, we constructed the point $q \in Q'$ that has the largest distance to the reference point. If, as it is the case precisely for right-complete distance measures, we always have $Q' = Q$, i.e., all points of Q are involved in a matching, this point does not need to be "guessed" by the algorithm (by trying every single point in Q), but it can be computed in advance.
As a result, one `for`-loop of Algorithm `Reference-Candidate-Match` can be eliminated and replaced by computing the point in Q that is farthest away from r_Q.

Algorithm 6.2.9
Input: $P \in V^{[1:m]}, Q \in V^{[1:n]}$ *for* $V = \mathbb{R}^2$*; relational right-comlete distance measure* $\mathbf{d}$*; reference points* $r_P = r(P), r_Q = r(Q) \in V$ *for some* $(V, G, \mathbf{d}, c)$*-reference mapping* r*.*
Output: *Analogous to Lemma 6.2.8.*

```
Ref-Right-Complete-Match(P, Q, r_P, r_Q, d)
    Compute the point q ∈ Q that has the largest distance to r_Q;
    B := (r_Q, q);
    D := ∞;
    for (i ∈ [1 : m])
        A := (r_P, p_i);
        Compute a candidate transformation g for A and B;
        d := d(P, gQ);
        if (d < D) then D := d; h := g;
    return h;
end.
```

Again, the proof that the same bounds hold as in Lemma 6.2.8 works very similar to the proof of Lemma 6.2.8 and is hence omitted. For analysing the running time, note that the point q farthest away from r_Q can be computed in $O(n)$ time in a straightforward way. After that, the algorithm obviously performs m distance computations, which amounts to a total time complexity of $O(n + mT_{\mathbf{d}}(m,n))$.

6.3 Matching under SE(3) Using Candidate Sets

The algorithms for matching under rigid motions in the plane from the last section can be generalized for matching under rigid motions in three dimensions. Throughout this section, let $V = \mathbb{R}^3$. Corresponding to the last section, let $P \in V^{[1:m]}$, $Q \in V^{[1:n]}$ and let $\mathbf{d}$ denote a relational distance measure. For matching under SE(3), we need to extend the definition of

candidate transformations — a major property of the definition from the last section was that the candidate transformation for two point pairs is uniquely defined. This essential property, however, is not valid anymore in three dimensions: if $A = (a_1, a_2)$ and $B = (b_1, b_2)$, there is one degree of freedom left to rotate around the axis corresponding to $[b_1; b_2]$ without affecting the property of a candidate transformation. In other words, (A, B)-candidate transformations as defined for point pairs are uniquely defined *modulo* $\mathrm{SO}_{[a_1;a_2]}(2)$ if we replace $\mathbb{R}^2$ by $\mathbb{R}^3$ in Definition 6.2.1. A straightforward way to make candidate transformations uniquely defined in three dimensions (at least in non-degenerate cases) is to extend the definition from the last section to *triplets of points* instead of pairs of points. Viewing this from a group theoretical point of view, this is motivated by the fact that if we let $G = \mathrm{SE}(3)$ act on triplets of points, we obtain *trivial stabilizers* for each (non-collinear) triplet of points.

Definition 6.3.1 *Let $V = \mathbb{R}^3$, and let $A, B \in V^3$, where $A = (a_1, a_2, a_3)$ and $B = (b_1, b_2, b_3)$. We say that $g \in \mathrm{SE}(3)$ is an (A, B)-*candidate transformation *iff*

(C1) $a_1 = gb_1$ *and*

(C2) $[a_1; a_2] = [gb_1; gb_2]$ *and*

(C3) *If neither a_1, a_2, a_3 nor b_1, b_2, b_3 are collinear, we have $[a_1; a_2; a_3] = [gb_1; gb_2; gb_3]$.*

If neither the three points of A nor the three points of B are collinear, the candidate transformation is uniquely defined. In the degenerate case that either the points in A or the points in B are collinear, we will be content to find a single transformation from the set of all (A, B)-candidate transformations. As for the two dimensional case, we start with an algorithm for arbitrary relational distance measures and show how reference points and right-completeness can be used to state faster algorithms.

6.3.1 Arbitrary Relational Distance Measures

The basic idea for our basic pattern matching algorithm based on candidate transformations in three dimensions is the canonical extension of the algorithm in two dimensions: instead of generating all point pairs (p_{i_1}, p_{i_2}), we generate all triplets $(p_{i_1}, p_{i_2}, p_{i_3})$ contained in P. Just as well, we generate all triplets of points $(q_{j_1}, q_{j_2}, q_{j_3})$ in place of the pairs (q_{j_1}, q_{j_2}) contained in Q. Then, for all $A = (p_{i_1}, p_{i_2}, p_{i_3})$ and for all $B = (q_{j_1}, q_{j_2}, q_{j_3})$, we compute an (A, B)-candidate transformation g. For each candidate transformation g, we compute $\mathbf{d}(P, gQ)$ and determine the candidate transformation h that yields the smallest distance.

Algorithm 6.3.2
Input: *$P \in V^{[1:m]}$ and $Q \in V^{[1:n]}$; relational distance measure* $\mathbf{d}$.
Output: *As stated below.*

```
Candidate-Match(P, Q, d)
    D := ∞;
    for (i1, i2, i3) ∈ {(μ1, μ2, μ3) ∈ [1 : m]^3 | μ1 ≠ μ2, μ1 ≠ μ3, μ2 ≠ μ3}
       for (j1, j2, j3) ∈ {(ν1, ν2, ν3) ∈ [1 : n]^3 | ν1 ≠ ν2, ν1 ≠ ν3, ν2 ≠ ν3}
          A := (p_{i1}, p_{i2}, p_{i3});
          B := (q_{j1}, q_{j2}, q_{j3});
          Compute an (A, B)-candidate transformation g;
```

```
            d := d(P, gQ);
            if (d < D) then D := d; h := g;
    return h;
end.
```

Again, the fact that the output of the algorithm is a transformation $h \in \mathrm{SE}(3)$ such that $\mathbf{d}(P, hQ) \leq 16\varepsilon$, where $\varepsilon := \inf_{g \in \mathrm{SE}(3)} \mathbf{d}(P, gQ)$ can be shown similar to the proof of Lemma 6.2.8. The proof for the three dimensional case, however, is somewhat more involved. The running time obviously is $O(m^3 n^3 T_{\mathbf{d}}(m, n))$, since $O(m^3)$ triplets of points in P and $O(n^3)$ triplets of points in Q need to be examined.

Lemma 6.3.3 *Let $G = \mathrm{SE}(3)$. Given input as specified, Algorithm 6.3.2 computes a transformation $g \in \mathrm{SE}(3)$ such that*

$$\mathbf{d}(P, gQ) \leq 16\varepsilon,$$

for any $\varepsilon > \inf_{g \in G} \mathbf{d}(P, gQ)$.

The proof of this lemma is based on the following apparent generalization of Remark 6.2.4.

Remark 6.3.4 *Let $a_1, a_2, a_3 \in \mathbb{R}^3$ such that $\|a_1 - a_2\| \leq \|a_1 - a_3\|$. Furthermore, let $b_0, b_1, b_2, b_3 \in \mathbb{R}^3$ such that b_2 is closer to $[b_0; b_1]$ than b_3. Then, the following holds:*

(1) *If $h_1 \in \mathrm{SO}_{a_1}(3)$, then*

$$\|a_2 - h_1 a_2\| \leq \|a_3 - h_1 a_3\|.$$

(2) *If $h_2 \in \mathrm{SO}_{b_0, b_1}(3)$, then*

$$\|b_2 - h_2 b_2\| \leq \|b_3 - h_2 b_3\|.$$

Proof of Lemma 6.3.3. Let $\varepsilon > \inf_{h \in G} \mathbf{d}(P, hQ)$, and let $g \in G$ satisfy $\mathbf{d}(P, gQ) = \varepsilon$. Starting with g, we succesively construct group elements g_1, g_2 and g_3 such that g_1 satisfies property (C1), g_2 satisfies (C1) and (C2), and finally g_3 satisfies (C1)–(C3) for some pair of triplets (p_a, p_b, p_e) and (q_c, q_d, q_f).
The construction of g_1 and g_2 is almost the same as in the proof of Lemma 6.2.3. Hence, we only sketch these two steps.
Constructing g_1. Let $Q' := \{q \in Q \mid \exists i \in [1:m] \mid \|p_i - gq\| \leq \varepsilon\}$, i.e., the set of all points in Q that are matched with some point in P. Furthermore, let $c, d \in [1:n]$ denote indices such that the pair (q_c, q_d) is a diameter pair of Q'. Since $q_c, q_d \in Q'$ and $\mathbf{d}$ is relational, there are indices $a, b \in [1:m]$ such that $\|p_a - gq_c\| \leq \varepsilon$ and $\|p_b - gq_d\| \leq \varepsilon$.
We now construct g_1 by defining t as the translation that maps gq_c to p_a. Obviously, we have $\|gq_c - p_a\| \leq \varepsilon$. Translating each gq_j by t for all $j \in [1:n]$, we obtain

$$\|p_i - tgq_j\| \leq \varepsilon + \varepsilon = 2\varepsilon, \tag{6.8}$$

for all $(i, j) \in R(P, gQ, \varepsilon)$. Since we have $tgq_c = p_a$, i.e., $g_1 := tg$ satisfies property (C1) of an (A, B)-candidate transformation for $A := (p_a, p_b, p_e)$ and $B := (q_c, q_d, q_f)$ for any indices $e \in [1:m]$ and $f \in [1:n]$.

Constructing g_2. Using Eq. (6.8), we get $\|p_b - g_1 q_d\| \leq 2\varepsilon$. We now consider a rotation $h \in \mathrm{SO}_{p_a}(3)$ with the property that $[p_a; p_b] = [hg_1 q_c; hg_1 q_d]$. (Note that $[hg_1 q_c; hg_1 q_d] = [p_a; hg_1 q_d]$

and, furthermore, h is not uniquely defined.) An elementary geometric argument (cf. Figure 6.5) shows that $\|g_1 q_d - h g_1 q_d\| \leq 4\varepsilon$. By Remark 6.3.4(1) and the fact that q_d is the point in Q' farthest away from q_c, this implies $\|g_1 q - h_1 g_1 q\| \leq 4\varepsilon$ for all $q \in Q'$. Using the triangle inequality and Eq. (6.8), we obtain

$$\|p_i - g_2 q_j\| \leq 6\varepsilon \tag{6.9}$$

for all $(i,j) \in R(P, gQ, \varepsilon)$. Since $h \in \mathrm{SO}_{p_a}(3)$ leaves p_a unchanged (i.e., $hp_a = p_a$), $g_2 := hg_1$ satisfies (C1) as well as (C2) of an (A,B)-candidate transformation for $A = (p_a, p_b, p_e)$ and $B = (q_c, q_d, q_f)$ for any indices $e \in [1:m]$ and $f \in [1:n]$.

Constructing g_3. Let q_f be the point in Q' such that $g_2 q_f$ has the largest distance to the straight line corresponding to $[g_2 q_c; g_2 q_d]$. Due to $q_f \in Q'$, there is an index e such that $(e,f) \in R(P, gQ, \varepsilon)$, and hence (using Eq. (6.9)) we get $\|p_e - g_2 q_f\| \leq 4\varepsilon$. We now distinguish two cases:

(i) *p_a, p_b, p_e collinear or q_c, q_d, q_f collinear:* In this case, g_2 satisfies all criteria of a candidate transformation, and we are done by defining $g_3 := g_2$.

(ii) *Neither p_a, p_b, p_e nor q_c, q_d, q_f are collinear:* There is a uniquely defined transformation $h_2 \in \mathrm{SO}_{r_P, p_{i_1}}(3)$ so that $g_3 := h_2 g_2$ satisfies (C3). An elementary geometric argument yields

$$\begin{aligned} \|g_2 q_f - h_2 g_2 q_f\| &\leq \|g_2 q_f - p_e\| + \|p_e - h_2 g_2 q_f\| \\ &\leq \|g_2 q_f - g_1 q_f\| + \|g_1 q_f - p_e\| + \|p_e - h_2 g_2 q_f\| \\ &\leq 2\varepsilon + 6\varepsilon + 2\varepsilon \\ &= 10\varepsilon. \end{aligned} \tag{6.10}$$

Since $g_2 q_f$ is the point in Q' with the largest distance from $[p_a; p_b]$, Eq. (6.10) in combination with Remark 6.3.4(2) implies that for all $q \in Q'$, we have

$$\|q - h_2 q\| \leq 10\mathbf{d}(P, gQ).$$

Using this in combination with Eq. (6.9) and the triangle inequality, we get

$$\begin{aligned} \|p_i - g_3 q_j\| &\leq \|p_i - g_2 q_j\| + \|g_2 q_j - g_3 q_j\| \\ &\leq 6\varepsilon + 10\varepsilon \\ &= 16\varepsilon. \end{aligned}$$

for all $(i,j) \in R(P, gQ, \varepsilon)$. Since the algorithm examines all candidate transformations for all triplets of points from P and Q, our claim follows. □

Generalizing the idea of Chakraborty et al. [20], this algorithm can also be used for finding largest common point sets with respect to relational distance measures.

6.3.2 Making Use of Reference Points and Right-Completeness

In the case of rigid motions in the plane, reference points as well as right-completeness of a distance measure allowed us to state asymptotically faster algorithms than for arbitrary relational distance measures. The ideas used in the two dimensional case also carry into the three dimensional case. Since carrying these ideas to the three dimensional case is straightforward, we only provide sketches of the resulting algorithms in this section.

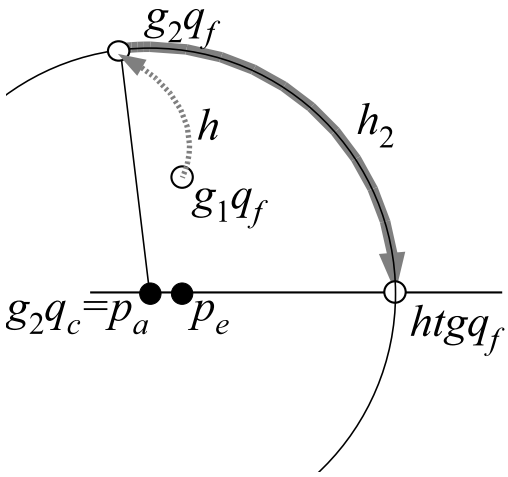


Figure 6.6: *Sketch for the bound claimed in Eq.* (6.10). By Eq. (6.8), we have $\|p_b - g_1 q_d\| \leq 2\varepsilon$, where $\varepsilon := \mathbf{d}(P, gQ)$. Furthermore, by Eq. (6.9), we have $\|g_1 q_f - h g_1 q_f\| \leq 6\varepsilon$. We also know that $\|p_b - htq_f\| = \|p_a - p_e\| - \|h_2 g_2 q_c - h_2 g_2 q_f\| = \|gq_c - gq_f\| \leq 2\varepsilon$ (where the last inequality follows from $\|p_a - gq_c\| \leq \varepsilon$ and $\|p_e - gq_f\| \leq \varepsilon$ using Remark 6.2.5 with $p_1 := p_a, p_2 := p_e, q_1 := gq_c, q_2 := gq_f$ and $\delta := \varepsilon$). Using the triangle inequality, we get $\|g_2 q_f - h_2 g_2 q_f\| \leq 10\varepsilon$.

Right-Complete Distance Measures. Consider the triplet (q_c, q_d, q_e) constructed in the proof of Lemma 6.3.3. These three points are points from the set $Q' \subseteq Q$ (which is the set of all points involved in the matching) having the following properties:

- (q_c, q_d) is a diameter pair of Q'
- q_e is a point that is farthest away from the straight line corresponding to $[q_c; q_d]$.

For a right-complete distance measure, we have $Q' = Q$. Hence, the triplet (q_c, q_d, q_e) does not need to be guessed by the algorithm (by trying all $O(n^3)$ possibilities) but can be computed in advance — and again, in doing so, we obtain the algorithm proposed in [36] for the special case of matching with respect to d_{H} under three dimensional rigid motions.
It remains to analyze the time complexity of the algorithm. To this end, we need to know an upper bound for computing a diameter pair of a three dimensional point set. Although diameter pair algorithms in three dimensions are much more involved than algorithms for the two dimensional case, the problem has been shown to be solvable in $O(n \log n)$ time for three dimensional point sets as well. We refer to [56] and [38] for further theoretical and practical results on this (surprisingly difficult) problem. Now, given a diameter pair (q_c, q_d), the point q_e can be found in linear time in an obvious way.
Since the inner `for`-loop can be eliminated completely, the remaining running time of the algorithm is determined by $O(m^3)$ distance computations, and altogether we get an asymptotical running time of $O(m^3 T_{\mathbf{d}}(m, n) + n \log n)$. The quality of approximation is the same as stated in Lemma 6.3.3.

Distance Measures with a Reference Mapping. Just as in the two dimensional case, the reference points $r(P)$ and $r(Q)$ obtained from a reference mapping r yield a translation that,

in terms of the proof of Lemma 6.3.3, is used for constructing the transformation g_1. The points $r(P)$ and $r(Q)$ serve as a substitute for the points p_a and q_c, and, in analogy to Lemma 6.2.8, the algorithm computes a transformation $h \in \mathrm{SE}(3)$ such that

$$\mathbf{d}(P, hQ) \leq 8(c+1)\varepsilon,$$

where $\varepsilon := \inf_{g \in G} \mathbf{d}(P, gQ)$. As can be easily seen, the running time of the algorithm is $O(m^2 n^2 T_{\mathbf{d}}(m,n))$, since only all pairs (instead of all triplets) of points from P and Q need to be tested.

Right-Complete Distance Measures with a Reference Mapping. In this case, we compute a triplet of points $(r(Q), q_d, q_f)$ such that

- q_d is a furthest neighbor of $r(Q)$ in Q
- q_f is the point farthest away from the straight line corresponding to $[r(Q); q_d]$.

Given these points, one `for`-loop needs to cycle over all point pairs (p_b, p_e) in P, since the candidate transformations can be computed for the triplets $A := (r(P), p_b, p_e)$ and $B := (r(Q), q_d, q_f)$. This leads to an overall running time of $O(m^2 T_{\mathbf{d}}(m,n) + n \log n)$.

Chapter 7

Matching with Respect to the Fréchet Distance

For all distance measures considered previously, it was not necessary to work with point sequences. The Hausdorff distance as well as the bottleneck distance can be defined for sets of points instead of sequences of points as well. In some applications, geometric shapes are not represented by point sets, but by *polygonal curves* that can be described by point sequences rather than sets of points. For polygonal curves, a wide range of distance measures has been studied; for example, the Hausdorff distance between the set of points on a (polygonal) curve P and the set of points on another (polygonal) curve Q can be used as a distance measure. For a survey on further distance measures, see [65]. The distance measure to be examined in this chapter is the *Fréchet distance* that is based on reparametrizations of the curves to be compared. In the context of curve simplification, it has been first examined by Alt and Godau [4]. For polygonal curves P and Q of lengths m and n, respectively, the Fréchet distance can be computed in $O(mn)$ time [5]. More recent work is concerned with relations to the Hausdorff distance [8] as well as matching polygonal curves with respect to the Fréchet distance under the group of translations in the plane [9]. In [66], the idea underlying the result of [9] is generalized to larger classes of transformations using techniques from real algebraic geometry, similar to those by Basu, Pollack and Roy summarized in Theorem 2.3.6.
In this chapter, a discrete version of the Fréchet distance is proposed, as well as bounds between this discrete version and the continuous Fréchet distance. Since the discrete version turns out to be a *relational* distance measure, the results from Chapters 4 and 5 can be applied to find efficient matching algorithms. Furthermore, the discrete version of the Fréchet distance (as well as the continuous version) has a reference point. In the last parts of this chapter, the problem of *matching subcurves* is addressed. In the terminology of Chapter 5, matching subcurves with respect to the discrete Fréchet distance leads to a problem involving a right-complete distance measure.

7.1 Notation and Basic Concepts

Let V denote the Euclidean vector space $\mathbb{R}^k$ with the Euclidean norm $\|.\| := \|.\|_2$ for some $k > 0$. A *curve in* V is a continuous mapping $f\colon [a, b] \to V$. A *polygonal curve of length* $m \in \mathbb{N}$ is defined as a curve $P\colon [0, m] \to V$ with the property that for all $i \in [0 : m - 1]$ the curve $P|_{[i,i+1]}$ is affine, i.e., $P(i + \lambda) = (1 - \lambda)P(i) + \lambda P(i + 1)$ for $\lambda \in [0, 1]$. Since

$x \in X^{[a:b]}$ is completely described by a sequence of $b-a+1$ values in X, we also write $x = \langle x_a, \ldots, x_b \rangle \in X^{[a:b]}$.
For $f \in V^I$, let $\|f\|_\infty := \sup_{t \in I} \|f(t)\|$. The *Fréchet distance* between $P \in V^{[0,m]}$ and $Q \in V^{[0,n]}$ (for some $m, n > 0$) is defined as

$$d_{\mathrm{F}}(P, Q) = \min_{(\alpha,\beta)} \|P \circ \alpha - Q \circ \beta\|_\infty,$$

where (α, β) ranges over all continuous, weakly increasing and surjective mappings $\alpha \in [0, m]^{[0,1]}$ and $\beta \in [0, n]^{[0,1]}$. In the sequel, we denote the set of all continuous, weakly increasing and surjective mappings from a subset X of $\mathbb{R}$ to another subset Y of $\mathbb{R}$ by $\mathrm{Mon}(X, Y)$ and write $\mathrm{Mon}_{m,n} := \mathrm{Mon}([0,1],[0,m]) \times \mathrm{Mon}([0,1],[0,n])$. In case $m \in \mathbb{N}$ and $P \in V^{[0,m]}$ denotes a polygonal curve, we can identify P with the mapping $[0:m] \ni i \mapsto P(i) =: p_i$, and hence we also write $P \in V^{[0:m]}$. More generally, we have the following definition.

Definition 7.1.1 *Let $v\colon [a, a'] \to V$ and $w\colon [b, b'] \to V$ be two curves. The* Fréchet-distance *$d_{\mathrm{F}}(v, w)$ of v and w is defined as*

$$d_{\mathrm{F}}(v, w) := \inf_{\alpha,\beta} \max_{t \in [0,1]} \|v(\alpha(t)) - w(\beta(t))\|,$$

where the infimum is taken over all $(\alpha, \beta) \in \mathrm{Mon}([0,1],[a,a']) \times \mathrm{Mon}([0,1],[b,b'])$.

The Fréchet-distance is a *pseudo metric*, i.e., d_{F} has all properties of a metric except for $d_{\mathrm{F}}(a, b) = 0 \Rightarrow a = b$, see [34, 31].
Given two polygonal curves $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$, we define the *discrete Fréchet distance* as

$$\mathbf{d}_{\mathrm{F}}(P, Q) = \min_{(\kappa,\lambda)} \|P \circ \kappa - Q \circ \lambda\|_\infty,$$

where the pairs (κ, λ) range over the set

$$\mathbf{Mon}_{m,n} := \mathrm{Mon}([0 : m+n],[0 : m]) \times \mathrm{Mon}([0 : m+n],[0 : n]).$$

(Note that for finite subsets X and Y of $\mathbb{R}$, every map $f\colon X \to Y$ is continuous, and hence in this case $\mathrm{Mon}(X, Y)$ is the set of all weakly increasing surjective mappings.) Correspondingly, one can define $\mathbf{d}_{\mathrm{F}}$ for polygonal curves $P \in V^{[a:b]}$ and $Q \in V^{[c:d]}$ for integers a, b, c, d by adapting domain and range of the reparametrizations. Note that the integer interval $[0 : m+n]$ substitutes the real interval $[0, 1]$ as the common domain for the two reparametrizations. The discrete Fréchet distance is similar to the *dynamic time warping distance* that is defined as

$$\mathbf{d}_{\mathrm{W}}(P, Q) := \min_{(\kappa,\lambda)} \|P \circ \kappa - Q \circ \lambda\|_2,$$

where (κ, λ) ranges over $\cup_{K \in [\max(m,n):m+n]} \mathbf{Mon}_{m,K}$ and $\|f\|_2 := (\|\sum_{i \in I} (f(t))^2\|)^{1/2}$ for $f \in V^I$ with $|I| < \infty$.
Dynamic time warping has been considered in the context of speech signal processing and time series databases [55], in both cases for $V = \mathbb{R}$. More recently, dynamic time warping has been used for matching polygonal curves in the plane under the group of translations [54]. The results presented in this chapter can be viewed as a bridge between these works and the results obtained in the area of computational geometry.
We can compute the discrete Fréchet distance between $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$ in $O(mn)$ time using dynamic programming:

Algorithm 7.1.2
Input: $P \in V^{[0:m]}, Q \in V^{[0:n]}$ *and* $V = \mathbb{R}^k$ *for some integers* $k, m, n > 0$.
Output: $\mathbf{d}_{\mathrm{F}}(P, Q)$

$d_{0,0} := \|p_0 - q_0\|$;
`for` $j := 1$ `to` n `do` $d_{0,j} := \max\{d_{0,j-1}, \|p_0 - q_j\|\}$;
`for` $i := 1$ `to` m
    $d_{i,0} := \max\{d_{i-1,0}, \|p_i - q_0\|\}$
    `for` $j := 1$ `to` n
        $d_{i,j} := \max\{\min\{d_{i,j-1}, d_{i-1,j}, d_{i-1,j-1}\}, \|p_i - q_j\|\}$;
    `end`
`end`
`return` $d_{m,n}$.

The verification of Algorithm 7.1.2 is based on the fact that

$$d_{i,j} := \mathbf{d}_{\mathrm{F}}(P|_{[0:i]}, Q_{[0:j]})$$

satisfies

$$d_{i,j} = \max\{\min\{d_{i,j-1}, d_{i-1,j}, d_{i-1,j-1}\}, \|p_i - q_j\|\}.$$

(Put $d_{a,b} = \infty$ if $a < 0$ or $b < 0$, and let $\min \emptyset := 0$.)
Computing d_{F} (as proposed in [5]) can be done in $O(mn)$ time as well; the algorithm, however, is more involved. We require some basic results that have been developed in the context of computing the Fréchet distance and that have also been used for matching under the group of translations. Certainly the most important concept is the *ε-free space* of two polygonal curves $P \in V^{[0,m]}$ and $Q \in V^{[0,n]}$, which is defined as

$$F_\varepsilon(P, Q) := \{(s, t) \in [0, m] \times [0, n] \mid \|P(s) - Q(t)\| \leq \varepsilon\}.$$

The decision problem of determining whether $d_{\mathrm{F}}(P, Q) \leq \varepsilon$ can be reduced to the problem of finding a curve through $F_\varepsilon(P, Q)$ that is monotonic in both coordinates:

Theorem 7.1.3 ([6]) *Let P and Q be polygonal curves and let $\varepsilon \geq 0$. The following holds:*

(1) *If P and Q are line segments, $F_\varepsilon(P, Q)$ is the intersection of the unit square $[0, 1]^2$ with a (possibly degenerate) ellipse. In particular, $F_\varepsilon(P, Q)$ is convex. For curves P and Q of length > 1, the ε-free space can be described as*

$$\cup_{(i,j) \in [1:m] \times [1:n]} F_\varepsilon(P|_{[i-1,i]}, Q|_{[j-1,j]}).$$

(2) *We have $d_{\mathrm{F}}(P, Q) \leq \varepsilon$ iff there is a curve within $F_\varepsilon(P, Q)$ that starts at $(0, 0)$, ends at (m, n) and is monotonic in both coordinates.*

Analogously, we can define the *discrete ε-free space* of two polygonal curves as

$$\mathbf{F}_\varepsilon(P, Q) = \{(i, j) \in [0 : m] \times [0 : n] \mid \|p_i - q_j\| \leq \varepsilon\}.$$

Using $\mathbf{F}_\varepsilon(P, Q)$ instead of $F_\varepsilon(P, Q)$, one can state properties analogous to Theorem 7.1.3. As a discrete analogue to curves in the free space that are monotonic in both coordinates, we work with *monotonic paths* in $\mathbb{Z} \times \mathbb{Z}$. A monotonic path in $\mathbb{Z} \times \mathbb{Z}$ is a sequence $\langle A_1, \ldots, A_L \rangle$, $A_i = (a_i, b_i) \in \mathbb{Z} \times \mathbb{Z}$, with the property that both sequences $\langle a_1, \ldots, a_L \rangle$ and $\langle b_1, \ldots, b_L \rangle$ are weakly increasing.

Theorem 7.1.4 *Let $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$ be polygonal curves and let $\varepsilon \geq 0$. Then, we have $\mathbf{d}_{\mathrm{F}}(P,Q) \leq \varepsilon$ iff there is a monotonic path of length $K \leq m+n$ within $\mathbf{F}_{\varepsilon}(P,Q)$ that starts at $(0,0)$ and ends at (m,n).*

Sketch of Proof. Let $\mathbf{d}_{\mathrm{F}}(P,Q) = \|P \circ \kappa - Q \circ \lambda\|_{\infty} \leq \varepsilon$, for suitable reparametrizations $(\kappa,\lambda) \in \mathbf{Mon}_{m,n}$. Then the monotonic path is given by all pairs $(\kappa(i),\lambda(i))$, omitting pairs that yield loops (i.e., $(\kappa(i),\lambda(i)) = (\kappa(i-1),\lambda(i-1))$). Conversely, given a monotonic path, we obtain suitable reparametrizations by introducing a loop for every diagonal step of the path (i.e., $(\kappa(i),\lambda(i)) = (\kappa(i-1)+1,\lambda(i-1)+1)$). □

7.2 Weakly Increasing Integer Sequences

One important concept in conjunction with the discrete Fréchet distance are weakly increasing integer sequences. In this section, we provide some basic notation for such sequences as well as some of their properties that are required in the rest of this chapter.
For $\kappa \in \mathrm{Mon}([a:b],[c:d])$, let $\kappa^{-1}[j] := \{i \in [a:b] \mid \kappa(i) = j\}$. Furthermore, let

$$\begin{array}{rcl} \kappa^{-1} : [c:d] & \to & [a:b] \\ j & \mapsto & \kappa^{-1}(j) := \min \kappa^{-1}[j], \end{array} \tag{7.1}$$

and for $X \subseteq [a:b]$ define $\kappa[X] := \{\kappa(x) \mid x \in X\}$. Note that the mapping $j \mapsto \kappa^{-1}(j)$ is weakly increasing but not necessarily surjective.

Remark 7.2.1 *Let $\kappa \in \mathrm{Mon}([a:b],[c:d])$. For all $i \in [a:b]$, we have*

$$i \in \kappa^{-1}[\kappa(i)] \quad \textit{and} \quad \kappa^{-1}(\kappa(i)) \leq i. \tag{7.2}$$

Analogously, we have

$$j \in \kappa[\kappa^{-1}[j]] \quad \textit{and} \quad \kappa(\kappa^{-1}(j)) = j \tag{7.3}$$

for all $j \in [c:d]$.

We usually denote some $\kappa \in \mathrm{Mon}([a:b],[c:d])$ by the sequence $\langle \kappa(a), \ldots, \kappa(b) \rangle$. A basic operation that can be performed on two weakly increasing (not necesarily surjective) integer sequences is *merging* the two integer sequences to one single sequence: given two weakly increasing integer sequences κ and λ, let $\kappa \mathbin{/\!\!\backslash} \lambda$ denote the likewise weakly increasing integer sequence that contains the elements of κ and λ, including multiplicities.
Merging will be particularly interesting for a certain type of integer sequences:

Definition 7.2.2 *Let $k : [a:b] \to [c:d]$ and $\ell : [c:d] \to [a:b]$ be weakly increasing integer sequences. We say that a pair (k,ℓ) is* crossing free *if it satisfies*

$$\begin{array}{lrcl} \text{(CF1)} & j < k(i) & \implies & \ell(j) \leq i \quad \textit{and} \\ \text{(CF2)} & k(i) < j & \implies & i \leq \ell(j) \end{array}$$

for all $i \in [a:b]$ and $j \in [c:d]$.

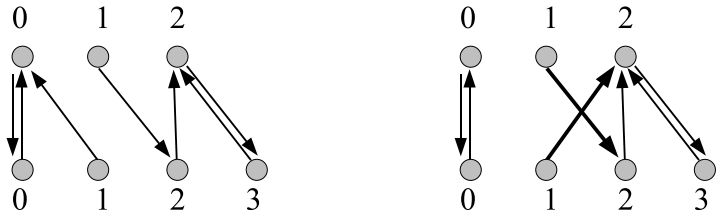


Figure 7.1: *Left:* a pair of weakly increasing, crossing free mappings $k : [0:2] \to [0:3], \ell : [0:3] \to [0:2]$. *Right:* a pair of mappings that is not crossing free. The arrows violating the crossing free condition are marked bold.

For examples of pairs of crossing free integer sequences as well as non-crossing-free integer sequences, see Figure 7.1. In this figure, a directed edge from vertex i to vertex $k(i)$ represents a pair $(i, k(i))$, as well as a pair $(\ell(j), j)$ is represented by an edge from j to $\ell(j)$. In the following, we use such *sequence diagrams* for illustrating pairs of crossing free integer sequences. Sequence diagrams motivate the use of the term crossing free, since two sequences are crossing free if and only if the corresponding sequence diagram does not contain edges that cross each other.

Remark 7.2.3 *The above definition can be stated equivalently by the two conditions*

$$\begin{array}{lrcl} (\mathrm{CF1}') & i < \ell(j) & \Longrightarrow & k(i) \leq j \quad \textit{and} \\ (\mathrm{CF2}') & \ell(j) < i & \Longrightarrow & j \leq k(i), \end{array}$$

since (CF1) $\Leftrightarrow$ (CF1′) *and* (CF2) $\Leftrightarrow$ (CF2′).

Crossing free pairs of sequences are closely related to the discrete Fréchet distance.

Theorem 7.2.4 *Let $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$ denote two polygonal curves. The following two statements are equivalent:*

(1) $\mathbf{d}_{\mathrm{F}}(P, Q) \leq \varepsilon$.

(2) *There is a crossing free pair of weakly increasing mappings $k\colon [1:m] \to [0:n]$ and $\ell\colon [1:n] \to [0:m]$ such that $\|p_i - q_{k(i)}\| \leq \varepsilon$ and $\|p_{\ell(j)} - q_j\| \leq \varepsilon$ for all $i \in [1:m]$ and $j \in [1:n]$.*

The proof of this close relation between crossing free pairs of mappings and $\mathbf{d}_{\mathrm{F}}$ relies on some further properties of crossing free pairs of mappings that we investigate first. The key concept to be studied in conjunction with crossing free integer sequences is the process of merging and the reverse process of *decomposing* sequences:

Definition 7.2.5 *Let $k \in [0:n]^{[1:m]}$ and $\ell \in [0:m]^{[1:n]}$ be crossing free weakly increasing mappings. We say that the pair (k, ℓ) is a* crossing free decomposition of $(\kappa, \lambda) \in \mathbf{Mon}_{m,n}$ *if and only if*

$$\begin{array}{rcccl} \kappa & = & \langle \ell(1), \ldots, \ell(n) \rangle & \mathbin{/\!\!\backslash\!\!\backslash} & \langle 0, \ldots, m \rangle \\ \lambda & = & \langle 0, \ldots, n \rangle & \mathbin{/\!\!\backslash\!\!\backslash} & \langle k(1), \ldots, k(m) \rangle. \end{array} \tag{7.4}$$

Lemma 7.2.6 *The following holds:*

(1) *Every* $(\kappa, \lambda) \in \mathbf{Mon}_{m,n}$ *has a crossing free decomposition* (k, ℓ).

(2) *Let* (k, ℓ) *be a crossing free decomposition of* (κ, λ). *Then, for all* $s \in [1 : m+n]$, *we have* $(\kappa(s), \lambda(s)) \in \{(\ell(j), j), (i, k(i))\}$ *for some* $i \in [1 : m]$ *or* $j \in [1 : n]$.

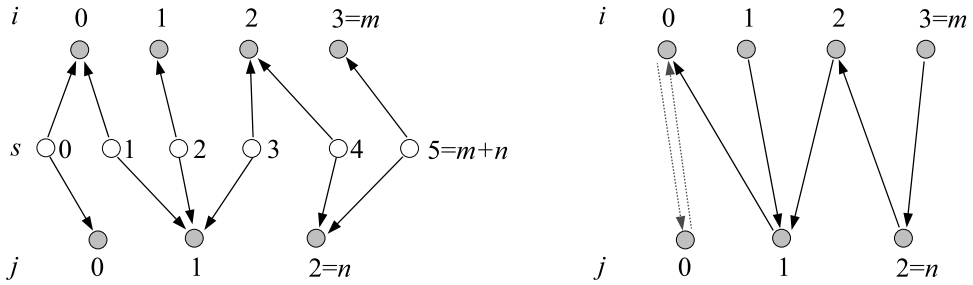


Figure 7.2: Example of a crossing free decomposition: In the left figure, an arrow indicates a pair $(s, \kappa(s))$ or $(s, \lambda(s))$. The right hand figure shows the crossing free decomposition (k, ℓ) of (κ, λ) as constructed in the proof of Lemma 7.2.6. Here, arrows indicate pairs $(i, k(i))$ or $(j, \ell(j))$. Note that the pairs $(0, k(0))$ and $(0, \ell(0))$ do not belong to the crossing free decomposition.

Proof. Let $(\kappa, \lambda) \in \mathbf{Mon}_{m,n}$. We claim that the mappings $k := (\lambda \circ \kappa^{-1})|_{[1:m]}$ and $\ell := (\kappa \circ \lambda^{-1})_{[1:n]}$ define a crossing free decomposition of (κ, λ). To this end, suppose that (k, ℓ) does not comply with (CF1), i.e., there is an $i \in [1 : m]$ and some $j \in [0 : n]$ such that $j < k(i)$ as well as $\ell(j) > i$.
On the other hand, we know that $(i, k(i)) = (\kappa(s), \lambda(s))$ for $s = \kappa^{-1}(i)$ and $(\ell(j), j) = (\kappa(s'), \lambda(s'))$ for $s' = \lambda^{-1}(j)$. This yields $\lambda(s) > \lambda(s')$ as well as $\kappa(s') > \kappa(s)$. Since we have either $s < s'$ or $s' < s$, either the monotonicity of κ or the monotonicity λ is violated, contradicting $(\kappa, \lambda) \in \mathbf{Mon}_{m,n}$.
Proving that (k, ℓ) complies with (CF2) works analogously.

For the proof of (2), we have to study the process of merging in more detail by supplying a procedure that computes $\langle 0, \ldots, m \rangle \between \ell$ and $\langle 0, \ldots, n \rangle \between k$ simultaneously. This merge procedure will help to prove our claim. We introduce some definitions that are useful for this merge procedure. For two integer sequences x and y, we define $x \circ y$ as the concatenation of x and y; moreover, let $x_\perp := x \circ \langle \infty \rangle$. We now consider the following algorithm for simultaneously computing κ and λ:

Input: $m, n \in \mathbb{N}$; crossing free weakly increasing mappings $k\colon [1 : m] \to [0 : n]$ and $\ell\colon [1 : n] \to [0 : m]$.
Output: $\kappa = \langle 0, \ldots, m \rangle \between \ell$ and $\lambda = \langle 0, \ldots, n \rangle \between k$

`Merge`(k, ℓ, m, n)
1 $k := k_\perp; \ell := \ell_\perp$
2 $\kappa(0) := 0; \lambda(0) := 0;$
3 $i := 1;\ j := 1;\ s := 1$

```
4   while (s ≤ m + n)  // i + j = s − 1
5     if (j < k(i) ∨ ℓ(j) < i)   //  (CF1) ∨ (CF2′)
6         κ(s) := ℓ(j); λ(s) := j;
7         j++; s++
8     else if (k(i) < j ∨ i < ℓ(j))   //  (CF1′) ∨ (CF2)
9         κ(s) := i; λ(s) := k(i);
10        i++; s++
11    else   //  j = k(i) ∧ i = ℓ(j)
12        κ(s) := ℓ(j); λ(s) := j;
13        j++; s++
14        κ(s) := i; λ(s) := k(i);
15        i++; s++
16  end
17  return ⟨κ(0), ..., κ(m + n)⟩, ⟨λ(0), ..., λ(m + n)⟩
end.
```

Appending ∞ as a last element to each of the integer sequences causes the algorithm not to read further elements from a sequence whose last element has already been read. By induction on $s = i + j - 1$, we can show that the loop invariants

$$\begin{array}{l} \langle \kappa(0), \ldots, \kappa(s-1)\rangle = \langle \ell(1), \ldots, \ell(j-1)\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle 0, \ldots, i-1\rangle \text{ and} \\ \langle \lambda(0), \ldots, \lambda(s-1)\rangle = \langle 0, \ldots, j-1\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle k(0), \ldots, k(i-1)\rangle \end{array} \tag{7.5}$$

hold w.r.t. Line 4: for $s = 1$, the claim obviously holds true. Now, let $s > 1$. By induction hypothesis, the claim holds before processing any of the Lines 6,9,12 or 14. Using properties (CF1) and (CF2) from the definition of crossing free pairs of sequences and their equivalences (CF1') and (CF2') from Remark 7.2.1, we find that
after processing Line 6, we have

$$\begin{array}{rcl} \langle \kappa(0), \ldots, \kappa(s)\rangle & = & \langle \ell(1), \ldots, \ell(j)\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle 0, \ldots, i-1\rangle \\ \langle \lambda(0), \ldots, \lambda(s)\rangle & = & \langle 0, \ldots, j\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle k(1), \ldots, k(i-1)\rangle, \end{array}$$

after processing Line 9, we have

$$\begin{array}{rcl} \langle \kappa(0), \ldots, \kappa(s)\rangle & = & \langle \ell(1), \ldots, \ell(j-1)\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle 0, \ldots, i\rangle \\ \langle \lambda(0), \ldots, \lambda(s)\rangle & = & \langle 0, \ldots, j-1\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle k(1), \ldots, k(i)\rangle, \end{array}$$

after processing Line 12, we have

$$\begin{array}{rcl} \langle \kappa(0), \ldots, \kappa(s)\rangle & = & \langle \ell(1), \ldots, \ell(j)\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle 0, \ldots, i-1\rangle \\ \langle \lambda(0), \ldots, \lambda(s)\rangle & = & \langle 0, \ldots, j\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle k(1), \ldots, k(i-1)\rangle, \end{array}$$

and after processing Line 14, we have

$$\begin{array}{rcl} \langle \kappa(0), \ldots, \kappa(s)\rangle & = & \langle \ell(1), \ldots, \ell(j-1)\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle 0, \ldots, i\rangle \\ \langle \lambda(0), \ldots, \lambda(s)\rangle & = & \langle 0, \ldots, j-1\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle k(1), \ldots, k(i)\rangle. \end{array}$$

Since in Lines 7, j and s are increased, the loop invariant (7.5) holds after processing Line 7. Analogously, increasing i, j and s in Lines 10,13 and 15 restores the loop invariant (7.5) in all other cases.

This shows that **Merge** computes κ and λ as defined in (7.16) and (7.17), proving the monotonicity of κ and λ as well as the properties $\kappa(0) = 0, \kappa(m+n) = m, \lambda(0) = 0$ and $\lambda(m+n) = n$.
Looking at Lines 6,9,12 and 13, each pair of indices $(\kappa(s), \lambda(s))$ is assigned either a pair $(i, k(i))$ or a pair $(\ell(j), j)$, which proves our claim. □

Proof of Theorem 7.2.4. Since we have $\mathbf{d}_{\mathrm{F}}(P,Q) \leq \varepsilon$ iff there is some $(\kappa,\lambda) \in \mathbf{Mon}_{m,n}$ such that $\|p_{\kappa(s)} - p_{\lambda(s)}\| \leq \varepsilon$ for all $s \in [0:m+n]$, the theorem follows immediately from Lemma 7.2.6. □

7.3 Metric Properties of the Discrete Fréchet Distance

It is well known [34, 31] that the continuous version of the Fréchet distance d_{F} is a pseudo metric, i.e., d_{F} satisfies all properties of a metric except for $d_{\mathrm{F}}(P,Q) = 0 \Rightarrow P = Q$. For the discrete version, we provide a proof here that $\mathbf{d}_{\mathrm{F}}$ is a pseudo metric as well. Proving the triangle inequality for $\mathbf{d}_{\mathrm{F}}$ is more involved than proving the triangle inequality for d_{F}, since the continuous version of the Fréchet distance between any two curves is based on the general reference interval $[0,1]$, whereas there is no such general reference interval in case of $\mathbf{d}_{\mathrm{F}}$.

Theorem 7.3.1 $\mathbf{d}_{\mathrm{F}}$ *is a pseudo metric.*

Proof. Obviously, we have $\mathbf{d}_{\mathrm{F}}(P,P) = 0$ and $\mathbf{d}_F(P,Q) = \mathbf{d}_{\mathrm{F}}(Q,P)$. It remains to prove the triangle inequality.
Let $A = \langle a_0, \ldots, a_\alpha\rangle \in V^{[0:\alpha]}$, $B = \langle b_0, \ldots, b_\beta\rangle \in V^{[0:\beta]}$ and $C = \langle c_0, \ldots, c_\gamma\rangle \in V^{[0:\gamma]}$. We show that $\mathbf{d}_{\mathrm{F}}(A,C) \leq \mathbf{d}_{\mathrm{F}}(A,B) + \mathbf{d}_{\mathrm{F}}(B,C)$. By Theorem 7.2.4, it suffices to construct a pair of crossing free integer sequences (u,v) such that $\|a_i - c_{u(i)}\| \leq \mathbf{d}_{\mathrm{F}}(A,B) + \mathbf{d}_{\mathrm{F}}(B,C)$ for all $i \in [0:\alpha]$ and $\|a_{v(j)} - c_j\| \leq \mathbf{d}_{\mathrm{F}}(A,B) + \mathbf{d}_{\mathrm{F}}(B,C)$ for all $j \in [1:\gamma]$.
The proof proceeds in three steps:

(1) Partition both intervals $[0:\alpha]$ and $[0:\gamma]$ into β many (possibly empty) intervals, so that

$$\begin{array}{rcl} [0:\alpha] & = & \sqcup_{t\in[0:\beta]}[\alpha_t : \alpha_{t+1}) \quad \text{and} \\ [0:\gamma] & = & \sqcup_{t\in[0:\beta]}[\gamma_t : \gamma_{t+1}). \end{array} \tag{7.6}$$

(2) construct u and v blockwise between the intervals $[\alpha_t : \alpha_{t+1})$ and $[\gamma_t : \gamma_{t+1})$ using the partitionings from Step (1).

(3) Show that $\|p_i - q_{u(i)}\| \leq \mathbf{d}_{\mathrm{F}}(A,B) + \mathbf{d}_{\mathrm{F}}(B,C)$ and $\|p_{v(j)} - q_j\| \leq \mathbf{d}_{\mathrm{F}}(A,B) + \mathbf{d}_{\mathrm{F}}(B,C)$.

Step (1): By definition of $\mathbf{d}_{\mathrm{F}}$, there is a pair $(\mu,\nu) \in \mathbf{Mon}_{\alpha,\beta}$ such that

$$\|a_{\mu(s)} - b_{\nu(s)}\| \leq \mathbf{d}_{\mathrm{F}}(A,B) \quad \text{for all } s \in [0:\alpha+\beta]. \tag{7.7}$$

We use this pair (μ,ν) to construct the integers α_t ($t \in [0:\beta]$) as follows:

$$\alpha_t := \min\{r \in [0:\alpha] \mid \exists s \in [0:\alpha+\beta] \colon \mu(s) = r \wedge \nu(s) = t\}. \tag{7.8}$$

Note that $\alpha_t = \min \mu[\nu^{-1}[t]]$. Furthermore, we define $\alpha_{\beta+1} := \alpha + 1$.

The construction of the integers γ_t ($t \in [0:\beta]$) works analogously. By definition of $\mathbf{d}_{\mathrm{F}}$, there is a pair $(\kappa, \lambda) \in \mathbf{Mon}_{\beta,\gamma}$ such that

$$\|b_{\kappa(s)} - c_{\lambda(s)}\| \leq \mathbf{d}_{\mathrm{F}}(B, C) \quad \text{for all } s \in [0:\beta+\gamma]. \tag{7.9}$$

We use this pair (κ, λ) for constructing the integers γ_t ($t \in [0:\beta]$) as follows:

$$\gamma_t := \min\{r \in [0:\gamma] \mid \exists s \in [0:\beta+\gamma]\colon \lambda(s) = r \wedge \kappa(s) = t\}. \tag{7.10}$$

Furthermore, we define $\gamma_{\beta+1} := \gamma + 1$.
Due to the monontonicity of μ and ν, the sequence $\langle \alpha_t \rangle_{t\in[0:\beta+1]}$ is weakly increasing. Analogously, $\langle \gamma_t \rangle_{t\in[0:\gamma+1]}$ is weakly increasing due to the monotonicity of κ and λ.
Defining $A_t := [\alpha_t : \alpha_{t+1})$ and $C_t := [\gamma_t : \gamma_{t+1})$, the unions from Eq. (7.6) can be written as $[0:\alpha] = \sqcup_t A_t$ and $[0:\gamma] = \sqcup_t C_t$. (Note that A_t is empty if $\alpha_t = \alpha_{t+1}$ and B_t is empty if $\gamma_t = \gamma_{t+1}$). However, by construction, $A_\beta \neq \emptyset \neq C_\beta$. As a consequence, each $i \in [0:\alpha]$ is contained in one uniquely defined A_t, and each $j \in [0:\gamma]$ is contained in one uniquely defined C_t.

Step (2): We are now prepared to construct u and v. For each $t \in [0:\beta]$, we assign

$$\begin{array}{lcl} u(i) & := & \gamma_t \quad \text{for } i \in A_t \\ v(j) & := & \max\{\alpha_t, \alpha_{t+1} - 1\} \text{ for } j \in B_t. \end{array} \tag{7.11}$$

Taking the maximum of α_t and $\alpha_{t+1}-1$ instead of simply defining $v(j) := \alpha_{t+1}-1$ is required for the case $\alpha_t = \alpha_{t+1}$. See Figure 7.3 for an illustration of this construction. Obviously, each block constructed this way is crossing free.

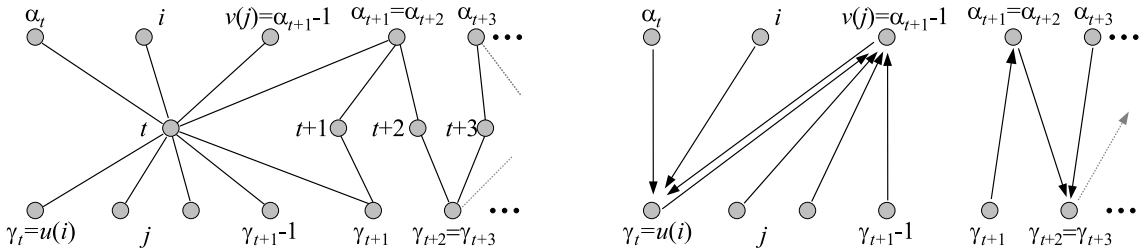


Figure 7.3: *Blockwise construction of $u(i)$ and $v(j)$ for $i \in A_t = [\alpha_t, \alpha_{t+1})$ and $j \in B_t = [\gamma_t, \gamma_{t+1})$.* In the left figure, an undirected edge (i,t) corresponds to a pair $(\mu(s), \nu(s))$ as well as an undirected edge (t,j) corresponds to a pair $(\kappa(s), \lambda(s))$. The figure on the right shows the corresponding pairs $(i, k(i))$ and $(\ell(j), j)$ constructed in the proof of Theorem 7.3.1 indicated by directed edges.

Due to Eq. (7.6), we have now defined $u(i)$ for all $i \in [0:\alpha]$ as well as $v(j)$ for all $j \in [0:\beta]$.

Step (3): In order to prove (3), recall that for each $i \in [0:\alpha]$, there is a uniquely defined $t \in [0:\beta]$ such that $i \in A_t$. Just as well, for each $j \in [0:\gamma]$, there is a uniquely defined t such that $j \in B_t$. Note that $[\alpha_t : \alpha_{t+1}) \subseteq \mu[\nu^{-1}[t]]$. Thus for each $i \in [\alpha_t : \alpha_{t+1})$ there is an $s \in \nu^{-1}[t]$ such that $\mu(s) = i$. On the other hand $\nu(s) = t$. Hence $(i,t) = (\mu(s), \nu(s))$ for some $s \in \nu^{-1}(t) \subseteq [0:\alpha+\beta]$.
Since (by construction) $(t, \gamma_t) = (\kappa(s'), \lambda(s'))$ for some s', we can use Eqs. (7.7) and (7.9) as well as the triangle inequality for the Euclidean distance, which proves the first part of (3).

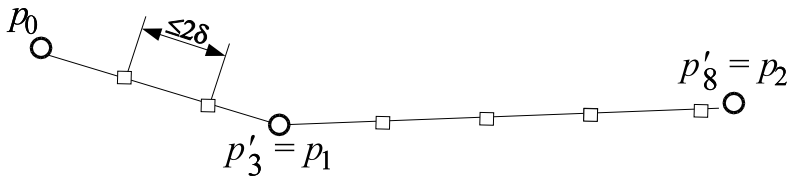


Figure 7.4: Example of a δ-sampled version P' of a polygonal curve P.

Analogously, for each $j \in [\gamma_t, \gamma_{t+1})$, there is an $s \in [0 : \beta + \gamma]$ such that $(t, j) = (\kappa(s), \lambda(s))$. Together with $(\alpha_t, t) = (\mu(s'), \nu(s'))$ for some s', this proves (3) using Eq. (7.9) and the triangle inequality for the Euclidean distance. □

7.4 Bounding the Discrete Fréchet Distance by the Fréchet Distance

We now show that one can approximate the Fréchet distance arbitrarily well by the discrete Fréchet distance by *oversampling* polygonal curves. We say that $P = \langle p_0, \ldots, p_m \rangle$ is *δ-sampled* iff

$$\forall i : \|p_{i-1} - p_i\| \leq 2\delta.$$

For arbitrary $\delta > 0$, we can construct a δ-sampled version P' of P by inserting extra points, as shown in Figure 7.4. Obviously, if P' is an oversampled version of P, we have $d_{\mathrm{F}}(P', P) = 0$.

Using the notion of samplings and oversamplings, we can state an immediate relation between the continuous and the discrete Fréchet distance:

Theorem 7.4.1 *Let $P = \langle p_0, \ldots, p_m \rangle, Q = \langle q_0, \ldots, q_n \rangle$ be δ-sampled polygonal curves. Then,*

$$d_{\mathrm{F}}(P, Q) \quad \leq \quad \mathbf{d}_{\mathrm{F}}(P, Q) \quad \leq \quad d_{\mathrm{F}}(P, Q) + \delta.$$

Note that by oversampling P and Q, we can bring $\mathbf{d}_{\mathrm{F}}$ arbitrarily close to d_{F}. The bounds stated are tight, as the following examples for $V = \mathbb{R}^2$ show: For $P = Q$, we have $d_{\mathrm{F}}(P, Q) = \mathbf{d}_{\mathrm{F}}(P, Q) = 0$, so that the first bound is tight. For the second bound, set $P := \langle (0,0), (0,1) \rangle$ and $Q := \langle (0,0), (0,1), (0,0), (0,1) \rangle$. Both P and Q are $\frac{1}{2}$-sampled, and we have $d_{\mathrm{F}}(P, Q) = \frac{1}{2}$ as well as $\mathbf{d}_{\mathrm{F}}(P, Q) = 1$, so that $\mathbf{d}_{\mathrm{F}}(P, Q) = d_{\mathrm{F}}(P, Q) + \frac{1}{2}$. See figure 7.5 for an interpretation of Theorem 7.4.1 in terms of the discrete and continuous free space of the two curves.
We prepare for the proof of Theorem 7.4.1. The proof demonstrated in the following is based on the results on crossing free mappings, mainly on Lemma 7.2.6. Another way of proving this bound is presented in [53].

Lemma 7.4.2 *Let $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$ be polygonal curves, P δ-sampled, and let $d_{\mathrm{F}}(P, Q) \leq \varepsilon$. Then for every vertex q_j of Q there is a vertex p_k of P with $\|p_k - q_j\| \leq \varepsilon + \delta$.*

Proof. As $d_{\mathrm{F}}(P, Q) \leq \varepsilon$, there is some real $\mu \in [0, m]$ satisfying $\|P(\mu) - q_j\| \leq \varepsilon$. Now P is δ-sampled. Thus for a suitable $k \in \{\lfloor \mu \rfloor, \lceil \mu \rceil\}$ we get $\|p_k - P(\mu)\| \leq \delta$. Hence $\|p_k - q_j\| \leq \|p_k - P(\mu)\| + \|P(\mu) - q_j\| \leq \varepsilon + \delta$. See also Figure 7.6 for an illustration. □

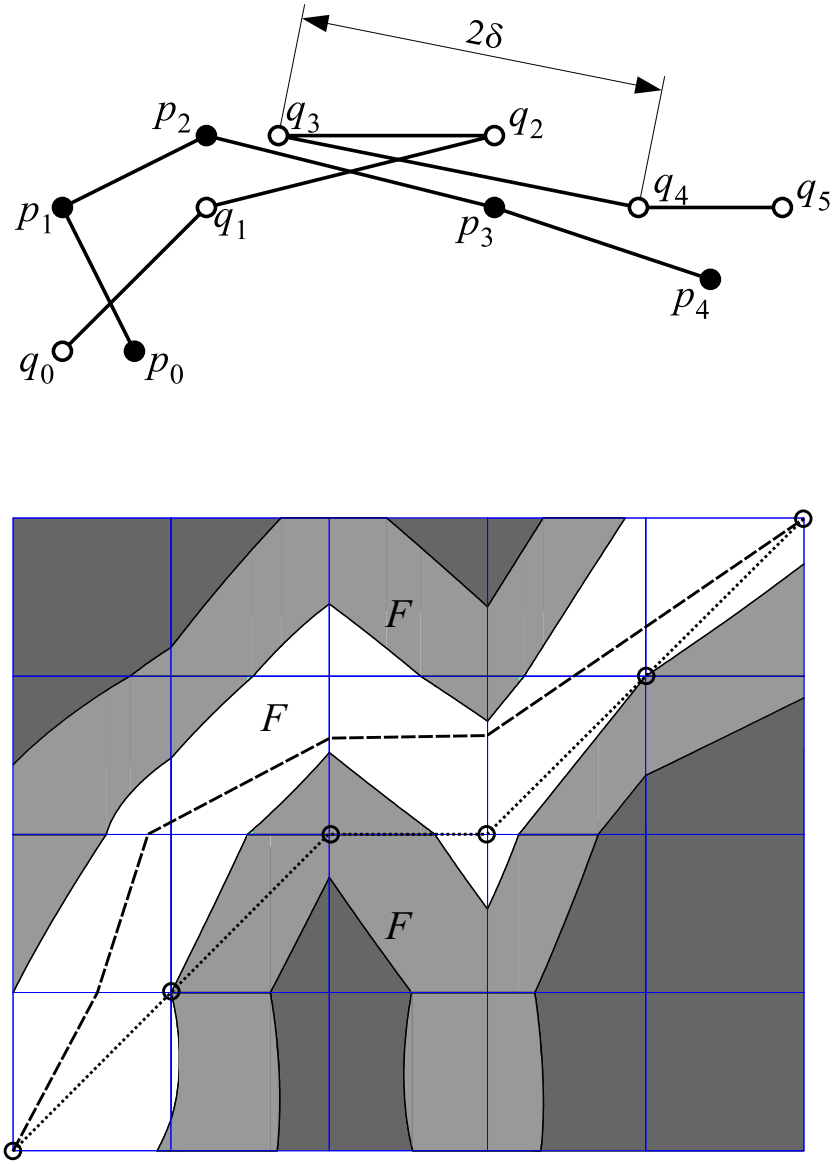


Figure 7.5: *Interpretation of Theorem 7.4.1 in terms of free space diagrams:* The lower part of the figure shows the free spaces $F_\varepsilon(P,Q)$ and $F_{\varepsilon+\delta}(P,Q)$ for the two δ-sampled curves P and Q shown above. Recall that for the discrete free spaces, we have $\mathbf{F}_\varepsilon(P,Q) = F_\varepsilon(P,Q) \cap [1:m] \times [1:n]$ and and $F_{\varepsilon+\delta} \cap [1:m] \times [1:n]$. As indicated by the dashed curve, there is a curve that is monotonic in both coordinates through $F_\varepsilon(P,Q)$ starting in $(0,0)$ and ending in (m,n), in other words, we have $d_{\mathrm{F}}(P,Q) \leq \varepsilon$. According to Theorem 7.4.1, we have $\mathbf{d}_{\mathrm{F}}(P,Q) \leq \varepsilon + \delta$. In terms of the free space, this can be seen from the fact that there is a monotonic path through $\mathbf{F}_{\varepsilon+\delta}(P,Q)$, as indicated by the dotted curve. Note that one cannot find a montonic path through $\mathbf{F}_\varepsilon(P,Q)$, i.e., $\mathbf{d}_{\mathrm{F}}(P,Q) > \varepsilon$.

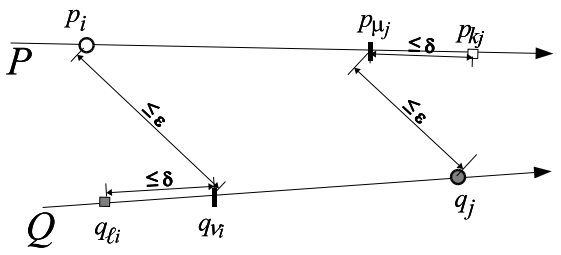


Figure 7.6: Sketch for the proof of Lemma 7.4.2

This latter result allows us to construct an integer sequence $k = \langle k(0), \ldots, k(n)\rangle$ so that every q_j is contained in the $(\varepsilon+\delta)$ neighborhood of $p_{k(j)}$. Now, in the context of the Fréchet-distance, the order of points on curves is very important. Thus, it is our next concern to show that we can guarantee the integer sequence k to be weakly increasing.
The following monotonicity properties are closely related to the fact that for every $\alpha \in \mathrm{Mon}([0,1],[0,m])$ both mappings $I \mapsto \alpha[I]$ and $J \mapsto \alpha^{-1}[J]$, where I and J are closed subintervals of $[0,1]$ and $[0,m]$, respectively, are order-preserving morphisms when partially ordering compact intervals by $I \sqsubseteq I' :\Longleftrightarrow \max I \leq \min I'$.
Another order-preserving map is the rounding operator, which assigns the nearest integer to every $\mu \in \mathbb{R}$:

$$\lfloor\mu\rceil := \begin{cases} \lfloor\mu\rfloor & \text{if } 0 \leq \mu - \lfloor\mu\rfloor < \frac{1}{2} \\ \lceil\mu\rceil & \text{if } 0 \leq \lceil\mu\rceil - \mu \leq \frac{1}{2}. \end{cases}$$

Observe that for a δ-sampled curve,

$$\|P(\mu) - p_{\lfloor\mu\rceil}\| \leq \delta. \tag{7.12}$$

Remark 7.4.3 *If $i \in \mathbb{Z}$ and $\mu \in \mathbb{R}$ then*

(1) $i < \lfloor\mu\rceil \Rightarrow i < \mu$ and

(2) $\lfloor\mu\rceil < i \Rightarrow \mu < i$.

Lemma 7.4.4 *Let P, Q be two δ-sampled polygonal curves with $d_{\mathrm{F}}(P,Q) \leq \varepsilon$. Then there are weakly increasing integer sequences*

$$k = \langle k(1), \ldots, k(n) = m\rangle \quad \text{and}$$
$$\ell = \langle \ell(1), \ldots, \ell(m) = n\rangle$$

such that for all $i \in [1:m]$ and all $j \in [1:n]$ the following holds:

$$\|p_{k(j)} - q_j\| \leq \varepsilon + \delta \quad \text{and} \quad \|p_i - q_{\ell(i)}\| \leq \varepsilon + \delta.$$

Proof. Obviously, we have $\|p_0 - q_0\| \leq \varepsilon$. As $d_{\mathrm{F}}(P,Q) \leq \varepsilon$, there are $\alpha \in \mathrm{Mon}([0,1],[0,m])$ and $\beta \in \mathrm{Mon}([0,1],[0,n])$ with $\|P(\alpha(t)) - Q(\beta(t))\| \leq \varepsilon$ for $t \in [0,1]$. This implies for arbitrary $s \in [0,n]$ and $t \in \beta^{-1}[s]$ that $\|P(\alpha(t)) - Q(s)\| \leq \varepsilon$.
Now define

$$\mu_j := \max \alpha[\beta^{-1}[j]] \in [0,m] \quad \text{and} \quad k(j) := \lfloor\mu_j\rceil \tag{7.13}$$

for $j \in [1:n]$. Note that $k(n) = m$. We claim that $\|p_{k(j)} - q_j\| \leq \varepsilon + \delta$ for all $j \in [1:n]$. We get, by Eq. (7.12), $\|p_{k(j)} - P(\mu_j)\| \leq \delta$; furthermore, the definition of μ_j yields $\|P(\mu_j) - q_j\| \leq \varepsilon$. By the triangle equality our claim follows.
As both α and β^{-1} induce order-preserving morphisms we know that $\alpha[\beta^{-1}[j]] \sqsubseteq \alpha[\beta^{-1}[j+1]]$; in particular, $\mu_j \leq \mu_{j+1}$. This implies the monotonicity of the sequence $\langle k(1), \ldots, k(n)\rangle$, since $\lfloor . \rceil$ also is an order-preserving map.
Exchanging the roles of P and Q, the proof for the existence of a weakly increasing integer sequence $\langle \ell(1), \ldots, \ell(m)\rangle$ is exactly the same; in place of μ_j, we put

$$\nu_i := \max \beta[\alpha^{-1}[i]] \in [0:n] \quad \text{and} \quad \ell(i) := \lfloor \nu_i \rceil. \tag{7.14}$$

□

Note that

$$j \in \beta[\alpha^{-1}[\mu_j]] \quad \text{and} \quad i \in \alpha[\beta^{-1}[\nu_i]]. \tag{7.15}$$

The two weakly increasing sequences k and ℓ will help to closer relate the curves P and Q. However, we have to solve two problems. First of all, k and ℓ typically have different lengths. Secondly, both sequences may have *gaps*, i.e., $k(j) - k(j-1) > 1$ and $\ell(i) - \ell(i-1) > 1$ for some j, i. Instead of k and ℓ we will work with the sequences

$$\kappa := \langle \ell(1), \ldots, \ell(n)\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle 0, \ldots, m\rangle \tag{7.16}$$

$$\lambda := \langle 0, \ldots, n\rangle \mathbin{/\!\!\backslash\!\!\backslash} \langle k(1), \ldots, k(m)\rangle, \tag{7.17}$$

Remark 7.4.5 *From the facts that the integer sequences $\kappa \in \mathbf{Mon}([0:m+n],[0:m])$ and $\lambda \in \mathbf{Mon}([0:m+n],[0:n])$ are surjective and weakly increasing, we may conclude that $\kappa(s+1) - \kappa(s) \in \{0,1\}$ and $\lambda(s+1) - \lambda(s) \in \{0,1\}$ for all $s \in [0:n+m-1]$.*

As for the proof of Theorem 7.3.1, the fact that k and ℓ are crossing free helps to prove Theorem 7.4.1.

Lemma 7.4.6 *Let P and Q be δ-sampled polygonal curves satisfying $d_{\mathrm{F}}(P,Q) \leq \varepsilon$. Using the notation introduced in Equations (7.13) to (7.14), the sequences k and ℓ are crossing free.*

Proof. In order to prove that k and ℓ satisfy condition (i) of Definition 7.2.2, let $i < k(j)$. Then the above remarks and the minimality of $k(j)$ imply that $\mu_j > i$. As α and β are weakly increasing, we get (using (7.15)) $\nu_i = \max \beta[\alpha^{-1}[i]] \leq \min \beta[\alpha^{-1}[\mu_j]] \leq j$. If ν_i is an integer, $\ell(i) = \nu_i \leq j$, and we are done. Otherwise, $\nu_i < j$. But then, by Remark 7.4.3, $\ell(i) = \lfloor \nu_i \rceil \leq j$. Altogether, this proves (i).
The proof that k and ℓ satisfy condition (ii) of Definition 7.2.2 is very similar: Let $k(j) < i$. Then $\mu_j < i$ and hence, by (7.15), $j \in \beta[\alpha^{-1}[\mu_j]] \sqsubseteq \beta[\alpha^{-1}[i]] \ni \nu_i$, thus $j \leq \nu_i$. If ν_i is an integer, then $j \leq \nu_i = \ell(i)$, otherwise $j \leq \lfloor \nu_i \rceil \leq \ell(i)$, which proves (ii). □

Proof of Theorem 7.4.1. We start with the proof of the right hand inequality. Let $\varepsilon := d_{\mathrm{F}}(P,Q)$. Thus, by Definition 7.1.1, there are reparametrizations α and β such that $\|P(\alpha(t)) - Q(\beta(t))\| \leq \varepsilon$ for all $t \in [0,1]$. These reparametrizations allow us to construct integer sequences k and ℓ for P and Q according to Lemma 7.4.4 so that we can define κ and λ as in Eqs. (7.16) and (7.17).

From the fact that k and ℓ are crossing free and by Lemma 7.2.6, we know that every pair $(\kappa(s), \lambda(s))$ equals $(k(j), j)$ or $(i, \ell(i))$ for some j or i. According to Lemma 7.4.4, we know that the inequalities $\|p_i - q_{\ell(i)}\| \leq \varepsilon + \delta$ and $\|p_{k(j)} - q_j\| \leq \varepsilon + \delta$ hold for $i \in [0:m]$ and $j \in [0:n]$, which proves $\mathbf{d}_{\mathrm{F}}(P,Q) \leq \varepsilon + \delta$. Altogether, we have shown that $d_{\mathrm{F}}(P,Q) = \varepsilon$ implies $\mathbf{d}_{\mathrm{F}}(P,Q) \leq \varepsilon + \delta$.

For the proof of the left hand inequality, let $(\kappa, \lambda) \in \mathbf{Mon}_{m,n}$ be optimal in the sense that $\mathbf{d}_{\mathrm{F}}(P,Q) = \|P \circ \kappa - Q \circ \lambda\|_\infty$. By affine interpolation, one obtains $(\alpha, \beta) \in \mathrm{Mon}_{m,n}$ with $\alpha(\frac{i}{m+n}) = \kappa_i$ and $\beta(\frac{i}{m+n}) = \lambda_i$. Then,

$$\begin{aligned} d_{\mathrm{F}}(P,Q) &= \min_{\alpha',\beta'} \max_{t\in[0,1]} \|P(\alpha'(t)) - Q(\beta'(t))\| \\ &\leq \max_{t\in[0,1]} \|P(\alpha(t)) - Q(\beta(t))\| \\ &= \max_{s\in[1:m+n]} \|P(\kappa(s)) - Q(\lambda(s))\| = \mathbf{d}_{\mathrm{F}}(P,Q), \end{aligned}$$

where the last but one equality follows from the fact that for line segments $L = \langle L_0, L_1\rangle$ and $L' = \langle L'_0, L'_1\rangle$, $d_{\mathrm{F}}(L,L') = \max\{\|L_0 - L'_0\|, \|L_1 - L'_1\|\}$. This proves the left hand inequality. □

7.5 Matching with Respect to the (discrete) Fréchet Distance

The discrete version of the Fréchet distance is defined by an assignment between the vertices of two polygonal curves. As one can easily see, the discrete Fréchet distance is a relational distance measure. Hence, we can apply the tools developed in Chapter 5. Alt, Knauer and Wenk [9] use the fact that the starting point of a polygonal curve is a $(\mathbb{R}^k, T(k), d_{\mathrm{F}}, 2)$-reference point, yielding a trivial matching algorithm with respect to d_{F} under translations in the plane as follows: Given $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$, define $t := p_0 - q_0$ and decide whether $d_{\mathrm{F}}(P, tQ) \leq \varepsilon$. This can be put in even more general terms:

Lemma 7.5.1 *Let $V = \mathbb{R}^k$ and $P = \langle p_0, \ldots, p_m\rangle \in V^{[0:m]}$. Then, the mapping*

$$r\colon P \mapsto p_0$$

is a $(V, \mathrm{AGL}(k), \mathbf{d}_{\mathrm{F}}, 2)$-reference point.

The proof is trivial and hence omitted. This result finally allows us to state the time bounds shown in Table 7.1.
Theorem 7.4.1 allows us to set up a relation between $G(P,Q,\varepsilon,\mathbf{d}_{\mathrm{F}})$-matches and $G(P,Q,\varepsilon,d_{\mathrm{F}})$-matches, as stated in Corollary 7.5.2. Hence, the time bounds stated in Table 7.1 can also be seen as time bounds for approximately matching with respect to d_{F}. The quality of approximation, however, depends on the sampling rate and hence on the Euclidean length of the two curves.

Corollary 7.5.2 *Let P and Q be δ-sampled polygonal curves in $V = \mathbb{R}^k$. Then, for all transformation groups $G \leq \mathrm{AGL}(k)$ and $\varepsilon > 0$,*

$$G(P,Q,\varepsilon,d_{\mathrm{F}}) \subseteq G(P,Q,\varepsilon+\delta,\mathbf{d}_{\mathrm{F}}) \subseteq G(P,Q,\varepsilon+\delta,d_{\mathrm{F}}).$$

Transformation group	time	c	Algorithm
$T(k)$	$O(mn)$	2	5.3.5
$\mathrm{SO}(2), \mathrm{SC}(2)$	$O((mn)^2)$	1	5.2.4
$\mathrm{SE}(2), \mathrm{HT}(2)$	$O((mn)^4)$	2	5.3.5
$\mathrm{SM}(2)$	$O((mn)^5)$	2	5.3.5
$\mathrm{SE}(3)$	$O((mn)^6)$	2	5.3.5

Table 7.1: Time complexities for c-approximate solutions for deciding the emptiness of $G(P, Q, \varepsilon, \mathbf{d}_{\mathrm{F}})$, where $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$.

Proof. Let $g \in G(P, Q, \varepsilon, d_{\mathrm{F}})$. Then, $d_{\mathrm{F}}(P, gQ) \leq \varepsilon$. Thus, by Theorem 7.4.1, $\mathbf{d}_{\mathrm{F}}(P, gQ) \leq \varepsilon + \delta$, i.e., $g \in G(P, Q, \varepsilon + \delta, \mathbf{d}_{\mathrm{F}})$.
The second inclusion follows from $d_{\mathrm{F}} \leq \mathbf{d}_{\mathrm{F}}$. □
Put in simple terms, this result states that matching with respect to the *discrete* Fréchet distance yields an approximate solution for the problem of matching with respect to the *continuous* Fréchet distance; the quality of approximation can be improved arbitrarily by oversampling the curves to be matched. Thus, we focus on algorithms for matching with respect to the discrete version.

7.5.1 Using Ordered Cell Enumeration

In Chapter 5, it has been shown that one can obtain faster algorithms for matching with respect to the Hausdorff and the bottleneck distance when using *ordered cell enumeration*. There is no immediate way of making use of ordered cell enumerations when matching with respect to $\mathbf{d}_{\mathrm{F}}$. In fact, $\mathbf{d}_{\mathrm{F}}$ seems to be more resistant against setting up a dynamic data structure (in the sense of Definition 5.2.5) than the Hausdorff and the bottleneck distance. However, we can at least improve practical running times by using the undirected Hausdorff distance as a lower bound for the discrete Fréchet distance. As can be seen easily, we have

$$\mathbf{d}_{\mathrm{H}}(P, Q) \leq \mathbf{d}_{\mathrm{F}}(P, Q).$$

We can state this bound equivalently as an implication

$$\mathbf{d}_{\mathrm{H}}(P, Q) > \varepsilon \quad \Longrightarrow \quad \mathbf{d}_{\mathrm{F}}(P, Q) > \varepsilon. \tag{7.18}$$

Now, we apply an ordered cell enumeration to solve the decision problem whether the set of matches $G(P, Q, \varepsilon, \mathbf{d}_{\mathrm{F}})$ is non-empty as follows: during an ordered cell enumeration, we use a dynamic data structure $\mathbf{S}$ for $\mathbf{d}_{\mathrm{H}}$. Whenever we encounter a cell representative g such that $\mathbf{d}_{\mathrm{H}}(P, gQ) > \varepsilon$, we do not need to compute $\mathbf{d}_{\mathrm{F}}(P, Q)$ due to Eq. 7.18 — we only compute $\mathbf{d}_{\mathrm{F}}(P, gQ)$ if $\mathbf{d}_{\mathrm{H}}(P, gQ) \leq \varepsilon$. Although this does not lead to asymptotically faster algorithms, the algorithm runs significantly faster in practice. For experimental results, we refer to Chapter 8.

7.5.2 Matching with Respect to $\mathbf{d}_{\mathrm{W}}$

In Section 5.6, an algorithm for matching with respect to the mean-value Hausdorff distance and the mean-square Hausdorff distance have been introduced. The distance measures $\mathbf{D}_{\mathrm{H}}$

and $\mathbf{D}_{\mathrm{H}}^2$ evolve from the directed Hausdorff distance by exchanging the "inner" norm from an ℓ_∞-norm to an ℓ_1- or ℓ_2-norm, respectively. We have the same situation in case of the discrete Fréchet distance and the dynamic time warping distance — the inner norm is changed from an ℓ_∞-norm to an ℓ_2-norm. In fact, this allows us to carry the ideas underlying the algorithms from Section 5.6 to design an algorithm that solves the problem of matching with respect to $\mathbf{d}_{\mathrm{W}}$.
To be more presise, the algorithm works as follows: Instead of considering the family of all (G, ε)-transporters from some q_j to some p_i, we consider the family of all $(G, \ell\varepsilon^2)$-transporters from q_j to p_i, for all $i \in [0:m]$, $j \in [0:n]$ and $\ell \in [1:m+n]$.
Hence, we need to deal with the arrangement defined by the family of transporters

$$\langle \tau_{p_i,q_j}^{G,\ell\varepsilon^2} \rangle_{i\in[0:m];j\in[0:n];\ell\in[1:m+n]}, \tag{7.19}$$

which is not defined by mn many transporter sets (as for matching with respect to $\mathbf{d}_{\mathrm{F}}$), but by $mn(m+n)$ many. The matching algorithm then proceeds in the same way as Algorithm 5.2.4 — for each representative g from the suptransversal of the arrangement, we compute $\mathbf{d}_{\mathrm{W}}(P, gQ)$. If for some representative g, $\mathbf{d}_{\mathrm{W}}(P, gQ) \leq \varepsilon$, g is returned as an output. If for all representatives g, $\mathbf{d}_{\mathrm{W}}(P, gQ) > \varepsilon$, *false* is returned as an output. Proving that the output of the algorithm is

$$\begin{cases} g \in G(P, Q, 2\varepsilon, \mathbf{d}_{\mathrm{W}}) & \text{if } G(P, Q, \varepsilon, \mathbf{d}_{\mathrm{W}}) \neq \emptyset, \\ \mathit{false} & \text{if } G(P, Q, 2\varepsilon, \mathbf{d}_{\mathrm{W}}) = \emptyset, \\ \mathit{false} \text{ or } g \in G(P, Q, \varepsilon, \mathbf{d}_{\mathrm{W}}) & \text{otherwise.} \end{cases} \tag{7.20}$$

works in the same way as the proof of the analogous statement for $\mathbf{D}_{\mathrm{H}}$ from Section 5.6. It should be noted that there is no obvious way to apply the technique of transporter projection for matching with respect to $\mathbf{d}_{\mathrm{W}}$. Hence, the algorithm described in this section can be considered practical only under the groups SC(2) and SO(2).

7.6 Matching Subcurves

We now turn to the *partial Fréchet distance* $\vec{\mathbf{d}}_{\mathrm{F}}$ for measuring the resemblance of $Q \in V^{[0:m]}$ as a subcurve of $P \in V^{[0:n]}$ and the discrete Fréchet distance for closed polygonal curves, $\mathbf{d}_{\mathrm{F}}^\circ$. As for the discrete Fréchet distance, we first show how to compute $\vec{\mathbf{d}}_{\mathrm{F}}$ as well as $\mathbf{d}_{\mathrm{F}}^\circ$, and then propose algorithms for matching with respect to these distance measures.
To determine whether $Q \in V^{[0:n]}$ is a subcurve of $P \in V^{[0:m]}$, we define the *partial Fréchet distance* as

$$\vec{\mathbf{d}}_{\mathrm{F}}(P, Q) := \min_{[a:b]\subseteq[0:m]} \mathbf{d}_{\mathrm{F}}(P|_{[a:b]}, Q).$$

In order to adapt the discrete Fréchet distance to closed curves, we view P as cyclically continued, i.e., for $i > m$ we let $P(i) := P(i \bmod (m+1))$. In analogy to the continuous Fréchet distance for closed curves in [6], we define

$$\mathbf{d}_{\mathrm{F}}^\circ(P, Q) := \min_{a\in[0:m]} \mathbf{d}_{\mathrm{F}}(P|_{[a:a+m],}, Q).$$

We now propose algorithms for deciding $\vec{\mathbf{d}}_{\mathrm{F}}(P, Q) \leq \varepsilon$ and $\mathbf{d}_{\mathrm{F}}^\circ(P, Q) \leq \varepsilon$. In terms of the discrete free space $\mathcal{F}_\varepsilon(P, Q)$, we need to determine whether there is a monotonic path from

$(a,0)$ to (b,n) for some $[a:b] \subseteq [0:m]$ (in case of $\vec{\mathbf{d}}_{\mathrm{F}}$) or whether there is a monotonic path from $(a,0)$ to $(a+m,n)$ for some a (in case of $\mathbf{d}_{\mathrm{F}}^{\circ}$). This idea is crucial for the decision algorithms we propose.

Theorem 7.6.1 *Let $P \in V^{[0:m]}$,$Q \in V^{[0:n]}$ and $\varepsilon \geq 0$. Then, $\vec{\mathbf{d}}_{\mathrm{F}}(P,Q) \leq \varepsilon$ as well as $\mathbf{d}_{\mathrm{F}}^{\circ}(P,Q) \leq \varepsilon$ can be decided in $O(mn)$ time.*

Proof. We start with a decision algorithm for $\vec{\mathbf{d}}_{\mathrm{F}}(P,Q) \leq \varepsilon$. Consider the following algorithm: for $i \in [0:2m]$ and $j \in [0:n]$, define $r_{i,j} := \max\{a \in [0:i] \mid \mathbf{d}_{\mathrm{F}}(P|_{[a:i]}, Q|_{[0:j]}) \leq \varepsilon\}$ (and $r_{i,j} := -\infty$ if no such a exists). Using dynamic programming as in the algorithm from Section 7.1 for computing $\mathbf{d}_{\mathrm{F}}$, we can compute in $O(mn)$ time the values $r_{i,n}$ for each $i \in [0:2m]$. We have $\vec{\mathbf{d}}_{\mathrm{F}}(P,Q) \leq \varepsilon$ if and only if $r_{i,n} > -\infty$, for some $i \in [0:2m]$.
Deciding $\mathbf{d}_{\mathrm{F}}^{\circ}(P,Q) \leq \varepsilon$ works similar as deciding $\vec{\mathbf{d}}_{\mathrm{F}}(P,Q) \leq \varepsilon$; in addition to $r_{i,j}$, we compute $R_{i,j} := \max\{b \in [i:2m] \mid \mathbf{d}_{\mathrm{F}}(P|_{[i:b]}, Q|_{[j:n]}) \leq \varepsilon\}$ for all $i \in [0:2m]$ and $j \in [0:n]$, which can also be done in $O(mn)$ time using dynamic programming as follows: We start with computing $R_{m,n}$. Then, the algorithm continues as the algorithm for computing $\mathbf{d}_{\mathrm{F}}(P,Q)$, except for the `for`-loops: these run from m down to 0 rather than running from 0 to m and, analogously, from n down to 0 rather than from 0 to n.
We claim that $\mathbf{d}_{\mathrm{F}}^{\circ}(P,Q) \leq \varepsilon$ if and only if for some i, $r_{i+m,n} \geq i$ and $R_{i,0} \geq i+m$. The necessity of this condition follows immediately from the definitions of $\mathbf{d}_{\mathrm{F}}^{\circ}(P,Q)$, $r_{i,j}$ and $R_{i,j}$. For the proof that the condition is sufficient, we follow the construction shown in Figure 7.7: By definition of $\mathbf{d}_{\mathrm{F}}^{\circ}$, it suffices to regard the free space restricted to $[0:2m] \times [0:n]$ instead of the complete free space in $\mathbb{Z} \times \mathbb{Z}$. We have two paths contained in $\mathbf{F}_{\varepsilon}(P,Q)$, one from $(i+c,0)$ to $(i+m,n)$ for some $c \geq 0$ (since $r_{i+m,n} \geq i$) and one from $(i,0)$ to $(i+m+e,n)$ for some $e \geq 0$ (since $R_{i,0} \geq i+m$). As demonstrated in Figure 7.7, these two paths intersect in some point (i',j'). Hence we can construct a path from $(i,0)$ via (i',j') to $(i+m,n)$ that is completely contained in $\mathbf{F}_{\varepsilon}(P,Q)$, which proves the claim. □
The stated upper bounds of $O(mn)$ for deciding $\vec{\mathbf{d}}_{\mathrm{F}}(P,Q) \leq \varepsilon$ and $\mathbf{d}_{\mathrm{F}}^{\circ}(P,Q) \leq \varepsilon$ are slightly smaller than the upper bounds of $O(mn\log(mn))$ from [6] for deciding whether the continuous Fréchet distance for closed curves or partial correspondences is at most ε.

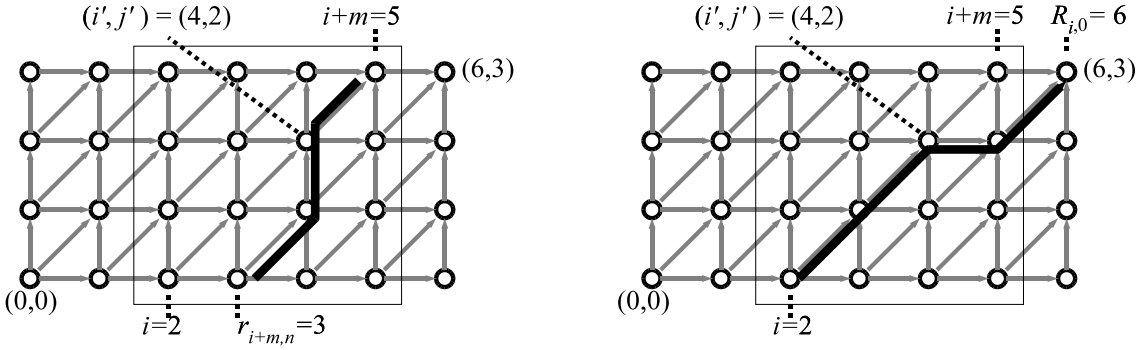


Figure 7.7: Deciding $\mathbf{d}_{\mathrm{F}}^{\circ}(P,Q) \leq \varepsilon$ can be done using $r_{i+m,n}$ and $R_{i,0}$ for $i \in [0:m]$. In the given example, we have $m = n = 3$ and $r_{i+m,n} \geq i$ (left) as well as $R_{i,0} \geq i+m$ (right) for $i = 2$.

Next, we consider matching with respect to $\vec{\mathbf{d}}_{\mathrm{F}}$ and $\mathbf{d}_{\mathrm{F}}^{\circ}$. For matching under the groups SO(2) and SC(2), we can use the idea of enumerating (P,Q,G,ε)-cells as in the first section. Since

both $G(P, Q, \varepsilon, \vec{\mathbf{d}}_{\mathrm{F}})$ and $G(P, Q, \varepsilon, \mathbf{d}_{\mathrm{F}}^{\circ})$ are unions of (P, Q, G, ε)-cells, we can apply Algorithm 5.2.4. Instead of testing $\mathbf{d}_{\mathrm{F}}(P, gQ) \leq \varepsilon$ for each cell, we test $\vec{\mathbf{d}}_{\mathrm{F}}(P, gQ) \leq \varepsilon$ or $\mathbf{d}_{\mathrm{F}}^{\circ}(P, gQ) \leq \varepsilon$, respectively. Hence, we obtain exactly the same time bounds as for matching with respect to $\mathbf{d}_{\mathrm{F}}$.
We now apply the technique of projecting transporters for matching approximately with respect to $\vec{\mathbf{d}}_{\mathrm{F}}$ and $\mathbf{d}_{\mathrm{F}}^{\circ}$. Let $G = T(k) \rtimes H$, $H \leq \mathrm{GL}(k)$, be a transformation group. Our goal is to decide approximately whether $G(P, Q, \varepsilon, \mathbf{d}) \neq \emptyset$, where $\mathbf{d} \in \{\vec{\mathbf{d}}_{\mathrm{F}}, \mathbf{d}_{\mathrm{F}}^{\circ}\}$. To this end, note that both $\vec{\mathbf{d}}_{\mathrm{F}}$ and $\mathbf{d}_{\mathrm{F}}^{\circ}$ are *right-complete* distance measures. Hence, we can apply Algorithm 5.4.2 in order to obtain the time bounds summarized in Table 7.2 for solving the matching tasks approximately.

G	Time	c	Algorithm
$T(k)$	$O(n)$	2	5.4.2
$\mathrm{SO}(2), \mathrm{SC}(2)$	$O(m^2n^2)$	1	5.2.4
$\mathrm{SE}(2), \mathrm{HT}(2)$	$O(m^2n^3)$	2	5.4.2
$\mathrm{SM}(2)$	$O(m^4n^5)$	2	5.4.2
$\mathrm{SE}(3)$	$O(m^5n^6)$	2	5.4.2

Table 7.2: Time bounds for c-approximate solutions using Algorithm 5.4.2 for the problem of matching with respect to $\vec{\mathbf{d}}_{\mathrm{F}}$ and $\mathbf{d}_{\mathrm{F}}^{\circ}$.

Finally, we propose a method for finding common subcurves of P and Q. We restrict our considerations to curves that are not cyclically continued and define

$$\mathrm{LCSC}(P, Q, \varepsilon) := \max\{b \in [0:m] \mid \exists\, a, c, d \in \mathbb{N}\colon \quad a + b \leq m, c + d \leq n, \\ \mathbf{d}_{\mathrm{F}}(P|_{[a:a+b]}, Q|_{[c:c+d]}) \leq \varepsilon\}$$

as the *length of the longest common subcurve of P and Q*. Stated as a matching problem, we want to find $\mathrm{LCSC}(P, Q, G, \varepsilon) := \max_{g \in G} \mathrm{LCSC}(P, gQ, \varepsilon)$. $\mathrm{LCSC}(P, Q, \varepsilon)$ can be computed in $O(mn)$ time: we define $L_{i,j} := \max\{b \in [0:m] \mid \exists\, c \in \mathbb{N}\colon \mathbf{d}_{\mathrm{F}}(P|_{[i-b:i]}, Q|_{[c:j]}) \leq \varepsilon\}$ if such b exists and $L_{i,j} := -\infty$ otherwise. Using dynamic programming, we can compute $L_{i,j}$ for all $(i, j) \in [0:m] \times [0:n]$ in $O(mn)$ time. Determining the maximum $L_{i,j}$ yields $\mathrm{LCSC}(P, Q, \varepsilon)$. The longest common subcurve is nothing but a special case of a largest common point set with respect to a relational distance function. $\mathrm{LCSC}(P, Q, \varepsilon)$ defines a suitable weight function W_{F} (that can be computed in $O(mn)$ time) such that computing $\mathrm{LCP}(P, Q, W_{\mathrm{F}}, G, \varepsilon)$ using Algorithm 5.5.2 yields $\mathrm{LCSC}(P, Q, G, \varepsilon)$ under any linear algebraic group G.
For $G = \mathrm{SC}(2)$ and $G = \mathrm{SO}(2)$, we obtain a running time of $O(m^2n^2)$. For finding longest common subcurves under the groups $\mathrm{SE}(2)$ or $\mathrm{HT}(2)$, the total time complexity obtained is $O(m^3n^3)$.

Chapter 8

Implementations and Practical Results

An important aspect of any algorithm is how difficult it is to implement, and whether theoretical improvements carry on to lower running times in practice. This aspect is particularly important for all algorithms described in this thesis that rely on *cell enumeration* methods. For this type of algorithm, one needs to be very diligent in order to obtain implementable algorithms at all: it was argued in Chapter 2 that the theoretical running times one obtains cannot be achieved in practice if the cell enumeration involves polynomials in more than one variable. Recall that an implementation of the results stated in Theorem 2.3.6 can be considered as an open research problem. All implementations presented in this chapter use cell enumeration for univariate polynomials only; algorithms requiring higher dimensional cell enumeration have not been implemented at all. However, if the cell enumeration involves univariate polynomials of low degree only, the algorithms are easy to implement. In particular, this is the case for the group of rigid motions if one applies the technique of eliminating translation components – the cell enumeration then needs to be performed for arrangements of circular arcs only. Within the scope of this thesis, some of the algorithms described in Chapters 2–7 have been implemented. In order to give the reader an impression as to what extent these algorithms are applicable, this chapter provides some experimental results of these implementations.
Throughout this chapter, let $G := \mathrm{SE}(2)$ denote the group of rigid motions in the plane. The following algorithms have been implemented for matching under the group of rigid motions:

(1) STANDARD-MATCH: An approximate algorithm for matching with respect to the discrete Fréchet distance based on Algorithm 5.3.5 (using a curves' starting point as a reference point) in combination with Algorithm 5.2.4. The asymptotical running time of this algorithm is $O(m^2n^2)$.

(2) BOUNDED-MATCH: An improved version of the approximate algorithm for matching with respect to the discrete Fréchet distance based on Algorithm 5.3.5 and 5.2.6 and the Hausdorff distance as a lower bound, as described in Section 7.5.1. The asymptotical running time of this algorithm is $O(m^2n^2)$ as well.

(3) PARTIAL-MATCH: An approximate algorithm for matching with respect to the *partial* discrete Fréchet distance based on Algorithms 5.2.6 and 5.4.2. The implementation uses

the directed Hausdorff distance as a lower bound, as described in Section 7.5.1; the asymptotical running time is $O(m^3n^2)$.

(4) HAUSDORFF-MATCH: An approximate algorithm for matching with respect to the directed Hausdorff distance, based on Algorithms 5.2.6 and 5.4.2; the running time is $O(m^2n\log(mn))$.

In the remaining part of this chapter, we examine different aspects concerning running times and quality of approximation of these implementations. In case of HAUSDORFF-MATCH, we compare the results with an implementation of the GRID-method from [33]. The asymptotical running time of GRID is $O(m^{3/4}n^{5/4}\sqrt{\Delta})$, where Δ denotes the diameter of Q.
The goal of all experiments is to determine in what respects the *running times* of the algorithms developed in the previous chapters can be considered practical. There is no intention to determine whether the (discrete) Fréchet distance is a reasonable measure of similarity for real world shape matching applications. In particular, the quality of the results was not compared to any other state-of-the-art methods in shape matching. The results presented here can be viewed as a first step towards using the Fréchet distance as a distance measure for shape matching in real world applications.

8.1 Results for Matching under Fréchet Distance

8.1.1 Data Sets

A popular data set for testing shape matching algorithms is the *Squid* data set that was compiled by Mokhtarian, Abbassi and Kittler [51]. This data set contains 1100 boundary shapes of different maritime animals, stored as polygonal curves. For our needs, some preprocessing steps needed to be performed on this data set:

(1) Each polygonal curve of the squid data set contains 400 to 1000 points. This number of points turned out to be too large to obtain reasonable running times. However, all shapes in the *Squid* data set are significantly *oversampled* (see Figure 8.1 (top)). Suitable data were obtained by *downsampling* the curves in two different ways. First, the polygonal curves were downsampled by taking only every 5th (or every 20th) vertex, yielding 2200 very regular polygonal curves (see Figure 8.1 (middle)) with two different sampling rates.

In order to obtain realistic polygonal curves as query curves, a second way of downsampling was performed. In this second downsampling process, *randomized* downsampling was performed: Each vertex of a polygonal curve was removed with a probability of 4/5 (or 19/20), yielding 2200 unregularly sampled polygonal curves. In addition, some noise was added to each point (see Figure 8.1 (bottom)).

We always used a pair of one regularly downsampled curve P and one randomly downsampled curve Q (with equal sampling rate) for computing $G(P, Q, \varepsilon, \mathbf{d}_F)$-matches.

(2) In order to obtain polygonal curves of different lengths, the curves obtained in Step (1) were cut off after 20, 30, 40, ..., 190 vertices. These truncated curves were used for testing STANDARD-MATCH and BOUNDED-MATCH.

(3) From each randomly downsampled curve of length m ($m = 20, 30, 40, \ldots, 190$) a subcurve of length $m/2$ was generated as a query curve for PARTIAL-MATCH. This query subcurve was obtained by cutting off approximately $m/4$ vertices at both ends of each curve.

(4) The boundary shapes from the Squid data set should usually be considered as closed curves. For our purposes, however, we neglected this and used them as test data for curves with a fixed starting point which are not closed. The algorithm for matching with respect to the discrete Fréchet distance for closed curves was not implemented because in practice, it turns out that PARTIAL-MATCH can be used for matching two closed curves P and Q as well if one considers two cycles of P as one non-closed polygonal curve.

From these data, a number of test instances (i.e., pairs of curves P and Q to be matched) were selected randomly. All plots of running times presented in the following result from average running times in the sense that each point in a plot was acquired as an average running time from at least 9 different instances, each of which was repeated several times. The instances were chosen so that only few instances (at most every third instance) yields a non-empty set of matches. This was done because finding out that *no* matches exist usually takes longer than finding out that matches do exist, so that data sets with too many pairs of curves matching each other could produce unrealistically small running times.

Figure 8.1: Preprocessing of the *Squid* data set. The original *Squid* curve is shown in the top. The two curves in the middle are regularly downsampled versions, while the two curves on the bottom represent randomly downsampled curves with noise added to the points.

Remark 8.1.1 *All experiments presented here were conducted on a 1GHz-Pentium III processor with 512 MByte memory and running the Windows 2000 operating system.*

8.1.2 Running Times in Practice

STANDARD-MATCH versus BOUNDED-MATCH. We start with experimental results for the implementations of STANDARD-MATCH and BOUNDED-MATCH. As was pointed out in Section 7.5.1, it is a difficult problem to determine whether using the Hausdorff distance as a lower bound for the discrete Fréchet distance yields asymptotically smaller time bounds. At least in practice, the lower bound yields significantly smaller running times, as demonstrated in the comparison of the implementations of STANDARD-MATCH and BOUNDED-MATCH in Figures 8.2 and 8.3. We now discuss these results in some more detail.

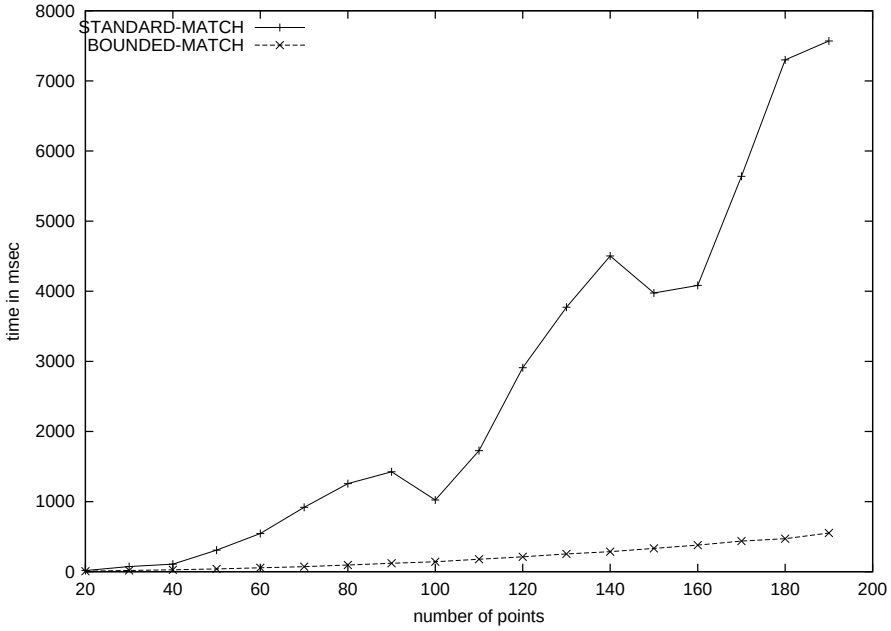


Figure 8.2: Running times of STANDARD-MATCH versus running times of BOUNDED-MATCH. For a fixed fault tolerance ε, the emptiness of $G(P, Q, \varepsilon, \mathbf{d}_{\mathrm{F}})$ was decided, with $|P| = |Q|$ ranging from 20 to 190 in steps of 10.

In Figure 8.2, note that BOUNDED-MATCH is always 7 to 14 times faster than STANDARD-MATCH. For matching two curves consisting of 190 points each, the running time of BOUNDED-MATCH is below 600 msec. Some explanation is needed for the decrease in running times occuring between instances of lengths 90 and 100 as well as between instances of lengths 140 and 150. This phenomenon turns out to be due to an implementation detail of the distance computation in STANDARD-MATCH. Recall that the running time of algorithm STANDARD-MATCH depends on the number of distance computations to be performed as well as the time needed for one single distance computation. The number of distance computations turns out to be strictly increasing with the lengths of the polygonal curves to be matched. What leads to this rather surprising decrease in running times is the time needed for single distance computations: since STANDARD-MATCH only solves a decision problem, we only need to *decide* whether the Fréchet distance is at most ε instead of actually computing $\mathbf{d}_{\mathrm{F}}$. To this end, the dynamic proprамming Algorithm 7.1.2 can be terminated early if during one cycle of the inner `for`-loop, we have $d_{i,j} > \varepsilon$ for all j. This small improvement of the algorithm has a dramatic influence on some instances of a matching problem. In terms of

average running times, this increases the variance of the running times, and, in some cases, even leads to such decrease of running times in spite of increasing curve lengths — and in spite of the fact that each point of the plot is an average running time of 25 different instances. The asymptotical running time, however, is not affected by this phenomenon and clearly shows a superlinear increase in running times. It should be mentioned that the early-termination improvement was observed to yield an average improvement of running times by a factor of about 10. Without this improvement, STANDARD-MATCH could have been tested on much smaller instances only.

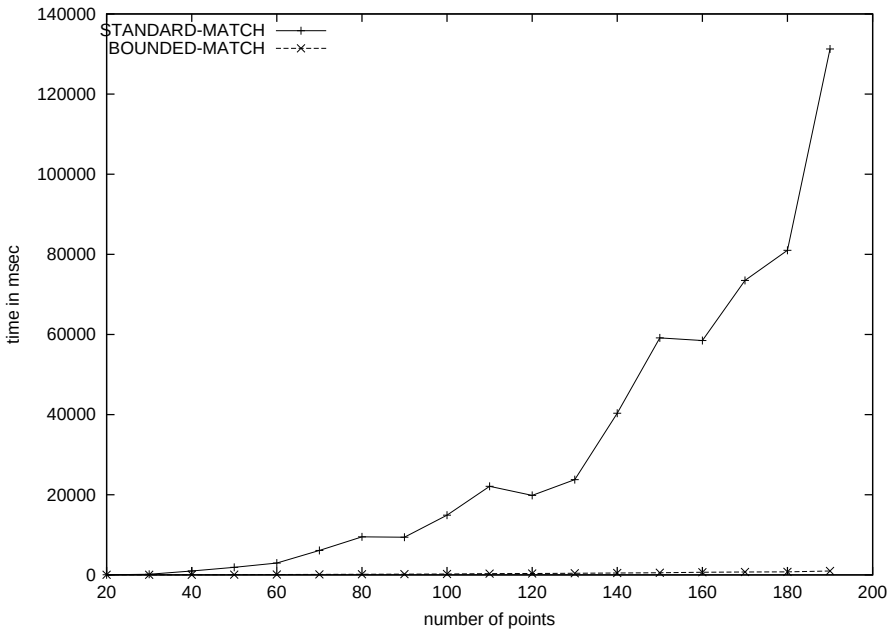


Figure 8.3: Running times of STANDARD-MATCH versus running times of BOUNDED-MATCH. For a fixed fault tolerance ε and sampling rate δ, a 2-approximate solution to the emptiness of $G(P', Q', \varepsilon + \delta, \mathbf{d}_{\mathrm{F}})$ was computed, with P' and Q' denoting δ-sampled versions of P and Q, respectively. Note that we have $\varepsilon = \delta$. By Corollary 7.5.2, this yields a 4-approximate solution for deciding the emptiness of $G(P, Q, \varepsilon, d_{\mathrm{F}})$.

Although BOUNDED-MATCH also uses the early-termination improvement for computing $\mathbf{d}_{\mathrm{F}}$, the effect does not carry noticeably into the running times measured. This is due to the fact that the overall amount of time spent on distance computation is much smaller. Recall that BOUNDED-MATCH uses an ordered cell enumeration. Hence, a significant amount of time is spent on sorting the $O(mn)$ transporters' borders and on the dynamic data structure which keeps track of the Hausdorff distance during ordered cell enumeration.
We now get to the discussion of Figure 8.3. This plot shows running times of almost the same test data as Figure 8.2; the only difference is that that the curves were δ-sampled for $\delta = \varepsilon$, where the fault tolerance ε was fixed over all instances. Looking at the inclusions from Corollary 7.5.2, this yields a 4-approximate solution to the decision problem whether $G(P, Q, \varepsilon, d_{\mathrm{F}}) = \emptyset$.
Testing the Quality of Approximation. Using the inclusions from Corollary 7.5.2, we have seen that any algorithm for matching with respect to the discrete Fréchet distance yields an approximate solution for matching with respect to the continuous Fréchet distance. Fur-

thermore, the approximation factor can be made arbitrarily small by *oversampling* the two polygonal curves to be matched. Hence, an issue that needs to be considered carefully for the implementations of STANDARD-MATCH and BOUNDED-MATCH is to examine what approximation factors can be achieved within acceptable running times.

We now set $\delta := \varepsilon$ as our sampling rate, and consider two polygonal curves P and Q as well as their δ-sampled versions P' and Q'. By Corollary 7.5.2, deciding the emptiness of $G(P', Q', \varepsilon + \delta, \mathbf{d}_{\mathrm{F}})$ yields a 2-approximate solution for the problem of deciding the emptiness of $G(P, Q, \varepsilon, d_{\mathrm{F}})$. This means that for a smaller fault tolerance ε, the curves P and Q need to be oversampled more densely in order to obtain a 2-approximate solution for matching with respect to d_{F}. As a consequence, the number of points which need to be inserted into P and Q to obtain oversampled versions depends linearly on the *Euclidean length* of P and Q (which in turn depends exponentially on the number of vertices in P and Q). Hence, the time for computing 2-approximate solutions depends exponentially on the reciprocal of the fault tolerance ε. This is a somewhat more differentiated statement than saying that the running time depends exponentially on the reciprocal of the quality of approximation. In fact, this observation gives a good explanation why in most experiments conducted in the context of this thesis, 2-approximate solutions for matching with respect to d_{F} could be found within a reasonable amount of time, i.e., within less than 10 seconds or even within less than one second.

This observation motivates the examination of the following experimental setup: varying our fault tolerance ε over some reasonable values (i.e., ε should not exceed the diameter of the polygonal curve as an upper bound and should not be much smaller than the minimum distance between any two vertices as a lower bound), we fix $\delta := \varepsilon$. This guarantees a constant quality of approximation. Furthermore, we fix $m = n = 50$, obtaining the running times shown in Figure 8.4. For $\varepsilon = \delta$ approaching 0, the exponential increase in running time can be clearly observed in this figure (keeping in mind the logarithmic scale on the running time axis).

Looking at Figure 8.4, one also observes that the running time is not strictly decreasing with increasing ε. This phenomenon can be explained as follows: for small values of ε and δ, many of the transporter sets (i.e., the circular arcs defining the arrangement the algorithm is based on) are empty. Increasing fault tolerance means that more of these transporter sets become *non-empty*. Now, recall that essentially for each non-empty transporter set the matching algorithm performs one distance computation. Hence, more non-empty transporter sets induce more distance computations. This phenomenon of increasing running time for larger values for ε "competes" with the phenomenon of smaller running times resulting from fewer points in the sampled versions of the curves.

For $\varepsilon = \delta \geq 24$, we see a steady increase in running times for increasing fault tolerance ε. This is due to the fact that P itself is already δ-sampled for $\delta = 24$. Hence, the increase in running time results from more non-empty transporter sets that have to be considered.

PARTIAL-MATCH. For the implementation of PARTIAL-MATCH, we provide some practical running times for matching polygonal curves with respect to the partial Fréchet distance $\vec{\mathbf{d}}_{\mathrm{F}}$ and, again, we examine the influence of the approximation factor on practical running times. A special feature of the implementation of PARTIAL-MATCH is that it also detects partial occurences of Q if P is a closed curve. This is achieved by computing $G(\tilde{P}, Q, \varepsilon, \vec{\mathbf{d}}_{\mathrm{F}})$ rather than $G(P, Q, \varepsilon, \vec{\mathbf{d}}_{\mathrm{F}})$, with $\tilde{P}$ denoting the curve P concatenated with itself. Further-

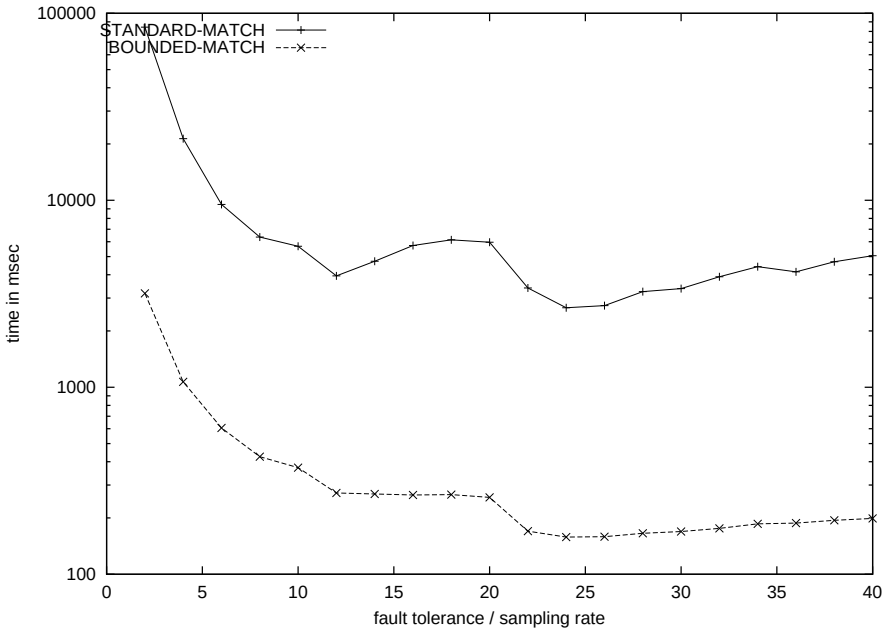


Figure 8.4: Dependance of the running time of STANDRAD-MATCH and BOUNDED-MATCH on the fault tolerance ε and the quality of approximation δ, choosing $\varepsilon := \delta$. The lengths of the original (non-oversampled) curves P and Q to be matched were both set to 50 for all matches performed for this plot. Note the logarithmic scale on the running time axis.

more, PARTIAL-MATCH uses ordered cell enumeration by taking the directed Hausdorff distance as a lower bound for $\vec{\mathbf{d}}_{\mathrm{F}}$, as proposed in Section 7.5.1.
For PARTIAL-MATCH, we evaluated the analougous experimental setup to the one shown in Figure 8.3, i.e., we fixed $\varepsilon = \delta$ and varied the length m of the first curve in steps of 10. The length n of the second curve (which usually is shorter than the first curve when considering partial matches) was set to $m/2$. As in all other plots displayed in this chapter, the resulting running times shown in Figure 8.5 are average running times of several (in this case nine) different instances with identical parameters. The results show a clearly superlinear increase of running times with increasing m; the increase is somewhat smoother than in in Figures 8.2 and 8.3. This is due to the fact that early-termination (which caused some unsteady behaviour in Figure 8.2) can only be applied in a very limited way when computing $\vec{\mathbf{d}}_{\mathrm{F}}$.
As for STANDARD-MATCH and BOUNDED-MATCH, we also examined the influence of simultaneously decreasing the fault tolerance ε, with the sampling rate δ set equal to the fault tolerance. The results shown in Figure 8.6 demonstrate the same exponential growth of running times with decreasing fault tolerance.

8.2 Results for Matching under Hausdorff Distance

In Chapter 5, we obtained an $O(m^2 n \log(mn))$-time decision algorithm for matching point sets with respect to the directed Hausdorff distance. This running time results from Algorithm `Right-Complete-Match` (i.e., Algorithm 5.4.2) in combination with ordered cell enumeration; from now on, we refer to this implementation as HAUSDORFF-MATCH. We compared the

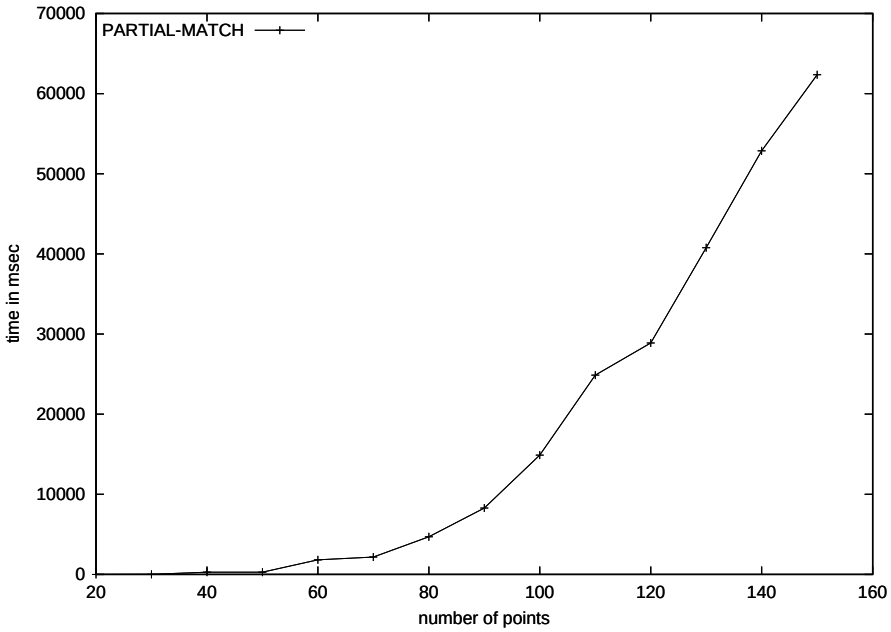


Figure 8.5: Running times of PARTIAL-MATCH. For a fixed fault tolerance ε and sampling rate δ, the emptiness of $G(P, Q, \varepsilon, \mathbf{d}_{\mathrm{F}})$ was decided, with $|P|$ ranging from 20 to 190 in steps of 10, where $|Q| = |P|/2$.

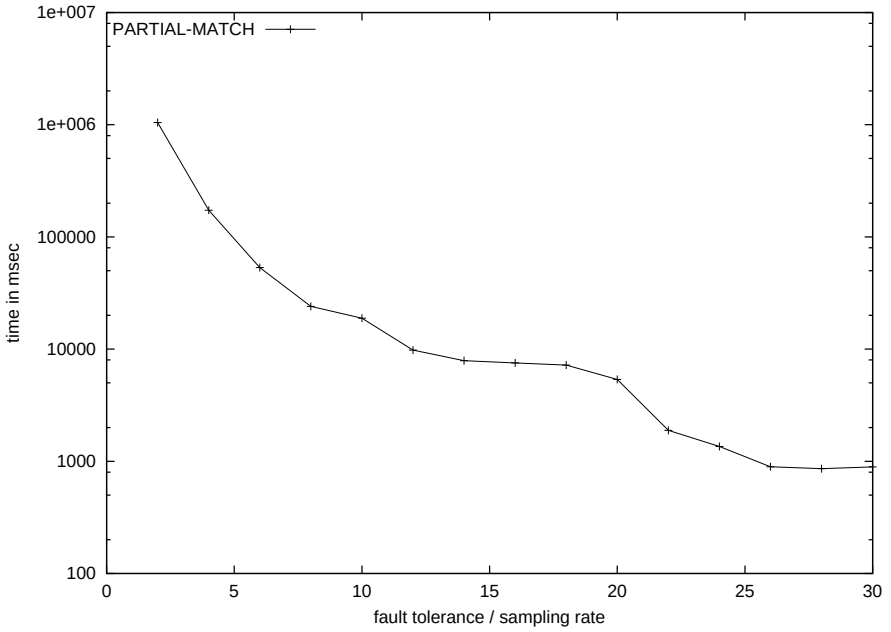


Figure 8.6: Dependance of the running time of PARTIAL-MATCH on the fault tolerance ε and the quality of approximation δ, choosing $\varepsilon := \delta$ for partially matching polygonal curves. The lengths of the original (non-oversampled) curve P was set to 50, and the length of the shorter query curve Q was set to 25 for all matches performed for this plot. Note the logarithmic scale on the running time axis.

running times of HAUSDORFF-MATCH with the implementation of the GRID-method which is described and studied in practice in [33]. In the following, we refer to this implementation as GRID. With the following experiments, we investigate two aspects:

- *Compare the running time of* HAUSDORFF-MATCH *with state-of-the-art algorithms*: The asymptotical running time of HAUSDORFF-MATCH is similar to the time bounds achieved by Goodrich et al. in [36] (which was generalized by means of candidate sets in Chapter 6), which is $O(m^2 n \log n)$. The GRID-algorithm is a significant improvement of Goodrich's method based on a disretization of the transformation space; for typical values of the fault tolerance ε for the decision algorithm, the GRID method performed best in the experiments conducted in [33].

- *Determine potential improvements for matching partially w.r.t. the discrete Fréchet distance:* The significant speed-up achieved by BOUNDED-MATCH over STANDARD-MATCH indicates that using the undirected Hausdorff distance as a lower bound for the Fréchet distance is a powerful way to obtain faster algorithms for matching w.r.t. the Fréchet distance $\vec{\mathbf{d}}_{\mathrm{F}}$. The same holds true for PARTIAL-MATCH, which uses the *directed* Hausdorff distance as a lower bound for $\vec{\mathbf{d}}_{\mathrm{F}}$.

  One can easily see that essentially, PARTIAL-MATCH performs the same steps as HAUSDORFF-MATCH — the only difference is that whenever HASUDORFF-MATCH would return a transformation g, PARTIAL-MATCH computes $\vec{\mathbf{d}}_{\mathrm{F}}(P, gQ)$. This observation suggests that faster algorithms for matching w.r.t. the directed Hausdorff distance might yield faster algorithms for matching w.r.t. $\vec{\mathbf{d}}_{\mathrm{F}}$. In fact, the GRID-algorithm can be viewed as a (significantly) improved version of pattern matching using candidate sets. Hence, GRID can be modified in a similar way as HAUSDORFF-MATCH for matching w.r.t. $\vec{\mathbf{d}}_{\mathrm{F}}$: whenever GRID has found a suitable transformation g with the property that $d_{\mathrm{H}}(P, gQ) \leq \varepsilon$, we determine whether we have $\vec{\mathbf{d}}_{\mathrm{F}}(P, gQ) \leq \varepsilon$ as well — if so, g is returned as a match; if not, the algorithm continues with examining further candidate transformations.

  Such modifications seem to be applicable to many algorithms that are used for matching w.r.t. the directed Hausdorff distance. However, proving whether the aforementioned procedure yields an approximate solution for deciding whether $G(P, Q, \varepsilon, \vec{\mathbf{d}}_{\mathrm{F}}) = \emptyset$ becomes very lengthy and technical — hence, we only compare the running times of algorithms for matching under the directed Hausdorff distance in order to demonstrate the potential of further improvements for matching under $\vec{\mathbf{d}}_{\mathrm{F}}$; the proof of correctness is left as an open problem (cf. Chapter 9).

8.2.1 Data Sets

As our data set, we used the same data as for testing STANDARD-MATCH, BOUNDED-MATCH and PARTIAL-MATCH and as described in Section 8.1.1.
Note that the *Squid* data set has some special properties: since all objects are described by closed polygonal curves, and since most objects' boundaries are — very roughly speaking — ellipsoidal, it makes sense to consider the points as evenly distibuted around an ellipse. Hence, the distribution of points is significantly different from the data used in [33] — for these experiments, Venkatasubramanian et al. used point sets which are evenly distributed in the Euclidean plane. Note that the results from [46] suggest that the distribution of the

points has an important influence on running times; to the best of the author's knowledge, however, there are no theoretical results examining the complexity of matching w.r.t. d_{H} for special point set distributions.
We decided to use the same data set as for STANDARD-MATCH, BOUNDED-MATCH and PARTIAL-MATCH because one major goal was to determine whether other methods for matching w.r.t. the directed Hausdorff distance could improve the running times for matching w.r.t. $\vec{\mathbf{d}}_{\mathrm{F}}$. In general, whenever one considers shape matching applications involving polygonal curves, it appears to make more sense to consider the point set (i.e., the set of vertices of the polygonal curve) as evenly distributed around an ellipse rather than evenly distributed in the plane.

8.2.2 Running Times

For comparing HAUSDORFF-MATCH and GRID, we performeds two test series: first, we fixed the fault tolerance ε and varied the cardinality of the point sets P and Q (Figure 8.7). In a second test series, the cardinality of P and Q was fixed, while the fault tolerance ε varied (Figure 8.8). For both test series, we fixed $|Q| = |P|/2$.
We start with the discussion of Figure 8.7. Clearly, GRID performs much better — in fact, the observed increase of the running time of GRID is nearly linear, while the increase of HAUSDORFF-MATCH is rather quadratic. For $m = 150$ (and $n = 75$), GRID is approximately 100 times faster than HAUSDORFF-MATCH. In terms of finding faster algorithms (based on the GRID-algorithm) for matching with respect to $\vec{\mathbf{d}}_{\mathrm{F}}$, this can be seen as a promising result.
Looking at Figure 8.8, we observe that HAUSDORFF-MATCH and GRID behave rather contrarily: the running time of HAUSDORFF-MATCH grows with increasing fault tolerance ε, while the running time of GRID strictly decreases. The decrease of the running time of GRID is due to the fact that the larger we choose the fault tolerance ε, the coarser is the grid of the transformation space that GRID uses discretized; note that working with a coarse grid means that the number of distance computations performed is small.
In case of HAUSDORFF-MATCH, the running time increases with increasing fault tolerance ε. This phenomen can be explained easily: the runnging time of HAUSDORFF-MATCH mainly depends on the number of non-empty transporter sets, and as ε increases, more and more transporter sets become non-empty. The running time of HAUSDORFF-MATCH, however, is not *strictly* increasing. For example, the running time for $\varepsilon = 20$ is smaller than the running time for $\varepsilon = 18$. This suggests that the variance of the number of non-empty transporter sets (and hence the variance of the running times measured in Figure 8.8) is rather large. In spite of such locally decreasing running times, the overall increasing tendency can be clearly seen in Figure 8.7.

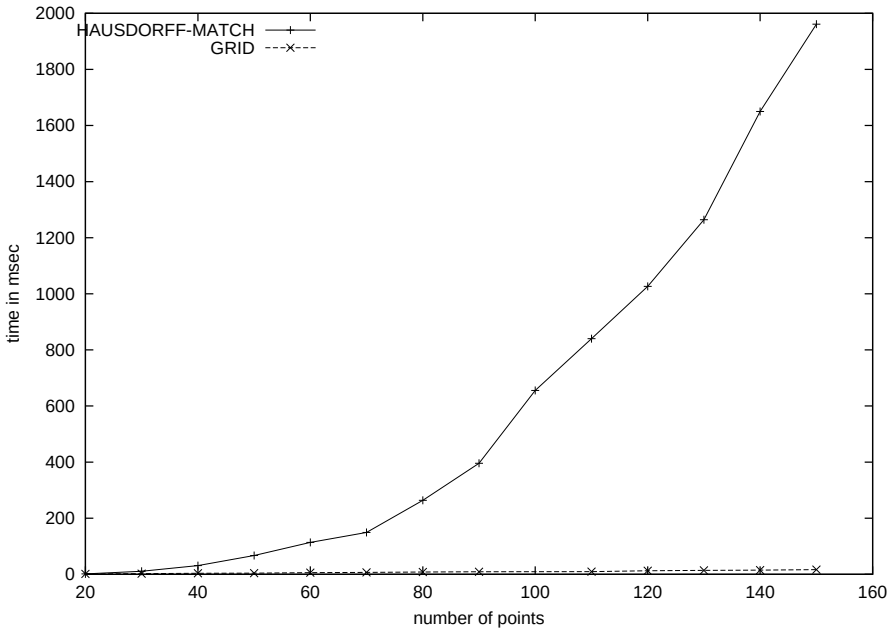


Figure 8.7: *Test series for fixed fault tolerance ε and varying cardinality of the point sets P and Q.* With m and n increasing, the running times of both, HAUSDORFF-MATCH and GRID, increase as well. The running time of GRID increases very slowly and almost linearly, while the running time of HAUSDORFF-MATCH increases much faster.

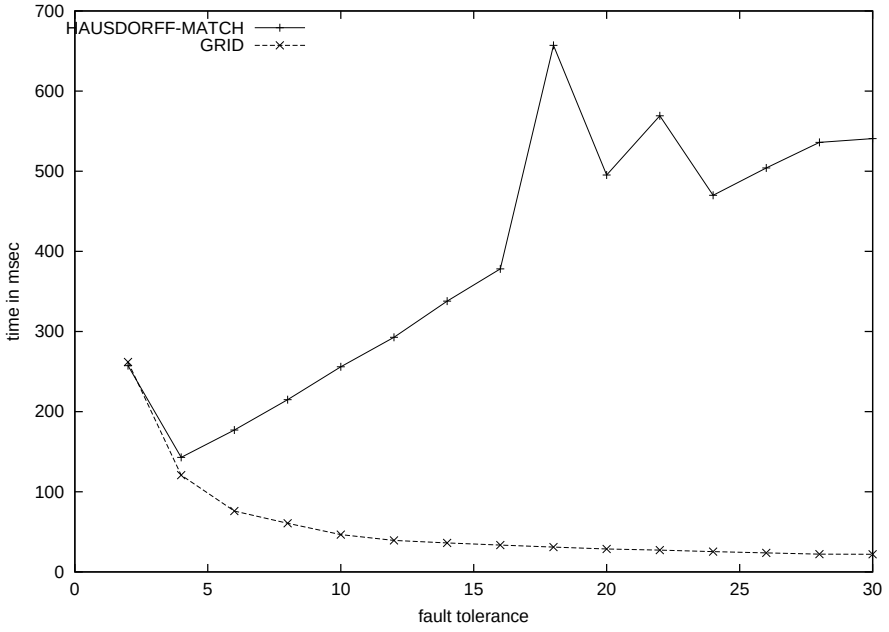


Figure 8.8: *Test series for fixed cardinality of the point sets P and Q and varying fault tolerance ε.*

Chapter 9

Summary and Conclusion

9.1 Summary

In this thesis, methods for matching geometric shapes and patterns with respect to numerous distance measures and under a large class of transformations have been introduced. Most of these algorithms originate in G-inverted lists which have been studied and generalized to (G, ε)-inverted lists in Chapter 3. In this generalization, *transporter sets* take an important place. Studying intersections of these transporter sets in Chapter 4 leads to generic pattern matching algorithms in Chapter 5. These algorithms allow sets (or sequences) of points with respect to a number of different distance measures and transformation groups to be matched. In this general setting, we have two patterns P and Q as well as a transformation group G acting on the set of all objects and a distance measure $\mathbf{d}$ providing a distance concept between two objects.

The suitable distance measures for this generic approach to geometric pattern matching have been characterized as *relational distance measures*, where admissible assignments between points are determined by sets of relations. Many well-known distance measures such as the Hausdorff- and the bottleneck distance belong to this class of distance measures. The class of transformation groups our generic approach applies to has been identified with the category of *linear algebraic groups*. Based on *transporter sets*, the affine variety underlying the linear algebraic group is partitioned into equivalence classes, and a suptransversal of this equivalence relation (i.e., a set containing at least one element from each equivalence class) is computed. For each element g of the suptransversal, one has to decide whether $\mathbf{d}(P, gQ) \leq \varepsilon$, where ε is a specified fault tolerance. Several techniques for making this method more efficient and practical have been studied, involving different properties of the distance measure $\mathbf{d}$. The cost of improving the efficiency by means of these techniques is that the solutions computed by the resulting algorithm are only *approximate* solutions.

In many applications, one is not interested in a decision procedure for matching where a fault tolerance ε needs to be specified as an input, but one rather wants to compute the *minimum* value that can be achieved. For solving this optimization problem, algorithms were proposed in Chapter 6 for matching under rigid motions in two and three dimensions by computing *sets of candidate transformations*. The resulting algorithms work for arbitrary relational distance measures and yield approximate solutions to the optimization problem. Again, certain properties of the distance measure $\mathbf{d}$ allow for improvements in the asymptotical running time of the matching algorithms. These properties coincide with those that occurred

in Chapter 5 for speeding up matching algorithms based on transporter sets.
Finally, in Chapter 7, a discrete variant of the Fréchet distance has been introduced and shown to be a relational distance measure. Further properties of this distance measure have been studied: using *crossing free integer sequences*, it has been shown to be a pseudo-metric. Furthermore, bounds between the discrete and the continuous Fréchet distance have been established. Since the discrete Fréchet distance is a relational distance measure, matching algorithms immediately result from the algorithms stated in Chapters 5 and 6. The bounds between the discrete and the continuous Fréchet distance finally show that any algorithm for matching with respect to the discrete Fréchet distance is an approximate algorithm for matching with respect to the continuous Fréchet distance.
Some of the algorithms described in Chapters 5 through 7 have been implemented. Different aspects of the implementations have been tested and the results have been discussed in Chapter 8.

9.2 Open Problems and Perspectives

Generalization of Candidate Sets: As introduced in Chapter 6, candidate sets allow for the formulation of approximate matching algorithms under rigid motions in two and three dimensions. Some ideas can be generalized easily so as to work under homothetic motions as well. This raises the question whether one can also define and compute suitable candidate sets under even larger groups, such as similarity motions in two and three dimensions or even under the affine general linear group. This would require a generalization of the elementary geometric properties (such as the bounds stated in Remark 6.2.4) used in Chapter 6. Furthermore, one needs to find certain k-tuples (for some suitable integer k) of points taking the place of the diameter pair in two dimensions or the place of the diameter pair with a third point having maximum distance to the straight line between the diameter pair. Based on these extensions, the generalized definition of a candidate transformation needs to be uniquely defined (at least for non-degenerate k-tuples of points).

Computing the discrete Fréchet distance in subquadratic time: For the continuous Fréchet distance, it is unknown whether one can decide in time $o(mn)$ whether $d_{\mathrm{F}}(P,Q) \leq \varepsilon$. The same problem holds for the discrete Fréchet distance introduced in Chapter 7. Along with the discrete Fréchet distance, a rich combinatorial framework — namely crossing free integer sequences and their relation to pairs of discrete reparametrizations — has been introduced. This framework might be a useful tool to either find faster algorithms for deciding whether $\mathbf{d}_{\mathrm{F}}(P,Q) \leq \varepsilon$, or they might be helpful in finding lower bounds for computing $\mathbf{d}_{\mathrm{F}}(P,Q) \leq \varepsilon$.

Faster algorithms for matching w.r.t. relational distance measures: For many specific transformation groups and specific distance measures, there are faster algorithms than the ones resulting from the generic approaches in Chapters 5 and 6. In many cases, it appears to be reasonable that faster algorithms for particular transformation groups or particular distance measures can be generalized to other transformation groups or arbitrary relational distance measures. As an example, consider the GRID-method for matching under rigid motions in the plane with respect to the directed Hausdorff distance from [33]. This algorithm works with a discrete grid of rigid motions, leading to significantly smaller asymptotical time bounds.

Since the GRID-method is based on the ideas by Goodrich et al. from [36], it appears to be reasonable that this approach generalizes to other relational distance measures as well. Note that improving the running time for matching under arbitrary relational distance measures immediately yields faster algorithms for matching with respect to the discrete Fréchet distance.

Generalizing the discrete Fréchet distance to higher-dimensional objects: The Fréchet distance, as it has been considered in Chapter 7, is only defined for one-dimensional objects, i.e., curves. By adapting the space of reparametrizations, the Fréchet distance can be generalized to higher dimensions. Godau [35] provides a generalized definition of the Fréchet distance and shows that for two-dimensional objects, every approximate solution to the decision problem whether the Fréchet distance of two simplicial complexes homeomorphic to a triangle is less than or equal to some $\varepsilon > 0$ is $\mathcal{NP}$-hard; however, the question whether this decision problem is in $\mathcal{NP}$ remains unresolved.

Thus, the question arises whether the *discrete* Fréchet distance can be generalized to higher dimensions and whether one can establish similar bounds as in Theorem 7.4.1. Such a result might in turn lead to a nondeterministic polynomial time algorithm showing that the decision problem is in $\mathcal{NP}$. A general problem arising for simplicial 2-complexes is the following: finding a reparametrizations for two surfaces with a common domain requires the two surfaces to be homeomorphic — and one needs to pick the reparametrizations from a suitable space of reparametrizations. (Note that the problem of determining whether two 2-complexes are homeomorphic is known to be graph-isomorphism-complete [60]). Even for the case that both surfaces are homeomorphic to a disc, the specification of a reasonable discrete space of reparametrizations is a non-trivial problem.

Being aware of the aforementioned $\mathcal{NP}$-hardness result by Godau, a further question arising is whether one can identify a set of instances of the decision problem that can be computed in polynomial time.

Generic Implementation: The approaches to pattern matching based on cell enumeration from Chapter 5 and candidate sets from Chapter 6 allow for pattern matching to be performed under different transformation groups and with respect to numerous distance measures. In terms of implementations, this means that implementations for a large number of pattern matching tasks can rely on a common basis of source code. For many libraries such as CGAL [32], Leda [50] or the *Standard Template Library* (see [13]), the technique of *generic programming* in the C++ programming language proved to be a successful way of obtaining reusable, robust and efficient implementations of algorithms from areas such as Computational Geometry, graph theory or elementary algorithmics. A generic implementation of the pattern matching algorithms proposed in this thesis appears to be a reasonable step towards better applicability of geometric pattern matching algorithms in real-world applications.

Applying Pattern Matching under Fréchet Distance in Real-World Applications: The results from Chapter 8 suggest that the running times of the algorithms implemented within the scope of this thesis suit the needs of real world applications. Besides applications of matching two-dimensional polygonal curves w.r.t. the Fréchet distance, there are interesting applications of matching polygonal curves in three dimensions in Molecular Biology. Here, one often encounters polygonal curves in the form of protein backbones: proteins are chains of amino

acids; such a protein chain adopts a (more or less rigid) spatial configuration which can be viewed as a polygonal curve in three dimensions. The polygonal curve describing this spatial configuration essentially determines the backbone of the protein.
Given two protein backbones, one can determine their similarity by matching them under SO(3) (since there is no fixed orientation of a backbone) with respect to the (discrete) Fréchet distance. Since in most cases, two proteins resemble each other only *partially*, a more interesting problem is to determine longest common subcurves of two protein backbones. Note that all this can be done by combining the techniques from Chapters 6 and 7.

Appendix A

Symbol Reference

$\sim_{P,Q,G,\varepsilon}$	Equivalence of transporter arrangements	41
$\triangle$	Symmetric difference between sets	43
$\sqcup$	Disjoint union	5
$\trianglelefteq$	Normal subgroup	6
$\rtimes$	Semidirect product	6
$\between$	Merge operator	74
$\mathrm{AGL}(k)$	Affine general linear group	6
$\mathrm{band}(f,g)$	Band between f and g	11
$\mathbf{d}_{\mathrm{B}}$	Bottleneck distance	38
$\mathbf{d}_{\mathrm{B}}^{p}$	Relaxed bottleneck distance	40
d_{H}	(directed) Hausdorff distance	17
$\mathbf{d}_{\mathrm{H}}$	undirected Hausdorff distance	17
d_{H}^{K}	Ranked Hausdorff distance	40
$\mathbf{D}_{\mathrm{H}}$	Mean-value Hausdorff distance	40
$\mathbf{D}_{\mathrm{H}}^{2}$	Mean-square Hausdorff distance	40
d_{F}	Fréchet distance	72
$\mathbf{d}_{\mathrm{F}}$	Discrete Fréchet distance	72
$\vec{\mathbf{d}}_{\mathrm{F}}$	Discrete partial Fréchet distance	86
$\mathbf{d}_{\mathrm{F}}^{\circ}$	Discrete closed Fréchet distance	86
d^{K}	Point sequence distance	28
$F_{\varepsilon}(P,Q)$	ε-free-space	73
$\mathbf{F}_{\varepsilon}(P,Q)$	discrete ε-free-space	73
$G(P,Q,\varepsilon,\mathbf{d})$	$(G,\varepsilon,\mathbf{d})$-matches of Q w.r.t. P	17
$\mathrm{GL}(k)$	General linear group	6
$\mathrm{graph}(f)$	Graph of f	9
$\Gamma(\mathbf{P})$	Cell-Graph of $\mathbf{P}$	13
$G_P(Q)$	G-matches of Q	16
$G_P(m)$	G-inverted list of m	16
$G_P^{\varepsilon}(Q)$	(G,ε)-matches of Q	17
$G_P^{\varepsilon}(m)$	(G,ε)-inverted list	18
$\mathrm{HT}(k)$	Group of homothetic motions	6
LCP	Largest common point set	51
$\mathbb{N}$	The set of all natural numbers $\{0,1,2,\dots\}$	

$\mathcal{P}_f(M)$	Finite subsets of M	6
$\mathbb{R}$	The set of all real numbers	
$\mathbb{R}[X]$	Ring of polynomials in X	9
$\mathbb{R}(X)$	Ring of rational functions in X	9
$R(P, Q, \varepsilon)$	ε-relation of P and Q	38
$\mathbf{R}(\mathbf{d}, m, n)$	Set of admissible relations	39
sign	Sign function	9
$\mathrm{SC}(k)$	Group of uniform scalings	6
$\mathrm{SE}(k)$	Group of rigid motions	6
$\mathrm{SO}(k)$	Special orthogonal group	6
$T(k)$	Group of translations	6
$\mathrm{Tran}_G(A, B)$	G-transporter	20
$\tau_{x,y}^{G,\varepsilon}$	(G, ε)-transporter	20
$\mathbb{Z}$	The set of all integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$	

Bibliography

[1] Pankaj K. Agarwal, Sariel Har-Peled, Michael Sharir, and Yusu Wang. Hausdorff distance under translation for points, disks, and balls. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 282–291.

[2] Tatsuya Akutsu. Protein structure alignment using dynamic programming and iterative improvement. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 1996.

[3] Helmut Alt, Oswin Aichholzer, and Günter Rote. Matching shapes with a reference point. In *Proceedings of the 10th Annual ACM Symposium on Computational Geometry*, pages 85–92, 1994.

[4] Helmut Alt and Michael Godau. Measuring the resemblance of polygonal curves. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 102–109, 1992.

[5] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995.

[6] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[7] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[8] Helmut Alt, Christian Knauer, and Carola Wenk. Bounding the Fréchet distance by the Hausdorff distance. In *Abstracts 17th European Workshop Comput. Geom.*, pages 166–169. Freie Universität Berlin, 2001.

[9] Helmut Alt, Christian Knauer, and Carola Wenk. Matching polygonal curves with respect to the Fréchet distance. In *Proc. 18th Int. Symp. on Theoretical Aspects of Computer Science*, pages 63–74, 2001.

[10] Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom.*, 3:237–256, 1988.

[11] Christoph Ambühl, Samarjit Chakraborty, and Bernd Gärtner. Computing Largest Common Point Sets under Approximate Congruence. In *Proc. ESA*, 2000.

[12] Esther M. Arkin, K. Kedem, Joseph S. B. Mitchell, J. Sprinzak, and M. Werman. Matching points into pairwise-disjoint noise regions: combinatorial bounds and algorithms. *ORSA J. Comput.*, 4(4):375–386, 1992.

[13] Matthew. H. Austern. *Generic Programming and the STL.* Addison-Wesley, München, Boston, Madrid, Amsterdamm, 1999.

[14] David Ballard. Generalized hough transform to detect arbitrary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(2):111–122, 1981.

[15] Jérémy Barbay and Claire Kenyon. Adaptive intersection and t-threshold problems. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 390–399. ACM Press, 2002.

[16] Saugata Basu, Richard Pollack, and Marie-Francoise Roy. On computing a set of points meeting every semi-algebraically connected component of a family of polynomials on a variety. *Journal of Complexity*, 13:28–37, 1997.

[17] Jacek Bochnak, Michel Coste, and Marie-Francoise Roy. *Geometrie algebrique reelle*, volume 12 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer Verlag, Berlin Heidelberg, 1987.

[18] John F. Canny. Some algebraic and geometric problems in PSPACE. In *Proceedings 20. ACM STOC*, pages 460–467, 1988.

[19] David E. Cardoze and Leonard J. Schulman. Pattern matching for spatial point sets. In *IEEE Symposium on Foundations of Computer Science*, pages 156–165, 1998.

[20] Samarjit Chakraborty and Somenath Biswas. Approximation algorithms for 3-d commom substructure identification in drug and protein molecules. In *Workshop on Algorithms and Data Structures*, pages 253–264, 1999.

[21] L. Paul Chew, Michael T. Goodrich, Daniel P. Huttenlocher, Klara Kedem, Jon M. Kleinberg, and Dina Kravets. Geometric pattern matching under Euclidean motion. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 151–156, 1993.

[22] Michael Clausen and Frank Kurth. A Unified Approach to Content-Based and Fault Tolerant Music Recognition. *IEEE Transactions on Multimedia*, 2002. Accepted for publication.

[23] Michael Clausen and Frank Kurth. Content-based information retrieval by group theoretical methods. In *Proceedings of the NATO Advanced Study Institute on Computational Noncommutative Algebra and Applications*, Dordrecht, NL, 2003. Kluwer Academic Publishers. To appear.

[24] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183, 1975.

[25] George E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, sep 1991.

[26] David A. Cox, John B. Little, and Don O'Shea. *Ideals, Varieties, and Algorithms.* Springer-Verlag, NY, 2nd edition, 1996. 536 pages.

[27] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Adaptive set intersections, unions, and differences. In *Proceedings of the ACM-SIAM Symposium in Data Structures and Algorithms*, Januar 2000.

[28] Marie-Pierre Duboisson and Anil K. Jain. A modified hausdorff distance for object matching. In *Proceedings of the International Conference on Pattern Recognition*, pages 566–568, Jerusalem, 1994.

[29] Herbert Edelsbrunner. A new approach to rectangle intersections. *International Journal of Computer Mathematics*, 13:221–229, 1983.

[30] Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 301–310, 1996.

[31] George M. Ewing. *Calculus of Variations with Applications.* Dover Publications, Dover edition, 1985.

[32] Andreas Fabri, Geert-Jan Giezeman, Lutz Kettner, Stefan Schirra, and Sven Schönherr. On the design of CGAL a computational geometry algorithms library. *Software Practice and Experience*, 30(11):1167–1202, 2000.

[33] Martin Gavrilov, Piotr Indyk, Rajeev Motwani, and Suresh Venkatasubramanian. Geometric pattern matching: A performance study. In *Symposium on Computational Geometry*, pages 79–85, 1999.

[34] Michael Godau. Die Fréchet-Metrik für Polygonzüge — Algorithmen zur Abstandsmessung und Approximation. Master's thesis, Freie Universität Berlin, Germany, 1991.

[35] Michael Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions.* PhD thesis, Freie Universität Berlin, 1999.

[36] Michael T. Goodrich, Joseph S. B. Mitchell, and Mark W. Orletsky. Practical methods for approximate geometric pattern matching under rigid motion. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 103–112, 1994.

[37] Philip Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10:26–30, 1935.

[38] Sariel Har-Peled. A practical approach for computing the diameter of a point-set. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 177–186, 2001.

[39] Paul J. Heffernan and Stefan Schirra. Approximate decision algorithms for point set congruence. *Computational Geometry: Theory and Applications*, 4:137–156, 1994.

[40] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, December 1973.

[41] Jeffrey E. Humphreys. *Linear Algebraic Groups.* Springer, New York, Heidelberg, Berlin, 1975.

[42] Daniel P. Huttenlocher, Klara Kedem, and Jon M. Kleinberg. On dynamic voronoi diagrams and the minimum hausdorff distance for point sets under euclidean motion in the plane. In *Symposium on Computational Geometry*, pages 110–119, 1992.

[43] Daniel P. Huttenlocher, George A. Klanderman, and William J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:850–863, 1993.

[44] Daniel P. Huttenlocher and Shimon Ullman. Object recognition using alignment. In *Proceedings, First International Conference on Computer Vision*, pages 102–111, London, UK, 1987.

[45] Alistair Moffat Ian H. Witten and Timothy C. Bell. *Managing Gigabytes.* Morgan Kaufmann Publishers, San Francisco, CA, zweite edition, 1999.

[46] Piotr Indyk, Rajeev Motwani, and Suresh Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.

[47] Piotr Indyk and Suresh Venkatasubramanian. Approximate congruence in nearly linear time. In *Symposium on Discrete Algorithms*, pages 354–360, 2000.

[48] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient modelbased recognition scheme. In 2nd Inter. Conf. on Comput. Vision, pages 238-249, 1988.

[49] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[50] Kurt Mehlhorn and Stefan Näher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, Cambridge, 1999.

[51] Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Robust and efficient shape indexing through curvature scale space. In *Proc. British Machine Vision Conference*, pages 53–62, Edinburgh, 1996.

[52] Axel Mosig. Algorithmen und Datenstrukturen zur effizienten Konstellationssuche. Master's thesis, Universität Bonn, Germany, 2001.

[53] Axel Mosig and Michael Clausen. Approximately matching polygonal curves with respect to the Fréchet distance. *submitted for publication*, 2003.

[54] Mario E. Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *IEEE International Conference on Computer Vision*, volume 1, pages 108–, Corfu, Greece, 1999.

[55] Lawrence Rabiner and Bilng-Hwang Juang. *Fundamentals of Speech Recognition.* Prentic Hall, Englewood Cliffs, N. J., 1993.

[56] Edgar A. Ramos. Deterministic algorithms for 3-D diameter and some 2-D lower envelopes. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 290–299, 2000.

[57] James Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13(3):255–299, 1992.

[58] Tido Röder. A group theoretical approach to content-based image retrieval. Master's thesis, University of Bonn, Department of Computer Science III, October 2002.

[59] Jürgen Schmitz. Datenstrukturen und Algorithmen zur effizienten Suche in polyphonen Musikstücken auf MIDI-Basis. Master's thesis, Universität Bonn, Germany, 2001.

[60] John Shawe-Taylor and Tomaž Pisanski. Homeomorphism of 2-complexes is graph isomorphism complete. *SIAM Journal on Computing*, 23(1):120–132, February 1994.

[61] Kevin Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19, 1995.

[62] Nora H. Sleumer. Output-sensitive cell enumeration in hyperplane arrangements. *Nordic Journal of Computing*, 6(2):137–147, 1999.

[63] George Stockman. Object recognition and localization via pose clustering. Computer Vision, Graphics, and Image Processing vol.40, no.3 (1987), pp. 361–87. 14.

[64] Adam Strzebonski. Solving algebraic inequalities. *The Mathematica Journal*, 7:525–541, 2000.

[65] Remco C. Veltkamp and Michiel Hagedoorn. Shape similarity measures, properties, and constructions. In *Advances in Visual Information Systems*, volume 1929 of *Lecture Notes in Computer Science*, pages 467–476, 2000.

[66] Carola Wenk. *Shape Matching in Higher Dimensions.* PhD thesis, Freie Universität Berlin, 2003.

Die im Vergleich zur ersten Fassung vorgenommenen redaktionellen Änderungen sind mit meiner Zustimmung erfolgt.

Bonn, den 12.01.2004