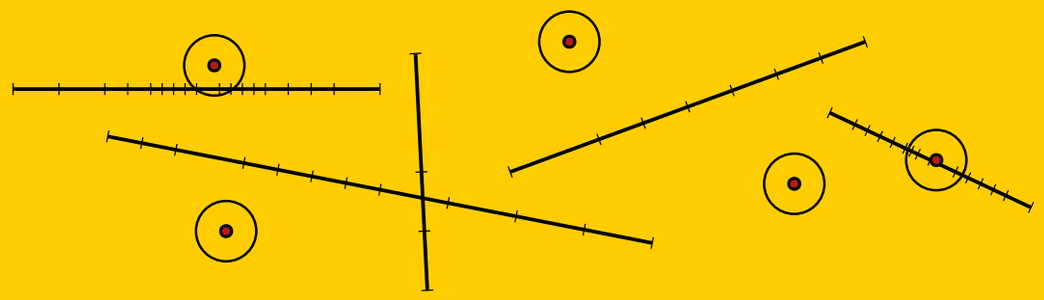


Similarity measures are used in many applications, from comparing and searching for images to procedural content generation. They let us rank objects in order of similarity to a query, and they let us guide many search-based algorithms by telling us how good a chess move is, or how much our generated content resembles a given example. In this thesis, we specifically explore geometric similarity measures, which are used to quantify the resemblance between two geometric objects. We expand on the theoretical understanding of similarity measures, showing how the earth mover's distance can be approximated for a variety of simple geometric objects. We also find new applications for well-studied similarity measures by showing how the Hausdorff distance can be used to interpolate between two shapes, creating a morph. Additionally, we take a detour to study a new model of indeterminacy in graphs.

GEOMETRIC SIMILARITY MEASURES AND THEIR APPLICATIONS Jordi Vermeulen

GEOMETRIC SIMILARITY MEASURES AND THEIR APPLICATIONS

Jordi Vermeulen

Geometric similarity measures and their applications

Jordi Vermeulen

Typeset with LaTeX. Figures drawn using Ipe, the extensible drawing editor. Cover designed in Inkscape.

Printing: Ridderprint, www.ridderprint.nl

This work is licensed under CC BY-NC-ND 4.0. To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/4.0/

DOI: 10.33540/1571
ISBN: 978-90-393-7531-0

Geometric similarity measures and their applications

Geometrische maten van gelijkenis en hun toepassingen
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. H.R.B.M. Kummeling, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op maandag 9 januari 2023 des ochtends te 10.15 uur

door

Jordi Luca Vermeulen

geboren op 15 mei 1991 te Amsterdam

Promotor:	Prof. dr. M. J. van Kreveld
Copromotoren:	Dr. F. Staals Dr. A. Vaxman
Beoordelingscommissie:	Prof. dr. R. C. Veltkamp Prof. dr. ir. A. C. Telea Dr. J. Nederlof Prof. dr. F. C. R. Spieksma Prof. dr. W. Mulzer

Dit proefschrift werd mogelijk gemaakt met financiële steun van NWO onder TOP-project nummer 612.001.651.

Contents

Chapter 1

Introduction

Computers and automation continue to fulfil an increasingly important role in our society. Advances in the research field of computer science are one of the driving forces behind this, and theoretical computer science lays the foundation of this field. In this subfield, we try to determine what it means for a problem to be solvable by a computer, and identify which problems these are. We develop algorithms and data structures that allow us to solve these problems in the best way possible. Here, "best" can refer to a variety of aspects, such as speed, accuracy, and reliability. A major focus is on mathematically proving that a given problem can be solved by a computer, or proving that a given algorithm solves a problem optimally.

In computational geometry, we specifically study these questions for problems with a geometric component. Instead of dealing only with abstract concepts such as numbers and equations, we focus on cases where the objects we study are geometric. The number of geometric problems we are interested in solving is virtually endless. Here are some examples:

1. Calculate the area of a given polygon.
2. Given two locations in an environment with obstacles, calculate a shortest path between them.
3. Given a set of items, calculate the box with smallest volume that they all can be packed into.
4. Given a set of disks, determine if there is a point that lies in all of them.
5. Given a set of lines, find all the points at which two lines intersect.

As can be seen, some problems have very practical applications (1–3), while others are more abstract (4, 5). Some can be relatively easily solved (1, 4, 5), while others have been studied for decades (2, 3).

In all the above cases, the problem can be solved by an *algorithm*. An algorithm is a formal specification of a sequence of steps that can be followed in order to find the solution to a given problem. Algorithms are meant for human consumption: they are written down in a way that allows humans to interpret and analyse them. However, due to their formal nature, they are usually well-suited to being converted into a program that can be executed by a computer.

Let us consider problem number 4. Here, we are given a set of n disks $C = \{c_1, \ldots, c_n\}$, and are tasked with determining whether there is a point lying in all of them. Observe that this equivalent to asking if the intersection $\bigcap_i c_i$ of all the disks is non-empty. We can use a theorem due to Helly [62] to help us: if each triple of three disks has a non-empty intersection, then the entire set of disks has non-empty intersection. We can now give an algorithm to solve problem 4: look at all $\binom{n}{3}$ triples of disks, and determine if their intersection is empty or not. If any of the intersections is empty, then $\bigcap_i c_i = \emptyset$. If none of the intersections are empty, then $\bigcap_i c_i \neq \emptyset$.

In some cases, we may not necessarily be interested or able to compute the exact solution to a problem, but instead are satisfied with approximating the solution. Consider, for instance, problem number 2. Due to the complexity of the obstacles, it may be difficult to compute the shortest path exactly. At the same time, for most practical applications, we may not care much if the calculated path is a little bit longer than the shortest path. In such cases, we can use approximation algorithms. There are several types of approximation algorithm. The approximation may introduce some absolute error x, such that, if OPT is the cost of an optimal solution, the approximation algorithm calculates a solution with cost at most OPT $+ x$. The error may also be multiplicative. A constant-factor approximation algorithm calculates a solution with cost at most $c \cdot$ OPT, for some fixed constant c. In some cases, the user can specify the desired accuracy of the approximation. One such case is a $(1 + \varepsilon)$-approximation, where the algorithm, given some constant $\varepsilon > 0$, calculates a solution with cost at most $(1 + \varepsilon) \cdot$ OPT. Note that it is possible to combine these concept: we might have an algorithm that calculates a solution with cost $(1 + \varepsilon) \cdot \text{OPT} + x$. We will see such algorithms later in Chapter 2. For a more detailed overview of approximation algorithms as they apply to computational geometry, see Har-Peled's book [59].

In many cases, we are not only interested in the existence of an algorithm to solve a particular problem, but we also want to solve the problem as efficiently as possible. However, as an algorithm is not an actual computer program, we cannot measure how long it takes to compute the solution. Still, we want to be able to compare the efficiency of different algorithms. For this, we typically perform *asymptotic analysis*: we analyse how the performance of the algorithm scales as the size of the input becomes very large. A common tool we use here is "big-Oh notation". We say that a function $f(n) \in O(g(n))$ if there is some constant c and some number n_0 such that $f(n) \leq c \cdot g(n)$ for all $n > n_0$. This notation lets us focus on the most important terms in the running time of an algorithm. For instance, one algorithm for problem 4 might require $4n^3 + n^2 + 16n + 238$ steps to compute the solution, while another may take

$16n^3 + 8$ steps instead. Using big-Oh notation, both of these have a running time of $O(n^3)$, which indicates that we expect the performance of the two algorithms to scale roughly the same as the size of the input increases. However, a different algorithm may require $7n^2 \log n$ steps to solve the same problem. This algorithm would have an asymptotic running time of $O(n^2 \log n)$: we expect it to scale better as the input size increases.

A final word about the analysis of algorithms. Many algorithms require a different number of steps to compute the solution depending on the input, even for inputs of equal size. For example, the insertion sort algorithm for sorting a sequence of numbers takes $O(n)$ time to "sort" an already sorted sequence, but takes $O(n^2)$ time if the input is sorted, but in reverse. There are several ways to analyse such algorithms. We typically focus on the input that exhibits the worst running time, leading to what is known as worst-case analysis. However, other approaches exist: we may instead calculate the average running time over all possible inputs (average-case analysis), or analyse the average running time over a sequence of computations (amortised analysis). For a full overview of the design and analysis of algorithms, the reader is referred to one of the many excellent textbooks on the subject, see e.g. [40, 74, 76].

When the problem is very well defined, as in the examples above, it can often be solved optimally or approximately by an algorithm. However, there are many types of problems for which it is not clear how an optimal or even approximate solution can be calculated directly. Consider, for instance, the following example problems:

6. Given a chess position, determine the best move.
7. Given an example of a 3D model of a tree, generate new tree models of the same species.
8. Given an image, retrieve similar images from a database.

In these cases, it is not always possible to solve the problem optimally. In fact, in many of these cases, it is not even clear what "optimal" means. Is there a chess move that is objectively the best in a given position? An argument could be made in the case that a checkmate is possible, but otherwise it is not so easy.

Many techniques can be applied to solving these more complex problems. We might train an AI to learn good chess moves, or we might use genetic programming to generate a procedure for instantiating new tree models. These approaches typically need some way of describing how good a given solution is. For chess, we might devise an evaluation function that estimates how good a given position is. This evaluation function can then be used to search the space of possible moves more efficiently. For tree generation and image retrieval, we might devise a function that determines how similar a given model is to the input. This similarity function can then be used to steer the evolutionary process, or to sort the images in the database on similarity.

In all three cases, we want to come up with a function that somehow determines the quality or similarity of a given object, whether that is a chess move, a procedural

model, or a given image. Two mathematical concepts are central to such functions: *measures* and *metrics*. At a high level, measures assign a number to an object that reflects some property of that object. Typical everyday examples include the number of items in a collection, or the volume of an object. Metrics, on the other hand, assign a number to a pair of objects. Metrics typically reflect some notion of distance or similarity between objects. Typical everyday examples include the shortest distance between two points, or the time it takes to travel from one location to another.

Let us define these concepts more formally. Let X be a set, and let Σ be a collection of subsets of X.[1] A function $\mu : \Sigma \to \mathbb{R}$ is said to be a measure if and only if it satisfies the following properties:

1. $\forall E \in \Sigma : \mu(E) \geq 0$ (non-negativity);
2. $\mu(\emptyset) = 0$ (null empty set);
3. For all countable collections $\mathcal{E}$ of pairwise-disjoint elements of Σ, $\mu(\bigcup_{E \in \mathcal{E}} E) = \sum_{E \in \mathcal{E}} \mu(E)$ (countable additivity).

A function that fulfils only the second and third conditions is also called a *signed measure*. The triple (X, Σ, μ) is called a *measure space*. When $\mu(X) = 1$, we call it a *probability measure*. A common measure is the Lebesgue measure, which includes length, area and volume, and its generalisations to higher dimensions.

A function $d : M \times M \to \mathbb{R}$ is said to be a metric if and only if it satisfies the following properties:

1. $\forall x, y \in M : d(x, y) = 0 \Leftrightarrow x = y$ (identity of indiscernibles);
2. $\forall x, y \in M : d(x, y) = d(y, x)$ (symmetry);
3. $\forall x, y, z \in M : d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality).

Non-negativity (i.e. $d(x, y) \geq 0$) can be derived from these three properties, and is therefore usually not included as a separate requirement. A function that replaces the first condition with a weaker $d(x, x) = 0$ is called a *pseudometric*. A combination (M, d) of a set M and a metric d on that set is called a *metric space*. Depending on the application, metrics are sometimes (somewhat confusingly) called *distance measures* or *similarity measures*, although the terms are not always interchangable. In particular, there exist certain similarity measures that are not actually proper metrics. We do not study these cases in this thesis.

In many cases, the set M has some intrinsic notion of distance: the distance between two elements x and y is the length of the shortest path between them. A metric that is always equal to this length is called an *intrinsic metric*. Examples of intrinsic metrics include the Euclidean distance in $\mathbb{R}^n$ and the great-circle distance on the n-dimensional hypersphere S^n.

[1] Technically, Σ should be a σ-algebra, meaning it should be closed under complementation and countable unions and intersections, but this distinction is not important for our purposes.

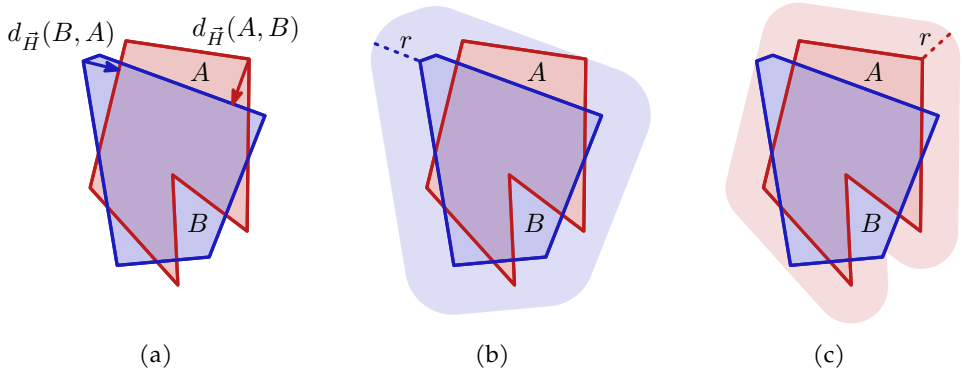


Figure 1.1: The Hausdorff distance from A (red) to B (blue), and vice versa. (a) shows both directed Hausdorff distances. (b) shows the smallest value r such that $A \subseteq B \oplus D_r$, and (c) shows the smallest value r such that $B \subseteq A \oplus D_r$.

1.1 Hausdorff distance

One of the most widely used distance measures is the Hausdorff distance. Given two shapes, it is the maximum shortest distance from any point on one shape to the other shape. This can be formulated as an adversarial game: player one picks a point on one of the shapes, and player two walks to the closest point on the other shape. The Hausdorff distance between the two shapes is then the longest distance that player two can be forced to walk.

More formally, for two sets A and B in some metric space (M, d), we define the *directed Hausdorff distance* as

$$d_{\vec{H}}(A, B) \coloneqq \sup_{a \in A} \inf_{b \in B} d(a, b),$$

The *undirected Hausdorff distance* is defined as

$$d_H(A, B) \coloneqq \max(d_{\vec{H}}(A, B), d_{\vec{H}}(B, A)).$$

A point $a \in A$ is said to *realise* the Hausdorff distance if $\inf_{b \in B} d(a, b) = d_H(A, B)$. When the term Hausdorff distance is not qualified as directed or undirected, it is usually referring to the undirected version. We adopt this convention throughout the rest of this thesis.

An alternative, equivalent definition of the Hausdorff distance can be given as follows. Given sets A and B, we can take the Minkowski sum $B \oplus D_r$ of B with a disk of radius r:

$$B \oplus D_r \coloneqq \{b + d \mid b \in B, d \in D_r\}.$$

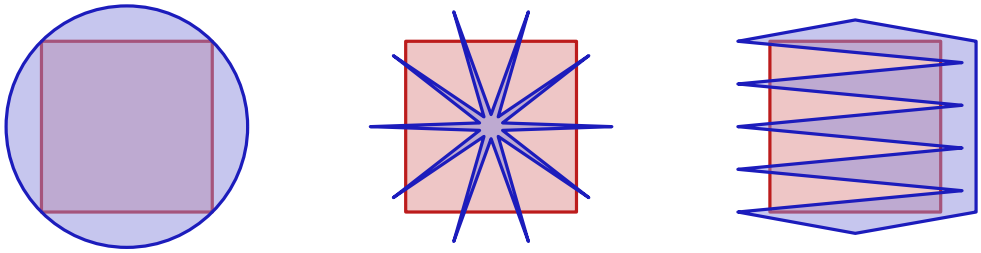


Figure 1.2: Very different shapes may have the same Hausdorff distance to the same shape. In all three cases, the undirected Hausdorff distance between the red and the blue shape is the same.

The directed Hausdorff distance $d_{\vec{H}}(A, B)$ from A to B is now the smallest value r such that $A \subseteq B \oplus D_r$. The undirected Hausdorff distance $d_H(A, B)$ is the smallest value r such that $A \subseteq B \oplus D_r$ and $B \subseteq A \oplus D_r$. Both definitions are illustrated in Figure 1.1.

Note that the Hausdorff distance is only a proper metric if we restrict A and B to be compact sets. We can see this easily from the following example: let A be a disk, and let B be the same disk with a single point removed. Clearly, $d_H(A, B) = 0$, but $A \neq B$. This violates the identity of indiscernibles, which is required for a metric. However, if $C(M)$ is the set of all non-empty compact subsets of M, then $(C(M), d_H)$ is itself a metric space.

As the Hausdorff distance between A and B depends only on the point that is furthest away from the points in the other shape, it is categorised as a *bottleneck metric*. In practice, this means that we can typically greatly modify A and/or B while maintaining the same Hausdorff distance (see Figure 1.2 for some examples). While this is not a desirable property for all applications, the simplicity of the Hausdorff distance makes it easily computable for a wide variety of inputs.

The Hausdorff distance has been applied in many different settings, such as computer vision [43] and computer graphics [16, 39], for tasks such as template matching, and error computation between a model and its simplification. Much of the research in computational geometry has focused on developing algorithms to compute the Hausdorff distance between different types of objects, such as convex polygons [18], points [46], line segments, polylines, and simple polygons [8], simplices in k-dimensional Euclidean space [9], certain classes of curves [15], and imprecise points [75]. However, computing the Hausdorff distance between general semialgebraic sets has been shown to be $\forall\exists_<\mathbb{R}$-complete [66].

Additionally, much attention has been devoted to minimising the Hausdorff distance under certain transformations. Here we typically define some set of allowed transformations $\mathcal{T}$, and are given two sets A and B of some geometric objects. The task is then to find the transformation $T \in \mathcal{T}$ that minimises $d_H(A, T(B))$. This has

been studied for point sets under translation [46, 65] or translation and rotation [38], and polygons under rigid motions (translation, rotation and scaling) [8].

In Chapters 3 and 4, we study how the Hausdorff distance can be applied to the morphing of shapes. In particular, given two shapes A and B with $d_H(A, B) = 1$, we investigate ways of finding a shape C that has $d_H(A, C) = \alpha$ and $d_H(B, C) = 1 - \alpha$ for any $\alpha \in [0, 1]$. In Chapter 3, we consider the maximal shape that fulfils these conditions, studying its geometric and combinatorial properties, and give algorithms for computing it. We also consider a natural generalisation to the Hausdorff middle of multiple sets, again studying properties and giving algorithms. This chapter is based on work that previously appeared in:

> M. van Kreveld, T. Miltzow, T. Ophelders, W. Sonke and J. L. Vermeulen. Between shapes, using the Hausdorff distance. *Computational Geometry*, 100:101817, 2022.

In Chapter 4, we consider a different shape that fulfils the conditions, studying the same properties and giving algorithms. We also give an extensive experimental analysis comparing the two morphs. This chapter is based on work that previously appeared in:

> L. de Kogel, M. van Kreveld and J. L. Vermeulen. Abstract Morphing Using the Hausdorff Distance and Voronoi Diagrams. *Proceedings of the 30th Annual European Symposium on Algorithms*, pages 74:1–74:16, 2022.

1.2 Fréchet distance

As mentioned, one downside of the Hausdorff distance is that is can be unresponsive to changes in the input sets that do not affect the points realising the Hausdorff distance. A distance measure that partially alleviates this problem is the Fréchet distance. The Fréchet distance is classically defined between two curves, although other versions exist. Intuitively, the Fréchet distance considers the maximum distance that two points attain while moving over each of the curves. The usual metaphor is that of a person walking a dog: each has a curve describing the path they take, but they can speed up and slow down as desired. The question is what the shortest possible leash is that allows the walk to be completed.

We now give a formal definition. Let $A, B : [0, 1] \to M$ be two curves. We say that $\alpha : [0, 1] \to [0, 1]$ is a reparameterisation of a curve if it is a continuous, non-decreasing function. The Fréchet distance $d_F(A, B)$ between A and B is then defined as

$$d_F(A, B) \coloneqq \inf_{\alpha,\beta} \max_t d(A(\alpha(t)), B(\beta(t))),$$

where α and β are reparameterisations, and d denotes the distance function of metric space M. For any pair of reparameterisations, we consider the maximum distance attained while traversing both curves. We then take the reparameterisation that

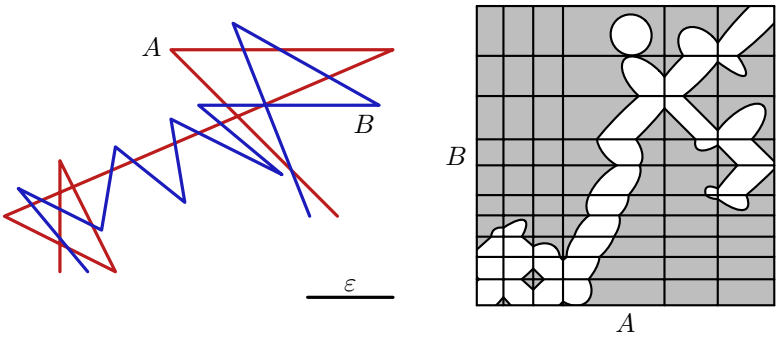


Figure 1.3: The free space diagram shows the valid configurations of points on polylines A and B for a given leash length ε. The grid denotes the vertices of each polyline, with the width/height of each cell proportional to the length of the segment on the polyline. White areas are configurations for which the leash length is sufficient, grey areas are those where the leash is too short.

minimises this maximum distance. We can view the Fréchet distance as turning the set of all curves in a metric space M into a metric space in its own right.

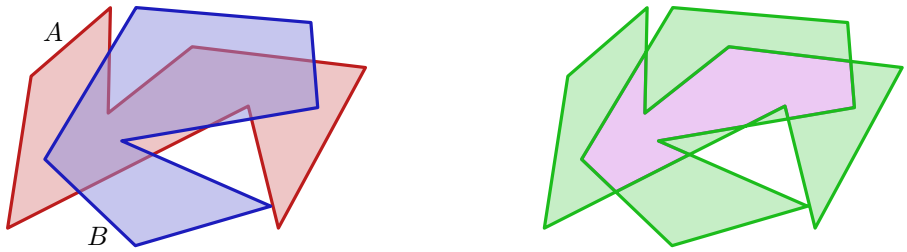
The typical tool for both reasoning about and calculating the Fréchet distance is the *free space diagram*. For a given value $\varepsilon \geq 0$, we define the following set:

$$F_\varepsilon(A, B) := \{(s, t) \in [0, 1] \times [0, 1] \mid d(A(s), B(t)) \leq \varepsilon\}.$$

That is, $F_\varepsilon(A, B)$ is the set of configurations in which the person and the dog are at most ε apart. It is then not hard to see that the decision version of the Fréchet distance calculation (i.e. "is $d_F(A, B) \leq \varepsilon$?") is equivalent to asking if there is an xy-monotone path from $(0, 0)$ to $(1, 1)$ in $F_\varepsilon(A, B)$. Parametric search then lets us find $d_F(A, B)$ using $F_\varepsilon(A, B)$ to solve the decision problem. An example of a free space diagram is given in Figure 1.3.

Like the Hausdorff distance, the Fréchet distance is a bottleneck metric: it depends only on the maximum distance attained during the traversal of both curves. This can again mean that the distance between two curves is not affected by significant modifications of one or both of the input curves. For example, a line segment A and the same line segment B with a single spike of length x in the middle will have the same Fréchet distance as two parallel line segments placed x apart. However, this problem is less pronounced than with the Hausdorff distance, as even small changes to the input may make a different reparameterisation more effective.

Many variants of the Fréchet distance exist. The *weak Fréchet distance* does not require the reparameterisations to be non-decreasing, allowing one or both of the traversals to backtrack. If the inputs are polylines, the *discrete Fréchet distance* measures the distance only at the vertices, ignoring the length of the leash while travelling



(a) (b)

Figure 1.4: (a) Two overlapping sets A (red) and B (blue). (b) Their intersection in purple, and their symmetric difference in green.

between them. The *homotopic Fréchet distance* [36] requires all the shortest paths between $A(\alpha(s))$ and $B(\beta(t))$ to be homotopic. The input is then augmented with obstacles that the leash cannot cross.

Instead of taking the maximum leash length, we can also consider the sum of all leash lengths. This is is referred to as *dynamic time warping* [26] when we consider only the positions at the vertices, and has applications in signal processing. Recently, the first polynomial time algorithm was found that computes the exact solution to *continuous dynamic time warping* for 1D curves [34]. In this setting, we integrate the length of the leash during the entire traversal of the curves.

As the definition of the Fréchet distance requires the objects being compared to be parameterised, algorithms are available only for objects that are already parameterised, or allow a natural parameterisation. Besides simply computing the Fréchet distance between two (polygonal) curves [13], work has been done in finding curves in some set or domain that minimise the Fréchet distance. For instance, Har-Peled and Raichel [60] show how to find two curves inside two simplicial complexes that minimise the weak Fréchet distance. Similarly, Alt et al. [11] show how to find a path in a geometric graph that minimises the Fréchet distance to a given polygonal curve.

The Fréchet distance has also been applied to higher-dimensional objects. Here the decision problem is NP-hard, even when A and B are non-intersecting planar polygons with holes or 2D terrains [32]. However, Alt and Buchin [10] show that the weak Fréchet distance between surfaces can be computed in polynomial time.

1.3 Area of symmetric difference

While the Fréchet distance is more sensitive to changes in the input than the Hausdorff distance, it is still a bottleneck metric. For applications in which such a metric is not suitable, we need metrics that depend on the entire input for their values. One such metric can be obtained by considering the *symmetric difference* between two sets.

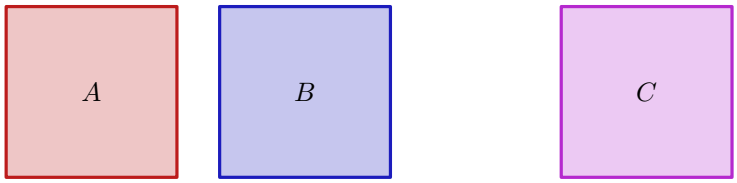


Figure 1.5: The area of symmetric difference cannot differentiate between distinct disjoint sets with the same area. In this example, $d_{\triangle}(A, B) = d_{\triangle}(A, C) = d_{\triangle}(B, C)$.

Given two sets A and B, the symmetric difference $A \triangle B$ is defined as

$$A \triangle B \coloneqq (A \cup B) \setminus (A \cap B).$$

That is, it is the union of the parts of A and B that do not overlap. See Figure 1.4 for an illustration.

If our sets live in a measure space (X, Σ, μ), then we can define a distance function as follows:

$$d_{\triangle}(A, B) \coloneqq \mu(A \triangle B).$$

For $d_{\triangle}$ to be a true metric though, we need to be somewhat careful with the types of sets we consider. Similar to the Hausdorff distance, we need A and B to be compact, as otherwise we could have $d_{\triangle}(A, B) = 0$ while $A \neq B$. However, even for compact sets, $d_{\triangle}$ may only be a pseudometric. Consider, for instance, taking B to be A plus a single point outside of A. This set is compact, but will have the same Lebesgue measure as A. One way to ensure $d_{\triangle}$ is a metric is by requiring that our sets are bounded and equal to the closure of their interior.

The area of symmetric difference is not a bottleneck metric. However, it can still assign the same distance to significantly different pairs of sets, particularly when the sets are disjoint. Consider, for instance, two unit squares placed close together, and two unit squares placed far apart. The area of symmetric difference will be 2 in both cases. See Figure 1.5 for an illustration. This is less of an issue when we are interested in comparing shapes only, but is undesirable when the location of the shapes is important.

There are two reasons we might be interested in finding a transformation that minimises the area of symmetric difference. First, when we are only interested in computing the similarity of shapes, as opposed to objects with some location, it seems to make sense to normalise the input by finding the relative alignment that minimises the distance. On the other hand, it may be that our task is precisely to align certain objects. We could naively place the centroids on top of each other, but this may create an unnatural alignment. A better approach may be to align the objects such that the area of symmetric difference is minimised. Note that this is the same as maximising the area of overlap.

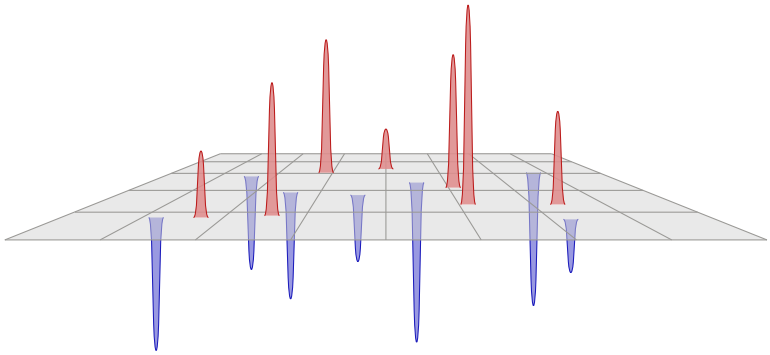


Figure 1.6: One way to look at the earth mover's distance is to consider one distribution (red) as piles of earth, and the other distribution (blue) as holes in the ground. The question is then to level the ground by filling the holes with earth from the piles in the most efficient way possible.

For convex shapes, aligning the centroids actually gives an $11/3$-approximation of the optimal area of symmetric difference [12]. Similarly, the area of the overlap is at least $9/25$ times the optimal area for convex polygons [24]. For general shapes, no exact polynomial-time algorithm is known, although approximation algorithms do exist for polytopes under rigid motion [5, 106].

1.4 Earth mover's distance

The Hausdorff and Fréchet distance are bottleneck measures, and the area of symmetric difference permits changes in the disjoint parts of the shapes without changing the value of the metric function, as long as they have the same area. A metric that has neither of these problems is the *earth mover's distance* (EMD). The earth mover's distance is typically defined between two weighted point sets in the plane of equal total weight. Its name derives from the following analogy: we can consider one of the point sets to be a set of piles of earth, and the other point set to be a set of holes. The weights of the points determine the height of the piles and the depth of the holes. Moving some quantity of earth over some distance has a cost associated with it that is the product of the amount of earth moved and the distance over which it is moved. The task is then to find the most cost-efficient way to fill all the holes with the earth from our piles. This is illustrated in Figure 1.6.

The general problem of optimally moving a distribution of mass was first described by Monge in 1781 [90], and was reformulated by Kantorovich in 1942 [69]. It is a special case of the more general *optimal transport problem*. For a full treatment of the problem's history and connections to other areas of mathematics, the reader is

referred to Villani's book [107].

Let us now define the earth mover's distance formally. Let $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$ be two point sets in some metric space (M, d). Additionally, let μ_P and μ_Q be two probability measures[2] assigning a weight to each element of P and Q, respectively. We define H to be the set of all mappings of weight from P to Q, i.e. for a given $\eta \in H$, $\eta(p_i, q_j)$ specifies how much mass moves from p_i to q_j. These mappings are usually known as *transport plans*. We can then define the earth mover's distance d_e as follows:

$$d_e(P, Q) = \min_{\eta \in H} \sum_i \sum_j d(p_i, q_j) \cdot \eta(q_i, q_j)$$

This is equivalent to a minimum cost flow problem in a Euclidean graph with P as the sources and Q as the sinks. When all weights are equal, this simplifies to an assignment problem, or finding a minimum-weight maximum matching in a bipartite graph.

It is not necessary to restrict the definition of the earth mover's distance to only point sets. In fact, a generalised version called the *Wasserstein metric* has been studied in mathematics. Let P and Q be two probability measures. The r-Wasserstein distance is then defined as follows:

$$W_r(P, Q) = \inf_{\eta \in H} \left(\int_P \int_Q d(p, q)^r \cdot \eta(p, q) \ \mathrm{d}p \ \mathrm{d}q \right)^{1/r}$$

The parameter r lets us control the influence of the distance on the cost. Note that our definition of d_e is equivalent to W_1 when P and Q are finite point sets.

The earth mover's distance has been widely applied in fields such as image retrieval [97], shape matching [57, 86, 103] and mesh reconstruction [54]. It has also been studied in many geometric contexts. Agarwal et al. [2] give both exact and approximation algorithms for the case where P and Q are point sets under some L_p metric. For sets of points with integer weights, Khesin at al. [73] give two algorithms running in $O(n\varepsilon^{-O(d)} \log(\Lambda)^{O(d)} \log n \log(1/\rho))$ and $O(n\varepsilon^{-O(d)} \log (U^{O(d)} \log(n/\rho)^2)$ time that compute a $(1+\varepsilon)$-approximation with probability at least $1-\rho$, where d is the dimension, Λ the aspect ratio of the input, and U the total mass. This result was improved by Fox and Lu [50], using a similar method to obtain, with high probability, a $(1+\varepsilon)$-approximation in $O(n\varepsilon^{-O(d)} \log^{O(d)} n)$ time.

The EMD was also studied when the input sets may be transformed: Cabello et al. [35] present algorithms that, given two weighted point sets of n and m points in $\mathbb{R}^2$, compute a $(1+\varepsilon)$-approximation of a translation that minimises the EMD, and a $(2+\varepsilon)$-approximation of a rigid motion that minimises the EMD. These algorithms run in $O((n^2/\varepsilon^4) \log^2 n)$ and $O((n^3 m/\varepsilon^4) \log^2 n)$ time, respectively.

[2] This ensures that the two measures have equal total measure, and simplifies some of the definitions by giving a total measure of one.

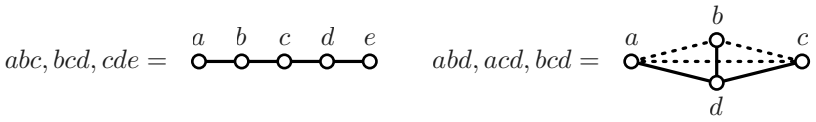


Figure 1.7: Two different sets of triples with their reconstructions. On the left, the only graph consistent with this set of triples is a path of five vertices. On the right, multiple graphs are possible: the solid edges plus any choice of at most one dotted edge.

For continuous distributions, rather than discrete point sets, many numerical algorithms are known (see e.g. De Goes et al. [53], Lavenant et al. [83], Mérigot [87, 88] and Solomon et al. [101]). For the case where one set consists of weighted points and the other is a bounded set $C \subset \mathbb{R}^d$, Geiß et al. [52] give a geometric proof that there exists an additively weighted Voronoi diagram such that transporting mass from each point p to the part of C contained in its Voronoi cell is optimal. The weights of this Voronoi diagram can be determined numerically.

In Chapter 2, we describe the first algorithms that compute the earth mover's distance between geometric objects other than points with provable approximation ratios. In particular, we give $(1+\varepsilon)$-approximations for the case where P is a set of points and Q is a set of line segments or triangles in $\mathbb{R}^2$. We also give approximation algorithms where both P and Q are sets of segments or sets of triangles. In these cases, there is an additional small additive term in our approximation ratio. Finally, we show how to extend these results to higher dimensions, giving approximation algorithms between points and simplices, as well as simplices and simplices, in $\mathbb{R}^d$. A preliminary version of this research appeared in

> M. van Kreveld, F. Staals, A. Vaxman and J. L. Vermeulen. Approximating the Earth Mover's Distance between sets of points and line segments. *Proceedings of the 35th European Workshop on Computational Geometry*, pages 24:1–24:6, 2019.

1.5 Graph reconstruction

In addition to the work on geometric similarity measures, we present results on an unrelated topic in graph reconstruction.

Imagine that we get information on a graph, but not its complete structure by a list of edges. One natural question that arises is whether we can determine the graph uniquely based on this information. This type of problem is usually referred to as *graph reconstruction*.

The problem of graph reconstruction arises naturally in many cases where some unknown graph is observed indirectly. For instance, we may have some (noisy)

measurement of the graph structure, or only have access to an oracle that answers specific types of queries. Much previous research has been done for specific cases, such as reconstructing metric graphs from a density function [42], road networks from a set of trajectories [4], graphs using a shortest path or distance oracle [68], labelled graphs from all r-neighbourhoods [93], or reconstructing phylogenetic trees [30]. A lot of research has been devoted to the *graph reconstruction conjecture*, which states that it is possible to reconstruct any graph (up to isomorphism) from all subgraphs obtained through the removal of one vertex [28, 82, 94, 108].

We explore the case where the input consists of all triples of vertices whose induced subgraph is connected. In other words, we know for each given triple of vertices that two or three of the possible edges are present, but we do not know which ones. We may be able to deduce the graph fully from all given triples.

As a simple example, assume we are given the (unordered) triples abc, bcd, and cde. Then the only (connected) graph that matches this specification by triples is the path a—b—c—d—e. On the other hand, if we are given all triples on four vertices a, b, c, d except for abc, then there are several graphs possible. We must have the edges ad, bd, and cd, and zero or one of the edges ab, ac, and bc. See Figure 1.7 for an illustration.

This model of indeterminacy of a graph is perhaps the simplest combinatorial model for partial information, a model that does not use probability. Normally a graph is determined by pairs of vertices which are the edges; now we are given triples of vertices with indeterminacy on the edges between them.

Many different types of uncertainty in graphs have been studied. Fuzzy graphs [95] are a generalisation of fuzzy sets to relations between elements of such sets. In a fuzzy set, membership of an element is not binary, but a value between zero and one. Fuzzy graphs extend this notion to the edges, which now also have a degree of membership in the set of edges. Uncertain graphs are similar to fuzzy graphs in that each edge has a number between zero and one associated with it, although here this number is a probability of the edge existing. Much work has been done on investigating how the usual graph-theoretic concepts can be generalised or extended to fuzzy and uncertain graphs [70, 92]. Methods for drawing these types of graphs have also been developed, see e.g. [98, 100].

In Chapter 5, we investigate the graph reconstruction model described above. We show that for any set of triples, we can efficiently test whether a graph consistent with the triples exists. We also show that this solution may not be unique in general, but that several classes of graphs allow for unique reconstruction. We conclude by studying which structures of the triples make the reconstruction not unique, although a full characterisation currently eludes us. This chapter is based on unpublished joint work with Jeff Erickson and Marc van Kreveld.

Chapter 2

Approximating the earth mover's distance between sets of geometric objects

2.1 Introduction

The earth mover's distance (EMD) is a metric that is widely used in fields such as image retrieval [97], shape matching [57, 86, 103] and mesh reconstruction [54]. It models two sets P and S as distributions of mass, and takes their distance $d_e(P, S)$ to be the minimum cost of transforming one distribution into the other, where cost is measured by the amount of mass moved multiplied by the distance over which it is moved. More formally,

$$d_e(P, S) = \inf_{\eta \in H} \int_P \int_S d(p, s) \cdot \eta(p, s) \ \mathrm{d}p \ \mathrm{d}s$$

where H is the set of all mappings of mass between P and S and $d(\cdot, \cdot)$ is any metric. In the case where P and S are finite sets of (weighted) points, we can rewrite this as

$$d_e(P, S) = \min_{\eta \in H} \sum_{p \in P} \sum_{s \in S} d(p, s) \cdot \eta(p, s)$$

For unweighted point sets, the solution can be obtained by solving an assignment problem; for weighted point sets, this is an instance of a minimum cost flow problem.

Recently, much attention has been devoted to computing the earth mover's distance when both P and S are sets of points [2, 50, 73, 99]. In this chapter we expand on this by letting P and S be sets of points, line segments, triangles or d-dimensional

simplices in $\mathbb{R}^d$. We describe a unified framework for calculating the EMD between points and segments, points and triangles, points and simplices, segments and segments, triangles and triangles and simplices and simplices. Our approach provides polynomial-time algorithms that give a $(1+\varepsilon)$-approximation to the earth mover's distance between P and S, for some arbitrarily small $\varepsilon > 0$. Moreover, our algorithms produce an assignment of mass that realises this cost. For triangles and simplices, the running time also depends on the largest edge length (note that we normalise the total area/volume of each set to one, so we cannot improve the running time by scaling the input). When neither set consists of discrete points, there is a small extra additive term in our approximation. For all our algorithms, our approach is to subdivide the elements of the input into sufficiently small pieces, and then approximate each piece by a point. The approximate optimal transport plan can then be obtained by solving a transport problem on these points. Our results are summarised in Table 2.1. Note that all our algorithms give the solution with high probability; this is simply a consequence of using Fox and Lu's algorithm [50] to solve the optimal transport problem on points. Substituting a deterministic algorithm here would make our results deterministic as well.

To our knowledge, these are the first combinatorial algorithms with a provable approximation ratio for the earth mover's distance when the objects are continuous rather than discrete points. We give algorithms for moving mass from points to segments (Section 2.5), points to triangles (Section 2.6), points to simplices (Section 2.9.1), segments to segments (Section 2.7), triangles to triangles (Section 2.8) and simplices to simplices (Section 2.9.2).

Objects	Running time	Additive term
Points to segments	$O\left(\frac{nm}{\varepsilon^c}\operatorname{polylog}\frac{nm}{\varepsilon}\right)$	-
Points to triangles	$O\left(\frac{nm}{\varepsilon^c}\operatorname{polylog}\frac{nm\Delta}{\varepsilon}\right)$	-
Points to simplices	$O\left(6^d d^2 d^d m + \frac{105^d d^2 d^{d/2} nm}{\varepsilon^{O(d)}}\log^{O(d)}\left(\frac{dnm\Delta}{\varepsilon^d}\right)\right)$	-
Segments to segments	$O\left(\frac{nm}{\varepsilon^c}\operatorname{polylog}\frac{nm}{\varepsilon}\right)$	$O\left(\frac{\varepsilon}{nm}\right)$
Triangles to triangles	$O\left(\frac{nm\Delta(n+m)}{\varepsilon^c}\operatorname{polylog}\frac{nm\Delta}{\varepsilon}\right)$	$O\left(\frac{\varepsilon}{\sqrt{nm}}\right)$
Simplices to simplices	$O\left(\frac{\sqrt{d}(nm)^{1/d}\Delta^d(n+m)}{\varepsilon^{O(d)}}\log^{O(d)}\left(\frac{d(nm)^{1/d}\Delta^d}{\varepsilon}\right)\right)$	$O\left(\frac{\sqrt{d}\varepsilon}{(nm)^{1/d}}\right)$

Table 2.1: A summary of our results for different choices of sets P and S of sizes n and m. d is the dimension, Δ is the largest diameter of any element of the sets.

2.2 Related work

The general problem of optimally moving a distribution of mass was first described by Monge in 1781 [90], and was reformulated by Kantorovich in 1942 [69]. It is known as the earth mover's distance due to the analogy of moving piles of dirt around; it is also known as the 1-Wasserstein distance, and is a special case of the more general optimal transport problem. For a full treatment of the problem's history and connections to other areas of mathematics, the reader is referred to Villani's book [107].

The earth mover's distance has been studied in many geometric contexts. Agarwal et al. [2] give both exact and approximation algorithms for the case where both sets are points under some L_p metric. When both sets are weighted points, Khesin at al. [73] give two algorithms running in $O(n\varepsilon^{-O(d)} \log(\Lambda)^{O(d)} \log n \log(1/\rho))$ and $O(n\varepsilon^{-O(d)} \log (U^{O(d)} \log(n/\rho)^2)$ time that compute a $(1+\varepsilon)$-approximation with probability at least $1-\rho$, where d is the dimension, Λ the aspect ratio of the input, and U the total mass. However, their algorithm assumes that the point weights are integers, whereas our weights can be arbitrary real numbers, as they correspond to lengths and areas. This result was improved by Fox and Lu [50]. They used a similar method to obtain, with high probability, a $(1+\varepsilon)$-approximation in $O(n\varepsilon^{-O(d)} \log^{O(d)} n)$ time. The EMD was also studied when the input sets may be transformed: Cabello et al. [35] present algorithms that, given two weighted point sets of n and m points in $\mathbb{R}^2$, compute a $(1+\varepsilon)$-approximation of a translation that minimises the EMD, and a $(2+\varepsilon)$-approximation of a rigid motion that minimises the EMD. These algorithms run in $O((n^2/\varepsilon^4) \log^2 n)$ and $O((n^3 m/\varepsilon^4) \log^2 n)$ time, respectively.

For continuous distributions, rather than discrete point sets, many numerical algorithms are known (see e.g. De Goes et al. [53], Lavenant et al. [83], Mérigot [87, 88] and Solomon et al. [101]). For the case where one set consists of weighted points and the other is a bounded set $C \subset \mathbb{R}^d$, Geiß et al. [52] give a geometric proof that there exists an additively weighted Voronoi diagram such that transporting mass from each point p to the part of C contained in its Voronoi cell is optimal. The weights of this Voronoi diagram can be determined numerically.

De Goes et al. [54] discuss a problem similar to our own, but in the context of the reconstruction and simplification of 2D shapes. Given a set of points, they want to reconstruct a simplicial complex of a given number of vertices that closely represents the shape of the point set. They start with computing the Delaunay triangulation of the point set, then iteratively collapse the edge that minimises the increase in the EMD between the point set and the triangulation. They use a variant of the EMD in which the cost is proportional to the square of the distance (2-Wasserstein distance). This allows them to calculate this variant of the EMD between a given set of points and a given edge of the triangulation exactly, as the squared distance can be decomposed into a normal and a tangential component. However, they determine the assignment of points to edges heuristically. In this work, we show how to obtain a $(1+\varepsilon)$-approximation to the true optimal solution.

2.3 Preliminaries

We are given a set of points $P = \{p_1, \ldots, p_n\}$ in the plane with weights $|p_i|$ and a set of geometric objects $S = \{s_1, \ldots, s_m\}$, with lengths, areas or volumes $|s_j|$. It is given that $\sum |p_i| = \sum |s_i|$. We assume the mass associated with an object s_i is distributed uniformly over the object, and that all objects have the same mass density. For convenience, and without loss of generality, we scale the input such that the total mass in either set is one. We want to compute a "transport plan" of mass from P to S that minimises the cost according to the earth mover's distance. We define for each pair $(p_i, s_j) \in P \times S$ a function $\eta_{i,j}(x, y)$, that describes the density of mass being moved from p_i to the point $(x, y) \in s_j$. All these functions together describe the function η used in the definition of $d_e(A, B)$. Such a set of functions needs to satisfy the following conditions to be a valid transport plan:

$$\begin{aligned} \forall i, j: \quad & 0 \leq \eta_{i,j}(x, y) \leq 1 \\ \forall i: \quad & \sum_{j=1}^{m} \int_{s_j} \eta_{i,j}(x, y) \ \mathrm{d}t = |p_i| \\ \forall j, (x, y) \in s_j: \quad & \sum_{i=1}^{n} \eta_{i,j}(x, y) = 1 \end{aligned}$$

We can then define the cost of a given transport plan η as

$$|\eta| = \sum_{i=1}^{n} \sum_{j=1}^{m} \int_{s_j} \eta_{i,j}(x, y) \cdot d(p_i, (x, y)) \ \mathrm{d}t$$

where $d(\cdot, \cdot)$ is any metric. Our problem is to find a transport plan η^* with minimal cost.

In the following section, we give an exact algorithm to calculate an optimal transport plan between a set of weighted points and line segments when $d(\cdot, \cdot)$ is the L_1 metric. However, the approach we use does not seem to generalise to Euclidean distances, objects with areas, or even two sets of segments. This motivated us to look towards approximation algorithms for more general versions of the EMD problem. In the rest of this chapter, we describe approximation algorithms and only consider the case where $d(\cdot, \cdot)$ is the L_2 metric.

2.4 Points to segments under the L_1 metric

When S is a set of line segments and distances are measured by the L_1 metric, we can solve the problem exactly by a convex quadratic program. We first subdivide all segments on the x- and y-coordinates of the points; call the set of subdivided segments $Q = \{q_1, \ldots, q_k\}$. Note that the horizontal and vertical strip induced by

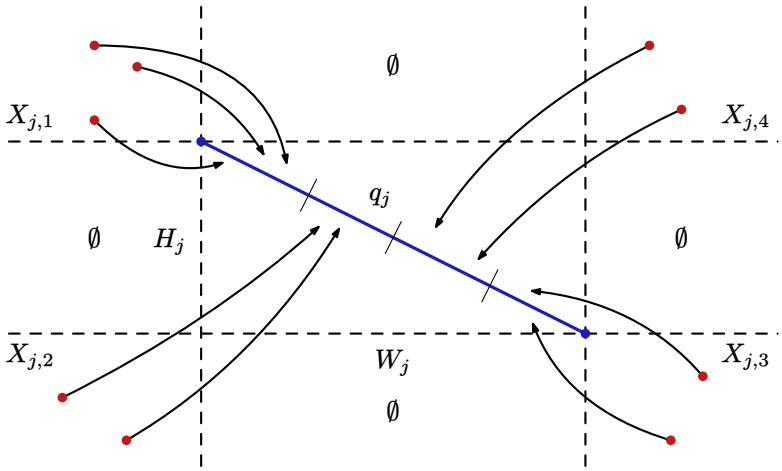


Figure 2.1: The regions of points for one subsegment q_j, and the parts of the segment they assign their mass to.

each segment is now empty of points, while the axis-aligned quadrants starting at each corner of its bounding box may contain points; see Figure 2.1 for an illustration. Let Q_1 be the set of segments in Q with slope between -1 and 1, and let Q_2 be $Q \setminus Q_1$.

We can now label the quadrants of points for each segment q_j: let $X_{j,1}$ and $X_{j,2}$ be the quadrants to the left of q_j, with $X_{j,1}$ being the quadrant starting at the leftmost endpoint of q_j, and $X_{j,2}$ being the other. Similarly, let $X_{j,3}$ be the quadrant starting at the rightmost endpoint of q_j, and let $X_{j,4}$ be the other quadrant on the right. In case of a horizontal or vertical segment, $X_{j,2}$ and $X_{j,4}$ are simply merged into $X_{j,1}$ and $X_{j,3}$, and it does not matter if $X_{j,1}$ is the top or bottom quadrant.

For all points in P, the L_1 distance to any point on the segment q_j is the same as the distance via one of the corners of the axis-aligned bounding box of q_j. Therefore, for each quadrant, we can separately consider the cost to reach the bounding box of q_j with a certain amount of mass, and the cost to spread that mass out over the segment. Furthermore, the order in which a segment receives mass from the different quadrants in an optimal solution is fixed depending on its slope, see Figure 2.1. A simple swapping argument shows that the cost of an assignment not following this order can be decreased by making it follow the order.

Let $u_{i,j}$ be the variable representing the amount of mass moved from p_i to q_j, let $d_{i,j}$ be the precomputed distance from p_i to the bounding box of q_j, let W_j and H_j be the width and height of the bounding box of q_j, and let w_j and h_j be constants such that $w_j \cdot \ell$ is the absolute difference in x-coordinate when moving a distance of ℓ along segment q_j, and $h_j \cdot \ell$ is the absolute difference in y-coordinate. Writing $\sum_{p_i \in X_{j,1}} u_{i,j}$ as $x_{j,1}$ for convenience, for a given segment $q_j \in Q_1$ we can write the

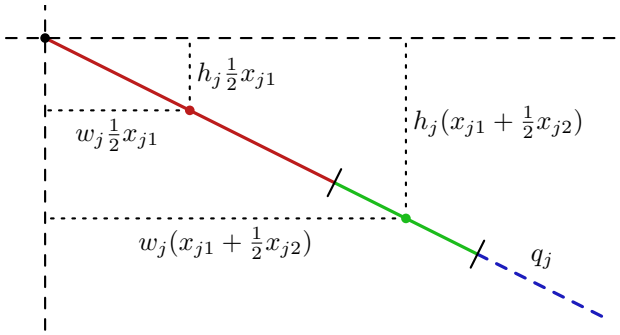


Figure 2.2: The calculations of the distances to the midpoints of the regions to which the mass from $X_{j,1}$ and $X_{j,2}$ will be assigned for a segment with slope between -1 and 1.

cost $c_{j,k}$ for moving the mass from the corner of the region $X_{j,k}$ to the segment as follows:

$$\begin{aligned}
c_{j,1} &= \tfrac{1}{2} x_{j,1}^2 (w_j + h_j) \\
c_{j,2} &= x_{j,2}(w_j x_{j,1} + w_j \tfrac{1}{2} x_{j,2} + (H_j - h_j x_{j,1} - h_j \tfrac{1}{2} x_{j,2})) \\
&= x_{j,2}((w_j - h_j)(x_{j,1} + \tfrac{1}{2} x_{j,2}) + H_j) \\
c_{j,3} &= \tfrac{1}{2} x_{j,3}^2 (w_j + h_j) \\
c_{j,4} &= x_{j,4}(w_j x_{j,3} + w_j \tfrac{1}{2} x_{j,4} + (H_j - h_j x_{j,3} - h_j \tfrac{1}{2} x_{j,4})) \\
&= x_{j,4}((w_j - h_j)(x_{j,3} + \tfrac{1}{2} x_{j,4}) + H_j)
\end{aligned}$$

Here we use the fact that under the L_1 distance, the cost of sending mass to some connected region of a segment is the same as the cost of sending everything to the midpoint of the connected region; see Figure 2.2 for an illustration of the calculations of the distances to these midpoints. Note that we omit the cost of sending the mass from the points to the corners of the bounding box; this will be accounted for later. Further note that $w_j - h_j$ is always positive here. Symmetrically, the costs $c'_k(q_j)$ for a given segment $q_j \in Q_2$ are as follows:

$$\begin{aligned}
c'_{j,1} &= \tfrac{1}{2} x_{j,1}^2 (w_j + h_j) \\
c'_{j,2} &= x_{j,2}((h_j - w_j)(x_{j,3} + \tfrac{1}{2} x_{j,2}) + W_j) \\
c'_{j,3} &= \tfrac{1}{2} x_{j,3}^2 (w_j + h_j) \\
c'_{j,4} &= x_{j,4}((h_j - w_j)(x_{j,1} + \tfrac{1}{2} x_{j,4}) + W_j)
\end{aligned}$$

Note that $h_j - w_j$ is always positive here. We can now formalise our problem as follows:

$$\eta^* = \arg\min_{\mathbf{u}} \left(\sum_j \sum_i d_{i,j} \cdot u_{i,j} \right) + \sum_k \left(\sum_{q_j \in Q_1} c_{j,k} + \sum_{q_j \in Q_2} c'_{j,k} \right)$$

$$\begin{aligned} \text{subject to} \quad & u_{i,j} \geq 0 && \forall i, j \\ & \sum_j u_{i,j} = |p_i| && \forall i \\ & \sum_i u_{i,j} = |q_j| && \forall j \end{aligned}$$

Since all $d_{i,j}$, $c_{j,k}$ and $c'_{j,k}$ are non-negative, this is a sum of convex quadratic functions, giving a quadratic program with a convex objective function.

Theorem 2.1. *Let P be a set of n weighted points and S be a set of m line segments with equal total weight. It is possible to construct an exact optimal transport plan between P and S under the L_1 metric by solving a convex quadratic program.*

When all the weights in our objective function and constraints are integers, a convex quadratic program can be solved in weakly polynomial time, see e.g. [55, 79, 91]. In our case, some of the weights may be real numbers. In particular, w_j and h_j may be square roots of rational numbers. It is not clear if such a program can be solved in polynomial time.

As square roots appear in many geometric settings, we are typically happy to assume a model of computation in which we can perform elementary operations on arbitrary real numbers in constant time. However, even in such a model of computation, the typical methods (cited above) for solving convex quadratic programs in polynomial time may fail. These methods generally rely on approximately solving a series of quadratic programs with increasing precision, and then argue that when the precision is high enough, the approximate solution can be rounded to the globally optimal solution. The argument that such rounding works eventually relies on the input being integral.

This problem can be addressed in several ways. First, we can employ different methods for solving the quadratic program, such as the simplex algorithm. This method takes exponential time in the worst case, but has been shown to be polynomial in practice through smoothed analysis [41]. Second, we can forego an exact algorithm and obtain a $(1 + \varepsilon)$-approximation by simply rounding the square roots in our program with enough precision. Given the value of ε, it suffices to simply approximate the values such that the ratio of the rounded value to the original is at most $(1 + \varepsilon)$. Third, we could apply the L_1 metric not only to distances, but also to the length of each segment. If we define the length of a segment to be equal to the L_1 distance between its endpoints, the equations for $c_{j,k}$ and $c'_{j,k}$ simplify significantly, and the square

roots disappear. Our entire program can then be made to have integer coefficients by simply requiring all points in P and endpoints of segments of S to lie on the integer lattice. Note that this solution also applies if we restrict ourselves to classes of segments for which no square roots show up in our program, such as segments that are axis-aligned.

It may seem that our exact algorithm only runs in provably polynomial time in quite restricted cases. However, it is worth noting here that it is not clear that an exact algorithm for the L_2 case even exists, let alone one that runs in polynomial time.

2.5 Points to segments

We now describe a polynomial-time algorithm that finds a transport plan with a cost that is at most $1 + \varepsilon$ times the cost of the optimal transport plan when one set consists of points with total weight one and the other of line segments with total length one. The main idea is to reduce our instance to a transport problem on two weighted sets of points. Our strategy is as follows: we subdivide each segment such that for each subsegment s' the ratio of the distance to the closest and furthest point on s' for every $p_i \in P$ is at most $1 + \delta$ for some appropriate choice of $\delta \in O(\varepsilon)$. We then approximate a minimum cost flow problem on a bipartite graph between P and the subdivided segments, where the cost of any edge is equal to the shortest distance between a point and a subsegment. Finally, we use the solution to this flow problem to build a discrete transport plan. For an appropriate choice of δ, this gives a $(1 + \varepsilon)$-approximation.

The naive approach to subdividing the segments would be to make all the pieces some equal, appropriately small length. However, we can reduce the number of subsegments required by subdividing the segments as follows[1]. We repeatedly perform the following procedure for each subsegment. If there exists a point in P such that the entire subsegment lies within distance δ/nm of that point, do nothing. Otherwise, if there is a point in P for which the ratio of the longest and shortest distance between that point and the current subsegment is more than $1 + \delta$, cut the subsegment in half. Call the resulting set of subsegments Q; see Figure 2.3 for an example.

We now define a complete bipartite graph $G = (P \cup Q, P \times Q)$, with edges between each point-subsegment pair (note that this graph is used for analysis only; our algorithm does not construct it). The cost of each edge will simply be the shortest distance between the point and segment it connects. A solution to a flow problem in G can be transformed into a transport plan by assigning a piece of subsegment to a point with length equal to the amount of flow along the corresponding edge. We will show that the EMD between P and S is approximated by the cost of any transport plan derived from a minimum cost flow in G.

First note the following general lower bound on the cost of an optimal solution:

[1]This reduces the total number of subsegments required from $O(nm/\varepsilon^2)$ to $O(\frac{nm}{\varepsilon} \log \frac{1}{\varepsilon})$.

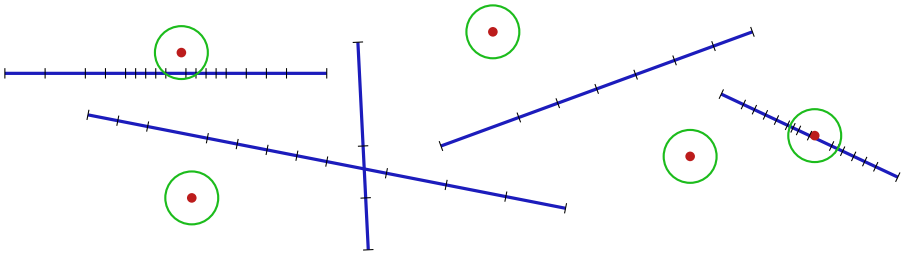


Figure 2.3: An example subdivision of a set of segments. Small perpendicular line segments delimit the generated subsegments. A green circle denotes the distance of δ/nm from each point. Note that ε is set to a very large value here for the clarity of the resulting image.

Lemma 2.2. *The earth mover's distance $|\eta^*|$ between P and Q is bounded from below by the cost $|\mathcal{W}|$ of a minimum cost flow $\mathcal{W}$ in G.*

Proof. Consider any transport plan η^* that minimises the earth mover's distance. If, for each point p_i that moves mass to some segment q_j, we modify the transport plan such that the mass is moved only to the point on q_j closest to p_i, we obtain a plan λ with cost $|\lambda| \leq |\eta^*|$. Such a plan is a solution to a flow problem in G, as it moves all available mass. It follows that the cost $|\mathcal{W}|$ of a minimum cost flow $\mathcal{W}$ in G satisfies $|\mathcal{W}| \leq |\eta^*|$. □

We also note the following lower bound on the value of $|\mathcal{W}|$:

Lemma 2.3. $|\mathcal{W}| \geq \dfrac{\delta - 2\delta^2 - 2\delta^3}{nm}$.

Proof. For a given point-segment pair $(p, s) \in P \times S$, consider the segments in Q derived from s that have a point within distance δ/nm of p. By construction, such a segment has its furthest point at distance at most $(1+\delta) \cdot \delta/nm = \delta/nm + \delta^2/nm$ to p. Therefore, the total length of these segments is at most $2(\delta/nm + \delta^2/nm)$ for a given p and s. Over all point-segment pairs, this gives a total length of at most $2\delta + 2\delta^2$. This means the total length of segments in Q with distance to the closest point in P at least δ/nm is at least $1 - 2\delta - 2\delta^2$. The cost of a minimum flow in $\mathcal{W}$ is therefore at least $(1 - 2\delta - 2\delta^2) \cdot \delta/nm = (\delta - 2\delta^2 - 2\delta^3)/nm$. □

We calculate a transport plan η between P and Q as follows. First, we approximate each segment $q \in Q$ by a point somewhere on that segment with weight equal to the length of q; call this set of points T. We obtain η by calculating an optimal transport plan ν between P and T, and then spreading the mass sent to each point $t \in T$ evenly over the segment in Q that point was derived from. We now bound the cost of η in terms of $|\mathcal{W}|$:

Lemma 2.4. $|\mathcal{W}| \leq |\eta| \leq (1+\delta)^2|\mathcal{W}| + \frac{4\delta^2}{nm} + \frac{2\delta^3}{nm}$.

Proof. We first bound the cost of ν. In $\mathcal{W}$, we measured all the distances to the closest point on each subsegment. Imagine that we picked all the points in T to be the furthest point on the subsegment. For the subsegments with furthest distance to a point in P of at least δ/nm, the ratio of these distances is at most $1+\delta$ by construction. We can therefore bound all the parts of ν where the furthest distance to a point in P is at least δ/nm by $(1+\delta)|\mathcal{W}|$. The total mass being moved over distance at most δ/nm in ν is at most 2δ, giving a cost of at most $2\delta^2/nm$. The total cost of ν is therefore at most $(1+\delta)|\mathcal{W}| + 2\delta^2/nm$.

Now consider the extra cost incurred when transforming ν into η by spreading the mass out evenly over all the segments. We can use the same argument as before: for the parts of ν with distance to a point in P of at least δ/nm, the cost increases by a factor of at most $1+\delta$, and the total cost of the part within distance δ/nm is at most $2\delta^2/nm$. We can therefore bound the cost of η by $(1+\delta)|\nu| + 2\delta^2/nm$.

We now obtain the upper bound stated in the lemma by plugging the bound on ν into the bound on η. The lower bound follows directly from the fact that none of the distances in η are smaller than the distances between the same objects in $\mathcal{W}$. □

We now show that $|\eta|$ approximates $|\eta^*|$.

Theorem 2.5. *$|\eta|$ is a $(1+17\delta)$-approximation to the earth mover's distance $|\eta^*|$ between P and S for $0 < \delta \leq \frac{1}{4}$.*

Proof. By Lemma 2.4, we know that

$$|\eta| \leq (1+\delta)^2|\mathcal{W}| + \frac{4\delta^2}{nm} + \frac{2\delta^3}{nm}$$

$|\mathcal{W}|$ is also a lower bound on $|\eta|$; the ratio between the upper and lower bound is

$$\frac{(1+\delta)^2|\mathcal{W}| + \frac{4\delta^2}{nm} + \frac{2\delta^3}{nm}}{|\mathcal{W}|}$$

This ratio is the largest for small values of $|\mathcal{W}|$, so we plug in the lower bound from Lemma 2.3:

$$\begin{aligned} & \frac{(1+\delta)^2 \cdot \frac{\delta - 2\delta^2 - 2\delta^3}{nm} + \frac{4\delta^2}{nm} + \frac{2\delta^3}{nm}}{\frac{\delta - 2\delta^2 - 2\delta^3}{nm}} \\ = & \frac{1 + 4\delta - 3\delta^2 - 6\delta^3 - 2\delta^4}{1 - 2\delta - 2\delta^2} \\ = & 1 + \delta + \frac{5\delta + \delta^2 - 4\delta^3 - 2\delta^4}{1 - 2\delta - 2\delta^2} \end{aligned}$$

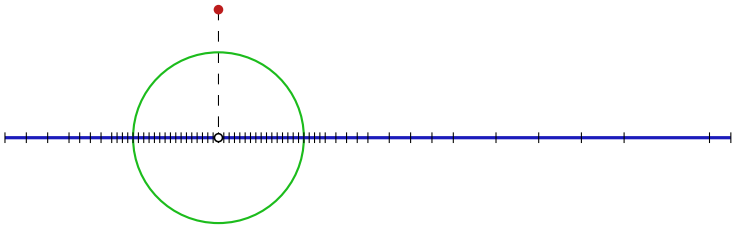


Figure 2.4: An example of a single $R_{i,j}$. Small perpendicular segments delimit the generated subsegments. The green circle denotes the distance of δ/nm from the projected point.

$$\begin{aligned} &\leq 1 + \delta + \frac{6\delta}{1 - 2\delta - 2\delta^2} \\ &\leq 1 + 17\delta && \text{(assuming } \delta \leq \tfrac{1}{4}\text{)} \end{aligned}$$

As $|\mathcal{W}|$ is also a lower bound for $|\eta^*|$ (Lemma 2.2), and η can obviously not have lower cost than the optimal transport plan, this gives a $(1 + 17\delta)$-approximation. □

Setting $\delta = \varepsilon/17$ gives a $(1 + \varepsilon)$-approximation. Note that the bound on δ is not restrictive: for any constant ε that would require a larger value of δ, we can simply use the value $1/4$ at the cost of a constant factor in the running time of our algorithm.

2.5.1 Running time analysis

We now analyse the number of subsegments in Q. We will count the number of subsegments in a different subdivision of S, and then show that Q has at most a constant factor more subsegments. The alternative subdivision of each s_j will be as follows: project each p_i onto the supporting line of s_j, call this point $p_{i,j}$. We construct the one-dimensional Voronoi diagram of all $p_{i,j}$ along the supporting line of s_j; let $s_{i,j}$ be the part of s_j inside the Voronoi cell of $p_{i,j}$. From each $p_{i,j}$, we subdivide $s_{i,j}$ into both directions. Up to a distance of δ/nm, we make subsegments of size δ^2/nm. Moving outward, we double the size of the subsegments whenever their ratio of distances to $p_{i,j}$ would still be below $1 + \delta$. Let $R_{i,j}$ be the resulting subdivision; see Figure 2.4 for an example.

Lemma 2.6. *$R = \bigcup R_{i,j}$ has $O\left(\frac{nm}{\delta} \log \frac{1}{\delta}\right)$ subsegments.*

Proof. We define $\beta = \frac{\delta}{nm}$ and $\gamma = \frac{\delta^2}{nm}$. In the following, we analyse only the case where $p_{i,j}$ is on $s_{i,j}$; if it lies outside, the number of subsegments will be smaller, as the size of the subsegments increases with distance. The length covered as we add

subsegments on $s_{i,j}$ can be written as

$$\beta + 2\sum_{i=0}^{k} \alpha_i 2^i \gamma$$

where k is the number of times we double the size of the subsegments, and α_i is the number of subsegments with a size that has been doubled i times. The number of subsegments can then be calculated by finding the values of k and α_i. We start with α_0, which can be found by considering the distance at which the next cell could be double the size:

$$\begin{aligned}
\frac{\beta + \alpha_0\gamma + 2\gamma}{\beta + \alpha_0\gamma} &\leq 1 + \delta \\
\frac{2\gamma}{\beta + \alpha_0\gamma} &\leq \delta \\
\alpha_0 &\geq \frac{2\gamma - \delta\beta}{\delta\gamma} = \frac{2}{\delta} - \frac{\beta}{\gamma} = \frac{1}{\delta} \\
\alpha_0 &\geq \frac{1}{\delta}
\end{aligned}$$

Per the procedure described above, we double the size of the subsegments as soon as this is allowed. This corresponds to taking the values of α_i as small as possible, so we take $\alpha_0 = \frac{1}{\delta}$. Next, we can show by induction that all α_i are equal:

$$\text{IH: } \alpha_j = \alpha^* = \frac{1}{\delta} \text{ for } j < i.$$

$$\begin{aligned}
\frac{\beta + 2^{i+1}\gamma + \sum_{j=0}^{i} \alpha_j 2^j \gamma}{\beta + \sum_{j=0}^{i} \alpha_j 2^j \gamma} &= 1 + \delta \\
\frac{2^{i+1}\gamma}{\beta + \sum_{j=0}^{i} \alpha_j 2^j \gamma} &= \delta \\
2^{i+1}\gamma &= \delta\beta + \delta\sum_{j=0}^{i} \alpha_j 2^j \gamma \\
&= \delta\beta + \delta\alpha_i 2^i \gamma + \delta\sum_{j=0}^{i-1} \alpha_j 2^j \gamma \\
2^{i+1} &= 1 + \delta\alpha_i 2^i + \delta\alpha^*(2^i - 1) \\
\alpha_i &= \frac{2^{i+1}}{\delta 2^i} - \frac{1}{\delta 2^i} - \frac{\delta\alpha^*(2^i - 1)}{\delta 2^i}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{2}{\delta} - \frac{1}{\delta 2^i} - \frac{2^i - 1}{\delta 2^i} \\
&= \frac{2}{\delta} - \frac{1}{\delta} \\
&= \frac{1}{\delta}
\end{aligned}
$$

Knowing that all α_i are equal to $\frac{1}{\delta}$, we can determine the value of k:

$$
\begin{aligned}
\beta + \sum_{i=0}^{k} \frac{1}{\delta} 2^i \gamma &\geq |s_{i,j}| \\
\beta + \frac{1}{\delta}\gamma(2^{k+1} - 1) &\geq |s_{i,j}| \\
\beta + \beta(2^{k+1} - 1) &\geq |s_{i,j}| \\
2^{k+1} &\geq \frac{|s_{i,j}|}{\beta} \\
k &\geq \log \frac{|s_{i,j}|}{\beta} - 1
\end{aligned}
$$

This gives a total number of subsegments of $O(\frac{1}{\delta} \log \frac{|s_{i,j}|}{\beta})$ for each point-segment pair. The sum over all pairs is largest when all $|s_{i,j}|$ are equal, i.e. $1/nm$. This gives us a total number of subsegments for all pairs of $O\Big(\frac{nm}{\delta} \log \frac{1/nm}{\beta}\Big) = O\big(\frac{nm}{\delta} \log \frac{1}{\delta}\big)$. □

Lemma 2.7. *The set Q has $O\left(\frac{nm}{\delta} \log \frac{1}{\delta}\right)$ subsegments.*

Proof. Consider any subsegment $r \in R$. Any subsegment $q \in Q$ that overlaps with r has $|q| \geq |r|/4$: otherwise q was subdivided unnecessarily. As the subsegments in Q are disjoint, it follows that r can overlap with at most 5 subsegments in Q. As such, Q contains at most 5 times more subsegments than R, which, by Lemma 2.6, is $O\big(\frac{nm}{\delta} \log \frac{1}{\delta}\big)$. □

Putting everything together, we obtain the following result:

Theorem 2.8. *Let P be a set of n weighted points and S be a set of m line segments with equal total weight, let $|\eta^*|$ be the cost of an optimal transport plan between them, and let δ be any constant > 0. Given an algorithm that constructs a $(1+\delta)$-approximation between weighted sets of k points in $f_\delta(k)$ time, we can construct a transport plan between P and S with cost $\leq (1 + 25\delta)|\eta^*|$ in $O\big(f_\delta\big(\frac{nm}{\delta} \log\big(\frac{1}{\delta}\big)\big)\big)$ time.*

Proof. In Theorem 2.5, we prove that an optimal transport plan ν between P and T approximates $|\eta^*|$. However, we may be able to compute a $(1+\delta)$-approximation to ν faster than we are able to compute it exactly. It remains to be shown that this approximation also suffices.

Plugging in a $(1+\delta)$-approximation to $|\nu|$, rather than the exact value, we obtain the ratio

$$\frac{(1+\delta)^3|\mathcal{W}| + \frac{4\delta^2}{nm} + \frac{4\delta^3}{nm} + \frac{2\delta^4}{nm}}{|\mathcal{W}|}$$

Following the same strategy as in the proof of Theorem 2.5, we derive the approximation ratio as follows:

$$\begin{aligned}
& \frac{(1+\delta)^3 \cdot (\delta - 2\delta^2 - 2\delta^3) + 4\delta^2 + 4\delta^3 + 2\delta^4}{\delta - 2\delta^2 - 2\delta^3} \\
= \; & \frac{(1+3\delta+3\delta^2+\delta^3)(1-2\delta-2\delta^2) + 4\delta + 4\delta^2 + 2\delta^3}{1-2\delta-2\delta^2} \\
= \; & \frac{1+5\delta-\delta^2-9\delta^3-8\delta^4-2\delta^5}{1-2\delta-2\delta^2} \\
= \; & 1 + \frac{7\delta+\delta^2-9\delta^3-8\delta^4-2\delta^5}{1-2\delta-2\delta^2} \\
= \; & 1 + \delta + \frac{6\delta+3\delta^2-7\delta^3-8\delta^4-2\delta^5}{1-2\delta-2\delta^2} \\
\leq \; & 1 + \delta + \frac{9\delta}{1-2\delta-2\delta^2} \\
\leq \; & 1 + 25\delta \qquad \text{(assuming } \delta \leq \tfrac{1}{4}\text{)}
\end{aligned}$$

As such, using an approximation of ν still gives us an approximation of η^*, albeit with a somewhat worse dependency on δ. □

To our knowledge, the current fastest algorithm to calculate a $(1+\delta)$-approximation to ν is that by Fox and Lu [50], which runs in $O(N\delta^{-O(1)}\operatorname{polylog} N)$ time, where N is the size of the input. Setting $\delta = \varepsilon/25$, this gives the following corollary to the previous theorem:

Corollary 2.9. *For any constant $\varepsilon > 0$, a transport plan between P and S with cost $\leq (1+\varepsilon)|\eta^*|$ can be constructed in $O\big(\frac{nm}{\varepsilon^c}\operatorname{polylog}\big(\frac{nm}{\varepsilon}\big)\big)$ time with high probability.*

2.6 Points to triangles

We consider the case where P is a set of weighted points with total weight one and S is a set of m triangles with total area one. We denote the longest edge of any triangle by Δ. Our strategy is similar to before: we subdivide the triangles such that for

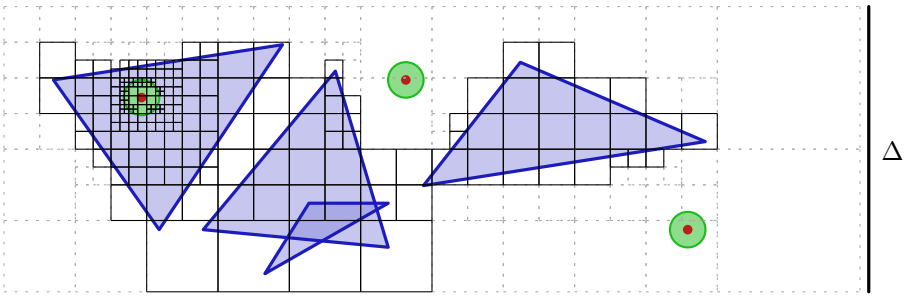


Figure 2.5: An example subdivision of a set of triangles. Each cell records the total area of triangles it intersects. Cells that are part of Q are shown in black; empty cells are shown in grey dashed lines. A green disk denotes the distance of $\delta/\sqrt{nm}$ from each point. Note that ε is set to a very large value here for the clarity of the resulting image.

each subregion, the ratio between its shortest and longest distance to each point is at most $1+\delta$ for some appropriate choice of $\delta \in O(\varepsilon)$. We then show that a solution based on an optimal transport plan between P and some points inside the subregions approximates an optimal solution.

We first overlay a uniform grid onto our triangles with grid cells of size $\Delta \times \Delta$. We can identify the cells of this grid that contain a triangle in $O(m \log m)$ time using point-location in a compressed quadtree where the smallest cell size is $\Delta \times \Delta$ [64]. As each triangle can intersect at most four cells, the total size of this set of cells is $O(m)$. We now recursively subdivide each cell as follows: if there is a point in P such that the whole cell is within distance $\delta/\sqrt{nm}$ of it, we stop; otherwise, if for any point the ratio of distances to the furthest and closest point in this cell is more than $1+\delta$, we subdivide this cell into four cells of one quarter the area. If the ratio holds for all points, we stop. Call the resulting set of cells Q; see Figure 2.5 for an example.

During each subdivision, we keep track of the total area of triangles contained inside that cell. We can then once again build a complete bipartite graph $G = (P \cup Q, P \times Q)$, with the capacity of each vertex set to the weight of the corresponding point or the total area of triangles contained in the corresponding cell, and the weight of each edge equal to the shortest distance between the point and the cell it connects. The cost of a minimum cost flow $\mathcal{W}$ is now once again a lower bound to the EMD, exactly as in Lemma 2.2. In an analogous way to Lemma 2.3, we obtain a lower bound on the cost of $\mathcal{W}$:

Lemma 2.10. $|\mathcal{W}| \geq \dfrac{\delta}{\sqrt{nm}} - \dfrac{\pi\delta^3 + 2\pi\delta^4 + \pi\delta^5}{\sqrt{nm}}$.

Proof. For a given point-triangle pair $(p, s) \in P \times S$, consider the cells in Q intersecting s that have a point within distance $\delta/\sqrt{nm}$ of p. By construction, such a cell has its

furthest point at distance at most $(1+\delta)\cdot\delta/\sqrt{nm} = \delta/\sqrt{nm} + \delta^2/\sqrt{nm}$. Therefore, the total area of these cells is at most $\pi(\delta/\sqrt{nm} + \delta^2/\sqrt{nm})^2 = \pi(\delta^2 + 2\delta^3 + \delta^4)/nm$. Over all point-triangle pairs, this gives a total area of at most $\pi(\delta^2 + 2\delta^3 + \delta^4)$. This leaves $1 - \pi(\delta^2 + 2\delta^3 + \delta^4)$ with distance at least $\delta/\sqrt{nm}$ in $\mathcal{W}$. The cost is therefore at least $(1 - \pi(\delta^2 + 2\delta^3 + \delta^2))\cdot\delta/\sqrt{nm} = \delta/\sqrt{nm} - \pi(\delta^3 + 2\delta^4 + \delta^5)/\sqrt{nm}$. □

We now once again approximate $|\mathcal{W}|$ by reducing the flow problem to a transportation problem between two sets of weighted points. Again, we pick any point in each cell $q \in Q$ and give it a weight equal to the area of triangles contained in q; call this set of points T.

Lemma 2.11. $|\mathcal{W}| \leq |\eta| \leq (1+\delta)^2|\mathcal{W}| + \frac{2\pi\delta^3}{\sqrt{nm}} + \frac{\pi\delta^4}{\sqrt{nm}}$

Proof. Let ν be an optimal transport plan between P and T, and let $|\nu|$ be its cost. We can upper bound $|\nu|$ by measuring all distances to the furthest point in each cell. We constructed Q such that the ratio of the closest and furthest distance between any point-cell pair is $1+\delta$ when the furthest distance is at least $\delta/\sqrt{nm}$. We can therefore bound all parts of ν where the distance is at least $\delta/\sqrt{nm}$ by $(1+\delta)|\mathcal{W}|$. The total mass being moved over a distance at most $\delta/\sqrt{nm}$ in ν is at most $\pi\delta^2$, giving a cost of $\pi\delta^3/\sqrt{nm}$. The total cost when measuring to the furthest point is therefore $(1+\delta)|\mathcal{W}| + \pi\delta^3/\sqrt{nm}$.

We now turn ν into a transport plan η between P and Q by spreading the mass sent to each point $t \in T$ out evenly over the parts of the triangles in the cell in Q that t was derived from. By construction, for cells with a distance of at least $\delta/\sqrt{nm}$, this increases the cost by at most a factor $1+\delta$. We can therefore bound the cost of this part of η by $(1+\delta)|\nu|$. The remaining part has a total mass of at most $\pi\delta^2$, giving a cost of $\pi\delta^3/\sqrt{nm}$. The total cost of η is then bound by $(1+\delta)|nu| + \pi\delta^3/\sqrt{nm}$.

Plugging in the bound on $|\nu|$ obtained above, we obtain an upper bound of $(1+\delta)^2|\mathcal{W}| + 2\pi\delta^3/\sqrt{nm} + \pi\delta^4/\sqrt{nm}$. The lower bound follows directly from the fact that none of the distance in η are smaller than the distances between the same objects in $\mathcal{W}$. □

Putting this all together, we can show that $|\eta|$ approximates $|\eta^*|$.

Theorem 2.12. *$|\eta|$ is a $(1+9\delta)$-approximation to the earth mover's distance $|\eta^*|$ between P and S for $0 < \delta \leq \frac{1}{2\pi}$.*

Proof. By Lemma 2.11 have that

$$|\eta| \leq (1+\delta)^2|\mathcal{W}| + \frac{2\pi\delta^3}{\sqrt{nm}} + \frac{\pi\delta^4}{\sqrt{nm}}$$

$|\mathcal{W}|$ is also a lower bound on $|\eta|$; the ratio between the upper and lower bound is

$$\frac{(1+\delta)^2|\mathcal{W}| + \frac{2\pi\delta^3}{\sqrt{nm}} + \frac{\pi\delta^4}{\sqrt{nm}}}{|\mathcal{W}|}$$

This ratio is largest for small values of $|\mathcal{W}|$, so we plug in the lower bound from Lemma 2.10:

$$\begin{aligned}
& \frac{(1+\delta)^2|\mathcal{W}| + \frac{2\pi\delta^3}{\sqrt{nm}} + \frac{\pi\delta^4}{\sqrt{nm}}}{|\mathcal{W}|} \\
\leq\ & \frac{(1+\delta)^2\left(\frac{\delta}{\sqrt{nm}} - \frac{\pi\delta^3+2\pi\delta^4+\pi\delta^5}{\sqrt{nm}}\right) + \frac{2\pi\delta^3}{\sqrt{nm}} + \frac{\pi\delta^4}{\sqrt{nm}}}{\frac{\delta}{\sqrt{nm}} - \frac{\pi\delta^3+2\pi\delta^4+\pi\delta^5}{\sqrt{nm}}} \\
\leq\ & \frac{(1+2\delta+\delta^2)(1-\pi\delta-2\pi\delta^2-\pi\delta^3) + 2\pi\delta^2 + \pi\delta^3}{1-\pi\delta-2\pi\delta^2-\pi\delta^3} \\
=\ & \frac{1+2\delta+\delta^2-2\pi\delta^2-5\pi\delta^3-4\pi\delta^4-\pi\delta^5}{1-\pi\delta-2\pi\delta^2-\pi\delta^3} \\
=\ & 1 + \frac{2\delta+\delta^2-\pi\delta-4\pi\delta^3-4\pi\delta^4-\pi\delta^5}{1-\pi\delta-2\pi\delta^2-\pi\delta^3} \\
=\ & 1 + \delta + \frac{\delta+\delta^2-\pi\delta-2\pi\delta^3-3\pi\delta^4-\pi\delta^5}{1-\pi\delta-2\pi\delta^2-\pi\delta^3} \\
<\ & 1 + \delta + \frac{\delta^2}{1-\pi\delta-2\pi\delta^2-\pi\delta^3} \\
\leq\ & 1 + \delta + \frac{\delta^2}{1-\frac{1}{2}-\frac{1}{\pi}-\frac{1}{2\pi^2}} && \text{(assuming } \delta \leq \tfrac{1}{2\pi}\text{)} \\
<\ & 1 + \delta + 8\delta^2 \\
<\ & 1 + 9\delta
\end{aligned}$$

As $|\mathcal{W}|$ is also a lower bound for $|\eta|$ (Lemma 2.2), and η can obviously not have lower cost than the optimal transport plan, this gives a $(1+9\delta)$-approximation. □

Setting $\delta = \varepsilon/9$ gives a $(1+\varepsilon)$-approximation.

2.6.1 Running time analysis

Our analysis will be the same as in Section 2.5.1; we just need to determine the size of Q. We will once again make an alternative subdivision of each $s_j \in S$, count the number of cells in that subdivision, and then argue that $|Q|$ differs by at most a constant factor. Our alternative subdivision is a direct adaptation of the one used in Section 2.5.1 to two dimensions: for each point p_i and triangle s_j, we fill a square with side length $2\delta/\sqrt{nm}$ centred on p_i with cells of size $\delta^2/\sqrt{nm}$. From there, we add rings of cells of side length $\delta^2/\sqrt{nm}$ around the square, until the next full ring could have cells double the size without violating the ratio of $1+\delta$ between the shortest and longest distance to p_i for any cell in the ring. We repeat this process until we have

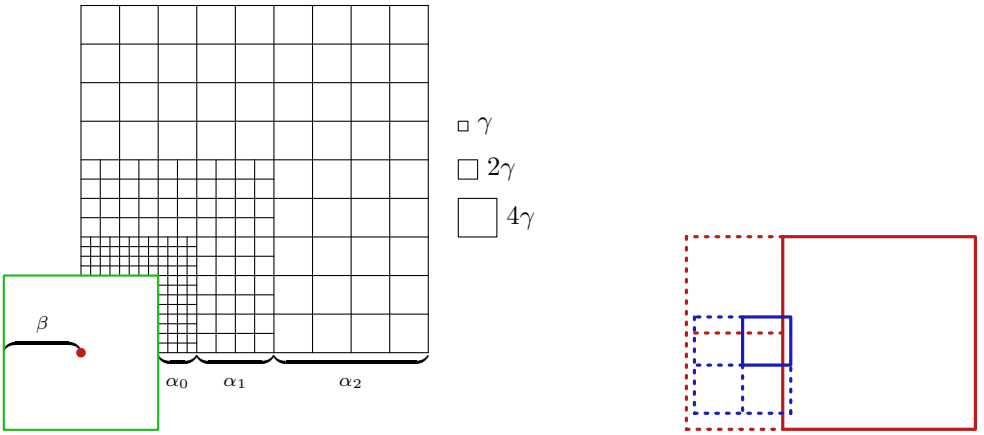


Figure 2.6: On the left, part of one quadrant of the construction of $R_{i,j}$. There are α_i layers of cells of size $2^i\gamma$ before the size is doubled. On the right, an illustration of the argument that a cell of Q (blue) has at least one quarter the edge length of a cell of R that it intersects (red).

covered a square of size $\Delta \times \Delta$. Let $R_{i,j}$ be the resulting set of cells; see Figure 2.6 for an example. The proof is similar to Lemma 2.6.

Lemma 2.13. *$R = \bigcup R_{i,j}$ has $O\left(\frac{nm}{\delta^2} \log \frac{nm\Delta}{\delta}\right)$ cells.*

Proof. We define $\beta = \frac{\delta}{\sqrt{nm}}$ and $\gamma = \frac{\delta^2}{\sqrt{nm}}$. In the following, we only analyse the case where $p_{i,j}$ is inside $s_{i,j}$; if it lies outside, the number of cells will be smaller, as the size of the cells increases with distance. We also analyse the number of cells in one quadrant only; the total number is simply four times as many. See Figure 2.6 for an illustration of a quadrant. The number of cells created as we add rings of cells on $s_{i,j}$ can then be written as

$$\frac{\beta^2}{\gamma^2} + \sum_{i=0}^{k} 2\alpha_i \cdot \frac{\beta + \sum_{j=0}^{i-1} \alpha_j 2^j \gamma}{2^i \gamma} + \alpha_i^2$$

where k is the number of times we double the size of the cells, and α_i is the number of rings containing cells of a size that has been doubled i times. The number of cells can then be calculated by finding the values of k and α_i. We take the values of α_i to be the same as in Lemma 2.6 (i.e. $1/\delta$): along a horizontal or vertical line through p_i these values give the exactly correct distance ratios, and cells not on this line can be made to have the correct ratio through one extra subdivision.

Let (x, y) be the vector from p_i to the closest point on the cell. Assume w.l.o.g. that $0 \leq y \leq x$; the other cases are symmetrical. By construction of our subdivision,

we know that the cell has size at most δx. We will now show that by dividing the cell one extra time (i.e. to a size of $\delta x/2$), the furthest point will have the desired ratio irrespective of the value of y.

$$\begin{aligned}
\frac{\sqrt{(x+\frac{\delta x}{2})^2+(y+\frac{\delta x}{2})^2}}{\sqrt{x^2+y^2}} &\leq 1+\delta \\
\frac{(x+\frac{\delta x}{2})^2+(y+\frac{\delta x}{2})^2}{x^2+y^2} &\leq (1+\delta)^2 \\
\frac{x^2+y^2+\delta x^2+\delta xy+\frac{\delta^2 x^2}{2}}{x^2+y^2} &\leq 1+2\delta+\delta^2 \\
\frac{\delta x^2+\delta xy+\frac{\delta^2 x^2}{2}}{x^2+y^2} &\leq 2\delta+\delta^2 \\
\delta xy &\leq \delta x^2+\frac{\delta^2 x^2}{2}+2\delta y^2+\delta^2 y^2
\end{aligned}$$

As $\delta xy \leq \delta x^2$, and the other terms on the right-hand side are positive, the inequality holds. As such, the construction described can be turned into one where all cells have the desired ratio with one extra subdivision.

Plugging the values of α_i, β, γ into our initial formula, we can obtain the number of cells as a function of k:

$$\begin{aligned}
& \left(\frac{\frac{\delta}{\sqrt{nm}}}{\frac{\delta^2}{\sqrt{nm}}}\right)^2+\sum_{i=0}^{k}\frac{2}{\delta}\cdot\frac{\frac{\delta}{\sqrt{nm}}+\sum_{j=0}^{i-1}\frac{1}{\delta}2^j\frac{\delta^2}{\sqrt{nm}}}{2^i\frac{\delta^2}{\sqrt{nm}}}+\frac{1}{\delta^2} \\
= \quad & \frac{1}{\delta^2}+\frac{k}{\delta^2}+\frac{2}{\delta}\sum_{i=0}^{k}\frac{\frac{\delta}{\sqrt{nm}}+\frac{\delta}{\sqrt{nm}}\sum_{j=0}^{i-1}2^j}{2^i\frac{\delta^2}{\sqrt{nm}}} \\
= \quad & \frac{1}{\delta^2}+\frac{k}{\delta^2}+\frac{2}{\delta}\sum_{i=0}^{k}\frac{\frac{\delta}{\sqrt{nm}}+\frac{\delta}{\sqrt{nm}}(2^i-1)}{2^i\frac{\delta^2}{\sqrt{nm}}} \\
= \quad & \frac{1}{\delta^2}+\frac{k}{\delta^2}+\frac{2}{\delta}\sum_{i=0}^{k}\frac{2^i\frac{\delta}{\sqrt{nm}}}{2^i\frac{\delta^2}{\sqrt{nm}}} \\
= \quad & \frac{1}{\delta^2}+\frac{k}{\delta^2}+\frac{2}{\delta}\sum_{i=0}^{k}\frac{1}{\delta} \\
= \quad & \frac{1}{\delta^2}+\frac{k}{\delta^2}+\frac{2k}{\delta^2} \\
\in \quad & O\left(\frac{k}{\delta^2}\right)
\end{aligned}$$

We can directly calculate the value of k by considering the number of doublings needed to cover a horizontal line segment of length Δ starting at p_i:

$$\begin{aligned}
\frac{\delta}{\sqrt{nm}} + \sum_{i=0}^{k} \frac{1}{\delta} \cdot 2^i \cdot \frac{\delta^2}{\sqrt{nm}} &= \Delta \\
\frac{\delta}{\sqrt{nm}} + \frac{\delta}{\sqrt{nm}} \sum_{i=0}^{k} 2^i &= \Delta \\
1 + \sum_{i=0}^{k} 2^i &= \frac{\sqrt{nm}\Delta}{\delta} \\
2^{k+1} &= \frac{\sqrt{nm}\Delta}{\delta} \\
k &\in O\left(\log \frac{nm\Delta}{\delta}\right)
\end{aligned}$$

This gives a total number of cells of $O\big(\frac{1}{\delta^2} \log \frac{nm\Delta}{\delta}\big)$ per point-triangle pair. Over all pairs, we obtain a total number of cells of $O\big(\frac{nm}{\delta^2} \log \frac{nm\Delta}{\delta}\big)$. □

Lemma 2.14. *The set Q has $O\left(\frac{nm}{\delta^2} \log \frac{nm\Delta}{\delta}\right)$ cells.*

Proof. Consider any cell $r \in R$. Any cell $q \in Q$ that overlaps with r has $|q| \geq |r|/16$: otherwise q was subdivided unnecessarily; see Figure 2.6. As the cells in Q are disjoint, it follows that r can overlap with at most 25 cells in Q. As such, Q contains at most 25 times more cells than R, which, by Lemma 2.13, is $O\big(\frac{nm}{\delta^2} \log \frac{nm\Delta}{\delta}\big)$. □

This leads to the following result:

Theorem 2.15. *Let P be a set of n weighted points and S be a set of m triangles with equal total weight, let Δ be the longest edge length in S after normalising its total area to one, let $|\eta^*|$ be the cost of an optimal transport plan between P and S, and let δ be any constant > 0. Given an algorithm that constructs a $(1+\delta)$-approximation between weighted sets of k points in $f_\delta(k)$ time, we can construct a transport plan between P and S with cost $\leq (1+9\delta)|\eta^*|$ in $O\big(f_\delta\big(\frac{nm}{\delta^2} \operatorname{polylog}\big(\frac{nm\Delta}{\delta}\big)\big)\big)$ time.*

We can again calculate a $(1+\delta)$-approximation to ν in $O(N\delta^{-O(1)} \operatorname{polylog} N)$ time using the algorithm by Fox and Lu [50], giving the following corollary to the previous theorem:

Corollary 2.16. *For any constant $\varepsilon > 0$ and some constant c, a transport plan between P and S with cost $\leq (1+\varepsilon)|\eta^*|$ can be constructed in $O\big(\frac{nm}{\varepsilon^c} \operatorname{polylog}\big(\frac{nm\Delta}{\varepsilon}\big)\big)$ time with high probability.*

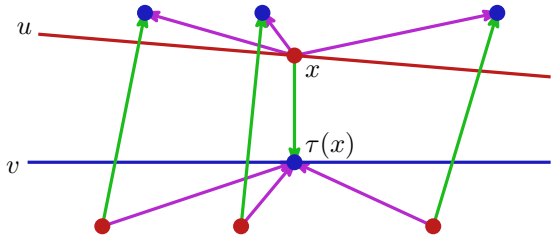


Figure 2.7: Two points, with their mass assignment in an optimal solution shown in purple. We greedily match a point x on u to $\tau(x)$ on v, and obtain the assignment of mass shown in green.

2.7 Segments to segments

In the previous section, we considered the case when one of our two input sets consists of points. We now describe an algorithm to compute the EMD between two sets of line segments. Here, we cannot directly apply our general approach of subdividing: the optimal transport plan may have a cost arbitrarily close to zero. As such, if we disregard everything within some radius of one of the sets, there may be nothing left. We solve this by introducing an additive term into the approximation. The cost of a plan generated by our algorithm is $(1+\varepsilon)|\eta^*| + A$, for some value A depending on ε. This allows us to greedily match parts of the input within a small distance of each other, and then solve the remainder with our previous approach.

Let $P = \{p_1, \ldots, p_n\}$ and $S = \{s_1, \ldots, s_m\}$ be sets of line segments with equal total length. Our algorithm is then as follows. First, we greedily match equal-length pieces of P and S that are within distance δ/nm of each other, until no such pieces remain; we describe this process in more detail later. Let P' and S' be the remaining parts of P and S, respectively. We subdivide P' and S' as before: for every $p \in P'$, if there is an $s \in S'$ such that the ratio between the closest and furthest distance is more than $1+\delta$, cut p in half; after processing P', do the same for S'. Call the resulting sets Q and R. We then choose a point on each $q \in Q$ and $r \in R$, with a weight equal to the length of the subsegment, and solve an optimal transport problem between these two point sets. Our final transport plan is then obtained by spreading the mass moved between any two points evenly over the segments they were chosen on.

We first prove that greedily matching parts of the input within distance δ/nm increases the cost of an optimal solution by at most an additive term. The proof for the approximation algorithm then follows the same structure as in the previous sections. Let η_M be a transport plan between the parts of the input that were greedily matched, in which the longest distance is at most δ/nm, and let η^*_G be an optimal transport plan for the remainder of the input.

Lemma 2.17. *Let u and v be two subsegments with length l of P and S, respectively. If all*

mass from u can be transported to v with distance at most κ, then an optimal transport plan between $P \setminus \{u\}$ and $S \setminus \{v\}$ has cost at most $|\eta^*| + l\kappa$.

Proof. We will construct a transport plan in which u and v are removed, having cost at most $|\eta^*| + l\kappa$. The cost of an optimal solution on the remainder is then not higher.

Let $\eta_s(x,t) : \mathbb{R}^2 \times [0,1] \to \mathbb{R}^2$ be a function describing, for a point $x \in s$, where its mass comes from or goes to (recall that each point sends or receives mass density one), let $d_s(x,t) : \mathbb{R}^2 \times [0,1] \to \mathbb{R}$ be defined as $d(x, \eta_s(x,t))$, and let $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ be a mapping of points on u to points on v such that for all $x \in u$, $d(x, \tau(x)) \leq \kappa$. The cost of the part of η^* involving segments u and v can then be written as

$$c^*(u) = \int_{x \in u} \int d_u(x,t) \ \mathrm{d}t \ \mathrm{d}x$$

$$c^*(v) = \int_{x \in u} \int d_v(\tau(x),t) \ \mathrm{d}t \ \mathrm{d}x$$

We modify η^* by removing u and v, and moving all mass that each point $\tau(x)$ receives in η^* to where x moved it in η^*; see Figure 2.7. We can distribute this mass in any way we like, as the total incoming and outgoing mass is one by definition. This gives a transport plan η with cost

$$\begin{aligned} |\eta| = |\eta^*| &- \left(\int_{x \in u} \int d_u(x,t) \ \mathrm{d}t \ \mathrm{d}x \right) - \left(\int_{x \in u} \int d_v(\tau(x),t) \ \mathrm{d}t \ \mathrm{d}x \right) \\ &+ \left(\int_{x \in u} \int d(\eta_u(x,t), \eta_v(\tau(x),t)) \ \mathrm{d}t \ \mathrm{d}x \right) \end{aligned}$$

By the triangle inequality, $d(\eta_u(x,t), \eta_v(\tau(x),t)) \leq d_v(\tau(x),t) + \kappa + d_u(x,t)$. It follows that

$$\begin{aligned} |\eta| &\leq |\eta^*| - \left(\int_{x \in u} \int d_u(x,t) \ \mathrm{d}t \ \mathrm{d}x \right) - \left(\int_{x \in u} \int d_v(\tau(x),t) \ \mathrm{d}t \ \mathrm{d}x \right) \\ &\quad + \left(\int_{x \in u} \int d_v(\tau(x),t) + \kappa + d_u(x,t) \ \mathrm{d}t \ \mathrm{d}x \right) \\ &= |\eta^*| + l\kappa \end{aligned}$$

□

Note that this bound is tight in the worst case: consider horizontal line segments of unit length with their left endpoints having x-coordinate 0. If we take P to consist of two such segments at $y = 0$ (p_1) and $y = 2$ (p_2), and S to consist of two segments at $y = 1$ (s_1) and $y = 3$ (s_2), the optimal solution would move mass from p_1 to s_1 and from p_2 to s_2, giving a total cost of 2. If we set $\kappa = 1$, we would greedily match p_2 and s_1, giving a total cost of 3, being exactly $l\kappa$ more than the optimal.

We can now bound the costs of η^*_G and η_M.

Lemma 2.18. $|\eta_G^*| \leq |\eta^*| + \frac{\delta}{nm}$.

Proof. Any subsegments of P and S with length l that are greedily matched increase the cost of an optimal solution in the remaining part by at most $\delta l/nm$ (Lemma 2.17). The total length that can be greedily matched is at most one, so the total extra cost is at most δ/nm. □

Lemma 2.19. $|\eta_M| \leq \frac{\delta}{nm}$.

Proof. By construction, the distance over which any mass is transported in η_M is at most δ/nm. The total mass transported is at most one, giving the bound. □

For each segment $p \in P$, we can straightforwardly compute a maximal subset that can be transported over distance at most δ/nm. Consider each segment $s \in S$: the supporting lines of p and s can intersect inside p or s, outside both, or not at all. If they don't intersect (i.e. are parallel), computation of the parts that can be transported within the required distance is trivial. If they intersect outside both, we can find the points on p and s furthest from the intersection point that are within the required distance, then find the largest distance we can move towards the intersection point while staying within the required distance. If they intersect inside one or both of the segments, we split the segments at the intersection point and handle both sides using the case for intersections outside the segments.

Let P' and S' be the parts of P and S that remain after the greedy matching, with $|P'| = |S'| = \ell$. We subdivide P' and S' into Q and R as described above. As before, we define a complete bipartite graph $G = (Q \cup R, Q \times R)$, where the weight of each edge is equal to the shortest distance between the two subsegments it connects, and the capacity of each vertex is equal to the length of the subsegment it represents. Let $\mathcal{W}$ be a minimum cost flow in G; we observe the following lower bound on its cost:

Lemma 2.20. $|\mathcal{W}| \geq \frac{\delta\ell}{nm}$.

Proof. By construction, the distances in G are at least δ/nm. As the total mass moved is ℓ, we obtain the bound stated in the lemma. □

Lemma 2.2 also still applies to the part of the input that remains after greedy matching. We now approximate $|\mathcal{W}|$ by reducing the flow problem to a transportation problem between two weighted point sets. We pick any point on each $q \in Q$ and $r \in R$, and give them weights equal to $|q|$ and $|r|$. Call these sets of points U and V. We can now bound the cost of η_G in terms of η_G^* using the flow problem.

Lemma 2.21. $|\eta_G^*| \leq |\eta_G| \leq (1+\delta)^2 |\eta_G^*|$.

Proof. Let ν be an optimal transport plan between U and V, and let $|\nu|$ be its cost. We can upper bound $|\nu|$ by measuring all distances to the furthest points inside the segments. By construction of Q and R, the ratio of longest to shortest distance is at most $1+\delta$. The cost $|\nu|$ of ν can therefore not be more than $(1+\delta)|\mathcal{W}|$.

We can turn η into a valid transport plan η_G between Q and R by spreading the mass moved to each point in U and V evenly over the segments in Q and R that they were derived from. Again, by construction, the distances increase by a factor of at most $1+\delta$, giving $\eta_G \leq (1+\delta)|\nu|$.

Plugging in the bound on $|\nu|$ obtained above, we obtain an upper bound of $(1+\delta)^2|\mathcal{W}|$. As $|\mathcal{W}| \leq |\eta_G^*|$, we obtain that $|\eta_G| \leq (1+\delta)^2|\eta_G^*|$. The lower bound follows directly from the fact that η_G^* is optimal, and therefore cannot have a cost higher than that of η_G. □

We can then show that, for a transport plan $\eta = \eta_G + \eta_M$, $|\eta|$ approximates $|\eta^*|$:

Theorem 2.22. $|\eta| \leq (1+3\delta)|\eta^*| + \dfrac{5\delta}{nm}$.

Proof. By Lemma 2.21, we know that $|\eta_G| \leq (1+\delta)^2|\eta_G^*|$. As $\delta \leq 1$, $(1+\delta)^2 \leq 1+3\delta$. By Lemma 2.18, we have that $|\eta_G^*| \leq |\eta^*| + (1-\ell)\delta/nm$. Combining the two results, we get that

$$\begin{aligned}
|\eta_G| &\leq (1+3\delta)|\eta_G^*| \\
&\leq (1+3\delta)\left(|\eta^*| + (1-\ell)\frac{\delta}{nm}\right) \\
&\leq (1+3\delta)|\eta^*| + (1-\ell)\frac{\delta+3\delta^2}{nm} \\
&\leq (1+3\delta)|\eta^*| + \frac{4\delta}{nm}
\end{aligned}$$

By Lemma 2.19, $|\eta_M| \leq \delta/nm$. As $|\eta| = |\eta_G| + |\eta_M|$, we obtain the bound stated in the lemma. □

Setting $\delta = \varepsilon/3$ gives a $(1+\varepsilon)$-approximation with an additive term of $5\varepsilon/3nm$

2.7.1 Running time analysis

During the greedy matching, each $p \in P$ may have been cut into m pieces, and each $s \in S$ into n pieces. As such, P' and S' (the parts remaining after greedy matching) both contain $O(nm)$ subsegments. In the worst case, P' and S' are close to each other everywhere, causing them to be subdivided into the smallest possible subsegments. As the minimum distance is δ/nm, and the ratio of the longest and shortest distance between any two subsegments is $1+\delta$, the smallest possible subsegment has size

$\Theta(\frac{\delta^2}{nm})$. Each subsegment of P' and S' may give rise to one extra subsegment in Q and R, as the length may not be exactly divisible by δ/nm. This gives sets Q and R a size of $O(\frac{nm}{\delta^2} + nm) \in O(\frac{nm}{\varepsilon^2})$, leading to the following result:

Theorem 2.23. *Let P and S be sets of n and m line segments in the plane, both having equal total length, let $|\eta^*|$ be the cost of an optimal transport plan between them, and let δ be any constant > 0. Given an algorithm that constructs a $(1+\delta)$-approximation between weighted sets of k points in $f_\delta(k)$ time, we can construct a transport plan between P and S with cost $\leq (1+c'\delta)|\eta^*| + \frac{5\delta}{nm}$ for some constant c' in $O\big(f_\delta\big(\frac{nm}{\delta^2}\big)\big)$ time with high probability.*

We can again calculate a $(1+\delta)$-approximation to ν in $O(N\delta^{-O(1)}\operatorname{polylog} N)$ time using the algorithm by Fox and Lu [50], giving the following corollary to the previous theorem:

Corollary 2.24. *For any constant $\varepsilon > 0$, a transport plan between P and S with cost $\leq (1+\varepsilon)|\eta^*| + O\big(\frac{\varepsilon}{nm}\big)$ can be constructed in $O\big(\frac{nm}{\varepsilon^c}\operatorname{polylog}\big(\frac{nm}{\varepsilon}\big)\big)$ time with high probability.*

2.8 Triangles to triangles

We consider the case where P and S are both sets of triangles with total area one and longest edge length Δ. The algorithm is completely analogous to the one for transport between sets of segments: we greedily match parts of the input within a certain distance, subdivide the remainder and approximate the optimal transport plan by reduction to a minimum cost flow. As the setup and proofs are exactly the same as in the previous section (just substitute the integrals over segments with integrals over area), this is omitted. All we need is an algorithm that can greedily match parts of the input within a given distance.

We do this greedy matching as follows. We can first remove the parts where P and S overlap: they have cost zero. We then overlay a grid with cells of size $\delta/(2\sqrt{nm})$ onto our input, and keep only the cells that contain an edge or are adjacent to one that does (the other cells already have the desired clearance from cells containing triangles from the other set). Inside each cell, we record the total area of triangles from P and S that lie inside it separately. For parts of P and S that lie inside the same cell, we match as much as possible, resulting in a grid where each cell only contains parts of P or S. We then match as much of each cell as possible to each of its eight neighbours. The maximum distance over which we have greedily matched weight is $\sqrt{2}\delta/\sqrt{nm}$, and the remaining parts of P and S have a minimum distance of $\delta/\sqrt{nm}$ to each other.

We can then combine the ideas of the point-to-triangle and segment-to-segment algorithm to approximate the optimal solution. First, we overlay a uniform grid with cells of size $\delta \times \delta$ separately for P and S, and identify the cells that contain a triangle. We then recursively subdivide each cell as long as there is any part of a triangle in the other set for which our distance ratio of $1+\delta$ is violated. We track the total area of

triangles contained in each cell, then approximate each cell by a point with a weight equal to this area. After each set is approximated by points in this way, we can run our algorithm as before. Lemma 2.17 can be straightforwardly modified to give the same result for triangles, except that l will be an area instead of a length. This means that we obtain a $(1+\varepsilon)$-approximation with an additive term of $O(\varepsilon/\sqrt{nm})$.

2.8.1 Running time analysis

The number of cells examined during the greedy matching is $O(\frac{\sqrt{nm}\Delta}{\delta})$ per triangle, so $O(\frac{\sqrt{nm}\Delta(n+m)}{\delta})$ in total. The part of the input remaining after greedy matching can be at distance $\delta/\sqrt{nm}$ from each other everywhere, causing it to be subdivided into cells of size $\Theta(\frac{\delta^2}{\sqrt{nm}})$ to maintain a distance ratio of $1+\delta$. There may be $O(\frac{\sqrt{nm}\Delta}{\delta^2})$ cells that intersect the boundaries of a triangle, or $O(\frac{\sqrt{nm}\Delta(n+m)}{\delta^2})$ in total; the other cells are interior to some triangle, and as the total area is one, there can be at most $O(\frac{nm}{\delta^4})$ of them. The total number of cells is therefore at most $O(\frac{\sqrt{nm}\Delta(n+m)}{\delta^2}+\frac{nm}{\delta^4}) \in O(\frac{\sqrt{nm}\Delta(n+m)}{\delta^4}$. This gives the following result:

Theorem 2.25. *Let P be a set of n and S a set of m triangles in the plane, both having equal total area and longest edge length at most Δ after normalising their total areas to one, let $|\eta^*|$ be the cost of an optimal transport plan, and let δ be any constant >0. Given an algorithm that constructs a $(1+\delta)$-approximation between weighted sets of k points in $f_\delta(k)$ time, we can construct a transport plan between P and S with cost $\leq (1+c'\delta)|\eta^*| + O(\frac{\delta}{\sqrt{nm}})$ for some constant c' can be constructed in $O\Big(f_\delta\Big(\frac{\sqrt{nm}\Delta(n+m)}{\delta^4}\Big)\Big)$ time.*

We can again calculate a $(1+\delta)$-approximation to ν in $O(N\delta^{-O(1)}\operatorname{polylog} N)$ time using the algorithm by Fox and Lu [50], giving the following corollary to the previous theorem:

Corollary 2.26. *For any constant $\varepsilon > 0$, a transport plan between P and S with cost $\leq (1+\varepsilon)|\eta^*| + O(\frac{\varepsilon}{\sqrt{nm}})$ can be constructed in $O\Big(\frac{\sqrt{nm}\Delta(n+m)}{\varepsilon^c}\operatorname{polylog}\big(\frac{nm\Delta}{\varepsilon}\big)\Big)$ time with high probability.*

2.9 Higher dimensions

In this section we show how our approach can be extended to work in d-dimensional space. We discuss the case of transporting mass from points to d-dimensional simplices, and from one set of simplices to another.

2.9.1 Points to simplices

The approach described here is a direct extension of the one detailed in Section 2.6. Let P be a set of n weighted points in d dimensions with total mass one, and let S be a set of m d-dimensional simplices with total volume one and longest edge length Δ. We start by overlaying an infinite grid of size Δ and identifying the cells intersected by any simplex in $O(dm\log(m) + 6^d d^2 d^d m)$ time using compressed quadtrees [64]. We then repeatedly subdivide each cell until the ratio between the shortest and longest distance is at most $1+\delta$ for all points in P, or until it is wholly within $\delta/(nm)^{1/d}$ of any point in P. Call the resulting set of cells Q. We can again show that picking one point in each cell of Q with weight equal to the total volume of simplices contained in it, and then solving a transport problem between P and the resulting set of points, approximates the transport problem between P and S.

As the structure of the proof is very similar to that contained in Section 2.6, we omit some of the intermediate lemmas here. We start with the lower bound on the cost of a minimum cost flow $\mathcal{W}$ in the bipartite graph $G = (P \cup Q, P \times Q)$:

Lemma 2.27. $|\mathcal{W}| \geq \dfrac{\delta}{(nm)^{1/d}} - \dfrac{\delta(2(\delta+\delta^2))^d}{(nm)^{1/d}}$.

Proof. For a given point-simplex pair $(p, s) \in P \times S$, consider the cells in Q intersecting s that have a point within distance $\delta/(nm)^{1/d}$ of p. Such a cell has its furthest point at most at distance $(\delta+\delta^2)/(nm)^{1/d}$. The total volume of these cells is then $(2(\delta+\delta^2))^d/nm$ (the volume of a d-dimensional hypercube with radius $(\delta+\delta^2)/(nm)^{1/d}$, which contains the hypersphere with the same radius). Over all point-simplex pairs, this gives a volume of at most $(2(\delta+\delta^2))^d$, leaving $1-(2(\delta+\delta^2))^d$ with distance at least $\delta/(nm)^{1/d}$ in $\mathcal{W}$. The cost is therefore at least $(1-(2(\delta+\delta^2))^d)\cdot\delta/(nm)^{1/d} = \delta/(nm)^{1/d} - \delta(2(\delta+\delta^2))^d/(nm)^{1/d}$. □

We again approximate $|\mathcal{W}|$ by reducing the flow problem to a transportation problem between two sets of weighted points. We do this by picking a point in each cell $q \in Q$ and giving it a weight equal to the total volume of simplices contained in q. Call this set of points T:

Lemma 2.28. *The cost $|\nu|$ of an optimal transport plan ν between P and T satisfies* $|\mathcal{W}| \leq |\nu| \leq (1+\delta)|\mathcal{W}| + \dfrac{\delta(2(\delta+\delta^2))^d}{(nm)^{1/d}}$.

Proof. The lower bound follows directly from the fact that none of the distances in ν are smaller than the distances between the same objects in $\mathcal{W}$ (recall that the distances in $\mathcal{W}$ are measured to the closest point on the cell). We can upper bound $|\nu|$ by measuring all distances to the furthest point in each cell. By construction, those distances are at most $1+\delta$ times the distance to the closest point when the furthest distance is at least $\delta/(nm)^{1/d}$. We can therefore bound all parts of ν where the distance is at least $\delta/(nm)^{1/d}$ by $(1+\delta)|\mathcal{W}|$. The total mass being moved over distance at most $\delta/(nm)^{1/d}$

in ν is at most $(2(\delta+\delta^2))^d$, giving a cost of $\delta(2(\delta+\delta^2))^d/(nm)^{1/d}$. The total cost when measuring to the furthest point is therefore $(1+\delta)|\mathcal{W}| + \delta(2(\delta+\delta^2))^d/(nm)^{1/d}$. □

As before, we turn ν into a valid transport plan η between P and Q by spreading the mass moved to each point in T out evenly over the parts of simplices contained in the cell of Q the point was derived from. By the same argument used in Lemma 2.28, we obtain the following bound on the cost of η:

Lemma 2.29. $|\eta| \leq (1+\delta)|\nu| + \dfrac{\delta(2(\delta+\delta^2))^d}{(nm)^{1/d}}$.

Putting this all together, we can show that $|\eta|$ approximates the cost of the optimal transport plan $|\eta^*|$:

Theorem 2.30. *$|\eta|$ is a $(1+21\delta)$-approximation to the earth mover's distance $|\eta^*|$ between P and S for $0 < \delta \leq \frac{1}{5}$.*

Proof. We follow the same structure as Theorem 2.12, obtaining the following ratio, into which we plug the lower bound from Lemma 2.27:

$$
\begin{aligned}
& \frac{(1+\delta)^2|\mathcal{W}| + \frac{2\delta(2(\delta+\delta^2))^d}{(nm)^{1/d}} + \frac{\delta^2(2(\delta+\delta^2))^d}{(nm)^{1/d}}}{|\mathcal{W}|} \\
\leq\ & \frac{(1+2\delta+\delta^2)\left(\frac{\delta}{(nm)^{1/d}} - \frac{\delta(2(\delta+\delta^2))^d}{(nm)^{1/d}}\right) + \frac{2\delta(2(\delta+\delta^2))^d}{(nm)^{1/d}} + \frac{\delta^2(2(\delta+\delta^2))^d}{(nm)^{1/d}}}{\frac{\delta}{(nm)^{1/d}} - \frac{\delta(2(\delta+\delta^2))^d}{(nm)^{1/d}}} \\
=\ & \frac{1+2\delta+\delta^2+(2(\delta+\delta^2))^d - \delta(2(\delta+\delta^2))^d - \delta^2(2(\delta+\delta^2))^d}{1-(2(\delta+\delta^2))^d} \\
=\ & 1+\delta+\frac{\delta+\delta^2+2(2(\delta+\delta^2))^d - \delta^2(2(\delta+\delta^2))^d}{1-(2(\delta+\delta^2))^d} \\
\leq\ & 1+\delta+\frac{\delta+\delta^2+2(2(\delta+\delta^2))((2(\delta+\delta^2)))^{d-1}}{1-(2(\delta+\delta^2))^d} \\
\leq\ & 1+\delta+\frac{\delta+\delta^2+8\delta}{1-\frac{1}{2^d}} && \text{(Assuming } (\delta+\delta^2) \leq \tfrac{1}{4}\text{)} \\
\leq\ & 1+\delta+\frac{9\delta+\delta^2}{\frac{1}{2}} \\
<\ & 1+21\delta
\end{aligned}
$$

As $|\mathcal{W}|$ is also a lower bound for $|\eta|$ (Lemma 2.2), and η can obviously not have lower cost than the optimal transport plan, this gives a $(1+21\delta)$-approximation. □

Note that the constant in our approximation is slightly worse than the one obtained in Theorem 2.12; this is because we approximate the volume of a hypersphere by the volume of its bounding cube, whereas before we could calculate the area of the disk exactly.

2.9.1.1 Running time analysis

We construct a structure similar to Section 2.6.1, then argue that the number of cells in our actual subdivision is similar. Our construction is the direct generalisation of the one described before: we build a layered structure of cells of increasing sizes. Let $R_{i,j}$ be the set of cells generated by point p_i and simplex s_j; we now analyse how many cells are created.

Lemma 2.31. *$R = \bigcup R_{i,j}$ has $O\left(\frac{dd^{d/2}nm}{\delta^d} \log \frac{(nm)^{1/d}\Delta}{\delta}\right)$ cells.*

Proof. We follow the structure of the proof of Lemma 2.14, again counting the number of cells in one "quadrant", and then multiplying by the number of quadrants (2^d). The number of cells in a quadrant is

$$\left(\frac{\beta}{\gamma}\right)^d + \sum_{i=0}^{k} d\alpha_i \cdot \left(\frac{\beta + \sum_{j=0}^{i-1} \alpha_j 2^j \gamma}{2^i \gamma}\right)^{d-1} + \alpha_i^d$$

where α_i is the number of layers of cells that have doubled in size i times, β is the distance inside of which we use the smallest cell size ($\delta/(nm)^{1/d}$), γ is the smallest cell size ($\delta^2/(nm)^{1/d}$), and k is the number of times we need to double the cell size.

The value of each α_i is $1/\delta$: this value is exact along any axis, and we can show that all cells can be made to have the correct ratio with a given number of extra subdivisions. For a given cell $r \in R_{i,j}$, let $\mathbf{v}$ be the vector from p_i to the closest point on r, and let γ' be the edge length of cell q. W.l.o.g. assume that $v_0 = \max v_i$; by construction this gives us that $(v_0+\gamma')/v_0 \leq 1+\delta$. Let $\mathbf{u}$ be the vector from the closest point on q to the furthest point; we want to find a value x such that $\frac{|\mathbf{v}+\mathbf{u}/x|}{|\mathbf{v}|} \leq 1+\delta$. Through the triangle inequality, we can upper bound the distance to the furthest point on q as $|\mathbf{v}| + |\mathbf{u}/x|$. We can now calculate the required value of x:

$$\begin{aligned}
\frac{|\mathbf{v}| + \left|\frac{\mathbf{u}}{x}\right|}{|\mathbf{v}|} &\leq 1+\delta \\
\frac{|\mathbf{u}|}{x|\mathbf{v}|} &\leq \delta \\
\frac{\gamma'\sqrt{d}}{x|\mathbf{v}|} &\leq \delta
\end{aligned}$$

$$\frac{\gamma'\sqrt{d}}{xv_0} \leq \delta$$
$$\frac{\sqrt{d}}{x}\delta \leq \delta$$
$$x \geq \sqrt{d}$$

So all cells have the correct ratio if their edge length is reduced by a factor of at least $\sqrt{d}$, which means each cell needs to be replaced by at most $O(d^{d/2})$ cells.

This gives us the following derivation for the number of cells:

$$\begin{aligned}
& \frac{1}{\delta^d} + \sum_{i=0}^{k} \frac{d}{\delta} \cdot \left(\frac{\frac{\delta}{(nm)^{1/d}} + \sum_{j=0}^{i-1} 2^j \frac{\delta}{(nm)^{1/d}}}{2^i \frac{\delta^2}{(nm)^{1/d}}} \right)^{d-1} + \frac{1}{\delta^d} \\
= \; & \frac{1}{\delta^d} + \frac{k}{\delta^d} + \frac{d}{\delta} \sum_{i=0}^{k} \left(\frac{\frac{\delta}{(nm)^{1/d}} + \frac{\delta}{(nm)^{1/d}}(2^i - 1)}{2^i \frac{\delta^2}{(nm)^{1/d}}} \right)^{d-1} \\
= \; & \frac{1}{\delta^d} + \frac{k}{\delta^d} + \frac{d}{\delta} \sum_{i=0}^{k} \left(\frac{\frac{\delta}{(nm)^{1/d}}}{\frac{\delta^2}{(nm)^{1/d}}} \right)^{d-1} \\
= \; & \frac{1}{\delta^d} + \frac{k}{\delta^d} + \frac{d}{\delta} \cdot \frac{k}{\delta^{d-1}} \\
= \; & \frac{1}{\delta^d} + \frac{k}{\delta^d} + \frac{kd}{\delta^d} \\
\in \; & O\left(\frac{kd}{\delta^d} \right)
\end{aligned}$$

The value of k is derived in the same way as before, giving $k \in O(\log((nm)^{1/d}\Delta/\delta))$. This gives $O(\frac{d}{\delta^d} \log \frac{(nm)^{1/d}\Delta}{\delta})$ cells per point-simplex pair, where each cell needs to be divided into $O(d^{d/2})$ smaller cells, for a total of $O(\frac{dd^{d/2}nm}{\delta^d} \log \frac{(nm)^{1/d}\Delta}{\delta})$ cells. □

Lemma 2.32. *Q has $O\left(\frac{5^d dd^{d/2}nm}{\delta^d} \log \frac{(nm)^{1/d}\Delta}{\delta} \right)$ cells.*

Proof. Consider any cell $r \in R$. As before, any cell $q \in Q$ that overlaps with r has $|q| \geq |r|/4^d$: otherwise q was subdivided unnecessarily. As the cells in Q are disjoint, it follows that r can overlap with at most 5^d cells in Q. As such, Q contains at most 5^d times more cells than R, which, by Lemma 2.31, is $O\left(\frac{dd^{d/2}nm}{\delta^d} \log \frac{(nm)^{1/d}\Delta}{\delta}\right)$. □

Combined with the time required to build the quadtree that we use to find the starting cells of our subdivision, this gives the following result:

Theorem 2.33. *Let P be a set of n weighted points and S be a set of m simplices in $\mathbb{R}^d$ with equal total weight, let Δ be the longest edge length in S after normalising its total volume to one, let $|\eta^*|$ be the cost of an optimal transport plan between P and S, and let δ be any constant > 0. Given an algorithm that constructs a $(1+\delta)$-approximation between weighted sets of k points in $f_\delta(k)$ time, we can construct a transport plan between P and S with cost $\leq (1+21\delta)|\eta^*|$ in $O\Big(6^d d^2 d^d m + f_\delta\Big(\frac{5^d d d^{d/2} nm}{\delta^d} \log \frac{(nm)^{1/d}\Delta}{\delta}\Big)\Big)$ time.*

Note that if we set $d = 2$, this is the same running time as the one obtained in Theorem 2.15. We can again calculate a $(1+\delta)$-approximation to ν in $O(N\delta^{-O(d)} \log^{O(d)} N)$ time using the algorithm by Fox and Lu [50]. Setting $\delta = \varepsilon/21$, this gives a total running time of

$$\begin{aligned} & O\Big(6^d d^2 d^d m + \frac{105^d d d^{d/2} nm}{\varepsilon^{O(d)}} \log^{O(d)}\Big(\frac{5^d dnm\Delta}{\varepsilon^d}\Big)\Big) \\ \in\ & O\Big(6^d d^2 d^d m + \frac{105^d d^2 d^{d/2} nm}{\varepsilon^{O(d)}} \log^{O(d)}\Big(\frac{dnm\Delta}{\varepsilon^d}\Big)\Big). \end{aligned}$$

Corollary 2.34. *For any constant $\varepsilon > 0$, a transport plan between P and S with cost $\leq (1+\varepsilon)|\eta^*|$ can be constructed in $O\Big(6^d d^2 d^d m + \frac{105^d d^2 d^{d/2} nm}{\varepsilon^{O(d)}} \log^{O(d)}\big(\frac{dnm\Delta}{\varepsilon^d}\big)\Big)$ time with high probability.*

2.9.2 Simplices to simplices

The approach from Sections 2.7 and 2.8 can also be extended to work on d-dimensional simplices in d dimensions. We take the same approach of overlaying a grid with cells of size $\delta/(4(nm)^{1/d})$ onto the input and greedily matching the parts of P and S that are close together. The maximum distance over which we greedily match weight is then $\sqrt{d}\delta/(2(nm)^{1/d})$, and the remaining parts of P and S have minimum distance $\delta/(2(nm)^{1/d})$ to each other. We then approximate the transport plan between the remaining cells with a minimum cost flow. The same analysis still works, and we obtain a $(1+\varepsilon)$-approximation with an additive term of $O(\sqrt{d}\varepsilon/(nm)^{1/d})$.

2.9.2.1 Running time analysis

The number of cells examined during the greedy matching is $O(\frac{(nm)^{1/d}\Delta^d}{\delta})$ per simplex, so $O(\frac{(nm)^{1/d}\Delta^d(n+m)}{\delta})$ in total (note that we simplify the analysis by simply considering the volume of a d-dimensional cube of side length Δ). The part of the input remaining after greedy matching can be close to each other everywhere, causing it to be subdivided into cells of size $\Theta(\frac{\delta^2}{\sqrt{d}(nm)^{1/d}})$. The total number number of these cells is then $O(\frac{\sqrt{d}(nm)^{1/d}\Delta^d(n+m)}{\delta^2})$. This gives the following result:

Theorem 2.35. *Let P be a set of n and S a set of m d-dimensional simplices in $\mathbb{R}^d$, both having equal total volume and longest edge length at most Δ after normalising their total volumes to one, let $|\eta^*|$ be the cost of an optimal transport plan between them, and let δ be any constant > 0. Given an algorithm that constructs a $(1+\delta)$-approximation between weighted sets of k points in $f_\delta(k)$ time, we can construct a transport plan between P and S with cost $\leq (1+c'\delta)|\eta^*| + O(\frac{\sqrt{d}\delta}{(nm)^{1/d}})$ for some constant c' can be constructed in $O\Big(f_\delta\Big(\frac{\sqrt{d}(nm)^{1/d}\Delta^d(n+m)}{\delta^2}\Big)\Big)$ time.*

We can again calculate a $(1+\delta)$-approximation to ν in $O(N\delta^{-O(d)}\log^{O(d)} N)$ time using the algorithm by Fox and Lu [50], giving the following corollary to the previous theorem:

Corollary 2.36. *For any constant $\varepsilon > 0$, a transport plan between P and S with cost $\leq (1+\varepsilon)|\eta^*| + O(\frac{\sqrt{d}\varepsilon}{(nm)^{1/d}})$ can be found in $O\Big(\frac{\sqrt{d}(nm)^{1/d}\Delta^d(n+m)}{\varepsilon^{O(d)}}\log^{O(d)}\Big(\frac{d(nm)^{1/d}\Delta^d}{\varepsilon}\Big)\Big)$ time with high probability.*

2.10 Conclusion

We have provided approximation algorithms to the earth mover's distance between sets of points, line segments, triangles and d-dimensional simplices. These are the first combinatorial algorithms with a provable approximation ratio for this problem when the objects are continuous rather than discrete points.

Here we described the case where the total mass is spread uniformly over the available length or area. However, our approach also works when this is not the case. If the ratio of densities is bounded, the same running times hold; otherwise, this ratio will show up in the running times the same way that the longest edge length does for cases involving triangles.

We note that for points and line segments (in any dimension), the approximation scheme is free from undesired parameters, whereas for points and triangles (or simplices), the maximum edge length Δ appears in the running time, and when neither set is a set of points, an additive term appears in the approximation. The most interesting open question is whether either of these two artifacts can be avoided.

Chapter 3

Hausdorff morphing

3.1 Introduction

For two sets A and B in $\mathbb{R}^2$, we define the *directed Hausdorff distance* as

$$d_{\vec{H}}(A, B) \coloneqq \sup_{a \in A} \inf_{b \in B} d(a, b),$$

where d denotes the Euclidean distance. The *undirected Hausdorff distance* is defined as

$$d_H(A, B) \coloneqq \max(d_{\vec{H}}(A, B), d_{\vec{H}}(B, A)).$$

If A and B are closed sets then $d_H(A, B) = r$ is equivalent to saying that r is the smallest value such that $A \subseteq B \oplus D_r$ and $B \subseteq A \oplus D_r$, where $\oplus$ denotes the Minkowski sum, and D_r is a disk of radius r centered at the origin. Recall that the Minkowski sum of sets A and B is the set $\{a + b \mid a \in A,\ b \in B\}$. In this chapter we consider only closed sets, and therefore we can freely use this containment property.

The Hausdorff distance has been widely used in computer vision [43] and computer graphics [16, 39] for tasks such as template matching, and error computation between a model and its simplification. At the same time, the Hausdorff distance is a classic mathematical concept. Our research motivation is to study this profound concept from a new perspective. Algorithms to compute the Hausdorff distance between two given sets are available for many types of sets, such as points, line segments, polylines, polygons, and simplices in k-dimensional Euclidean space [8, 9, 18]. However, computing the Hausdorff distance between general semialgebraic sets has been shown to be $\forall\exists_<\mathbb{R}$-complete [66].

In this chapter, we consider the natural problem of finding a set that lies "between" two or more input sets, in a Hausdorff sense. In Section 3.2 we investigate the Hausdorff middle of sets A and B; this is a set that has minimum undirected Hausdorff distance to A and B. Differently put, it minimizes the maximum of four directed

Figure 3.1: Hausdorff morphs between three shapes.

Hausdorff distances. We show that when the Hausdorff distance between A and B is assumed to be 1, there is always a Hausdorff middle that has Hausdorff distance $1/2$ to A and B, and we cannot have a lower distance. We relate the convexity of A and/or B to the convexity and connectedness of the Hausdorff middle, and study its combinatorial complexity.

We actually treat the middle more generally, by defining a class of sets that smoothly interpolate between A and B, giving a morph between them. Figure 3.1 shows two examples of such morphs. We prove that for two given intermediate shapes in the morph, the difference between the interpolation parameters bounds the Hausdorff distance between the shapes.

Algorithms for morphing, sometimes called *shape interpolation*, have been widely studied. A classical application is the reconstruction of a 3D object from 2D slices, a common problem in medical imaging. Many algorithms that solve this problem exist, based on straight skeletons [20, 22], curve matching and triangulations [21], and Delaunay triangulations [27]. When considering more abstract applications, a typical approach is to first transform each input shape into a cannonical form, and then morph between those. Alt and Guibas [14] give an overview of this approach. Finally, work has been done to ensure the interpolation of two simple polygons is itself a simple polygon [56].

A common thread in all these algorithms is that they are based on computing some kind of correspondence between features of the input shapes, either by explicitly matching parts of the boundary, or by computing some geometrical structure (like a Voronoi diagram or a straight skeleton). In addition, most of these morphing algorithms interpolate only the boundary of the input shapes, and keep all intermediate shapes polygonal. Our approach does not require any correspondence between features of the input to be calculated. However, our approach is unusual in the sense that the intermediate shapes when morphing between e.g. two polygons are not necessarily polygons themselves.

In Section 3.3 we extend the results of Section 3.2 to Hausdorff middles of more than two sets and generalize several results. We assume that the maximum Hausdorff distance over all pairs of input sets is 1 and examine the smallest Hausdorff distance for a middle set. That is, given sets $\mathcal{M} = \{A_1, \ldots, A_k\}$, we are interested in the value $\alpha(\mathcal{M}) = \min_S \max_{i=1,\ldots,k} d_H(A_i, S)$. This value $\alpha(\mathcal{M})$ is no longer $1/2$, but depends on the input. For convex sets, we show that a value ≈ 0.608 can always be achieved and is sometimes necessary, whereas for non-convex sets a value of 1 may be required.

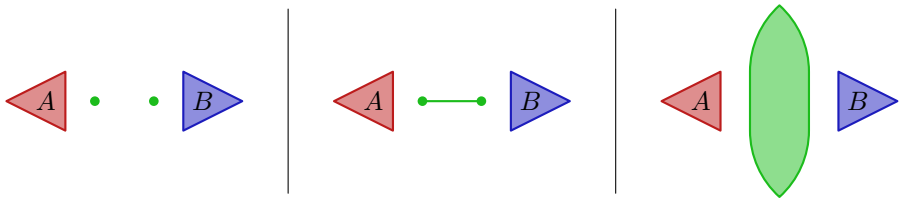


Figure 3.2: Three possible Hausdorff middles of A and B: two points, a line segment, and $S_{1/2}$.

For a given set of polygons with total combinatorial complexity n, we show that $\alpha(\mathcal{M})$ and the Hausdorff middle can be computed in $O(n^6)$ time, and, for any constant $\varepsilon > 0$, $(1+\varepsilon)$-approximated in $O(n^2 \log^2 n \log 1/\varepsilon)$ time. We note that other interpolation methods between two shapes do not have a natural generalization to a middle of three or more shapes.

Our proofs use three types of arguments. First, many of our arguments rely on simple manipulations of the formal definition of the Hausdorff distance. The second type of argument is of a topological nature. Using continuity and connectivity, we infer related properties to the output, by constructing topological structures or conclude that they cannot exist. The third type of argument uses 2-dimensional Euclidean geometry directly. We construct features, like vertices, edges and circular arcs, and argue about their existence, and give distance bounds. These arguments are often intricate and do not generalize. They are of particular value, as the 2-dimensional Euclidean plane is often the most interesting case in computational geometry.

3.2 The Hausdorff middle of two sets

Consider two compact sets A and B in $\mathbb{R}^2$; we are interested in computing a *Hausdorff middle*: a set C that minimizes the maximum of the undirected Hausdorff distances to A and B. That is,

$$C \in \arg\min_{C' \subset \mathbb{R}^2} \max(d_H(A, C'), d_H(B, C')).$$

Note that there may be many such sets that minimize the Hausdorff distance; see Figure 3.2 for a few examples. It might seem intuitive to restrict C to be the minimal set that achieves this distance, but such a set is not necessarily unique, and the common intersection of all minimal sets is not a solution itself (see Figure 3.3). However, the maximal set is unique. Let $d_H(A, B) = 1$. Then

$$S(A, B) := (A \oplus D_{1/2}) \cap (B \oplus D_{1/2})$$

is the unique maximal set with Hausdorff distance $1/2$ to A and B (we prove this below in Lemma 3.2; see the right of Figure 3.2 for an example of what S looks like).

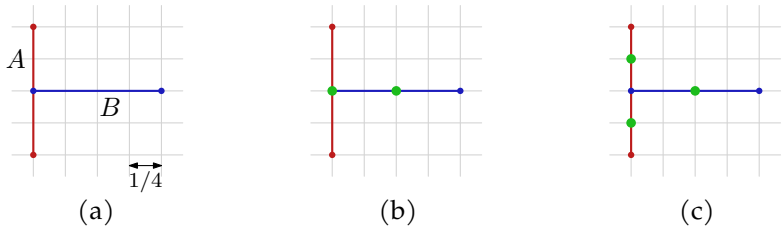


Figure 3.3: Two different minimal sets achieving minimal Hausdorff distance to A and B. Both the two green dots in Figure (b) and the three green dots in Figure (c) minimize the Hausdorff distance to A and B.

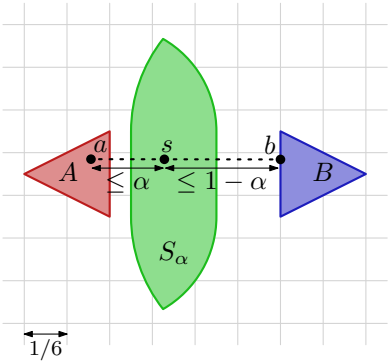


Figure 3.4: An arbitrary point $a \in A$ with its closest point b on B. The point s has distance at most α to a, and distance at most $1-\alpha$ to b.

Note that in the rest of this chapter we omit the arguments and simply write S, as the arguments are always clear from context. We want to show that $d_H(A,S) \leq 1/2$ and $d_H(B,S) \leq 1/2$. In fact, we can prove a more general statement.

We define

$$S_\alpha(A,B) \coloneqq (A \oplus D_\alpha) \cap (B \oplus D_{1-\alpha})$$

for $\alpha \in [0,1]$, and we use $\text{seg}(a,b)$ to denote the line segment connecting points a and b.

Theorem 3.1. *Let A and B be two compact sets in the plane with $d_H(A,B) = 1$. Then $d_H(A,S_\alpha) = \alpha$ and $d_H(B,S_\alpha) = 1-\alpha$.*

Proof. We first show that $d_H(A,S_\alpha) \leq \alpha$. The proof for $d_H(B,S_\alpha) \leq 1-\alpha$ is analogous and therefore omitted. We will infer $d_H(A,S_\alpha) \leq \alpha$ from $d_{\vec{H}}(A,S_\alpha) \leq \alpha$ and $d_{\vec{H}}(S_\alpha,A) \leq \alpha$; thereafter we will show equality.

Consider any point $a \in A$; by our assumption that $d_H(A,B) = 1$, there is a point $b \in B$ with $d(a,b) \leq 1$; see Figure 3.4. Now consider a point $s \in \text{seg}(a,b)$ with

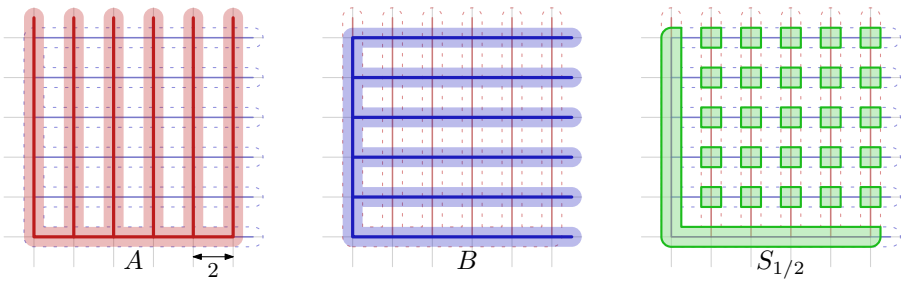


Figure 3.5: Sets A and B for which $S_{1/2}$ is disconnected. The shaded areas around A and B represent $A \oplus D_{1/2}$ and $B \oplus D_{1/2}$, respectively.

$d(a, s) \leq \alpha$ and $d(b, s) \leq 1 - \alpha$; clearly this point must be in S_α, as it is contained in both $A \oplus D_\alpha$ and $B \oplus D_{1-\alpha}$, and it has $d(a, s) \leq \alpha$. As this works for every $a \in A$, it holds that $d_{\vec{H}}(A, S_\alpha) \leq \alpha$. The fact that $d_{\vec{H}}(S_\alpha, A) \leq \alpha$ follows straightforwardly from S_α being a subset of $A \oplus D_\alpha$. Thus, $d_H(A, S_\alpha) \leq \alpha$.

To show equality, assume that the Hausdorff distance between A and B is realized by a point $\hat{a} \in A$ with closest point $\hat{b} \in B$, at distance 1. Consider the point $\hat{s} \in \text{seg}(\hat{a}, \hat{b})$ with $d(\hat{a}, \hat{s}) = \alpha$ and $d(\hat{b}, \hat{s}) = 1 - \alpha$. As observed, $\hat{s} \in S_\alpha$. Since $\hat{s}$ is the closest point of S_α to $\hat{a}$, and $\hat{b}$ is the closest point of B to $\hat{s}$, equality follows. □

Lemma 3.2. *S_α is the maximal set that satisfies $d_H(A, S_\alpha) = \alpha$ and $d_H(B, S_\alpha) = 1 - \alpha$.*

Proof. Consider any set T for which we have $d_{\vec{H}}(T, A) \leq \alpha$ and $d_{\vec{H}}(T, B) \leq 1 - \alpha$. As $A \oplus D_\alpha$ contains all points with distance at most α to A, we have that $T \subseteq A \oplus D_\alpha$; similarly, we have that $T \subseteq B \oplus D_{1-\alpha}$. By the definition of S_α, this implies that $T \subseteq S_\alpha$. As this holds for any T, we conclude that S_α is a unique maximal set. □

3.2.1 Properties of S_α

In this section, we study the convexity and connectedness of S_α. Recall that a set $A \subseteq \mathbb{R}^2$ is convex if for any two points $a, b \in A$, the segment $\text{seg}(a, b)$ between them is completely contained in A. Also, recall that a set $A \subset \mathbb{R}^2$ is connected if for any two points $a, b \in A$, there exists a continuous curve $c : [0, 1] \to A$ such that $c(0) = a$ and $c(1) = b$. This type of connectedness is known as path-connectedness, but we use the term connected for simplicity. We observe the following properties.

1. If A and B are convex, S_α is convex;
2. If A is convex and B is connected, S_α is connected;
3. For some connected sets A and B, S_α is disconnected.

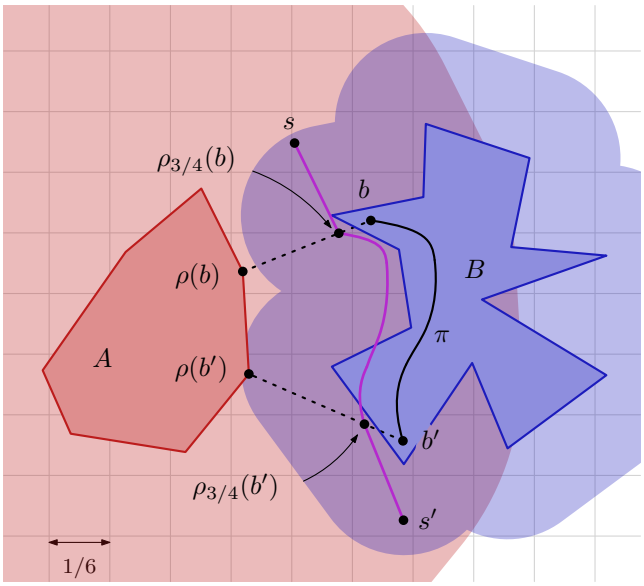


Figure 3.6: Illustration of the proof showing that S_α is connected if A is convex (sketched for $\alpha = 3/4$). The shaded areas around A and B represent $A \oplus D_{3/4}$ and $B \oplus D_{1/4}$, respectively, so that the doubly-shaded area is $S_{3/4}$.

Property 1 is straightforward: the Minkowski sum of A and B with a disk is convex, and the intersection of convex objects is itself also convex. The example in Figure 3.5 demonstrates Property 3; in fact, *any* Hausdorff middle will be disconnected for those input sets.

The next lemma establishes Property 2.

Lemma 3.3. *Let A and B be two compact connected regions of the plane with Hausdorff distance 1, and A convex. Then $S_\alpha = (A \oplus D_\alpha) \cap (B \oplus D_{1-\alpha})$ is connected for any $\alpha \in [0, 1]$.*

Proof. See Figure 3.6 for an illustration. Because A is convex, there is a continuous map $\rho\colon B \to A$ that maps each point of B to a closest point (within distance 1) in A. For $b \in B$, let $\rho_\alpha(b) = \alpha\rho(b) + (1 - \alpha)b$. We have that $\rho_\alpha\colon B \to S_\alpha$ is also continuous.

Now take any two points s and s' in S_α; respectively, they have points b and $b' \in B$ within distance $1 - \alpha$. The segments between s and $\rho_\alpha(b)$ and between s' and $\rho_\alpha(b')$ lie completely in S_α. Take a continuous curve π from b to b' inside B. The image of π under ρ_α connects $\rho_\alpha(b)$ to $\rho_\alpha(b')$ within S_α, so s and s' are connected inside S_α. □

We note that S_α may contain holes. Furthermore, S_α is not shape invariant when B is translated with respect to A. For example, let A be the union of the left and bottom sides of a unit square and let B_1 and B_2 be the left and right sides of that same

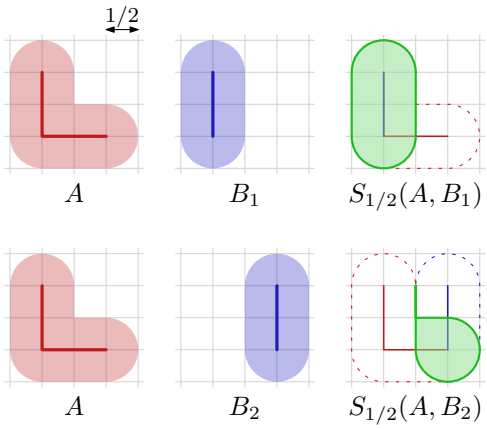


Figure 3.7: Although B_2 is a translate of B_1, the middle set between A and B_2 is not a translate of the middle set between A and B_1.

unit square. Then $(A \oplus D_{1/2}) \cap (B_1 \oplus D_{1/2})$ is not a translate of $(A \oplus D_{1/2}) \cap (B_2 \oplus D_{1/2})$. See Figure 3.7; note that $d_H(A, B_1) = d_H(A, B_2)$.

3.2.2 Complexity of S_α

In this section, we describe the complexity of S_α in terms of the number of vertices, line segments, and circular arcs that make up its boundary, for several types of polygonal input sets. Recall that ∂A denotes the boundary of set A.

Lemma 3.4. *Let A be a convex polygon with n vertices and B a simple polygon with m vertices. Then ∂S_α consists of $O(n+m)$ vertices, line segments and circular arcs, and this bound is tight in the worst case.*

Proof. For brevity we let $A^\oplus = A \oplus D_\alpha$ and $B^\oplus = B \oplus D_{1-\alpha}$.

There is a trivial worst-case lower bound of $\Omega(n+m)$ by taking $\alpha = 0$ or $\alpha = 1$, as $S_0 = A$ and $S_1 = B$. Note that if the boundaries of $A^\oplus$ and $B^\oplus$ would consist of only line segments, the upper bound is easy to show: $A^\oplus$ is convex, and its boundary can therefore intersect each segment of $\partial B^\oplus$ at most twice, making ∂S_α consist of (parts of) segments from $\partial A^\oplus$ and $\partial B^\oplus$ and at most $O(m)$ intersection points. The problem is that $\partial A^\oplus$ and $\partial B^\oplus$ also contain circular arcs, in which case $\partial A^\oplus$ may intersect an arc of $\partial B^\oplus$ $\Omega(n)$ times.

To show an upper bound of $O(n+m)$, we distinguish two cases. In the first case, we assume $\alpha \geq 1 - \alpha$. Note that in this case, the circular arcs that are part of the boundary of $A^\oplus$ have a radius larger or equal to those of $B^\oplus$. Additionally, $\partial A^\oplus$ is smooth and is an alternating sequence of circular arcs and segments, as A is convex. In this case, we do in fact have that any line segment or circular arc b of $\partial B^\oplus$ can

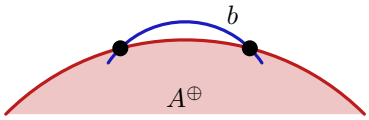


Figure 3.8: When $\alpha \geq 1 - \alpha$, an arc b of $\partial B^{\oplus}$ (blue) can only intersect $\partial A^{\oplus}$ (red) twice.

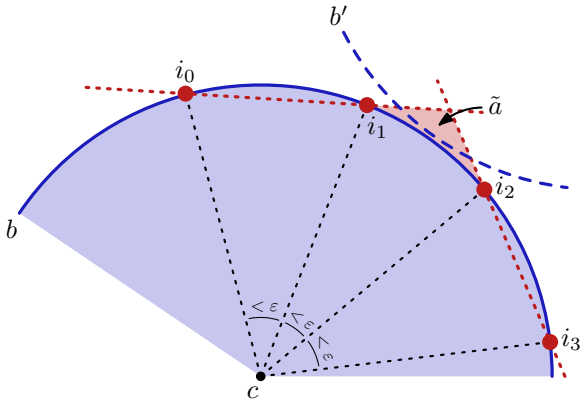


Figure 3.9: When $\alpha < 1 - \alpha$, a single arc b of $\partial B^{\oplus}$, shown in blue, can have many intersections with $\partial A^{\oplus}$, but no other arc b', shown as a dashed blue arc, can have many intersections with the same part of $\partial A^{\oplus}$. The intersections of b with $\partial A^{\oplus}$ are shown in red.

intersect $\partial A^{\oplus}$ at most twice. Consider two intersection points of b with $\partial A^{\oplus}$: as the curvature of $\partial A^{\oplus}$ is at most that of b, there can never be another intersection point between these two, or we would violate the convexity of $A^{\oplus}$. See Figure 3.8 for an illustration of this case.

For the second case, we assume $\alpha < 1 - \alpha$. We charge all the intersections to the arcs and line segments of $\partial A^{\oplus}$ and $\partial B^{\oplus}$. Each line segment of $\partial B^{\oplus}$ can intersect $\partial A^{\oplus}$ at most twice, as $A^{\oplus}$ is convex, so there can be at most $O(m)$ such intersections. Similarly, for arcs of $\partial B^{\oplus}$ that intersect $\partial A^{\oplus}$ at most three times, there can be at most $O(m)$ intersections in total. It remains to consider the arcs of $\partial B^{\oplus}$ that intersect $\partial A^{\oplus}$ more than three times.

Let b be such an arc of $\partial B^{\oplus}$. Consider any quadruple of consecutive intersection points i_0, i_1, i_2, i_3 with $\partial A^{\oplus}$ along b, see Figure 3.9, where the part of $\partial A^{\oplus}$ between i_1 and i_2 that does not contain i_0 and i_3 is outside the disk supporting b. This part is denoted $\tilde{a}$; note that $\tilde{a}$ must contain at least one circular arc, denoted a. Notice that we consider all intersection points between $\partial A^{\oplus}$ and b, except possibly for the first one or two and last one or two. These first and last ones can be charged to b, and

this charge is at most four per arc b. Let c be the center of the supporting disk of b. If any of the angles $\measuredangle i_0 c i_1$, $\measuredangle i_1 c i_2$, or $\measuredangle i_2 c i_3$, is larger than ε for some constant $\varepsilon > 0$, we again charge the intersection points i_1 and i_2 to b, and we have less than $360/\varepsilon$ of such charges. So we now assume that all three angles are at most ε. We charge the intersection points i_1 and i_2 to a, the arc of a disk that appears on $\tilde{a}$.

It remains to show that a is charged at most once. We can limit the distance by which $\tilde{a}$ can protrude outside of b: as $A^{\oplus}$ is convex, $\tilde{a}$ cannot cross the line through i_0 and i_1, nor the line through i_2 and i_3. This restricts $\tilde{a}$ to the shaded area in Figure 3.9. It is possible that $\tilde{a}$ intersects a different arc b' of $\partial B^{\oplus}$ in this shaded area. We observe that the disk that b' is a part of cannot contain the intersection points i_1 and i_2, as otherwise those points would not be intersections of $\partial A^{\oplus}$ and $\partial B^{\oplus}$. Now b' can intersect $\partial A^{\oplus}$ at most twice, as more intersections would violate the convexity of $A^{\oplus}$. In particular, b' cannot intersect $\partial A^{\oplus}$ four times, and hence b' cannot charge intersections on it to a. We conclude that a is charged only once. From this we conclude that there are at most $O(n+m)$ intersection points in total, and that ∂S_α therefore consists of at most $O(n+m)$ vertices, line segments and circular arcs. □

Lemma 3.5. *Let A and B be two simple polygons of n and m vertices, respectively. Then ∂S_α consists of $O(nm)$ vertices, line segments and circular arcs, and this bound is tight in the worst case.*

Proof. The worst-case lower bound of $\Omega(nm)$ follows by taking A and B to be two rotated "combs"; see Figure 3.5. For $\alpha = 1/2$, S_α consists of $\Omega(nm)$ distinct components. The upper bound follows directly from the fact that $A \oplus D_\alpha$ and $B \oplus D_{1-\alpha}$ have complexities $O(n)$ and $O(m)$, respectively. Each individual arc and edge on the boundaries of $A \oplus D_\alpha$ and $B \oplus D_{1-\alpha}$ intersect at most a constant number of times, so we cannot have more than $O(nm)$ intersection points. □

In fact, not just S_α, but *any* Hausdorff middle has complexity $\Theta(nm)$ for the example in Figure 3.5. Since S_α is maximal, the components cannot be connected without changing the Hausdorff distance to A or B, and other middles must have at least some point in every component of S_α to achieve Hausdorff distance $1/2$ to both A and B.

3.2.3 S_α as a morph

By increasing α from 0 to 1, S_α morphs from $A = S_0$ into $B = S_1$. (Examples of such morphs are presented in Figures 3.1 and 3.10.) The following lemma shows that this morph has a bounded rate of change.

Lemma 3.6. *Let S_α and S_β be two intermediate shapes of A and B with $d_H(A,B) = 1$ and $\alpha \leq \beta$. Then $d_H(S_\alpha, S_\beta) = \beta - \alpha$.*

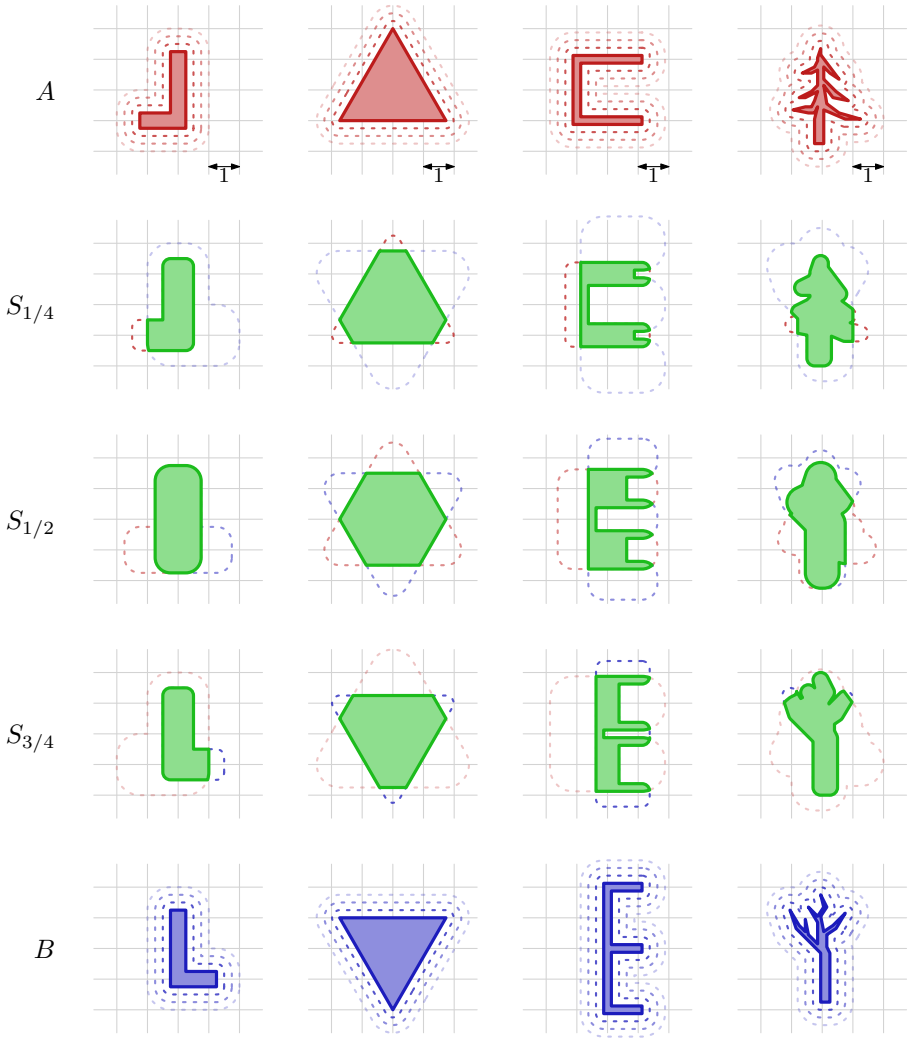


Figure 3.10: Some examples of morphs S_α between two shapes A and B.

Proof. We have $d_H(S_\alpha, S_\beta) \geq \beta - \alpha$ because, by the triangle inequality, $d_H(A, B) = 1 \leq d_H(A, S_\alpha) + d_H(S_\alpha, S_\beta) + d_H(S_\beta, B) \leq \alpha + d_H(S_\alpha, S_\beta) + 1 - \beta$.

It remains to show that $d_H(S_\alpha, S_\beta) \leq \beta - \alpha$. We show that $S_\beta \subseteq S_\alpha \oplus D_{\beta-\alpha}$; the proof that $S_\alpha \subseteq S_\beta \oplus D_{\beta-\alpha}$ is analogous. Let p be some point in S_β. Then, by definition of S_β, there exist some points $a \in A$ and $b \in B$ such that $d(a, p) \leq \beta$ and $d(b, p) \leq 1 - \beta$. Let $\bar{p}$ be the point obtained by moving p in the direction of a by $\beta - \alpha$. By the triangle inequality, we then have that $d(a, \bar{p}) \leq \beta - (\beta - \alpha) = \alpha$ and $d(b, \bar{p}) \leq (1-\beta)+(\beta-\alpha) = 1-\alpha$. This implies that $\bar{p} \in S_\alpha$. As p was an arbitrary point in S_β, and $d(p, \bar{p}) \leq \beta - \alpha$, we have that $S_\beta \subseteq S_\alpha \oplus D_{\beta-\alpha}$. So $d_H(S_\alpha, S_\beta) \leq \beta - \alpha$. □

The above lemma implies that, even though the number of connected components of S_α can change when α changes, new components arise by splitting and never "out of nothing", and the number of components can only decrease through merging and not by disappearance.

The morph $\langle S_\alpha \,|\, \alpha \in [0, 1]\rangle$ from A to B has a consistent submorph property, formalized below.

Lemma 3.7. *If a morph from $A = S_0$ to $B = S_1$ contains a shape C, then the morph from A to C concatenated with the morph from C to B is the same as the morph from A to B: they contain the same collection of shapes in between and in the same order.*

Proof. Let α be the value such that $S_\alpha(A, B) = C$. We define $S'_\beta(A, C) \coloneqq (A \oplus D_\beta) \cap (C \oplus D_{\alpha-\beta})$ for $\beta \in [0, \alpha]$, giving the morph from A to C. We need to show that $S_\beta(A, B) = S'_\beta(A, C)$. The case for the morph from C to B is analogous and therefore omitted.

Let x be any point in $S_\beta(A, B)$. By definition it has a distance of at most β to A, and Lemma 3.6 establishes that it has distance at most $\alpha - \beta$ to C. This implies that $x \in S'_\beta(A, C)$. As this works for any point x, we have that $S_\beta(A, B) \subseteq S'_\beta(A, C)$. Now let x' be any point in $S'_\beta(A, C)$. By definition it has distance at most β to A, and distance at most $\alpha - \beta$ to some point $c \in C$. As $d_H(B, C) = 1 - \alpha$, by the triangle inequality x' must have distance at most $(\alpha - \beta) + (1 - \alpha) = 1 - \beta$ to some point in B. This shows $x' \in S_\beta(A, B)$. As this works for any point x', we also have that $S'_\beta(A, C) \subseteq S_\beta(A, B)$. We conclude that $S_\beta(A, B) = S'_\beta(A, C)$. □

As a corollary of this lemma, $\{\alpha \in [0, 1] \mid S_\alpha \text{ is convex}\}$ is a connected interval.

3.2.4 The cost of connectedness

For some applications, it might be necessary to insist that the middle shape is always connected. However, in the worst case, the cost of connecting all components of S_α can be that the Hausdorff distance of the resulting shape to A and B becomes 1. See Figure 3.11 for an example where this is the case. In fact, any connected shape has distance at least 1 for this example.

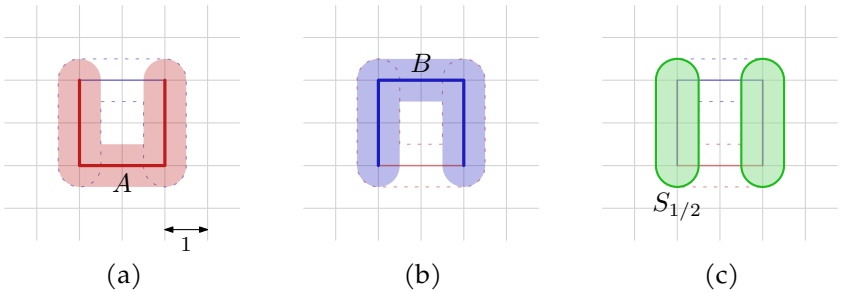


Figure 3.11: Figures (a) and (b) show the offsets of A, respectively B with distance $1/2$. Figure (c) shows the resulting $S_{1/2}$ in green. Any connected shape must cross the vertical middle line or stay on one side of it. In both cases, the Hausdorff distance doubles.

3.3 The Hausdorff middle of more than two sets

A natural question is whether the results from the previous section extend to more than two input shapes. There are several ways to formalize the notion of a Hausdorff middle between multiple shapes. Analogous to the case of two sets, we are interested in a middle shape that minimizes the maximum Hausdorff distance to each input set. Let $\mathcal{M} = \{A_1, \ldots, A_m\}$ be a collection of m input shapes with largest pairwise Hausdorff distance 1. We define T_α as $\bigcap_i (A_i \oplus D_\alpha)$; the (maximal) middle set is then given by the smallest value α for which $T_\alpha \oplus D_\alpha$ contains all input sets. We denote this smallest α by $\alpha(\mathcal{M}) := \min\{\, \alpha \mid \max_i d_H(A_i, T_\alpha) \leq \alpha \,\}$. If α is clear from the context, we use the notation $A^{\oplus}$ to mean $A \oplus D_\alpha$.

In this section, we first study the largest possible $\alpha(\mathcal{M})$ for general and convex input. We then study some general properties of T_α with respect to connectivity and convexity. After this, we consider whether there is some subset of $\mathcal{M}$ that requires the same value of α, and obtain a Helly-type property for convex input. Finally, we will give various algorithms to compute or approximate $\alpha(\mathcal{M})$ efficiently.

3.3.1 The largest $\alpha(\mathcal{M})$

In this section, we are interested in the largest possible value of $\alpha(\mathcal{M})$. We first discuss the general case and then study the case where all sets $A \in \mathcal{M}$ are convex. In both cases, we provide an exact answer. This section relies on some tedious calculations, which turn out to be easier if we do not normalize pairwise distances of our objects to 1.

As it turns out, for some inputs it may be the case that $\alpha(\mathcal{M}) = 1$; see Figure 3.12. Here, there can be no shape with Hausdorff distance less than 1 to all the input shapes, meaning any of the three input shapes can be chosen as "the middle". Hence, for two

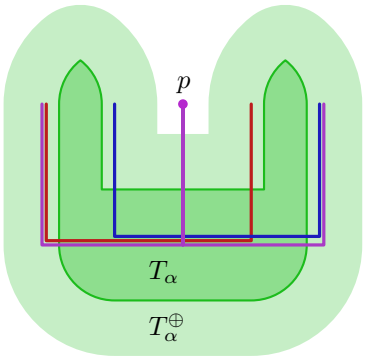


Figure 3.12: The pairwise Hausdorff distance in this construction is 1, and for any $\alpha < 1$, $T_\alpha^\oplus$ does not contain point p.

input sets, we always have $\alpha(\mathcal{M}) = 1/2$, but for more input sets, the value depends on the input, and $\alpha(\mathcal{M})$ will be in $[1/2, 1]$. The example in Figure 3.12 requires non-convex sets, raising the question of what the range of $\alpha(\mathcal{M})$ can be when all A_i are convex.

If we have three convex sets that are points, and they form the corners of an equilateral unit-side triangle, then we can easily see that $\alpha(\mathcal{M}) = 1/\sqrt{3} \approx 0.577$ and the middle shape is exactly the point in the middle of the triangle.

An example with three line segments shown in Figure 3.13 surprisingly achieves (for $\lambda \approx 0.253135$, $\theta \approx 123.37°$) a larger value $\alpha^* \approx 0.6068 = r$, which we call the *magic value*. Lemma 3.9 shows that no three convex sets achieve $\alpha(\mathcal{M}) > \alpha^*$. Thus the magic value is a tight upper bound for three convex sets.

We define the magic value as $\alpha^* = 1/z \approx 0.6068$, where the value of z is derived from Figure 3.14, and defined as $z := \min\{\lambda + 1 - \cos(2\theta) \mid \lambda \geq 0, \theta \in (90°, 180°),$ and $\lambda + 1 - \cos(2\theta) = \|(-\lambda\cot(2\theta) - \sin(2\theta) + \sin(\theta), \lambda - \cos(2\theta) + \cos(\theta))\|\} \approx 1.647986325231$ (at $\lambda \approx 0.253135$, $\theta \approx 123.37°$, verified using Wolfram Cloud).

Lemma 3.8. *Let $\mathcal{M} = \{A_1, \ldots, A_m\}$ be a collection of compact convex regions in the plane, and $\alpha := \alpha(\mathcal{M})$. There is some $A_i \in \mathcal{M}$ with $d_{\vec{H}}(A_i, T_\alpha) = \alpha$.*

Proof. By construction, we have $d_{\vec{H}}(T_\beta, A_i) \leq \beta$ for all i and all β. (Recall that this is equivalent to $T_\beta \subseteq A_i \oplus D_\beta$.) Moreover, if T_β is nonempty, then for any i, the map $\gamma \mapsto d_{\vec{H}}(T_\gamma, A_i)$ is continuous on the domain $[\beta, \infty)$, as T_γ changes continuously. We show that for some i, we have $d_{\vec{H}}(A_i, T_\alpha) = \alpha$. If instead $d_{\vec{H}}(A_i, T_\alpha) < \alpha$ for all i, then unless T_β is empty for all $\beta < \alpha$, we can decrease α, contradicting minimality of α. If instead α is the minimum value for which T_α is nonempty, then either $\alpha = 0$ and we are done because T_α contains all A_i, or $\alpha > 0$ and T_α has no interior (when viewed as a subset of the plane). Because T_α is the intersection of convex sets, it is

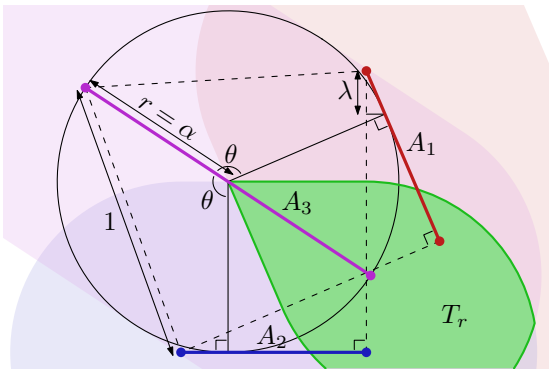


Figure 3.13: Three segments A_1, A_2, and A_3. Of these, A_3 is the diameter of a circle with radius r; the other two (A_1 and A_2) are tangent to the circle and are copies of one another reflected through A_3, such that all pairwise Hausdorff distances are at most 1 (length of dashed segments). The top left vertex of A_3 is furthest (at distance r) from the middle set T_r (green), so $\alpha(\{A_1, A_2, A_3\})$ is the radius r of the circle.

convex. If it has no interior, it is either a segment or a point, and by convexity it must lie on the boundary of $A_i^{\oplus}$ for some i, contradicting that $d_{\vec{H}}(A_i, T_\alpha) < \alpha$. □

Lemma 3.9. *Let $\mathcal{M} = \{A_1, A_2, A_3\}$ be compact convex regions in the plane. Let $\alpha := \alpha(\mathcal{M})$ and $d = \max_{i,j} d_H(A_i, A_j)$, then $d \geq \alpha/\alpha^*$ (equivalently $d \geq z\alpha$).*

Proof. By Lemma 3.8, we have $d_{\vec{H}}(A_i, T_\alpha) = \alpha$ for some i. If x is a point, we will write $\vec{d}(x, \cdot)$ to denote $d_{\vec{H}}(\{x\}, \cdot)$. Without loss of generality assume that $d_{\vec{H}}(A_3, T_\alpha) = \alpha$ and $\vec{d}(a, T_\alpha) = d(a, t) = \alpha$ with $a \in A_3$ and $t \in T_\alpha$. Let $T = A_1^{\oplus} \cap A_2^{\oplus} \supseteq T_\alpha$. There is no point $t' \in T$ with $d(t', a) < \alpha$, since then $\vec{d}(t', A_3) < \alpha$, in which case $t' \in A_3^{\oplus}$ and therefore $t' \in T_\alpha$, contradicting that $d_{\vec{H}}(a, T_\alpha) = \alpha$. So t is a point in T closest to a and hence $\vec{d}(a, T) \geq \alpha$.

Assume that $\alpha > 0$ (otherwise we are done) and let H_t be the half-plane (not containing a) bounded by the line through t that is perpendicular to $\text{seg}(t, a)$, see also Figure 3.14. The set T is convex, as it is the intersection of convex sets. Therefore, if T contains a point p, then T also contains $\text{seg}(t, p)$. Since t is a point of T closest to a, no such segment intersects the open disk of radius α centered at a, and therefore $T \subseteq H_t$.

Let C be the circle of radius α centered at t. For the remainder of the proof, let $i \in \{1, 2\}$. Let b_i be a point of A_i closest to t. Then b_i lies on or inside C. If $b_i \neq t$, we can define the half-plane H_i (not containing t) bounded by the line through b_i that is perpendicular to $\text{seg}(t, b_i)$. For $b_i \neq t$, we have by convexity of A_i and b_i being

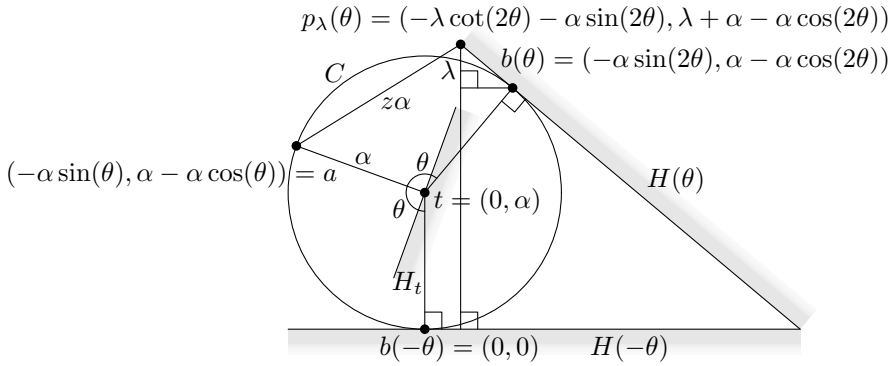


Figure 3.14: Derivation of the expression for z.

closest to t that $A_i \subseteq H_i$, so $\vec{d}(a, A_i) \geq \vec{d}(a, H_i)$. Without loss of generality, assume that $\vec{d}(a, H_i) < \alpha/\alpha^*$ (otherwise $d \geq d_{\vec{H}}(A_3, A_i) \geq \vec{d}(a, A_i) \geq \alpha/\alpha^*$).

If $d(a, b_i) \geq 2\alpha$, then b_i lies diametrically opposite to a on C, but then $\vec{d}(a, H_i) \geq 2\alpha > \alpha/\alpha^*$, which is a contradiction, so $d(a, b_i) < 2\alpha$. Let $t_i \in A_i^{\oplus}$ be the midpoint of b_i and a, then $d(a, t_i) < \alpha = d(a, t)$. If $d(b_1, t) < \alpha$, then T contains a point interior to $\text{seg}(t, t_2)$, contradicting that $\vec{d}(a, T) \geq \alpha$. So b_1 and (analogously) b_2 lie on C.

Let θ_i be the clockwise angle $\measuredangle atb_i \in (-180°, 180°)$. Define $b(\theta)$ to be the point on C for which θ is the clockwise angle $\measuredangle atb(\theta)$, so that $b_i = b(\theta_i)$. Similarly, let $H(\theta)$ be the half-plane (not containing C) bounded by the line tangent to C at $b(\theta)$, so that $H_i = H(\theta_i)$. Assume without loss of generality that $|\theta_1| \geq |\theta_2|$ (otherwise relabel A_1 and A_2). If θ_1 and θ_2 are both positive or both negative, consider the circle of radius $\alpha/2$ centered at the midpoint of a and t. Then t_1 lies on the (shorter) arc of this circle connecting t_2 and t. This arc lies entirely in $A_2^{\oplus}$, so t_1 lies in T_α, which contradicts that there is no point $t' \in T$ with $d(t', a) < \alpha$. So assume without loss of generality that $\theta_2 \leq 0 \leq \theta_1$ (otherwise mirror all points). If $\theta_1 - \theta_2 < 180°$, then T contains the segment between t and the midpoint of b_1 and b_2. This segment does not lie in H_t, which contradicts that $T \subseteq H_t$. Moreover, if $\theta_1 - \theta_2 = 180°$, then b_1 and b_2 are antipodal on C, so $d_H(A_1, A_2) \geq d_H(H(\theta_1), H(\theta_2)) = 2\alpha > \alpha/\alpha^*$. So consider the remaining case where $\theta_1 - \theta_2 > 180°$.

In fact, it will turn out that in the worst case, $\theta_2 = -\theta_1$. Suppose that $p \in A_1 \subseteq H(\theta_1)$ is the point of A_1 closest to a. We have $d \geq d(a, p)$ and $d \geq \vec{d}(p, A_2) \geq \vec{d}(p, H(\theta_2))$. Moreover, since $-\theta_1 \leq \theta_2 < \theta_1 - 180°$, the value of $\vec{d}(p, H(\theta))$ decreases as $\theta \in [-\theta_1, \theta_2]$ decreases. In particular, we have $\vec{d}(p, H(\theta_2)) \geq \vec{d}(p, H(-\theta_1))$. Since $|\theta_1| \geq |\theta_2|$, we have $\theta_1 \in (90°, 180°)$. Let $\lambda_p = \vec{d}(p, H(-\theta_1)) - \vec{d}(b(\theta_1), H(-\theta_1))$. If $\lambda_p < 0$, then $d(a, b(\theta_1)) < d(a, p)$, and p would not be a point of A_1 closest to a because the angle $\measuredangle ab(\theta_1)p$ would be at least 90 degrees. Combining the above lower bounds,

we obtain $d \geq \min\{\max\{d(a,p), \vec{d}(p, H(-\theta_1))\} \mid p \in H(\theta_1), \lambda_p \geq 0\}$. The right hand side of the above inequality is attained for some p on the boundary of $H(\theta_1)$. We parametrize such points p with parameters λ and θ_1: let $p_\lambda(\theta_1)$ be the unique point on the boundary of $H(\theta_1)$ with $\vec{d}(p_\lambda(\theta_1), H(-\theta_1)) = \vec{d}(b(\theta_1), H(-\theta_1)) + \lambda$. The above inequality becomes $d \geq \min_{\lambda \geq 0} \max\{d(a, p_\lambda(\theta_1)), \vec{d}(b(\theta_1), H(-\theta_1)) + \lambda\}$.

We need to minimize this quantity over all values of $\lambda \geq 0$ and $\theta_1 \in (90°, 180°)$. We will show that it is minimized when its terms $d(a, p_\lambda(\theta_1))$ and $\vec{d}(b(\theta_1), H(-\theta_1)) + \lambda$ are equal. The point $p_\lambda(\theta_1)$, and hence the two terms, vary continuously in λ and θ_1. For fixed θ_1, both terms are convex as a function of λ. Therefore, for any fixed θ_1, the function is minimized either when $\lambda = 0$, or the two terms are equal. As θ_1 approaches $180°$, the first term approaches at least 2α (for any λ), and as θ_1 approaches $90°$, the second term approaches at least 2α. Since the optimal value is less than 2α, there exists an optimal value of θ_1. Assume for a contradiction that the terms are not equal in an optimal solution. Fix $\lambda = 0$, and consider the two terms as a function of θ_1. For $\theta_1 \approx 90°$ and $\lambda = 0$, we have $d(a, p_\lambda(\theta_1)) \approx \alpha < 2\alpha \approx \vec{d}(b(\theta_1), H(-\theta_1)) + \lambda$. Conversely for $\theta_1 \approx 180°$ and $\lambda = 0$, we have $d(a, p_\lambda(\theta_1)) \approx 2\alpha > 0 \approx \vec{d}(b(\theta_1), H(-\theta_1)) + \lambda$. Hence, by the intermediate value theorem, the inequality as a function of θ_1 (with fixed $\lambda = 0$) is minimized when the terms are equal. We handled the case with $\lambda > 0$ above, so our inequality becomes $d \geq \min\{\vec{d}(b(\theta_1), H(-\theta_1)) + \lambda \mid \lambda \geq 0, \theta_1 \in (90°, 180°),$ and $d(a, p_\lambda(\theta_1)) = \vec{d}(b(\theta_1), H(-\theta_1)) + \lambda\}$. Following the derivation in Figure 3.14, this corresponds to $d \geq z\alpha = \alpha/\alpha^*$. □

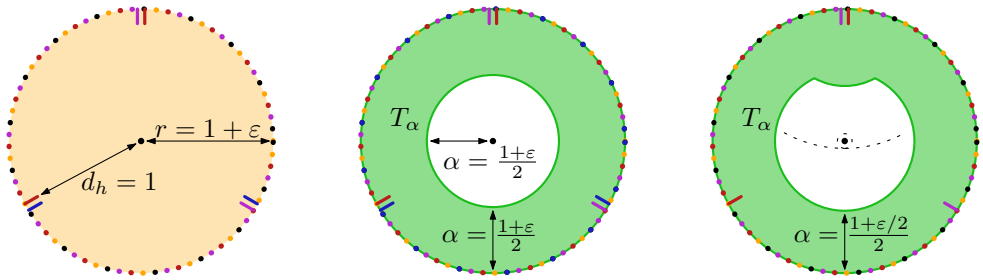
3.3.2 Convexity and connectedness of T_α

In this subsection, we use $\alpha := \alpha(\mathcal{M})$ for simplicity. Similarly to Section 3.2.1, we examine the properties of T_α for different types of input. We arrive at straightforward generalizations of the results obtained for two sets.

1. If all A_i are convex, then T_α is convex.
2. If one of the A_i is connected and the rest are convex, then T_α is connected.
3. For some input where each A_i is connected, and at least two are not convex, T_α is disconnected.

Property 1 follows from the same argument as before: T_α is the intersection of convex sets, and therefore itself convex. Property 3 can be shown by extending the construction from Figure 3.5 with some other sets: if the intersection of two of the sets is not connected, adding more sets will not make T_α connected as long as the pairwise Hausdorff distance does not increase. We establish Property 2 with the following lemma.

Lemma 3.10. *Let $\mathcal{M} = \{A_1, \ldots, A_m\}$ be a set of compact connected regions of the plane, with A_i convex for $i < m$. Then T_α is connected.*



(a) (b) (c)

Figure 3.15: When the input sets are not convex, all sets may be necessary to realize the value of α. Figure (a) shows our input construction, along with the radius of the circle and the Hausdorff distance. Figure (b) shows that when all sets are present, the required value of α is $(1 + \varepsilon)/2$. Figure (c) shows that with the blue set is removed, the required value of α is reduced to $(1 + \varepsilon/2)/2$.

Proof. Consider the set $T'_\alpha = \bigcap_{i=1}^{m-1} A_i^{\oplus}$. This set is convex, as it is the intersection of convex sets. Also note that by definition of T_α, A_m has directed Hausdorff distance at most α to T'_α. Let $A = T'_\alpha$ and $B = A_m$, normalized such that $d_{\vec{H}}(B, A) = 1$. We now apply Lemma 3.3 to A and B, using zero as the value for α. We obtain the result that $T_\alpha = (T'_\alpha \oplus D_0) \cap (A_m \oplus D_\alpha)$ is connected. Note that the Hausdorff distance from A to B may be bigger than one, but this does not matter for the proof of Lemma 3.3. □

3.3.3 Helly-type properties

An interesting question is whether there are any sets in the input that could be removed while maintaining the same optimal value of α. This type of property is sometimes called a Helly-type property, after the concept of a Helly family. To make this precise, we need some definitions. We say a collection $\mathcal{M}$ of m sets is *d-sufficient*, if there is a collection $\mathcal{M}_d \subset \mathcal{M}$ of d sets such that $\alpha(\mathcal{M}) = \alpha(\mathcal{M}_d)$. We remind the reader that we assume the maximum pairwise Hausdorff distance between our input sets is 1.

Lemma 3.11. *For every m, there is a collection $\mathcal{M}$ of m connected sets in the plane that is not $(m-1)$-sufficient.*

Proof. Figure 3.15 depicts a construction of four sets which are not 3-sufficient, which generalizes to more sets. The example has one set that is a disk of radius $1+\varepsilon$ (shown in orange in Figure 3.15(a)), and $m-1$ sets that are circles on the boundary of this disk with $m-1$ protrusions of some small length ε. These protrusions are evenly spaced along the boundary of the disk, and in each location there is a distinct set out

of the $m-1$ sets missing (each subset of size $m-2$ is represented by some protrusion). In the example, we have protrusions containing the red and the blue set, the blue and the purple set, and the red and the purple set. This way, for the case where all sets are present (Figure 3.15(b)), the protrusions don't have any influence on T_α, meaning that $\alpha \geq (1+\varepsilon)/2$ is required to let $T_\alpha^\oplus$ contain the entire disk. However, if we remove one set (other than the orange disk), there will be one protrusion where all sets are now present, meaning it will change the shape of T_α. Because of this, the center of the disk will already be covered with a smaller value of α, namely $(1+\varepsilon/2)/2$. This is shown in Figure 3.15(c): the dotted arc shows the dilation of the bump caused by the protrusion, which covers a part of the disk that would otherwise not be covered (shown as a dashed circle). Note that if we remove the orange disk, it is sufficient to use a value of $\alpha = \varepsilon/2$. Further note that with a minor adaptation, all sets become polygonal and simply connected. □

We have shown that in general, we cannot remove any sets from the input while maintaining the same value of α. However, when all input sets are convex, we can show that there is always a subset of size at most three that has the same optimal value of α.

Lemma 3.12. *Let $\mathcal{M} = \{A_1, \ldots, A_m\}$ be a collection of convex sets. Then there exists a subcollection $\mathcal{M}' \subseteq \mathcal{M}$ of size at most three such that $\alpha(\mathcal{M}) = \alpha(\mathcal{M}')$.*

Proof. Consider growing some value β from $1/2$ to 1. At some point, $T_\beta^\oplus := T_\beta \oplus D_\beta$ contains all sets in $\mathcal{M}$ (i.e. when $\beta = \alpha(\mathcal{M})$). There are two ways in which this can happen: (1) T_β is non-empty for the first time, and immediately the condition holds, or (2) T_β grows, and its dilation now covers the last point of all sets in $\mathcal{M}$. As T_β is convex no new components can appear except for the first, and thus we have only those two cases.

In Case 1, T_β is either a segment or a point; otherwise, $T_{\beta'}$ would have been non-empty for some $\beta' < \beta$. If it is a segment, it is generated by two parallel edges of some $A_i, A_j \in \mathcal{M}$ such that we have $\alpha(\{A_i, A_j\}) = \alpha(\mathcal{M})$. If it is a point, it is the common intersection of the dilation of some number of sets from $\mathcal{M}$; we argue that you can always pick three sets for which β is optimal. Let a be the single point in T_β; consider the vectors $\mathcal{V}$ perpendicular to the boundaries of the dilated input sets intersecting in this point. The vectors $\mathcal{V}$ must positively span the plane:[1] otherwise, all vectors would lie in a common half-plane, and a would not be the first point to appear in T_β. As we are in the plane, there must be a subset $\mathcal{U} \subset \mathcal{V}$ of three vectors that positively span the plane by themselves. The three corresponding sets $A_i, A_j, A_k \in \mathcal{M}$ satisfy $\alpha(\{A_i, A_j, A_k\}) = \alpha(\mathcal{M})$.

In Case 2, as our input sets are convex, T_β itself is also convex. Let $a \in A_i$ be one of the last points of $\mathcal{M}$ to be covered by $T_\beta^\oplus$. As $T_\beta^\oplus$ is convex, a must be on its

[1] We say $v_i \in \mathbb{R}^2$ *span the plane positively*, if for every point $p \in \mathbb{R}^2$ there are some numbers $a_i \in \mathbb{R}^+$ such that $\sum a_i v_i = p$.

boundary; let c be the piece of boundary curve a lies on. This piece of curve is either generated by the dilation of some boundary curve in T_β, or by the dilation of one of its vertices. If it is the dilation of a boundary curve, it can be traced back directly to a boundary curve of some A_j, in which case A_i and A_j have Hausdorff distance 2β, and $\alpha(\{A_i, A_j\}) = \alpha(\mathcal{M})$ for any choice of k. If it can be traced back to a vertex of T_β, this vertex is generated by the intersection of the boundaries of some $A_j^\oplus$ and $A_k^\oplus$, in which case we also have that $\alpha(\{A_i, A_j, A_k\}) = \alpha(\mathcal{M})$. □

Combining the previous lemma with Lemma 3.9, we obtain the following result.

Theorem 3.13. *Let $\mathcal{M} = \{A_1, \ldots, A_m\}$ be a collection of convex regions in the plane, and let $T_\alpha = \bigcap_i A_i^\oplus$. Then $\alpha(\mathcal{M})$ is at most the magic value $\alpha^* \approx 0.6068$.*

3.3.4 Algorithms

For any given collection $\mathcal{M} = \{A_1, \ldots, A_m\}$ of polygons, we want to compute $\alpha(\mathcal{M})$. We present two algorithms, a simple approximation algorithm and a more complex exact algorithm. They both use the same decision algorithm as a subroutine. To be precise, given $\mathcal{M}$ and some α, the decision algorithm decides if $\alpha \leq \alpha(\mathcal{M})$. We first present an algorithm for the decision problem. Then we sketch how they are used in the approximation algorithm and the exact algorithm. We denote all vertices and edges of the A_i as *features* of $\mathcal{M}$.

Decision algorithm Assuming the input has total complexity n, we can test whether a given value of $\alpha \leq \alpha(\mathcal{M})$ as follows. Compute the intersection T_α of the dilations $A_1^\oplus, \ldots, A_m^\oplus$ in $O(n^2 \log n)$ time, using the construction of an arrangement of straight and circular arcs [45, 58]. The set T_α will always have at most quadratic complexity, but it can be disconnected. Next we compute $T_\alpha^\oplus$. We take every connected component T of T_α separately, compute $T^\oplus$, and then compute their union. Since the connected components of T_α are disjoint and can be partitioned into $O(n^2)$ convex pieces, the Minkowski sums of these pieces with D_α form a set of pseudo-disks with total complexity $O(n^2)$, see [71]. It is known that such a union has $O(n^2)$ complexity and can be computed in $O(n^2 \log^2 n)$ time [3, 71]. Thus, we can compute T_α in $O(n^2 \log^2 n)$ time.

Note that $T_\alpha \subseteq A_i^\oplus$, by definition. It remains to test $A_i \subseteq T_\alpha^\oplus$, for each A_i. We test all those containments by a standard plane sweep [25] in $O(n^2 \log n)$ time. As soon as we find any proper intersection between an arc of $\partial(T_\alpha^\oplus)$ and some edge of some ∂A_i, we can stop the sweep and conclude that α needs to be larger. If there were no proper intersections of this type, there were only $O(n^2)$ events (and not $O(n^3)$), including the ones between edges of different ∂A_i. When there are no proper intersections, each shape A_i lies fully inside or outside $T_\alpha^\oplus$. We can test this in $O(n^2 \log n)$ time (replace each A_i by a single point and then test by a plane sweep or planar point

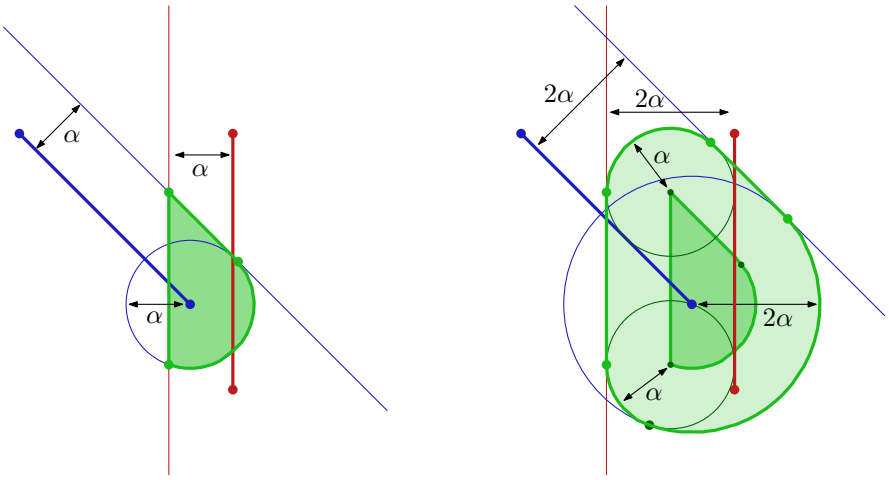


Figure 3.16: Left, two sets shown by red and blue line segments, and the construction of T_α from lines parallel to edges of $\mathcal{M}$ and circles centered at vertices of $\mathcal{M}$. Right, construction of $T_\alpha^\oplus$ from lines at distance 2α from edges of $\mathcal{M}$, circles of radius 2α centered at vertices of $\mathcal{M}$, and circles of radius α centered at certain vertices of T_α.

location [25]), and conclude that α must be larger or smaller than the one tested. Thus this decision algorithm takes $O(n^2 \log^2 n)$ total time.

Approximation algorithm The decision algorithm leads to a simple approximation algorithm to find a value of α that is at most a factor $1+\varepsilon$ from the optimum. We can perform $\lceil \log 1/\varepsilon \rceil$ steps of binary search in the range $[1/2, 1]$, testing if $T_\alpha^\oplus$ contains all A_i using the above decision algorithm. This takes $O(n^2 \log^2 n \log 1/\varepsilon)$ time in total.

Exact computation We can compute an exact value of $\alpha(\mathcal{M})$ in polynomial time. To this end, we imagine a continuous process where we grow α from $1/2$, and keep track of $T_\alpha^\oplus$. The first time (smallest α) $T_\alpha^\oplus$ covers all A_i, we have found the Hausdorff distance $\alpha(\mathcal{M})$ corresponding to the Hausdorff middle, and we can construct T_α explicitly as the Hausdorff middle. Such an approach is sometimes called *wavefront propagation* or *continuous Dijkstra*; it has been used before to compute Voronoi diagrams [25, 49], straight skeletons [6] and shortest paths on terrains [89]. This approach is combinatorial if there are finitely many events and we can determine each on time, before it occurs. Instead of explicitly maintaining $T_\alpha^\oplus$ when α grows, we will determine a polynomial-size set of critical α values that contains the sought one, and find it by binary search, using the decision algorithm described above.

The value $\alpha(\mathcal{M})$ that we aim to compute occurs when $T_\alpha^\oplus$ has grown just enough to cover all A_i. This can happen in three ways, roughly corresponding to a vertex of

A_i becoming covered, an edge of A_i becoming covered at some point "in the middle", or a hole of $T_\alpha^\oplus$ collapsing and disappearing interior to A_i. We call the vertices, edges, and arcs of $\mathcal{M}$ and $T_\alpha^\oplus$ the *features* (of their boundaries). The three ways of covering all A_i, expressed in the features of $\mathcal{M}$ and $T_\alpha^\oplus$, are now: (1) a feature of $T_\alpha^\oplus$ coincides with a vertex of some A_i, (2) a vertex of $T_\alpha^\oplus$ lies on a feature of some A_i, or (3) features of $T_\alpha^\oplus$ collapse and cause a hole of $T_\alpha^\oplus$ to disappear. In the last case, when that hole was inside some A_i, this can be the event where A_i is covered fully for the first time. In all cases, one, two, or three features of $T_\alpha^\oplus$ and zero or one feature of some A_i are involved, and at most three features in total. When three edge or circular arc features pass through a single point for some value of α, we say that these features are *concurrent*. Similarly, when an edge or circular arc passes through a vertex for some α, we say they are concurrent.

It can be that more than three features of $T_\alpha^\oplus$ pass through the point where, e.g., a hole in $T_\alpha^\oplus$ disappears, but then we can still determine this critical value by examining just three features of $T_\alpha^\oplus$, and computing the α value when the curves of these three features are concurrent.

Let us analyze which features make up the boundary of $T_\alpha^\oplus$, see Figure 3.16. There are four types: (1) straight edges, which are at distance 2α from an edge of $\mathcal{M}$, and parallel to it, (2) circular arcs of radius 2α, which are parts of circles centered at vertices of $\mathcal{M}$, (3) circular arcs of radius α, centered at a vertex of T_α, and (4) vertices where features of types (1)–(3) meet. Every one of the features of the boundary of $T_\alpha^\oplus$ is determined by one or two features of $\mathcal{M}$. In particular, each arc of type (3) is centered on an intersection point which is a vertex of T_α, of which there can be $\Theta(n^2)$ in the worst case (Figure 3.5). Depending on the type of intersection point, its trace may be linear in α, or may follow a low-degree algebraic curve (when the intersection has equal distance α to an edge and a vertex of $\mathcal{M}$).

Since any critical value can be determined as a concurrency of two (vertex and edge or arc) or three features (three edges or arcs) from $\mathcal{M}$ and $T_\alpha^\oplus$, and features of $T_\alpha^\oplus$ in turn are determined by up to two features of $\mathcal{M}$, every critical value depends on at most six features of the input $\mathcal{M}$. If we choose any tuple with up to six features of $\mathcal{M}$, and compute the α values that may be critical, we obtain a set of $O(n^6)$ values that contain all critical α values, among which $\alpha(\mathcal{M})$. We can compute this set in $O(n^6)$ time, as it requires $O(1)$ time for each tuple of up to six features of $\mathcal{M}$.

Theorem 3.14. *Let $\mathcal{M}$ be a collection of m polygonal shapes in the plane with total complexity n, such that the Hausdorff distance between any pair is at most 1, and let $\varepsilon > 0$ be a constant. The Hausdorff middle of $\mathcal{M}$ can be computed exactly in $O(n^6)$ time, and approximated within ε in $O(n^2 \log^2 n \log 1/\varepsilon)$ time.*

Parametric search could result in a faster exact algorithm, but for this one would need to express whether input features are close to a given S_α in terms of low-degree polynomials. This is nontrivial given that S_α as a function of α varies in a complex manner.

3.4 Discussion and future research

We have defined and studied the Hausdorff middle of two planar sets, leading to a new morph between these sets. We also considered the Hausdorff middle for more than two sets. While we assumed that the input sets are simply connected, our definition of middle and the morph immediately generalize to more general sets, like sets with multiple components and holes. In this sense our definition of middle is very general. Other interpolation methods between shapes do not generalize to more than two input sets and cannot easily handle sets with multiple components.

There are many interesting open questions. For example, when both input sets are one-dimensional curves, is there a natural way to define a Hausdorff middle curve that is also 1-dimensional?

Besides the maximal middle set, there are other options for a Hausdorff middle. For example, we can choose S_α clipped to the convex hull of $A \cup B$, which is also a valid Hausdorff middle. In Figure 3.11, the green shape would be reduced to the part inside the square, which may be more natural. This Hausdorff middle can also be used in a morph.

Another interesting question could be if, for two shapes A and B, we can find a translation or rigid motion of A such that some measure on the Hausdorff middle (e.g. area, perimeter, diameter) is minimized.

For two or more shapes in the plane, we could also define a middle based on area of symmetric difference. Here we may want to average the areas for the middle shape, and possibly choose the middle that minimizes perimeter. This problem is related to minimum-length area bisection [78].

Similarly, for a set of curves, we could define a middle curve based on the Fréchet distance. This appears related to the Fréchet distance of a set of curves rather than just a pair [44].

Finally, it is worth looking into faster algorithms for finding $\alpha(\mathcal{M})$. As mentioned, using parametric search seems to be a good candidate, but faster algorithms for the decision problem could also be investigated.

Chapter 4

Abstract morphing using the Hausdorff distance and Voronoi diagrams

4.1 Introduction

Morphing, also referred to as shape interpolation, is the changing of a given shape into a target shape over time. Applications include animation and medical imaging. Animation is often motivated by the film industry, where morphing can be used to create cartoons or visual effects. In medical imaging, the objective is a 3D reconstruction from cross-sections, such as those from MRI or CT scans. Reconstruction between two 2D slices is essentially 2D interpolation between shapes, which is a form of morphing. We regard morphing itself as the change of one shape into another shape by a parameter, or, more precisely, a function from the interval $[0, 1]$ to shapes in a space, such that the image at 0 is the one input shape and the image at 1 is the other input shape. It is often convenient to see the morphing parameter as time. In the rest of this chapter, we will refer to the shape of the morph at any particular time value as an *intermediate shape*. See Figure 4.1 for an example of two halfway shapes between polygons resembling a butterfly and a spider.

The quality of a morph depends on the application. For medical imaging, the implied 3D reconstruction must be anatomically plausible. For morphing between two drawings of a cartoon character, the shapes in between must keep the dimensions of the limbs, for example. Furthermore, semantically meaningful features (nose, chin) should morph from their position in the one shape to their position in the other shape.

In this chapter we concentrate on abstract morphing of shapes. A morphing task

Figure 4.1: The intermediate shapes of two different morphing methods at time value $1/2$ when morphing between the input shapes on the left. The middle shows the dilation morph (introduced in the previous chapter), the right shows the Voronoi morph (introduced in this chapter).

is abstract if there is no (semantic) reason to transform certain parts of a starting shape into certain parts of a goal shape. In the previous chapter, we presented a new type of abstract morphing based on the Hausdorff distance. It takes any two compact planar shapes A and B as input, and produces a morphed shape that interpolates smoothly between them. For a time value $\alpha \in [0, 1]$, this morph is equal to A at $\alpha = 0$ and to B at $\alpha = 1$. For any value of α it has Hausdorff distance α to A and Hausdorff distance $1 - \alpha$ to B, if the initial Hausdorff distance is 1 (the input can be scaled to make this true without changing intermediate shapes). Morphs with this property are called *Hausdorff morphs* [80]. The Hausdorff morph introduced in Chapter 3 is based on Minkowski sums with a disk, and hence we refer to this specific one as the *dilation morph*.

While the dilation morph has nice theoretical properties, in practice it will often grow intermediate shapes from A until $\alpha = 1/2$, at which point the greatly dilated shape will shrink back towards B. For α close to $1/2$, the morphed shape typically resembles neither of the input shapes unless they already looked alike. We can see this in Figure 4.1.

In this chapter we present a new Hausdorff morph called *Voronoi morph* that gives a subjectively more visually convincing morph, while maintaining many of the properties of the dilation morph. Our morph uses Voronoi diagrams to partition each input shape into regions with the same closest point on the other shape, and then scales and moves each such region to that closest point based on the value of α. We show that the Voronoi morph is also a Hausdorff morph. It interpolates smoothly between A and B, but does not have the same problem of significantly increasing the area during the morph. We also present a variant called *mixed morph* that reduces the

problem of unnecessarily increasing the perimeter of the interpolated shape. It uses dilation and erosion to overcome some shortcomings of the Voronoi morph.

4.1.1 Related work

The Hausdorff distance is a widely used distance metric that can be used for any two subsets of a space. It is a bottleneck measure: only a maximum distance determines the Hausdorff distance. Efficient algorithms to compute the Hausdorff distance between two simple polygons or their higher-dimensional equivalents exist [8, 9, 18]. The Hausdorff distance is used in computer vision [43] and computer graphics [16, 39] for template matching, and the computation of error between a model and its simplification.

Several algorithmic approaches to morphing have been described. Many of these are motivated by shape interpolation between slices (e.g., [7, 20, 21, 27], an overview can be found in [23]). Other papers discuss morphing explicitly and not as an interpolation problem. Many of these results use compatible triangulations [56, 84], in particular those that avoid self-intersections. It is beyond the scope of this chapter to give a complete overview of morphing methods. For (not so recent) surveys of shape matching, interpolation, and correspondence, see [14, 67]. Our method builds upon the morphing approach given in Chapter 3, which introduced Hausdorff morphs as a new technique for abstract morphing, and the dilation morph as a specific example of a Hausdorff morph.

Another shape similarity measure than the Hausdorff distance, the *Fréchet* distance, can also be used to define a morph. In particular, *locally correct Fréchet matchings* [31] immediately imply a smooth transition of one shape outline into another, because they match all pairs of points on the two curves. Similar approaches were given in [85, 96]. During the transition, however, the outline may be self-intersecting. This problem was addressed in [33, 37]. A more important shortcoming of morphing using the Fréchet distance is that it is unclear how to morph between shapes with different numbers of components and holes.

Much of the commercial software for morphing applies to images, with or without additional human control. Other software is meant as toolkits for designers to design their own morphs, most notably Adobe After Effects.

4.1.2 Our results

We introduce two new abstract morphs based on the Hausdorff distance. They are—just like the dilation morph—conceptually simple and easy to implement if one has code for Minkowski sum and difference with a disk, Voronoi diagrams, and polygon intersection and union. We examine basic properties of the two new morphs and compare how they relate to the dilation morph. In particular, we show that the Voronoi morph is a Hausdorff morph and that it is 1-Lipschitz continuous. We also

show that for any morphing parameter (time), the Voronoi-morph intermediate shape is a subset of the mixed-morph intermediate shape, which in turn is a subset of the dilation-morph intermediate shape.

We then proceed with an extensive experimental analysis where we compare four basic quantities: area, perimeter, number of components, and number of holes. We show how these quantities develop throughout the three morphs. We also present visual results. As data we use simple drawings of animals, country outlines, and text (letters and whole words).

4.2 Preliminaries

Given two sets A and B, we can define the directed Hausdorff distance from A to B as

$$d_{\vec{H}}(A, B) \coloneqq \sup_{a \in A} \inf_{b \in B} d(a, b),$$

where d denotes the Euclidean distance. The *undirected Hausdorff distance* between A and B is then defined as the maximum of both directed distances:

$$d_H(A, B) \coloneqq \max(d_{\vec{H}}(A, B),\ d_{\vec{H}}(B, A)).$$

When A and B are closed sets, we can alternatively define the Hausdorff distance using Minkowski sums. Recall that the Minkowski sum $A \oplus B$ is defined as $\{a+b \mid a \in A,\ b \in B\}$; the directed Hausdorff distance between A and B is then the smallest value r for which $A \subseteq B \oplus D_r$, where D_r is a disk of radius r.

In Chapter 3 we defined a function that interpolates between two shapes in a Hausdorff sense: for any time parameter $\alpha \in [0, 1]$, we defined the dilation morph

$$S_\alpha(A, B) \coloneqq (A \oplus D_\alpha) \cap (B \oplus D_{1-\alpha}),$$

and proved that this shape has Hausdorff distance α to A and $1 - \alpha$ to B, and that it is the maximal shape with this property. Additionally, we showed that this morph is 1-Lipschitz continuous: for two time parameters α and β, $d_H(S_\alpha(A, B), S_\beta(A, B)) \leq |\beta - \alpha|$. Note that we will omit the arguments A, B when they are clear from context.

Structurally, it turns out that the intermediate shapes may have quadratic complexity, even when the input is two simple polygons with n vertices each. For instance, if the input consists of a horizontal comb and an overlapping vertical comb, each with $n/4$ prongs, S_α will consist of $\Omega(n^2)$ components for any $\alpha \in (0, 1)$ (see Figure 3.5). In fact, this is not limited to the dilation morph: *any* intermediate shape with Hausdorff distance α to A and $1 - \alpha$ to B will have $\Omega(n^2)$ components, so every Hausdorff morph has this feature.

4.3 Voronoi morph

As demonstrated in Figure 4.1, one of the problems with the dilation morph is that the intermediate shapes tend to lose any resemblance to the input during the morphing process. The main reason for this is that the dilated shapes we are intersecting contain many points that do not influence the Hausdorff distance in any way, because they are not on the shortest path from a point on one shape to the closest point on the other. In other words, much of S_α can be removed without changing the Hausdorff distance to and from the input. That said, there is no obvious "correct" way to determine which parts should be removed to obtain the greatest resemblance to the input.

We propose a morph in which we only take the points of S_α that are on the shortest path between points in one input shape and the closest point on the other. Specifically, we only take the points where the ratio of distances to the one shape and the closest point on the other is $\alpha : 1-\alpha$. More formally, we define our new morph T_α as follows:

$$T_\alpha(A, B) \coloneqq \{a + \alpha(c(a, B) - a) \mid a \in A\} \cup \{b + (1-\alpha)(c(b, A) - b) \mid b \in B\},$$

where $c(a, B)$ denotes the point on B closest to a. In other words, we move each point in A closer to the closest point in B by a fraction α of that distance, and each point in B closer to the closest point in A by a fraction $1-\alpha$, and take the union of those two shapes. If a point is equidistant to multiple points in the other shape, we include all options. We can prove that this morph has the desired Hausdorff distances to the input.

Theorem 4.1. *Let A and B be two compact sets in the plane with $d_H(A, B) = 1$. Then for any $0 \leq \alpha \leq 1$, we have $d_H(A, T_\alpha) = \alpha$ and $d_H(B, T_\alpha) = 1 - \alpha$.*

Proof. We first show that $d_H(A, T_\alpha) \leq \alpha$, and then show strict equality. The case for $d_H(B, T_\alpha)$ is analogous and therefore omitted.

By construction, any point $a \in A$ has a point at distance at most α in T_α, showing that $d_{\vec{H}}(A, T_\alpha) \leq \alpha$. Similarly, by construction, for each point $b \in B$ there is a point $t_b \in T_\alpha$ such that $t_b = (1-\alpha)(c(b, A) - b)$. As $d(b, c(b, A)) \leq 1$, it must be the case that t_b has distance at most α to $c(b, A)$. It follows that all points in T_α have distance at most α to a point in A, thereby showing that $d_{\vec{H}}(T_\alpha, A) \leq \alpha$. As we have both $d_{\vec{H}}(A, T_\alpha) \leq \alpha$ and $d_{\vec{H}}(T_\alpha, A) \leq \alpha$, it follows that $d_H(A, T_\alpha) \leq \alpha$.

To show strict equality, assume the Hausdorff distance is realised by some point $\hat{a} \in A$ with closest point $\hat{b} \in B$, i.e., $d(\hat{a}, \hat{b}) = 1$. By construction, there is a point $\hat{t} \in T_\alpha$ at distance α from $\hat{a}$ and at distance $1-\alpha$ from $\hat{b}$. As $\hat{t}$ is the closest point to $\hat{a}$ in T_α, we have $d_H(A, T_\alpha) = \alpha$, and as $\hat{b}$ is the closest point to $\hat{t}$ in B, we have $d_H(B, T_\alpha) = 1 - \alpha$. If the Hausdorff distance is realised by a point on B, we use a symmetric argument. □

We can additionally show that the Voronoi morph, like the dilation morph, is 1-Lipschitz continuous in a Hausdorff sense:

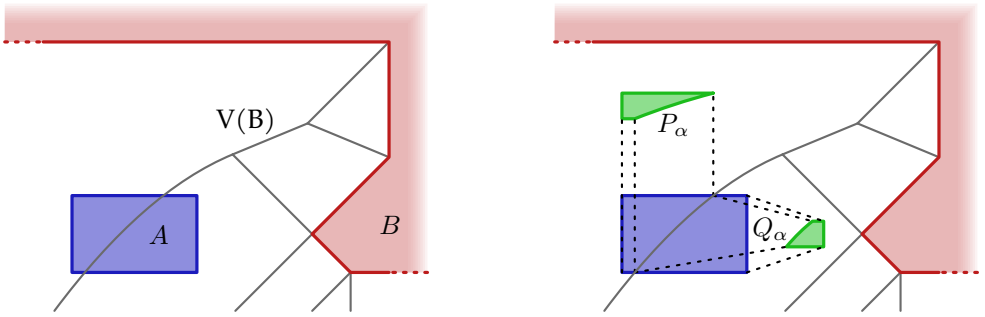


Figure 4.2: On the left, A is partitioned by the Voronoi diagram $V(B)$ of B. On the right, each partitioned part of A, shown in green, is scaled towards the closest point on B by a factor α.

Lemma 4.2. *Let $\alpha, \beta \in [0,1]$. Then $d_H(T_\alpha, T_\beta) \leq |\beta - \alpha|$.*

Proof. Let t_α be any point on T_α. Assume without loss of generality that there is some $a \in A$ such that $t_\alpha = a + \alpha(c(a,B) - a)$ (the case for t_α being included due to a point in B is analogous). Now consider the point $t_\beta = a + \beta(c(a,B) - a)$: t_α and t_β are on the same straight line segment between a and $c(a,B)$, and have distance $|\beta - \alpha| \cdot |c(a,B) - a|$ to each other. As $d_H(A,B) = 1$, we know that $|c(a,B) - a| \leq 1$, and therefore that $|t_\beta - t_\alpha| \leq |\beta - \alpha|$. This holds for any $t_\alpha \in T_\alpha$, and the argument is symmetric for T_β. □

Note that this type of continuity implies that components of T_α can only form or disappear by merging with or splitting from another component.

In addition to the Hausdorff distance-related properties, it is also interesting to study the general geometric and topological properties of T_α. We first show that the number of components $\#C(T_\alpha)$ of T_α does not change during the morph, except possibly at $\alpha = 0$ and $\alpha = 1$. We prove this for the case of polygonal input; the proof can likely be generalised, but the formalisation is somewhat tedious and not particularly interesting.

Let $V(A)$ be the Voronoi diagram of the vertices, open edges and the interior components of A. We now define $\mathrm{Par}(A,B)$ to be the input shape A partitioned into regions by $V(B)$. Note that $\mathrm{Par}(A,B)$ is a set of regions of A that each have the closest point of B on the same vertex, edge or face of B. For some region $P \in \mathrm{Par}(A,B)$, let P_α be the region obtained by scaling P towards the site of the Voronoi cell of B it is in by a factor α. If this site is a vertex, we simply scale P uniformly towards it; if it is an edge, we scale it perpendicular to the supporting line of that edge; and if it is a face, it does not scale or move at all; see Figure 4.2 for an illustration. Now let $\mathrm{Par}_\alpha(A,B) := \{P_\alpha \mid P \in \mathrm{Par}(A,B)\}$. Note that T_α is the union of all elements of $\mathrm{Par}_\alpha(A,B)$ and $\mathrm{Par}_{1-\alpha}(B,A)$.

Lemma 4.3. *Let* $0 < \alpha < \beta < 1$. *Then* $\#C(T_\alpha) = \#C(T_\beta)$.

Proof. Assume that $\#C(T_\alpha) \neq \#C(T_\beta)$. We can assume without loss of generality that $\#C(T_\alpha) > \#C(T_\beta)$, as in the other case we can take $T_\alpha(B, A)$ instead of $T_\alpha(A, B)$ and get the same morph, but parametrised in reverse. We can also assume that for fixed α, β is the smallest value such that $\#C(T_\alpha) > \#C(T_\beta)$. In this case, there are two regions P and Q of $\mathrm{Par}(A, B)$ or $\mathrm{Par}(B, A)$ that are disjoint and in different components of T_α, but intersect and are in the same component of T_β. This is because, as a consequence of Lemma 4.2, components cannot appear or disappear. In the following we assume $P, Q \in \mathrm{Par}(A, B)$; the arguments for when one or both are in $\mathrm{Par}(B, A)$ are identical.

As $P_\beta \cap Q_\beta \neq \emptyset$, there must be some point p in both P_β and Q_β. As both P_β and Q_β are formed by regions moving towards the closest point on the other shape, this point is then on the intersection of two shortest paths between A and B. Let a_1, b_1, a_2 and b_2 be the endpoints of these paths intersecting in p. One of the two segments $\mathrm{seg}(p, b_1,)$, $\mathrm{seg}(p, b_2,)$ will be the shortest; assume without loss of generality that it is $\mathrm{seg}(p, b_1,)$. In this case the path $a_2 p b_1$ is shorter than $a_2 p b_2$, and by the triangle inequality b_1 must be closer to a_2 than b_2.

This contradicts the assumption that b_2 was the closest point to a_2. We conclude that such shortest paths can never intersect, and therefore $P_\alpha \cap Q_\alpha = \emptyset$ for any $\alpha \in (0, 1)$. As such, components can never merge or split for $\alpha \in (0, 1)$, and as they also cannot appear or disappear by Lemma 4.2, the statement in the lemma follows. □

Note that the number of components can change at $\alpha = 0$ or $\alpha = 1$, as in these limit cases elements of $\mathrm{Par}_\alpha(A, B)$ and $\mathrm{Par}_\alpha(B, A)$ turn into points or line segments. Using the strategy from this proof, it also follows that $\mathrm{Par}_\alpha(A, B)$ and $\mathrm{Par}_{1-\alpha}(B, A)$ are interior-disjoint. An interesting corollary of this observation is that the area $|T_\alpha|$ of T_α is bounded from below by $(1 - \alpha)^2|A| + \alpha^2|B|$, which is attained when both shapes are disjoint and all parts are moving to a finite number of points (vertices) on the other shape.

4.3.1 A variant morph

One problem with the Voronoi morph is that it can introduce many slits into the boundary, thereby greatly increasing the perimeter of the shape. This is because parts of the input that have different closest points on the other shape will tend to move away from each other. We present a variant of the Voronoi morph that tries to reduce these problems. As it uses both the Voronoi morph and the dilation morph, we call this variant the *mixed morph*. The mixed morph $M_{\alpha,\varphi}$ is defined as follows:

$$M_{\alpha,\varphi}(A, B) \coloneqq ((T_\alpha(A, B) \oplus D_\varphi) \ominus D_\varphi) \cap S_\alpha,$$

where $\ominus$ is the Minkowski difference, defined as $A \ominus B \coloneqq (A^c \oplus B)^c$, where A^c is the complement of A. Taking a Minkowski sum with a disk is also known as *dilation*, and the Minkowski difference with a disk is known as *erosion*. Performing first a dilation and then an erosion with disks of the same radius is known as *closing*, and can be used to close small gaps and holes in a shape without modifying the rest too much. The closing operator is widely used and studied in the field of image analysis [61].

The resulting morph may no longer be a Hausdorff morph: we may have increased the Hausdorff distance by closing certain gaps or holes. We therefore intersect the closed version of T_α with the dilation morph S_α, so that gaps that are necessary to obtain the appropriate Hausdorff distance are maintained. This results in the mixed morph $M_{\alpha,\varphi}$.

The mixed morph has a new parameter, φ, being the radius of the disk used in the closing. Note that $M_{\alpha,0} = T_\alpha$. We can show that $M_{\alpha,\varphi}$ contains all shapes obtained with the same α but smaller value of φ:

Lemma 4.4. *Let $\varphi, \psi \in \mathbb{R}^+$ and $\varphi \leq \psi$. Then $M_{\alpha,\varphi} \subseteq M_{\alpha,\psi}$.*

Proof. Let us assume that $M_{\alpha,\varphi} \supset M_{\alpha,\psi}$ instead. Then there is some point p such that $p \in M_{\alpha,\varphi}$, but $p \notin M_{\alpha,\psi}$. There are two reasons why p might not be in $M_{\alpha,\psi}$: either $p \notin T_\alpha \oplus D_\psi$, or $p \in T_\alpha \oplus D_\psi$ but $p \notin (T_\alpha \oplus D_\psi) \ominus D_\psi$.

It can clearly not be the case that $p \in M_{\alpha,\varphi}$ but $p \notin T_\alpha \oplus D_\psi$: $M_{\alpha,\varphi}$ is a subset of $T_\alpha \oplus D_\varphi$, and as $\varphi \leq \psi$, we have that $T_\alpha \oplus D_\varphi \subseteq T_\alpha \oplus D_\psi$.

It must then be the case that $p \in T_\alpha \oplus D_\psi$ but $p \notin (T_\alpha \oplus D_\psi) \ominus D_\psi$. In this case, the distance between p and the boundary $\partial T_\alpha^\oplus$ of $T_\alpha \oplus D_\psi$ must be less than ψ. Let $q \in \partial T_\alpha^\oplus$ be the point on the boundary closest to p. As $p \in T_\alpha \oplus D_\varphi$ and $T_\alpha \oplus D_\varphi \subseteq T_\alpha \oplus D_\psi$, the segment $\text{seg}(p, q,)$ must intersect the boundary of $T_\alpha \oplus D_\varphi$ in some point q'. We must have that $d(p, q') \geq \varphi$, or p would not be in $M_{\alpha,\varphi}$, and we must have $d(q, q') \geq \psi - \varphi$, as $T_\alpha \oplus D_\psi = (T_\alpha \oplus D_\varphi) \oplus D_{\psi-\varphi}$. But then, by the triangle inequality, $d(p, q) \leq d(p, q') + d(q, q') \geq \psi$, which is a contradiction. Hence, $p \in M_{\alpha,\psi}$. As this holds for all $p \in M_{\alpha,\varphi}$, the statement in the lemma follows. □

Note that this means we now have the following hierarchical containment of morphs: $T_\alpha \subseteq M_{\alpha,\varphi} \subseteq M_{\alpha,\psi} \subseteq S_\alpha$, for $\varphi \leq \psi$. As T_α is a Hausdorff morph, and S_α is the maximal Hausdorff morph, this shows that $M_{\alpha,\varphi}$ is a Hausdorff morph as well. However, $M_{\alpha,\varphi}$ is not 1-Lipschitz continuous: components may suddenly merge when their distance falls below 2φ.

4.3.2 Algorithm

To give an algorithm for computing T_α, we assume A and B are (sets of) polygons, possibly with holes. As T_α is based on moving all points on the one shape to the closest point on the other shape, we can compute the Voronoi diagram of each input shape, and then use these to partition the other shapes. This gives us a partitioning of A into pieces that overlap B, or have the same closest point or edge on B, and vice

versa. Pieces of A completely inside B are unchanged, pieces with a vertex as closest element are scaled uniformly towards that vertex by a factor α, and pieces with an edge as closest element are scaled perpendicular to the supporting line of that edge by a factor α. For pieces of B we do the same, except that we scale them with a factor $1 - \alpha$. Figure 4.2 shows an example of how a shape A is partitioned by the Voronoi diagram $V(B)$ of B, and each piece is scaled towards the closest point on B.

Given this algorithm, we can also straightforwardly compute $M_{\alpha,\varphi}$ by computing T_α and S_α, dilating and eroding T_α by a distance φ, and then intersecting the result with S_α.

Our computations rely solely on Voronoi diagrams of segments, Minkowski sums and differences with disks, intersections and unions of polygons, all of which can be found in standard books or surveys [3, 19, 25] and an intermediate shape can be calculated in $O(n^2 \log n)$ time.

4.4 Experiments

We compare the dilation, Voronoi and mixed morphs experimentally on three data sets. The first data set is a collection of outlines of animals taken from [29]. The second is a selection of the outlines of European countries obtained from the Thematic Mapping World Borders data set;[1] we use the outlines of Austria, Belgium, Croatia, Czechia, France, Germany, Greece, Ireland, Italy, the Netherlands, Poland, Spain and Sweden. For these two sets we compute the morphs for all pairs of animals and all pairs of countries in the sets. None of the three morphs is translation-invariant or scale-invariant, so it matters where we place the shapes with respect to each other and what sizes they initially have. We choose to scale the shapes to have the same area and translate them to have a common centroid.

The third data set is a small collection of words and letters manually traced as polygons. We use three pairs of words (wish/luck, kick/stuff, try/it), and the letters f, i and u in a serif and a sans serif font. Observe that our morphs could in theory be used to define an infinite family of fonts by interpolating between the glyphs of each element. For these experiments we do not scale the shapes but use the font size, and we align them manually.

For each experiment, we measure the area, perimeter, number of components and number of holes of the morph for α values starting at zero and increasing in steps of $1/8$. The parameter φ of the mixed morph was universally set to 0.02 based on preliminary experimentation.

It is not necessarily insightful to compare areas and especially perimeters between experiments. To make the results more comparable, we make the assumption that an ideal morph linearly interpolates the area and perimeter between those of the input shapes. For each experiment, we can then give the ratio between the measured area

[1] `http://www.thematicmapping.org/downloads/world_borders.php`

and perimeter and these "ideal" values. For the number of components and holes this is less meaningful, as these are discrete values, so we simply record the numbers directly.

Each morphing method was implemented in C++, using Boost[2] to calculate intersections and unions of polygons, Voronoi diagrams, and Minkowski sums. Although efficiency is not the focus of our study, running all our experiments only took a few minutes in total.

4.5 Results

A summary of our measurements of area and perimeter can be seen in Tables 4.1 and 4.2. A summary of the number of components and holes for only the animals data set can be seen in Table 4.3; we exclude the other data sets because the inputs have different numbers of components. Topological measurements for all experiments can be viewed in Table 4.4. We note that the Voronoi and mixed morphs sometimes have spurious holes caused by numerical precision issues (e.g., the Voronoi morph should not have an intermediate shape with five holes in our experiment with the letter i). Animations of the different morphs for each experiment can be viewed online.[3]

In Figure 4.3 we can see that the average area of the dilation morph quickly grows as α increases, until reaching its peak at $\alpha = 1/2$, to about three times the desired size. For the perimeter we see the opposite trend, with the dilation morph typically having a lower perimeter than desired. This is a consequence of the dilation erasing details in the boundary of the input shapes. We can see in Figure 4.4 that this happens more quickly in the experiments with country shapes. This is expected, as most of the country shapes have more sharp coastline features and islands that quickly disappear, whereas the animal shapes are generally smoother and only have one component.

Our Voronoi morph on average has an area that is much closer to the desired value, and with much lower variance than the dilation morph. However, we see that on average the perimeter is much higher than the desired value. This is because points on opposite sides of a Voronoi edge move in different directions, causing new boundaries to appear in the interior of a shape as soon as $\alpha > 0$. We can see this happen in the middle column of Figure 4.5, and this is reflected in Figure 4.4, where we see the perimeter sharply increase and then stay mostly the same, before sharply dropping back down.

Our mixed morph achieves its purpose of reducing the perimeter of the Voronoi morph: the measured perimeters are close to the desired values, while the measured areas stay comparable to those of the Voronoi morph. In Figure 4.4, we see that the perimeter typically still increases during the morphing process, but does not jump up sharply as soon as $\alpha > 0$. This is because the small value of φ lets us close only

[2] `https://www.boost.org`

[3] `https://hausdorff-morphing.github.io`

Table 4.1: The distributions of areas for each morphing method over all experiments for all nine tested values of α, separated by experiment category.

	Dilation		Voronoi		Mixed	
Category	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Animals	1.977	0.763	0.969	0.024	0.986	0.019
Countries	2.249	1.498	0.960	0.039	0.987	0.039
Text	2.118	1.046	0.980	0.035	0.989	0.028

Table 4.2: The distributions of perimeters for each morphing method over all experiments for all nine tested values of α, separated by experiment category.

	Dilation		Voronoi		Mixed	
Category	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Animals	0.857	0.137	1.725	0.432	1.183	0.155
Countries	0.876	0.237	1.610	0.471	1.129	0.184
Text	0.955	0.142	1.401	0.418	1.155	0.192

Table 4.3: The distributions of the number of components and holes for each morphing method for all tested values of α except 0 and 1. This only includes the animals data set, as these shapes have only one component and no holes.

	Dilation		Voronoi		Mixed	
Category	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Components	1.004	0.063	18.556	8.089	5.317	3.213
Holes	0.218	0.602	2.544	2.699	0.218	0.532

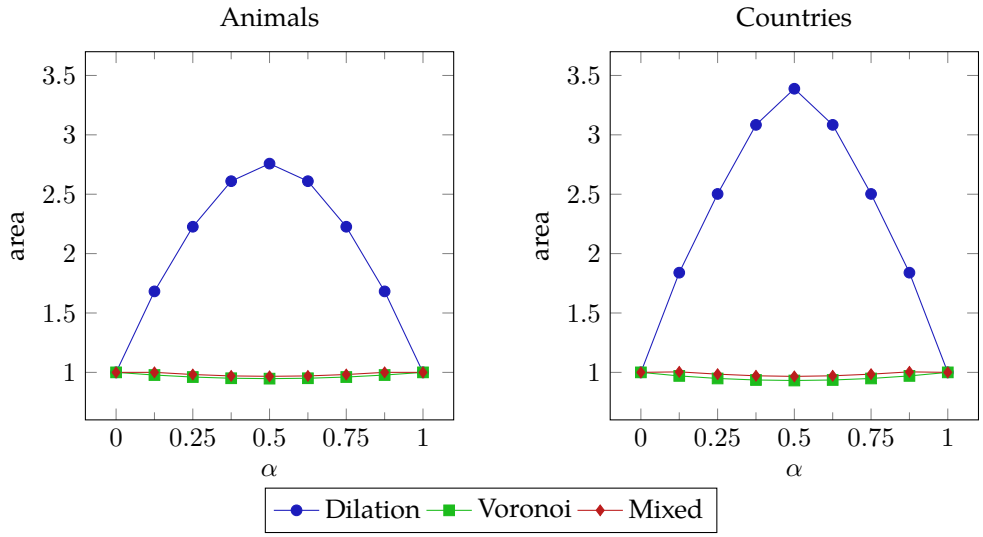


Figure 4.3: The average area over all experiments as a function of α, for both the animals and countries data sets.

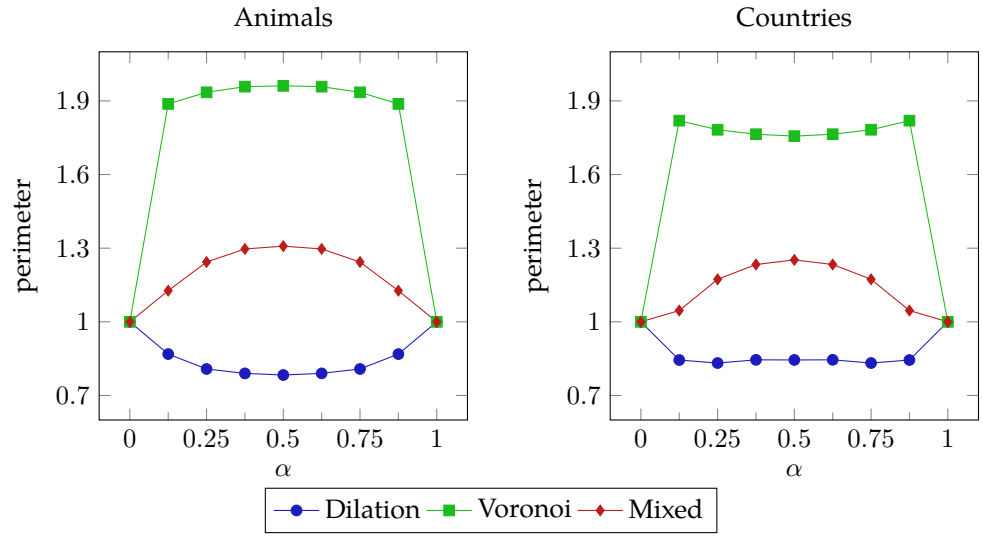


Figure 4.4: The average perimeter over all experiments as a function of α, for both the animals and countries data sets.

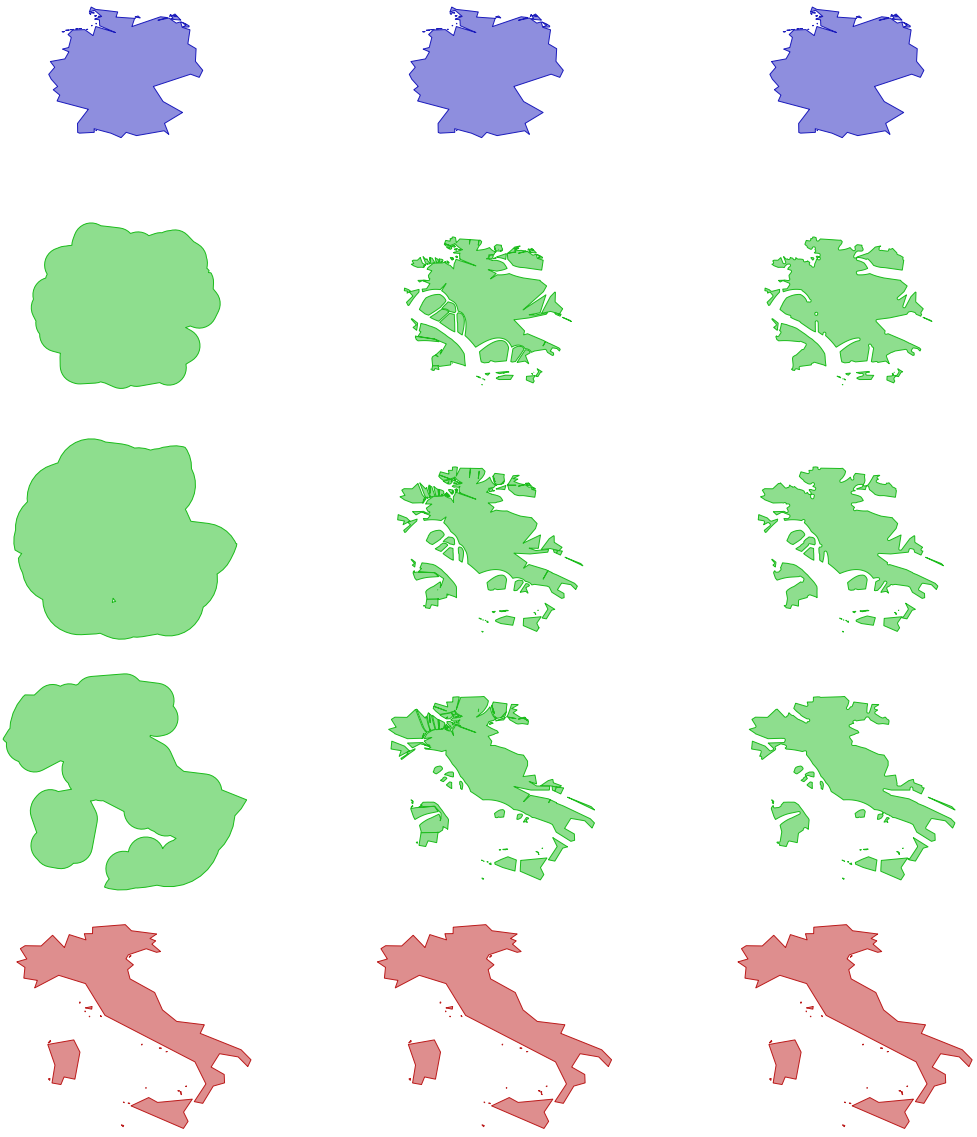


Figure 4.5: Intermediate shapes for $\alpha \in \{0, 1/4, 1/2, 3/4, 1\}$ when morphing between the outlines of Germany and Italy. The columns show the dilation morph, Voronoi morph and mixed morph from left to right.

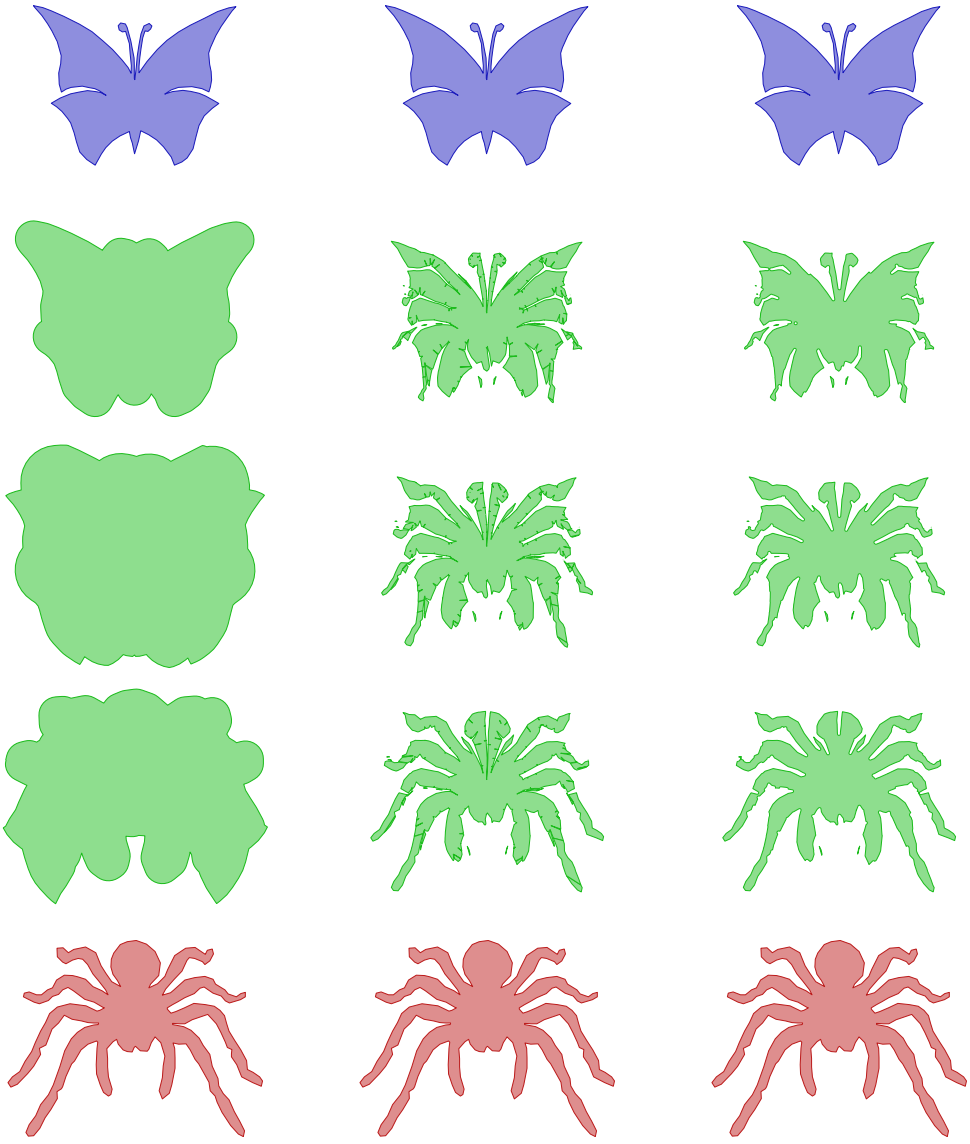


Figure 4.6: Intermediate shapes for $\alpha \in \{0, 1/4, 1/2, 3/4, 1\}$ when morphing between the outlines of a butterfly and a spider. The columns show the dilation morph, Voronoi morph and mixed morph from left to right.

Figure 4.7: Intermediate shapes for $\alpha \in \{0, 1/4, 1/2, 3/4, 1\}$ when morphing between the outlines of a bird and an ostrich. The columns show the dilation morph, Voronoi morph and mixed morph from left to right.

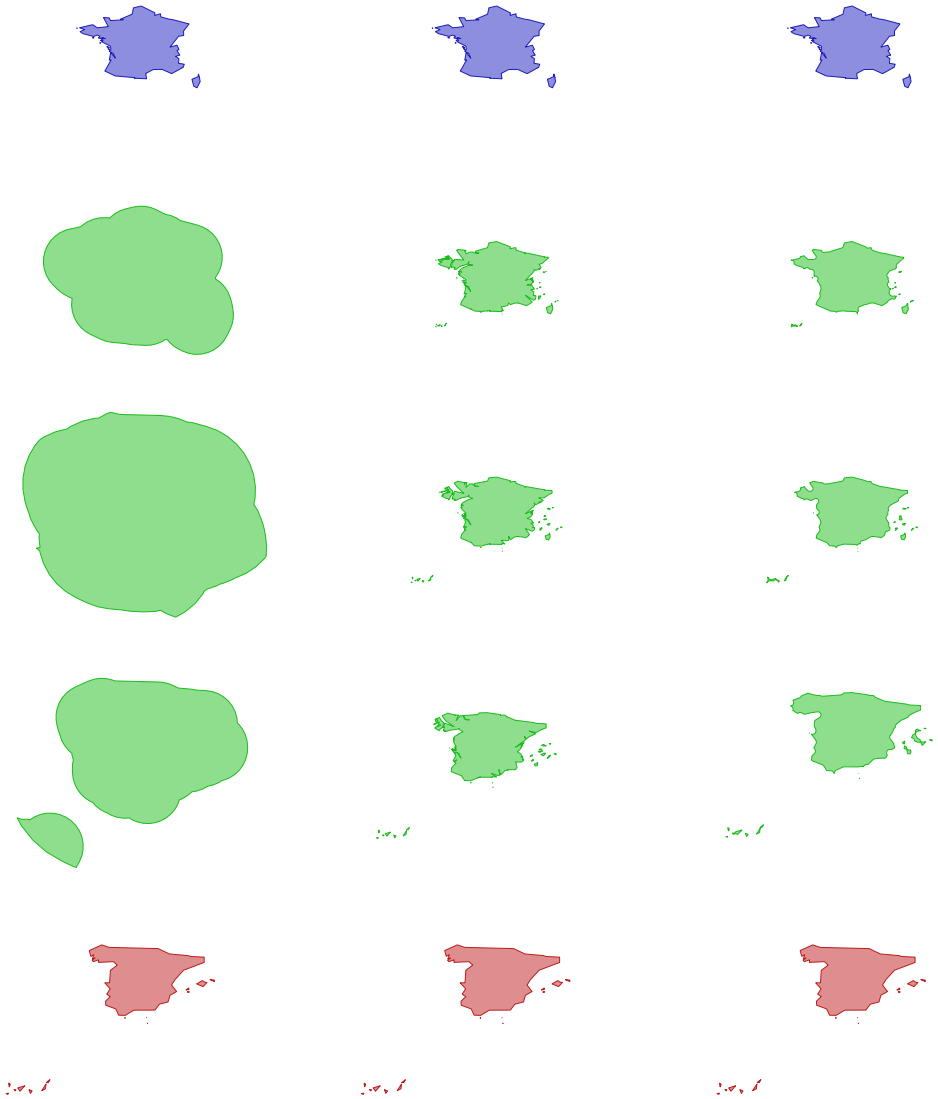


Figure 4.8: Intermediate shapes for $\alpha \in \{0, 1/4, 1/2, 3/4, 1\}$ when morphing between the outlines of France and Spain. The columns show the dilation morph, Voronoi morph and mixed morph from left to right.

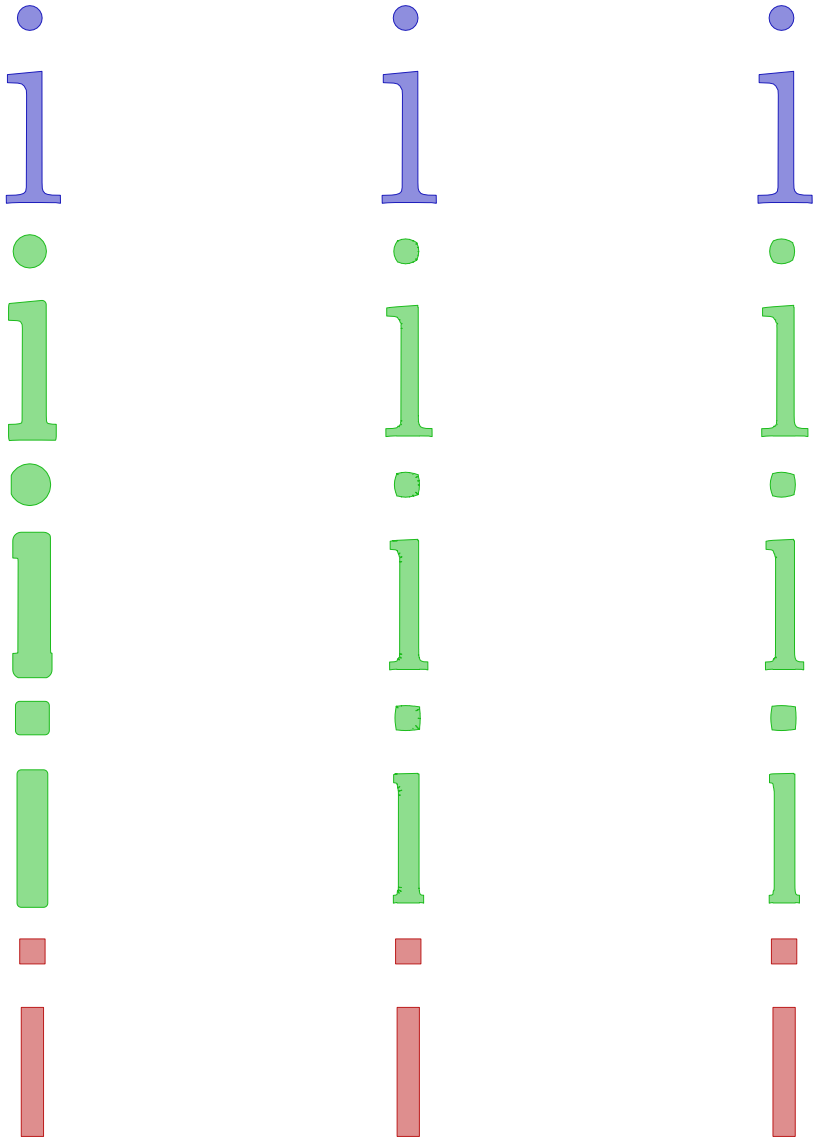


Figure 4.9: Intermediate shapes for $\alpha \in \{0, 1/4, 1/2, 3/4, 1\}$ when morphing between the outlines of the letter i in two different fonts. The columns show the dilation morph, Voronoi morph and mixed morph from left to right. Note that some artefacts in the Voronoi and mixed morphs, such as on the i's dot, are caused by having polygonal input instead of smooth curves.

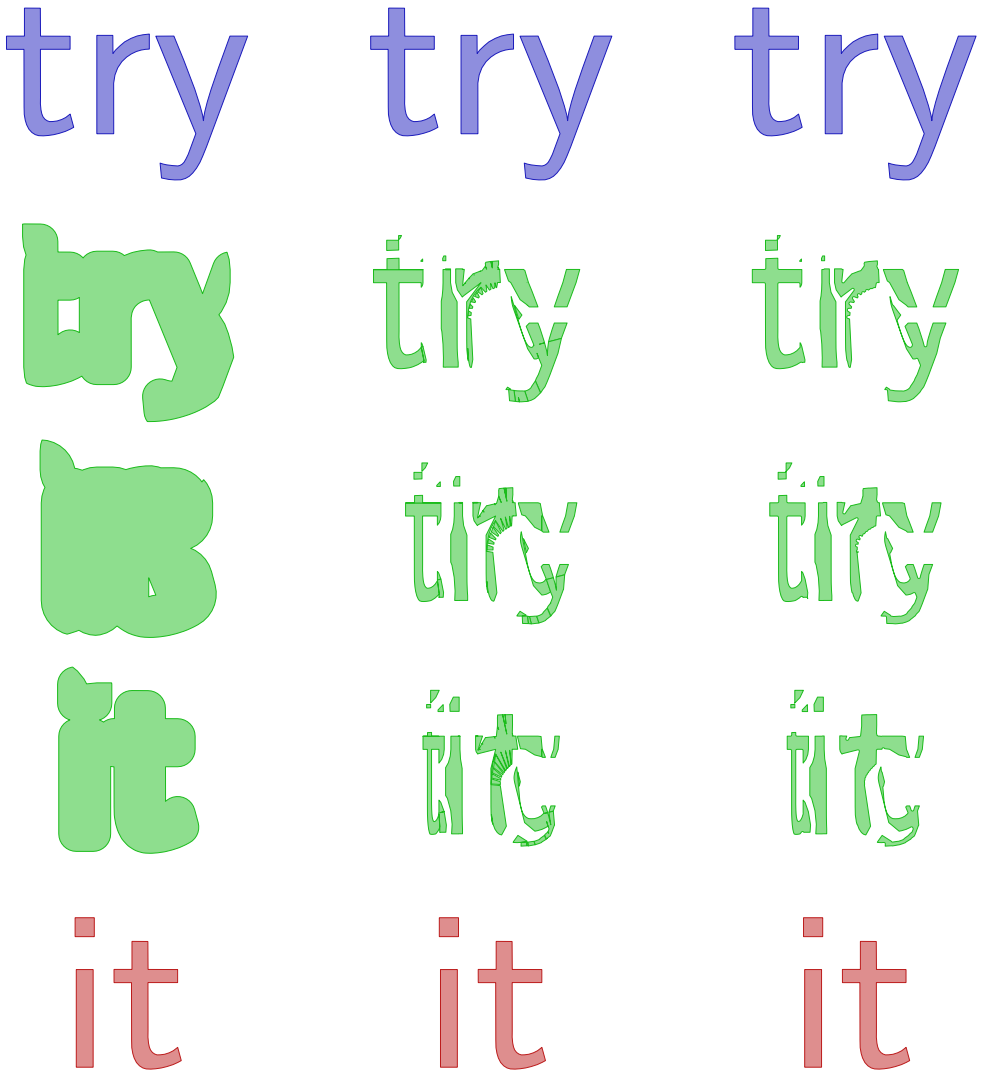


Figure 4.10: Intermediate shapes for $\alpha \in \{0, 1/4, 1/2, 3/4, 1\}$ when morphing between the outlines of the words try and it. The columns show the dilation morph, Voronoi morph and mixed morph from left to right. Note that some artefacts in the Voronoi and mixed morphs, such as in the curved part of the letter r, are caused by having polygonal input instead of smooth curves.

the narrow gaps that appear around the edges of the Voronoi diagram, but not the gaps that develop as pieces of the shapes move apart significantly. We can see this when comparing the middle and right columns of Figure 4.5: fewer gaps are closed at $\alpha = 1/2$ than at the other time values.

In addition to area and perimeter, we also tracked the number of components and holes for each morph type. We observe that for the dilation morph, there is an intermediate shape with only one component in all but one of our experiments (see Table 4.4), showing that this morph tends to turn everything into a blob during the morphing process. On the other hand, the Voronoi morph tends to have an intermediate shape with a number of components much larger than either of the input shapes. The mixed morph exhibits neither of these behaviours. This is illustrated in Figure 4.5.

Table 4.4: The minimum and maximum number of components and the maximum number of holes for each experiment, separated by morph type.

	Dilation			Voronoi			Mixed		
Experiment	min	max	holes	min	max	holes	min	max	holes
bird → butterfly	1	1	0	1	11	6	1	4	1
bird → cat	1	1	1	1	12	6	1	5	0
bird → dog	1	1	1	1	23	4	1	7	1
bird → horse	1	2	1	1	27	6	1	8	2
bird → ostrich	1	1	0	1	26	5	1	12	0
bird → shark	1	1	0	1	14	4	1	6	0
bird → spider	1	1	0	1	30	8	1	9	1
bird → turtle	1	1	0	1	21	3	1	9	1
butterfly → cat	1	1	1	1	6	2	1	3	1
butterfly → dog	1	1	1	1	9	4	1	4	1
butterfly → horse	1	1	1	1	28	3	1	9	2
butterfly → ostrich	1	1	1	1	11	7	1	4	2
butterfly → shark	1	1	0	1	8	4	1	3	1
butterfly → spider	1	1	1	1	30	9	1	10	2
butterfly → turtle	1	1	0	1	13	6	1	3	2
cat → dog	1	1	1	1	17	1	1	3	1
cat → horse	1	1	1	1	17	2	1	5	1
cat → ostrich	1	1	0	1	13	4	1	4	0
cat → shark	1	1	0	1	9	0	1	5	0
cat → spider	1	1	2	1	29	3	1	7	3
cat → turtle	1	1	0	1	14	1	1	7	0
dog → horse	1	1	2	1	31	4	1	9	1

Table 4.4: (continued from last page)

	Dilation			Voronoi			Mixed		
Experiment	min	max	holes	min	max	holes	min	max	holes
dog → ostrich	1	1	1	1	22	2	1	7	1
dog → shark	1	1	1	1	17	3	1	6	1
dog → spider	1	1	3	1	32	2	1	14	2
dog → turtle	1	1	1	1	16	1	1	6	0
horse → ostrich	1	1	2	1	27	7	1	15	1
horse → shark	1	1	1	1	22	5	1	7	1
horse → spider	1	1	4	1	38	3	1	9	1
horse → turtle	1	1	1	1	22	4	1	12	0
ostrich → shark	1	1	0	1	15	8	1	5	0
ostrich → spider	1	1	4	1	38	9	1	17	0
ostrich → turtle	1	1	0	1	21	12	1	4	0
shark → spider	1	1	0	1	23	2	1	5	2
shark → turtle	1	1	0	1	11	3	1	3	0
spider → turtle	1	1	4	1	25	14	1	8	3
austria → belgium	1	1	0	1	2	2	1	1	0
austria → croatia	1	19	0	1	24	2	1	19	3
austria → czechia	1	1	0	1	2	2	1	1	0
austria → france	1	10	0	1	16	2	1	10	1
austria → germany	1	20	0	1	20	2	1	20	0
austria → greece	1	68	4	1	79	2	1	68	2
austria → ireland	1	4	0	1	12	1	1	4	1
austria → italy	1	22	1	1	33	1	1	22	1
austria → netherlands	1	9	1	1	15	2	1	9	2
austria → poland	1	1	0	1	2	2	1	2	0
austria → spain	1	15	0	1	23	2	1	15	1
austria → sweden	1	19	0	1	26	2	1	19	0
belgium → croatia	1	19	1	1	26	3	1	19	4
belgium → czechia	1	1	0	1	1	3	1	1	0
belgium → france	1	10	0	1	13	1	1	10	2
belgium → germany	1	20	0	1	21	2	1	20	2
belgium → greece	1	68	6	1	81	1	1	68	4
belgium → ireland	1	4	0	1	9	1	1	4	2
belgium → italy	1	22	1	1	30	1	1	22	1
belgium → netherlands	1	9	1	1	13	3	1	9	2
belgium → poland	1	1	0	1	2	2	1	1	0
belgium → spain	1	15	0	1	18	2	1	15	0

Table 4.4: (continued from last page)

	Dilation			Voronoi			Mixed		
Experiment	min	max	holes	min	max	holes	min	max	holes
belgium → sweden	1	19	2	1	25	1	1	19	1
croatia → czechia	1	19	0	1	24	3	1	19	4
croatia → france	1	19	2	10	47	1	8	22	1
croatia → germany	1	20	0	19	55	0	6	20	3
croatia → greece	1	68	3	19	121	4	19	68	4
croatia → ireland	1	19	0	4	34	5	3	19	4
croatia → italy	1	22	1	19	67	7	19	30	1
croatia → netherlands	1	19	1	9	39	2	7	19	6
croatia → poland	1	19	0	1	34	1	1	19	2
croatia → spain	1	19	0	15	51	1	6	19	0
croatia → sweden	1	19	1	19	61	8	10	19	3
czechia → france	1	10	0	1	13	1	1	10	1
czechia → germany	1	20	0	1	22	1	1	20	2
czechia → greece	1	68	4	1	85	0	1	68	2
czechia → ireland	1	4	0	1	7	1	1	4	2
czechia → italy	1	22	1	1	25	1	1	22	0
czechia → netherlands	1	9	2	1	12	1	1	9	2
czechia → poland	1	1	0	1	2	1	1	1	0
czechia → spain	1	15	0	1	16	1	1	15	0
czechia → sweden	1	19	1	1	30	1	1	19	2
france → germany	1	20	0	10	39	6	4	20	2
france → greece	1	68	2	10	108	6	10	68	5
france → ireland	1	10	0	4	20	3	3	10	2
france → italy	1	22	1	10	52	6	10	25	0
france → netherlands	1	10	1	9	25	4	6	11	2
france → poland	1	10	0	1	11	12	1	10	2
france → spain	1	15	0	10	38	5	4	16	0
france → sweden	1	19	1	10	39	4	8	19	1
germany → greece	1	68	4	20	104	30	9	68	2
germany → ireland	1	20	0	4	31	6	4	20	3
germany → italy	1	22	1	20	57	26	11	22	2
germany → netherlands	1	20	1	9	43	14	6	20	3
germany → poland	1	20	2	1	26	6	1	20	2
germany → spain	1	20	0	15	39	16	2	20	0
germany → sweden	1	20	1	19	48	18	11	20	5
greece → ireland	1	68	4	4	79	12	3	68	4

Table 4.4: (continued from last page)

	Dilation			Voronoi			Mixed		
Experiment	min	max	holes	min	max	holes	min	max	holes
greece → italy	1	68	3	22	131	19	22	68	3
greece → netherlands	1	68	6	9	85	21	7	68	5
greece → poland	1	68	6	1	84	17	1	68	2
greece → spain	1	68	0	15	111	11	14	68	3
greece → sweden	1	68	3	19	125	11	12	68	2
ireland → italy	1	22	1	4	36	4	4	22	2
ireland → netherlands	1	9	2	4	21	4	3	10	3
ireland → poland	1	4	0	1	7	8	1	4	2
ireland → spain	1	15	0	4	25	4	3	15	0
ireland → sweden	1	19	0	4	31	4	4	19	3
italy → netherlands	1	22	2	9	42	15	9	22	2
italy → poland	1	22	1	1	29	14	1	22	1
italy → spain	1	22	0	15	51	12	12	22	0
italy → sweden	1	22	1	19	68	13	13	22	0
netherlands → poland	1	9	3	1	13	6	1	9	2
netherlands → spain	1	15	0	9	29	5	5	15	2
netherlands → sweden	1	19	2	9	39	5	6	19	2
poland → spain	1	15	0	1	17	6	1	15	0
poland → sweden	1	19	1	1	29	5	1	19	1
spain → sweden	1	19	0	15	47	3	4	19	0
wish → luck	1	5	2	4	44	5	4	22	0
kick → stuff	1	5	3	5	29	6	5	18	0
try → it	1	3	1	3	27	4	3	11	0
f serif → f sans	1	1	0	1	1	3	1	1	0
i serif → i sans	2	2	0	2	2	5	2	2	0
u serif → u sans	1	1	0	1	1	2	1	1	0

Inspecting the morphs visually (Figures 4.5–4.10), our mixed morph looks quite reasonable, especially when the area of symmetric difference between the input shapes is small. In many cases, the intermediate shape at $\alpha = 1/2$ is a recognisable mix of the two input shapes. This is not the case for the dilation morph, where the Hausdorff distance needs to be very small compared to the size of the input shapes for it to look good. For instance, when one shape has some small islands far away, the dilation morph will grow to have a very large area, whereas with the Voronoi and mixed morphs, the islands just slowly move towards the closest point on the other

shape; see Figure 4.8. However, both the Voronoi and mixed morph can still look bad when the area of symmetric difference is large. It may therefore be best to align the input shapes such that the area of symmetric difference is minimised, rather than simply aligning the centroids.

The morphs generally look less convincing on our experiments with text, as the shapes can be very different. For single letters (Figure 4.9) the morphs can look convincing, but when morphing between words, especially of different numbers of letters, the intermediate shape at $\alpha = 1/2$ does not necessarily resemble both input shapes (Figure 4.10). However, the intermediate shapes at $\alpha = 1/4$ and $\alpha = 3/4$ still do clearly resemble input shapes A and B, respectively, for the Voronoi and mixed morphs, but less so for the dilation morph. A better approach to morphing text may be to morph on a per-letter basis, rather than treating the whole text as a single shape. Some strategy would then have to be devised that determines which letter will morph to which, and how to deal with different Hausdorff distances between the letter pairs.

Both the Voronoi morph and the mixed morph often have small parts separating, moving, and then merging somewhere else (for example, the beak in the bird-to-ostrich morphs on `https://hausdorff-morphing.github.io`). Such artifacts may be circumvented by choosing a slightly warped Voronoi diagram, but this upsets the simplicity of the current methods. We can sometimes notice in the animations that the mixed morph is indeed not Lipschitz continuous, but since φ is rather small, this does not show clearly.

4.6 Conclusion

We introduced a new abstract morphing method based on Voronoi diagrams. This new method satisfies the same bounds on the Hausdorff distance as the previously introduced dilation morph, and is also 1-Lipschitz continuous. We have shown experimentally that the intermediate shapes of the Voronoi morph have areas that more closely match those of the input shapes than the dilation morph, but tends to have a perimeter that is larger than desired. To remedy this, we introduced a variant morph, the mixed morph, that we experimentally show to reduce this problem of increasing the perimeter. This mixed morph still satisfies the bounds on the Hausdorff distance, but is no longer 1-Lipschitz continuous. Our experimental analysis is the first we are aware of that analyses the development of area, perimeter, number of components and number of holes throughout the morphs.

An interesting open question is whether we can prevent the increase in perimeter caused by the Voronoi morph without losing 1-Lipschitz continuity. This would require somehow anticipating the moment when two pieces of boundary will meet, and smoothly bridging the gap between them over time, instead of just instantly filling it. To optimise the mixed morph, we can study the effects of choosing different φ, or even changing φ throughout the morph. Another direction is to develop other morphs that guarantee a smooth change of some distance measure other than the

Hausdorff distance; we noted that it is unclear how to employ the Fréchet distance for morphing in the presence of multiple components.

A more practically oriented direction for further research would be to develop a less naive method of filling gaps than the mixed morph. It does not necessarily make sense to use the same radius for the closing operator everywhere, which sometimes closes gaps that will be opened again. However, any adaptation of this type will disrupt the conceptual simplicity of the Voronoi and mixed morphs.

Chapter 5

Reconstructing graphs from connected triples

5.1 Introduction

Imagine that we get information on a graph, but not its complete structure by a list of edges. One natural question that arises is whether we can determine the graph uniquely based on this information. In this chapter we explore the case where the input consists of all triples of vertices whose induced subgraph is connected. In other words, we know for each given triple of vertices that two or three of the possible edges are present, but we do not know which ones. We may be able to deduce the graph fully from all given triples.

As a simple example, assume we are given the (unordered) triples abc, bcd, and cde. Then the only (connected) graph that matches this specification by triples is the path a—b—c—d—e. On the other hand, if we are given all triples on four vertices a, b, c, d except for abc, then there are several graphs possible. We must have the edges ad, bd, and cd, and zero or one of the edges ab, bc, and ca.

This model of indeterminacy of a graph is perhaps the simplest combinatorial model for partial information, a model that does not use probability. Normally a graph is determined by pairs of vertices which are the edges; now we are given triples of vertices with indeterminacy on the edges between them. As such, we believe this model is interesting to study.

As pointed out, there are cases where reconstruction of the graph from the set T of triples is unique and there are cases where it is ambiguous. There are also cases where T is not consistent with any graph, like $T = \{abc, cde\}$. Can we characterize these cases, and what can we say if we have additional information, for example, when we know that we are reconstructing a tree?

5.1.1 Related work

The problem of graph reconstruction arises naturally in many cases where some unknown graph is observed indirectly. For instance, we may have some (noisy) measurement of the graph structure, or only have access to an oracle that answers specific types of queries. Much previous research has been done for specific cases, such as reconstructing metric graphs from a density function [42], road networks from a set of trajectories [4], graphs using a shortest path or distance oracle [68], labelled graphs from all r-neighbourhoods [93], or reconstructing phylogenetic trees [30]. A lot of research has been devoted to the *graph reconstruction conjecture*, which states that it is possible to reconstruct any graph (up to isomorphism) from all subgraphs obtained through the removal of one vertex [28, 82, 94, 108].

Many different types of uncertainty in graphs have been studied. Fuzzy graphs [95] are a generalisation of fuzzy sets to relations between elements of such sets. In a fuzzy set, membership of an element is not binary, but a value between zero and one. Fuzzy graphs extend this notion to the edges, which now also have a degree of membership in the set of edges. Uncertain graphs are similar to fuzzy graphs in that each edge has a number between zero and one associated with it, although here this number is a probability of the edge existing. Much work has been done on investigating how the usual graph-theoretic concepts can be generalised or extended to fuzzy and uncertain graphs [70, 92]. Methods for drawing these types of graphs have also been developed, see e.g. [98, 100].

5.1.2 Our results

After some preliminaries in Section 5.2, we provide two relatively straightforward, general algorithms for reconstruction in Section 5.3. One runs in $O(n^3)$ time when the triples use n vertex labels, and the other runs in $O(n \cdot |T|)$ time when there are $|T|$ triples in the input. These algorithms return a graph that is consistent with the given triples, if one exists, and decide on uniqueness. In Section 5.4 we show that trees can be reconstructed uniquely, provided that we know that the unknown graph is a tree. For this case we give an $O(|T|)$ time algorithm. Section 5.5 continues this study by showing unique reconstruction of 2-connected outerplanar graphs, triangulated planar graphs, and graphs where all cycles have length at least 5. Section 5.6 contains a further study of ambiguity and captures the structure of (non-)uniquely reconstructible graphs partially. In Section 5.7 we study a natural extension of the model where not triples, but larger constant-size subsets of vertices are given whose induced subgraph is connected. We show that for k-tuples, we can uniquely reconstruct any tree of size at least $2k - 1$, provided we know that the sought graph is a tree.

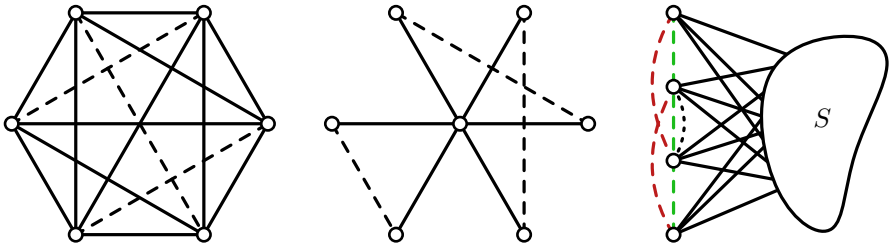


Figure 5.1: Three classes of ambiguous triples: a complete graph minus any independent set of edges, a star graph plus any (partial) matching of the leaves, and a path of length four in which all vertices are fully connected to some set S. In this last case, we cannot tell the difference between the red and green path.

5.2 Preliminaries

Let G be an unknown graph with n vertices and let T be the set of all triples of vertices that have a connected induced subgraph in G. We will use $\overline{T}$ to denote the complement of this set T, i.e. $\overline{T}$ is the set of all triples of vertices for which the induced subgraph is not connected. Note that $\left|T \cup \overline{T}\right| = \binom{n}{3} \in \Theta(n^3)$.

Observe that the presence of a triple gives the same amount of information as the absence of a triple: in the former case, at most one of the three possible edges is absent, whereas in the latter case, at most one of these edges is present.

The size of T is upper-bounded by the sum of squared degrees of the vertices of the graph. But we may be over-counting: if all three edges of a triple are present, then we would count the same triple three times. Hence:

$$\frac{1}{3}\sum_{v\in V}\binom{\text{degree}(v)}{2} \;\leq\; |T| \;\leq\; \sum_{v\in V}\binom{\text{degree}(v)}{2}$$

Hence, for a planar graph, $|T| \in O(n^2)$.

When multiple graphs yield the same set of triples, unique reconstruction is impossible. A simple example are a path and a cycle of three vertices: both contain the same triple. Three simple cases of sets of triples for potentially large graphs that are ambiguous are shown in Figure 5.1.

5.3 Algorithms

Given a set of triples T, we can find a graph G consistent with those triples by solving a 2-SAT formula. The main observation here is that the presence of a triple abc means that at least two of the edges ab, ac and bc must exist, whereas the absence of a triple means at most one of the edges can exist. We can then construct a 2-SAT

formula where each variable corresponds to an edge of the graph, and truth represents presence of that edge. For each triple $abc \in T$, we add clauses $(ab \vee ac)$, $(ab \vee bc)$ and $(ac \vee bc)$ to the formula. For each triple $abc \in \overline{T}$, we add clauses $(\neg ab \vee \neg ac)$, $(\neg ab \vee \neg bc)$ and $(\neg ac \vee \neg bc)$. A graph consistent with the set of triples can then be found by solving the resulting 2-SAT formula and taking our set of edges to be the set of true variables in the satisfying assignment. If the formula cannot be satisfied, no graph consistent with T exists.

We can solve the 2-SAT formula in linear time with respect to the length of the formula [17, 47]. We add a constant number of clauses for each element of T and $\overline{T}$, so our formula has length $O(|T \cup \overline{T}|)$. As $|T \cup \overline{T}| = \binom{n}{3}$, this gives us an $O(n^3)$ time algorithm to reconstruct a graph with n vertices. However, we prefer an algorithm that depends on the size of T, instead of also on the size of $\overline{T}$. We can eliminate the dependency on the size of $\overline{T}$ by observing that some clauses can be excluded from the formula because the variables cannot be true.

Lemma 5.1. *We can find a graph G consistent with T in $O(n \cdot |T|)$ time.*

Proof. The basic observation that allows us to exclude certain clauses from the formula is that if there is no triple containing two vertices a and b, the variable ab will always be false. Consequently, if we have a triple $abc \in \overline{T}$ for which at most one of the pairs ab, ac and bc appear together in some triple, we do not need to include its clauses in the formula, as at least two of the variables will be false, making these clauses necessarily satisfied.

We can construct the formula that excludes these unnecessary clauses in $O(n \cdot |T|)$ time as follows. We build a matrix $M(i, j)$ with each entry containing a list of all vertices with which i and j appear in a triple, i.e. $M(i, j) = \{x \mid ijx \in T\}$. This matrix can be constructed straightforwardly in $O(n^2 + |T|)$ time. We also sort each list in linear time using e.g. radix sort. As the total length of all lists is $O(|T|)$, this takes $O(n^2 + |T|)$ time in total.

Using this matrix, we can decide which clauses induced by triples from $\overline{T}$ to include as follows. For all pairs of vertices (a, b) that appear in some triple (i.e. $M(a, b) \neq \emptyset$), we find all x such that $abx \in \overline{T}$. As $M(a, b)$ is sorted, we can find all x in $O(n)$ time by simply recording the missing elements of the list. We then check if $M(a, x)$ and $M(b, x)$ are empty. If either one is not, we include the clause associated with $abx \in \overline{T}$ in our formula. Otherwise, we can safely ignore this clause, as it is necessarily satisfied by the variables for ax and bx being false. We do $O(n)$ work for each non-empty element of $M(i, j)$, of which there are $O(|T|)$, plus $O(n^2)$ time to traverse the matrix.

The total time to construct the formula is $O(n^2 + n \cdot |T|)$. As $|T| \in \Omega(n)$ for connected graphs, this simplifies to $O(n \cdot |T|)$ time. The resulting formula also has $O(n \cdot |T|)$ length, and can be solved in time linear in that length. The statement in the lemma follows. □

Observe that this is only an improvement on the naive $O(n^3)$ approach if $|T| \in$

$o(n^2)$. We also note that we can test the uniqueness of the reconstruction in the same time using Feder's approach for enumerating 2-SAT solutions [48].

5.4 Unique reconstruction of trees

We consider the case where we are given a set of triples T and we know that the underlying graph is a tree. We show that in this case the graph can be uniquely reconstructed if it has at least five vertices. Let us briefly examine trees with three or four vertices. A tree with three vertices is always a path and it will always have one triple with all three vertices. We do not know which of the three edges is absent. A tree with four vertices is either a path or a star. The path has two triples and the star has three triples. For the star, the centre is the one vertex that appears in all three triples, and hence the reconstruction is unique. For the path, we will know that the graph is a path, but we will not know in what order the middle two vertices appear. For the triples abc and bcd, both a—b—c—d and a—c—b—d are possible trees, and they have different edges.

Next we consider trees with at least five vertices. We first show that we can recognise all leaves and their adjacent vertices from the triples. In the following, we say that a vertex v *dominates* a vertex u if v appears in all the triples that u appears in. In addition, we say that v dominates u *directly* if there is no other vertex w such that v dominates u and w, and w dominates u.

Lemma 5.2. *A vertex u is a leaf, with neighbouring vertex v, if and only if v dominates u directly and u does not dominate any vertex, assuming the tree has at least five vertices.*

Proof. A leaf u can necessarily only appear in triples with its adjacent vertex v, as it is not adjacent to any other vertices by definition. A leaf is therefore always dominated by its neighbour v, and this domination is easily seen to be direct. Since $|V| \geq 5$, u does not dominate any vertex.

Conversely, assume v dominates u and u is not a leaf. Then the subtree $\mathcal{T}$ with u obtained by removing the edge uv has two or more vertices in it. If any neighbour of u is a leaf, u dominates it, so the domination of u by v is not direct. Otherwise, $\mathcal{T}$ has size at least 3 so v does not dominate u. □

We can use the lemma to prove that any tree can be reconstructed from its triples, provided that we know that the result must be a tree and $|V| \geq 5$. In order to derive an optimal, $O(|T|)$ time reconstruction algorithm, we will use a further characterisation of vertices of a tree using the triples. The main idea is that we can recognize not only leaves, but also other vertices where we can reduce the tree. If a vertex v has degree 2 in a tree, then there are two nodes w, w' such that every triple with v also contains w or w' (or both). The inverse is not true for two reasons: if v is a leaf, it also has the stated property, and if v has degree 3 where at least one neighbour is a leaf, then it has this property as well.

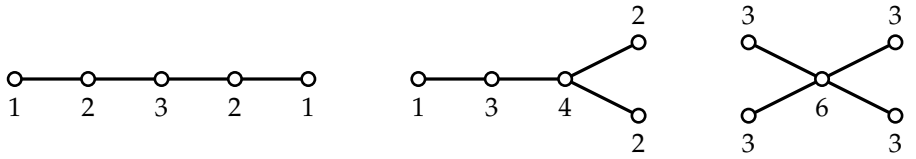


Figure 5.2: All trees of five vertices, and the number of triples each vertex occurs in.

Lemma 5.3. *A vertex v of a tree $\mathcal{T}$ of at least five vertices with triple set T is:*

(i) *a leaf if all triples with v also contain a vertex w, and there is no triple uvw such that u occurs only in uvw;*

(ii) *(a) a node of degree 2, or (b) a node of degree 3 with at least one leaf neighbour, if v is not a leaf and there are two nodes w_1, w_2 such that all triples with v also contain w_1 or w_2.*

Proof. The first characterisation is an alternative formulation of Lemma 5.2.

The second characterisation can be seen as follows. If v has degree at least 4, then no w_1, w_2 as in the lemma exist, which is easily verified by looking at the triples with these five vertices only. Furthermore, if v has degree 3 and none of its neighbours are leaves, then again there are no w_1, w_2. The characterisation in the lemma covers the remaining possibilities. □

With some straightforward testing, both characterisations can be checked in time $O(|T_v|)$, where T_v is the set of triples that include v. For testing (i), take any triple $vab \in T_v$, and test both a and b separately if they are the sought w. For testing (ii), take any triple $vab \in T_v$. If characterisation (ii) holds, then w_1 must be a or b. We try both; assume w.l.o.g. $w_1 = a$. Remove all triples with a from T_v. For any remaining triple, one of the two vertices must be w_2, and we test both. In total we get four options to test for w_1 and w_2; each option is easily checked in $O(|T_v|)$ time. Note that more than half of the vertices of $\mathcal{T}$ satisfy one of the two characterisations of the lemma.

The whole algorithm is therefore as follows: For each triple uvw in T, generate vwu and wuv as well. Collect the triples with the same first vertex to generate T_v for all $v \in V$. Then, for all $v \in V$, use T_v to test if v satisfies one of the conditions of Lemma 5.3. The vertices of V partition into V', V'', and V''', where V' contains the leaves, V'' contains the vertices that are not leaves but satisfy the second condition of the lemma, and $V''' = V \setminus (V' \cup V'')$. For all leaves v, record their neighbour and remove all triples with v. For all vertices in V'', note that they can no longer be vertices of degree 3, but they may have become leaves. We test this and consider the subset $W \subseteq V''$ of vertices that have not become leaves. These vertices appear as singletons or sequences of degree 2 vertices. Let $v \in W$ and let its neighbours w_1 and w_2 not be in W; we record these two edges of $\mathcal{T}$. Then we replace any triple vw_1x by w_2w_1x and any triple vw_2y by w_1w_2y; the triple vw_1w_2 is discarded. Let $v_1, \ldots, v_k$ be

a sequence of vertices in W such that v_i and v_{i+1} are adjacent, $1 \leq i \leq k-1$. Let w_1 be the other vertex neighbouring v_1 and w_2 the other vertex neighbouring v_k. We record all of these edges of $\mathcal{T}$. Then we replace each triple uw_1v_1 by uw_1w_2 and each triple v_kw_2x by w_1w_2x. Then we discard all triples that contain any of $v_1, \ldots, v_k$. Then we remove all leaves in $V'' \setminus W$ by discarding more triples.

This process takes time linear in $|T|$, and reduces the number of vertices occurring in T to half or less. We repeat the process on the remaining tree until it has size five, at which point we can uniquely identify the structure of the tree by simply looking at the number of triples each vertex occurs in (see Figure 5.2). We may not remove all vertices of V' or V'' if the remaining tree would be smaller than five vertices; in that case, we simply leave some of them in. A standard recurrence shows that the total time used is $O(|T|)$.

Theorem 5.4. *Let T be a set of triples, and let it be known that the underlying graph $G = (V, E)$ is a tree. If $n \geq 5$, then G can be uniquely reconstructed in $O(|T|)$ time.*

We note that if the tree contains no leaves that are siblings, we do not need to know that the graph is a tree for unique reconstruction.

5.5 Other families of graphs that are uniquely reconstructible

We show that any 2-connected outerplanar graph of at least six vertices can be uniquely reconstructed, as well as any triangulated planar graph with at least seven vertices. We also show that any graph in which all cycles consist of at least five vertices can be uniquely reconstructed.

We start with 2-connected outerplanar graphs of at least six vertices. Our approach is similar to the one for trees: we show that we can identify a vertex of degree two, and remove it from the graph by merging it with one of its neighbours. We keep doing this until we have a graph with six vertices, which can be distinguished by the number of triples in which each vertex and edge occurs. We first observe that we can recognise vertices of degree two.

Lemma 5.5. *Two vertices u and v have degree two and are neighbours, if and only if there are $w_1, w_2 \in V$ such that $uvw_1, uvw_2 \in T$, and u and v appear together in no other triples. We can also determine which of w_1 and w_2 is neighbour of u, and which is neighbour of v.*

Proof. If u and v are neighbours with degree two, they can clearly only appear together in a triple with the other neighbours of u and v (w_1 and w_2), as u and v are not connected to any other vertices. Conversely, if we have the triples uvw_1 and uvw_2, u and v must be neighbours, and as the graph is outerplanar, we can have at most three of the edges uw_1, uw_2, vw_1 and vw_2. Note that, as the graph is 2-connected, we must have $w_1 \neq w_2$, as all vertices of a 2-connected outerplanar graph appear in a simple

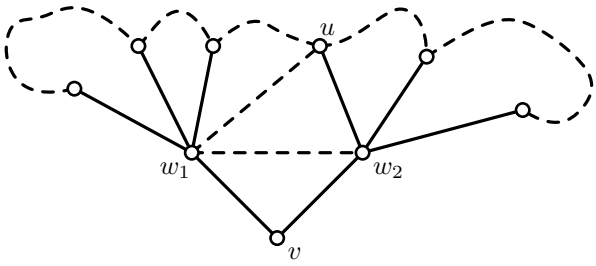


Figure 5.3: The local neighbourhood of a degree two vertex in a 2-connected outerplanar graph. The dashed edges w_1w_2 and w_1u may or may not be present.

cycle. Also, as the graph has at least six vertices, we know that w_1 and w_2 both have at least one additional neighbour. We can then tell which vertex neighbours which, from the absence of certain triples: if we do not have a triple uw_1x for $x \neq v, w_2$, u must neighbour w_2, if we do not have a triple vw_2x then v must neighbour w_1, etc. □

We can also identify a vertex of degree two that has neighbours with degree at least three.

Lemma 5.6. *A vertex v has degree two, and its neighbours are w_1 and w_2, if and only if all triples v appears in also contain w_1 or w_2, and there is no other vertex for which this is true.*

Proof. If v neighbours only w_1 and w_2, then it will clearly only appear in triples with these vertices, and if the graph is 2-connected outerplanar and has at least six vertices, there can be no other vertex for which the same holds. Conversely, if v would have degree at least three, it would appear in some triple without w_1 and w_2 (if the connected component of v after removal of w_1 and w_2 contains at least three vertices), or there would be some vertex v' that also only appears in triples with w_1 or w_2. Also, if v does have degree two, but w_1 or w_2 is not its neighbour, it would appear in some triple without w_1 or w_2. □

Using these two lemmas, we obtain the following result.

Theorem 5.7. *Let T be a set of triples, and let it be known that the underlying graph $G = (V, E)$ is 2-connected and outerplanar. Then G can be uniquely reconstructed from T if $n \geq 6$.*

Proof. If our graph has any two neighbouring vertices of degree two, we can recognise them and their neighbours by Lemma 5.5. Let u and v be the two degree two vertices, and let w_1 and w_2 be their respective other neighbours. We can then merge u and v by removing all triples they both occur in, and relabelling v to u in all other triples. We repeat this step until there are no neighbouring vertices of degree two left, or the remaining graph has six vertices.

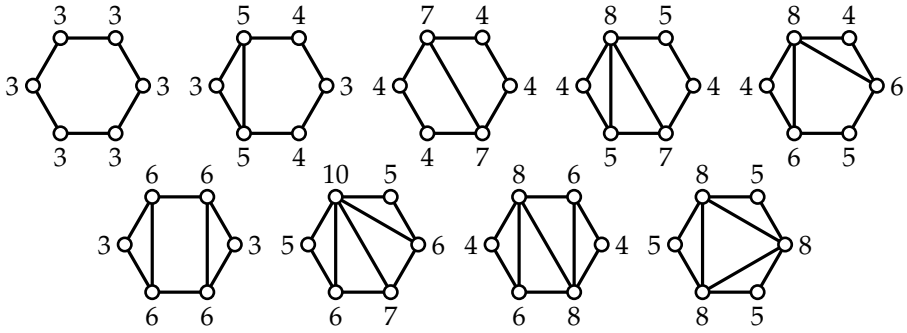


Figure 5.4: All base cases for a 2-connected outerplanar graph of six vertices, up to symmetries. Note that each case can be identified by the number of triples each vertex occurs in.

Otherwise, we know that any degree two vertex has two neighbours with degree at least three, which we can recognise by Lemma 5.6. This gives us a degree two vertex v with neighbours w_1 and w_2. We also know all neighbours of w_1 and w_2: they are exactly the vertices that appear in triples with u and w_1 or w_2. The only exception is that this does not tell us if w_1 and w_2 are neighbours, as we have the triple uw_1w_2 either way. However, we do know that, as the graph is 2-connected and outerplanar, w_1 and w_2 can share at most two neighbours, one of which is v (see Figure 5.3 for an illustration). Let u be the other neighbour; this means both w_1 and w_2 have at least one neighbour that is not a neighbour of the other vertex, as they are both degree at least three. We can then tell whether w_1 and w_2 are neighbours by the absence of any triple w_1w_2x with $vw_1x \in T$ or $vw_2x \in T$. If any such triple does not exist, w_1 and w_2 are not neighbours.

If the edge w_1w_2 exists, we can remove v and all triples in which it occurs, giving a new 2-connected outerplanar graph with one fewer vertex. If the edge does not exist, we can add it and add all the triples w_1w_2x for each neighbour x of w_1 or w_2, then remove v and all triples in which it occurs. This again gives a new 2-connected outerplanar graph with one fewer vertex.

Our base cases are the 2-connected outerplanar graphs of six vertices, each of which has a unique set of triples; see Figure 5.4 for an illustration. In all cases where an edge exists between two vertices u, v with k_u and k_v occurrences in triples, if a vertex w exists with $k_w = k_v$, we have that u, w occur less often together in a triple than u, v. So we can identify all labeled edges in each base case as well. □

To show that triangulated planar graphs of at least seven vertices can be uniquely reconstructed, we first show that unique reconstruction of such graphs is possible if they do not contain any separating triangles. A *triangulated planar graph*, also called a *maximal planar graph*, is a planar graph where every face (including the outer face)

is a triangle. Note that chordal graphs (graphs in which every cycle of more than three vertices has a chord) are sometimes also called triangulated graphs, but this is not the meaning we use here. A *separating triangle* is a triangle in the graph whose removal would result in the graph being disconnected.

We take a similar approach as before: we argue that we can recognise all vertices of degree at least five with their neighbours in cyclic order. We then argue that we can deduce the existence of all edges not incident to a vertex of degree at least five as well. In the following, a 6-wheel refers to a graph consisting of a cycle of five vertices, and a sixth vertex connected to all vertices on the cycle. A 6-fan refers to the same graph, but replacing the cycle with a path of five vertices.

Lemma 5.8. *We can uniquely identify the 6-fan and the 6-wheel as an induced subgraph from the triples, as well as which vertex is the apex, and the order in which the neighbours of the apex appear on the fan or wheel.*

Proof. Let v be the apex vertex, and $\mathcal{W} = \{w_1, \ldots, w_5\}$ be the other vertices in the fan or wheel in cyclic order. We can recognise v by the fact that it occurs in a triple with all ten pairs of vertices in $\mathcal{W}$. We can also identify the order in which w_1 through w_5 appear on the fan by their triples, as the triples $w_1w_2w_3$, $w_2w_3w_4$ and $w_3w_4w_5$ are characteristic of a path of five vertices. Similarly, the wheel additionally has the triples $w_1w_4w_5$ and $w_1w_2w_5$. It remains to be shown that these sets of triples are unique to the 6-fan and 6-wheel among all triangulated planar graphs without separating triangles.

We make a subdistinction on the degree of v in a hypothetical graph with the same triples. If v has degree five, we have a star graph in which v appears in the correct triples, but the vertices in $\mathcal{W}$ don't. The induced subgraph of $\mathcal{W}$ must be connected, as otherwise it would have one connected component of at most two vertices, which wouldn't add any triples for those vertices. For the 6-fan, the path $w_1w_2w_3w_4w_5$ would add exactly the triples required, and each other tree would add too many triples to at least one vertex, as seen in Figure 5.2. In addition, adding any more edges to any of the trees would add even more triples, so we conclude the path is the only way to get the required triples. Similarly, for the 6-wheel, the only way to make each vertex in $\mathcal{W}$ appear in three additional triples is to add a cycle to the graph, and the cycle in cyclic order is the only way to get the correct triples.

If v has degree four, we are missing an edge from v to some vertex in $\mathcal{W}$; w.l.o.g. let that vertex be w_1. As we have the triples with v, w_1 and each other vertex in $\mathcal{W}$, the absence of this edge means we must have edges from both v and w_1 to all other vertices in $\mathcal{W}$. For both the 6-fan and the 6-wheel, this immediately means we must now have triples not normally present, such as $w_1w_3w_4$, so we cannot have the same set of triples as the 6-fan or 6-wheel if v has degree four.

If v has degree three or less, there is at least one more edge between v and $\mathcal{W}$ missing, w.l.o.g. let that vertex be w_2. In that case the triple vw_1w_2 cannot be present, so we cannot have the same triples. We conclude that the 6-fan and the 6-wheel can be uniquely identified from the triples. □

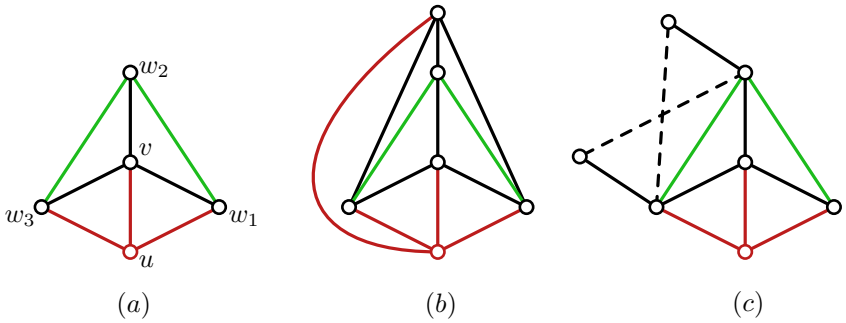


Figure 5.5: (a) The local structure around a degree four vertex with three neighbours of degree four. (b) Connecting all neighbours of v to the same vertex gives the octahedron. (c) Connecting any two neighbours of v to different vertices would give at least one of them a higher degree.

Using the lemma above, we can show that we can recognise vertices with degree at least five.

Lemma 5.9. *We can identify any vertex v of degree at least five with its neighbours, as well as the cyclic order of the neighbours around v.*

Proof. Using Lemma 5.8, we can identify all 6-fans with the same apex. By overlaying consecutive 6-fans, we can identify the k-wheel around v, where k is the degree of v. Note that any vertex of degree at least five and its neighbours must form a k-wheel, as the graph is triangulated. Also note that the cycle of neighbours around v cannot have any chords, as those would introduce separating triangles. □

This lets us reconstruct all triangles incident to a vertex of degree five. Next, let us examine the structure of vertices of degree four.

Lemma 5.10. *All edges of a triangulated planar graph without separating triangles of at least seven vertices lie on a 6-wheel or 6-fan.*

Proof. We show that induced subgraphs of vertices of degree four can only form paths and cycles. Consider a vertex v of degree four with more than two neighbours of degree four. This means it has either three or four neighbours of degree four. If it has four, the induced subgraph of v and its neighbours must be a 4-wheel, as each face incident to v must be a triangle, and the other triangulation option would give v a higher degree. All v's neighbours also need to be incident to triangular faces, and the only way to achieve this is to have them all connected to the same vertex. This is the octahedron, which cannot occur as an induced subgraph in a triangulated planar graph of at least seven vertices without separating triangles.

Figure 5.5 shows the situation if v has three neighbours of degree four, which we call w_1, w_2 and w_3. In this case v must have at least one additional neighbour u of degree at least five, shown in red. All faces must be triangles, so v's neighbours must be connected by the green edges, or v would have a higher degree. For the same reason, w_1 and w_3 must be neighbours of u. Now, because all faces of the graph must be triangles, neighbours of v must be connected to other vertices. We could connect all to the same vertex, as seen in Figure 5.5(b), but this would give us the octahedron as an induced subgraph again, as the quadrilateral $uw_1w_2w_3$ must be divided into triangles by the chord uw_2 (the other choice would give w_1 and w_2 a higher degree).

The only option that remains is to connect the neighbours of v to different vertices. However, if any two neighbours of v are connected to different vertices, as in Figure 5.5(c), at least one of them would need to have degree five to give triangular faces.

We conclude that vertices of degree four can only have two neighbours of degree four. This means they must form either a path or a cycle. However, as in Figure 5.5(c), if any neighbours of degree four don't share both their other neighbours, the graph is not triangulated. This implies that every path or cycle of degree four vertices has two vertices of degree at least five that are neighbours to all vertices on the path or cycle. As such, all paths or cycles of degree four vertices appear on some 6-fan or 6-wheel.

We cannot have degree one or two vertices in a triangulated graph of at least seven vertices, and the neighbours of a vertex of degree three would form a separating triangle. A triangulated planar graph of at least seven vertices without separating triangles therefore only has vertices of degree four or higher, for which all the edges lie on a 6-wheel or a 6-fan. □

Using this lemma we can obtain all edges of the graph, and therefore its reconstruction. It turns out that we can also recognize separators of a graph (we need this result only for separators of size 3):

Lemma 5.11. *A subset of vertices $S \subset V$ are a separator in G if and only if we can partition $V \setminus S$ into sets $V_1 \neq \emptyset$ and $V_2 \neq \emptyset$ such that any triple with an element from both V_1 and V_2 also contains an element of S, assuming $|V_1| + |V_2| \geq 3$.*

Proof. If S is a separator, it is clear that if we take V_1 and V_2 to be the two components of the graph after removing S, their vertices can only appear together in triples with elements of S. After all, if we would have a triple xyz with, say, $x, y \in V_1$ and $z \in V_2$, it would mean that x, y and z form a connected component of the graph, contradicting the fact that S is a separator.

Conversely, if S is not a separator, the remainder of the graph must form one connected component. If there are at least three remaining vertices, this means that, for any choice of V_1 and V_2, there must be a vertex $x \in V_1$ with a neighbour $y \in V_2$, and a third vertex z that neighbours x and/or y. We then have a triple containing vertices from V_1 and V_2 but not containing any element of S. □

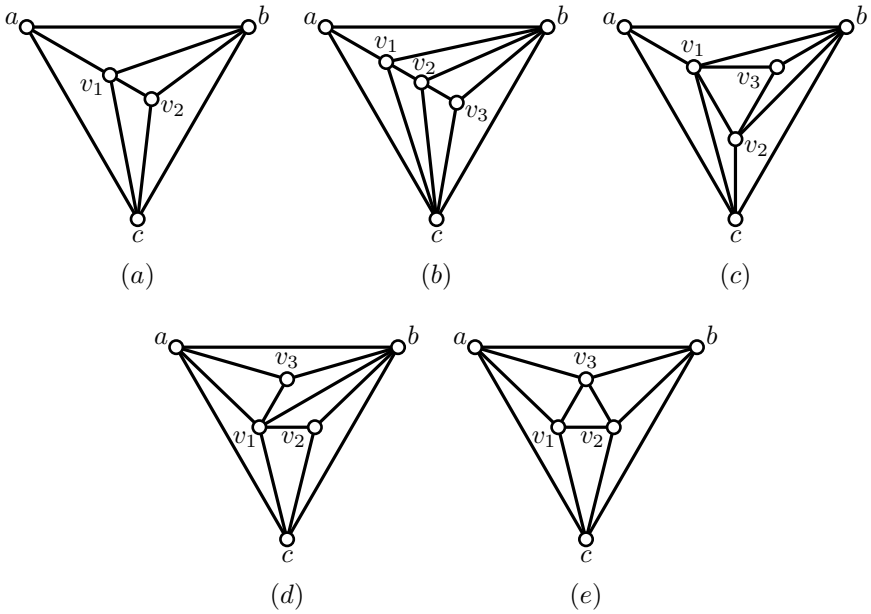


Figure 5.6: The cases for triangulated planar graphs of five or six vertices, modulo symmetries.

Using the fact that we can recognise separators, we now show that all triangulated planar graphs of at least seven vertices have a unique reconstruction.

Theorem 5.12. *Let T be a set of triples, and let it be known that the underlying graph $G = (V, E)$ is planar and triangulated. Then G can be uniquely reconstructed from T if $n \geq 7$.*

Proof. We decompose our graph using separating triangles, which by Lemma 5.11 we can detect as long as our induced subgraph has at least six vertices. Any separating triangle gives us a triangle S and a partition of V into V_1 and V_2, representing the two components of G after removing S. We can then consider the induced subgraphs given by $V_1 \cup S$ and $V_2 \cup S$, which are still planar and triangulated. We recurse on an induced subgraph G' as long as it contains a separating triangle and has at least seven vertices.

Any induced subgraph of at least seven vertices without any separating triangles can be reconstructed using Lemma 5.10. It remains to be shown that we can reconstruct the induced subgraphs of size at most six. In general, this is not possible: consider, for instance, the octahedron, where we cannot tell from the triples which vertex is connected to which. However, as our induced subgraphs are part of a larger graph, we are able to reconstruct them.

Let $G' = (V' \subset V, E' \subset E)$ be an induced subgraph of at most six vertices. We make a case distinction on the size of V'. As G' is the union of a separating triangle abc with one of the two components it separates the previous subgraph into, it must have at least four vertices. If it has four vertices, K_4 is the only possible triangulated structure. If it has five vertices, there is still only one structure, shown in Figure 5.6(a), but several cases depending on the labelling of the vertices on the separating triangle. As the full graph has at least seven vertices and is triangulated, there must be more vertices connected to a, b and c. We can then tell which vertex of the triangle and which of v_1 and v_2 have degree three in the induced subgraph by the absence of triples with those two vertices and a vertex outside the induced subgraph. With the labelling in the example figure, this would be triples of the form av_2x, where x is not in our induced subgraph. The induced subgraph is symmetric with regards to b and c, so we don't need to tell them apart.

If G' has six vertices, there are four structures possible, shown in Figure 5.6(b)-(e), but again there are several labellings possible for each structure. In cases (b) and (c), there are unique combinations of triples missing (av_1v_3 and av_2v_3 in (b), av_2v_3 and cv_2v_3 in (c)), and the graph is symmetric w.r.t b and c, so we don't need to distinguish them. In case (d), acv_3 and av_2v_3 are missing. We recognise a as it appears in both missing triples, c as it appears in one, and b as it appears in none. Finally, in case (e) all triples are present. However, just as in case (a), there must be more vertices connected to a, b and c. We can then tell which vertices on the outer triangle are neighbours of which vertices on the inner triangle by the absence of triples of the type av_2x, bv_1x and cv_3x, where x is any vertex outside the induced subgraph.

As these are all possible induced subgraphs of triangulated planar graphs of at most six vertices, we conclude that we can uniquely reconstruct all of them, allowing us to uniquely reconstruct any triangulated planar graph from the triples. □

Note that, while our proof is constructive, it does not lead to a faster algorithm than the general cubic or $O(n \cdot |T|)$ time algorithm described in Section 5.3.

When the cycles of our graphs all have at least five vertices (i.e. the graph has girth at least five), things become significantly easier. We can reconstruct such graphs in $O(|T|)$ time by exploiting the fact that for each triple exactly two of the edges exist, and that the edge that does not exist does not appear in any other triple. This last part is not true if there are cycles of four vertices, making this case more complicated.

Theorem 5.13. *Let T be a set of triples, and let it be known that all the cycles of the underlying graph $G = (V, E)$ have at least five vertices. Then G can be uniquely reconstructed from T in $O(|T|)$ time.*

Proof. When all cycles have at least five vertices, each triple abc represents exactly two edges. Assume ab is the edge that does not exist; we then know that ab only appears in the triple abc. This is because a and b are at distance 2, and any triple abd would then put d at distance 1 to both a and b. If $c \neq d$, we would then have a face $abdc$ of four vertices.

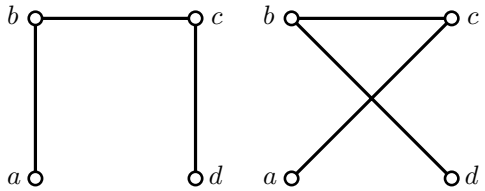


Figure 5.7: Two isomorphic graphs that give the same set of triples.

We can then identify all edges of the graph as follows. We build a list of length $3 \cdot |T|$ containing copies of all triples with each edge appearing at the start (e.g. for a triple abc, we include it as abc, acb and bca). We then sort this list in linear time. Now we walk through the list in order: for any item abc such that no other triple starts with ab, we know that the edges ac and bc must exist. All these edges together gives us the graph G. □

5.6 Results on ambiguity

As noted before in Section 5.2, it is not possible to uniquely reconstruct all graphs from their sets of triples, because we may obtain the same set of triples from some graph $G = (V, E)$ as we do for the graph $G' = (V, E \cup \{(x, y)\})$. In such cases we say that the edge xy is *ambiguous*. In other cases, we can reconstruct the structure of the graph, but there are multiple labellings of the vertices giving the same set of triples; see Figure 5.7 for an example. The problem of characterising all graphs that contain such ambiguities remains open; here we present some partial results about sets of triples that are not ambiguous.

To examine the structure that causes a particular edge xy to be ambiguous, it is useful to classify the vertices of the graph based on their relationship with the vertices x and y. We define the following classification:

$$
\begin{aligned}
A(x, y) &\coloneqq \{z \mid xyz \in T\} \\
B(x, y) &\coloneqq \{z \mid xz\cdot \in T \wedge yz\cdot \in \overline{T}\} \\
C(x, y) &\coloneqq \{z \mid yz\cdot \in T \wedge xz\cdot \in \overline{T}\} \\
D(x, y) &\coloneqq \{z \mid xz\cdot \in T \wedge yz\cdot \in T \wedge xyz \in \overline{T}\} \\
E(x, y) &\coloneqq \{z \mid xz\cdot \in \overline{T} \wedge yz\cdot \in \overline{T}\}
\end{aligned}
$$

where $xz\cdot \in T$ means x and z appear together in some triple. See Figure 5.8 for an illustration of these sets; we will omit the arguments where appropriate. Note that these sets are disjoint, and that their union contains all vertices in our graph minus

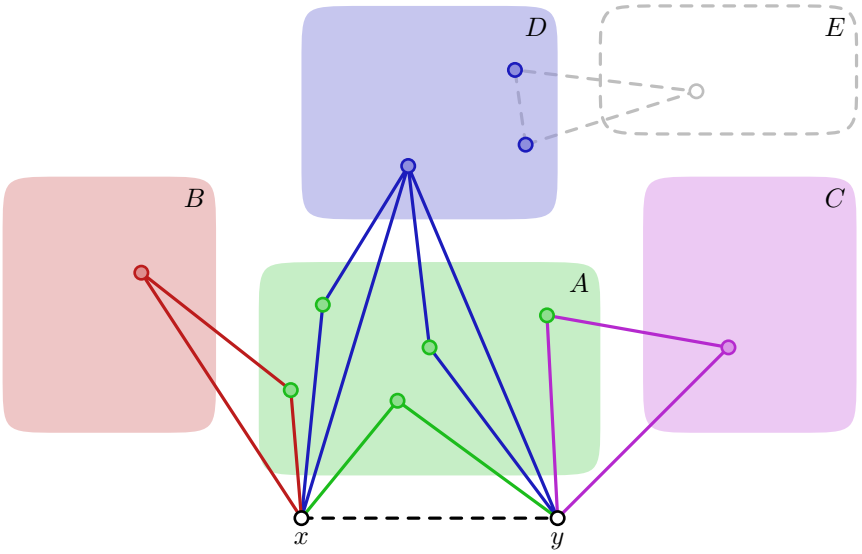


Figure 5.8: The subdivision of all vertices into sets A, B, C, D and E. The triangles visualise triples, not edges of the graph.

x and y. With these definitions in hand, we can make some statements about the ambiguity of xy in specific cases. We first make the following observations:

Observation 5.14. *If the edge xy is not ambiguous when only considering those triples formed by a subset of the vertices (i.e. all $abc \in T$ with $a, b, c \in V' \subseteq V$), it is not ambiguous for the full set of triples.*

Proof. The observation follows from the 2-SAT formula: there is a clause for each possible triple, positive if it exists and negative otherwise. Considering a subset of the vertices is the same as considering a subset of the clauses. If a subset of the clauses can only be satisfied by having a particular value for the variable xy, adding more clauses cannot change this. □

Observation 5.15. *If x and y do not appear together in some triple, the edge xy does not exist.*

Note that this last observation is only true for connected graphs of at least three vertices; otherwise, x and y may form their own component with an edge xy, but still appear in no triple. In such cases, we can then assume that A will not be empty. In our proofs, we will often also use the fact that if one edge of a triple does not exist, the other two must be in the graph. We can now show the following:

Lemma 5.16. *If some triple xaa' or yaa' with $a, a' \in A$ does not exist, the edge xy must exist.*

Proof. By definition of A, we know that xya and xya' are triples that exist. Assume that xy does not exist. In that case, the edges xa, ya, xa' and ya' must all exist, or we would not have triples xya or xya'. But the edges xa and xa' would yield a triple xaa', and the edges ya and ya' would yield a triple yaa', which contradicts our condition in the lemma statement. We conclude that xy must exist. □

We can further use this to prove that B and C must be empty for xy to be ambiguous, provided the graph is of sufficient size.

Lemma 5.17. *If B or C is not empty, and the graph contains at least five vertices, the edge xy is not ambiguous.*

Proof. We prove the case for B not being empty; the other case is symmetrical. By Observation 5.15, we know that there must be some triple xya with $a \in A$. As B is not empty, let b be a vertex in B. Call the fifth vertex z; we now differentiate three structures that our triples can have, based on the triples that z appears in:

1. xbz is a triple and $z \in A$.
2. xbz is a triple and $z \in B$ or $z \in D$.
3. xbz is not a triple.

Note that if xbz is a triple, z cannot be in C or E by definition.

We first consider Case 1. We show that xy is not an edge if and only if xab is a triple. The absence of xy implies xa, ya, xz and yz must all exist. The existence of yz means bz cannot exist, or we would have the triple ybz. Therefore, xb must exist, which together with the existence of xa implies the triple xab must be in our input. Conversely, if xab is in our input, we know xy does not exist: if it did, we know that xb cannot be an edge, so we must have ab and zb as edges. However, as by Lemma 5.16 we must have the triple yaz, we know that we have at least one of the edges ya and yz, which would then imply the triples yab or yzb. By the definition of B, these cannot exist, so we conclude xy cannot exist. We can therefore derive the existence of xy from the presence of the triple xab.

We now consider the Case 2. In this case, we know xy cannot exist: at least one of the edges xb and xz must exist, which would give triples xyb or xyz if xy exists. By the definitions of B and D, neither of these triples can exist when $z \in B$ or $z \in D$, so we conclude xy cannot be an edge.

Finally, we consider Case 3. In this case, we must have the triple xab: b must appear in some triple with x, and a is the only candidate vertex. We perform a further case analysis of all the pairs of vertices z can appear in a triple with. We can disregard some of the cases: ybz cannot be a triple because of the definition of B, xyz is identical

to Case 1 with the labels for a and z swapped, and xbz was already covered in Cases 1 and 2. The remaining triples we need to consider are xaz, yaz and abz. Note that z cannot be in A in any of the cases, as this would be equivalent to Case 1, or by Lemma 5.16 xy must exist.

Consider the case where z appears in a triple xaz. We further subdivide this case into $z \in B$ and $z \in D$. For $z \in B$, similar to Case 1, we show that $xy \leftrightarrow abz$. We first note that the edge xa must exist, as otherwise we would have edges ya and ab, which would give a triple yab, violating the definition of B. We now observe that there are two possible structures our graph can have: we have the edges xa, xb, xz and ya, or we have the edges xa, ab, az and xy. Any other combination of edges would cause y to appear in a triple with b or z. Edges ab and az would give a triple abz, showing $xy \rightarrow abz$. For the other direction, simply observe that the edge ya cannot exist, and therefore xy must.

We cannot use the same arguments when $z \in D$, as now the triple yaz is allowed to exist. It is therefore also possible to have the edges xa, xb, ya and az. However, note that the opposite case, with edges xa, xz ab, ya and xy is not valid, as it would give a triple xyz. We therefore conclude that if $z \in D$ and the structure of triples does not follow the ones detailed above for the case $z \in B$, xy cannot be an edge.

We now consider the case where z appears in a triple yaz. If $z \in A$, we are in Case 1 with the labels for a and z swapped. If $z \notin A$, xy cannot exist: if it did, xb and yz could not be edges, as that would give triples xyb and xyz. Because we have triples xab and yaz, this implies that edges xa, ab, yz and az must all exist, but then yab would be a triple, violating the assumption that $b \in B$.

Finally, we consider the case where abz is a triple. If $z \in A$, xy must exist, or otherwise xbz would be a triple, violating the assumption of this case. If $z \in B$, we must have the triple xaz (the only other option is xbz), and xy must exist: if it doesn't, we would have edges xb and xz, again giving triple xbz. If $z \in C$ or $z \in D$, we are in Case 3 with yaz being a triple, which is described above. Lastly, if $z \in E$, xy must be an edge: if it is not, we have edges xa, xb and ya, which would put z in D if az is an edge, or in B if bz is an edge, and one of these must exist. □

If B and C are empty, there are still cases in which we can deduce the existence of the edge xy:

Lemma 5.18. *If B and C are empty, and $\exists d, d' \in D : xdd' \in T \vee ydd' \in T$ or $\exists a, a' \in A, d, d' \in D : ((xad \in T \vee yad \in T) \wedge (xa'd' \in T \vee ya'd' \in T) \wedge ((a = a' \wedge add' \in \overline{T}) \vee (d = d' \wedge aa'd \in \overline{T}))$, then xy does not exist.*

Proof. If there is any triple containing x or y with two elements $d, d' \in D$, we know xy is not an edge: at least one of the edges xd, xd', yd or yd' would have to exist, which would create a triple xyd or xyd' if xy was also an edge. Such a triple would place d or d' in A, which is not possible by definition. The same applies if there are two triples $\cdot ad$ and $\cdot a'd'$, with $\cdot$ being x or y, and either $a = a'$ and add' is not a triple, or $d = d'$ and $aa'd$ is not a triple. In such a case, the absence of the indicated triple

means that one edge of the type ad does not exist, which in turn means xd or yd must exist, which precludes xy being an edge. □

What all these results have in common is that we can deduce the existence of the edge xy from the triples when some symmetry with respect to x and y is broken. However, there are many types of graphs where such symmetries exist, as illustrated in Figure 5.1. These situations make it hard to decide if an edge is ambiguous: it is not enough to look at only the neighbourhood of that edge, but we need to potentially consider large sections of the graph.

5.7 Extensions

There are several logical extensions to the concept of reconstructing a graph from triples. We could define a (k, ℓ)-representation, where T contains all k-tuples of vertices for which at least ℓ pairs are neighbours. The definition we used in previous sections would then be a $(3, 2)$-representation. Such a generalisation leads to some strange properties, however: in particular, for many types of connected graphs, some vertices may not appear in T, or T might even be empty altogether. This would happen for instance if our graph is a tree and $\ell \geq k$.

An interesting generalisation is to take all k-tuples for which the induced subgraph is connected for larger k. We can show that we can still recognise trees, as long as they have sufficient size. We use the same notion of *domination* as before: a vertex v *dominates* a vertex u if v appears in all the k-tuples that u appears in, and v dominates u *directly* if there is no other vertex w such that v dominates u and w, and w dominates u.

Lemma 5.19. *Assuming $n \geq 2k - 1$, a vertex u is a leaf, with neighbouring vertex v, if and only if v dominates u directly and u does not dominate any vertex.*

Proof. A leaf u is always directly dominated by its neighbour v. Conversely, if u is not a leaf, it will have some subtree $\mathcal{T}$ not containing v. The only way that u does not dominate all the vertices in $\mathcal{T}$ is if it has size at least k, in which case v would not dominate u. □

Theorem 5.20. *Let T be a set of k-tuples, and let it be known that the underlying graph $G = (V, E)$ is a tree. Then G can be uniquely reconstructed from T if $n \geq 2k - 1$.*

Proof. Similarly to Theorem 5.4, we remove leaves until $n = 2k - 1$ using Lemma 5.19. There will be exactly one vertex v that is not dominated by any vertex; removing v would leave subtrees of size at most $k - 1$ only. Note also that no vertex from one subtree can dominate a vertex in another subtree. Within each subtree, every vertex is dominated directly by its parent. This lets us reconstruct all the edges of each subtree, and thereby the entire tree. □

5.8 Conclusion

We have presented a new model of uncertainty in graphs, in which we only receive all triples of vertices that form a connected induced subgraph. In a way, this is the simplest model of combinatorial indeterminacy in graphs. We have studied some basic properties of this model, and provided an algorithm for finding a graph consistent with the observed indeterminacies. We also prove that some families of graphs are uniquely reconstructible, although we may need to know the family the sought graph belongs to. However, in general, edges of the graph may be ambiguous, meaning multiple graphs are consistent with the input. We analyse when such ambiguous edges occur, although we currently cannot give a full characterisation. Finally, we show a straightforward extension of our model to larger k-tuples, and show that trees of sufficient size are uniquely reconstructible.

There are several avenues for further research on this topic. One is to give a full characterisation of sets of triples that contain ambiguity. The model can also be extended further; for instance, we could give all tuples of connected vertices of size at least k. We could analyse a version where it is given whether two or three edges of a triple exist, which would likely simplify the situation. Another interesting question is how to visualise the possible graphs for an ambiguous set of triples. In some cases many different graphs are possible (e.g. a complete graph with any matching removed), but even for cases with only a few ambiguous edges it is not immediately clear how to place the vertices effectively.

Chapter 6

Conclusion and future outlook

Similarity measures allow us to quantify how alike two given objects are. As such, they are widely studied and have many applications, ranging from AI and image retrieval, to procedural content generation and computer vision. Geometric similarity measures in particular are used to quantify the similarity of geometric objects, such as point sets, polygons, three-dimensional polygonal meshes, and smooth curves. We may use such similarity measures to perform tasks like finding a geometric object in a given set that most closely resembles a query object, or to measure the similarity of a manufactured part with its (digital) master. In this thesis, we have made some steps to further understand such geometric similarity measures, to develop algorithms for computing them, and to find new applications for them.

In Chapter 2, we have developed and analysed algorithms for approximating the earth mover's distance between various types of geometric primitives. Previously, algorithms that are either exact or have provable approximation bounds were known only for the case of weighted point sets. In our work, we provided $(1+\varepsilon)$-approximations for the cases where one set contains weighted points, and the other set contains line segments, triangles, or higher-dimensional simplices. We also gave $(1+\varepsilon)$-approximations with a small additive term for the cases where both sets contain line segments, triangles or higher-dimensional simplices. All our algorithms rely on subdividing the continuous objects until they can be approximated sufficiently well by weighted points. We additionally gave an exact algorithm for calculating the earth mover's distance between a set of weighted points and a set of line segments under the L_1 distance. However, it is not clear if this algorithm runs in polynomial time, as it relies on solving a convex quadratic program that may contain square roots in its definition.

In Chapter 3, we applied the Hausdorff distance to the concept of shape morphing, introducing a new type of morph we call a Hausdorff morph. Such a morph interpolates the Hausdorff distance between two input shapes: at time $\alpha \in [0,1]$, it

has Hausdorff distance α to input shape A, and Hausdorff distance $1 - \alpha$ to input shape B. We showed that the maximal morph with this property, which we call the dilation morph, can be calculated using only Minkowski sums with disks and set intersections. We showed that the morph is convex if both input shapes are, that it is connected if at least one input shape is convex, and that there are pairs of connected input sets for which any Hausdorff morph is disconnected. We also analysed the complexity of the morphed shape in terms of the number of vertices, line segments and circular arcs that constitute its boundary. The complexity is linear in the size of the input when both input sets are simple polygons and at least one of them is convex, but quadratic in the worst case if neither is convex. We also showed that the dilation morph is 1-Lipschitz continuous with respect to the Hausdorff distance, meaning the Hausdorff distance between intermediate shapes at times α and β is bounded by (and in our case, equal to) $|\beta - \alpha|$.

Furthermore, we investigated how the notion of a Hausdorff middle can be extended to more than two input sets. In this case, we no longer consider morphs, but instead we simply try to find a single shape that minimises the maximum Hausdorff distance to any input shape. We showed that, in general, the best achievable distance may be the maximum of all pairwise Hausdorff distances between the input sets, but for convex sets we showed that, after normalisation, a distance of $\alpha^* \approx 0.6068$ is always possible, and that this distance is tight in the worst case. We further showed that the maximal Hausdorff middle for multiple sets is convex if all input sets are convex, connected if at most one input set is not convex but still connected, and that there are inputs with at least two non-convex but connected input sets for which any Hausdorff middle is disconnected. Next, we studied the question of whether we can remove any sets from the input without changing the Hausdorff distance our middle shape can achieve. We showed that, in general, for any given number of input sets, there is an input in which none of the sets may be removed without changing the Hausdorff distance. However, if all input sets are convex, then there always exists a subset of at most three input sets such that the optimal Hausdorff middle has the same distance for this subset as for the full input. Finally, we provided two algorithms that compute the optimal distance of a Hausdorff middle to a given set of polygons. One computes a $(1 + \varepsilon)$-approximation of the optimal distance in $O(n^2 \log^2 n \log 1/\varepsilon)$ time, the other computes the optimal distance exactly in $O(n^6)$ time.

In Chapter 4, we developed a different Hausdorff morph that addresses several practical concerns regarding the dilation morph. Instead of taking the maximal Hausdorff morph, we simply let each point on each shape move linearly towards the closest point on the other shape. In this way, the intermediate shapes do not grow into a uniform blob towards the middle of the morph, as is the case for the dilation morph. The morph can be calculated using Voronoi diagrams to find the closest point on each set, scaling the part inside each Voronoi cell towards the site of the cell, and then taking the union of all the scaled pieces. We showed that this morph, which we call the Voronoi morph, retains important properties of the dilation morph: it is

still a Hausdorff morph, and it is still 1-Lipschitz continuous. We also showed that the number of components of the intermediate shapes does not change during the morph, except possibly at $\alpha = 0$ and $\alpha = 1$.

We observed that, while the area of the intermediate shapes of the Voronoi morph more closely match the areas of the input shapes than is the case for the dilation morph, the perimeter of the Voronoi morph may still be too high. This is an artefact of the requirement that point move strictly to the closest point on the other shape: even a very unpronounced reflex vertex in one shape may split the other shape in two parts as soon as $\alpha > 0$, greatly increasing the perimeter. To remedy this, we introduced a third Hausdorff morph, which we call the mixed morph. This mixed morph modifies the Voronoi morph by taking the resulting shape and dilating it with a small radius φ, before eroding it again with the same small radius. This has the effect of closing any small gaps in the shape. To ensure our mixed morph is still a Hausdorff morph, we intersect the shape after erosion with the dilation morph at the same value of α. This reduces the increase in perimeter, but at the cost of having a morph that is no longer 1-Lipschitz continuous.

We performed an experimental comparison of the three morphing methods, looking at their behaviour regarding the area, perimeter, number of components, and number of holes during the morph. As test data, we used a set of outlines of European countries, a set of animal shapes, and a set of words and letters. Our results indicate that the mixed morph performs best on the aspects that we tested. It subjectively also has the most visually appealing results.

Finally, in Chapter 5, we studied the unrelated topic of graph reconstruction. In particular, we considered a situation in which a graph is not specified by all connected induced subgraphs of size two (edges), but by all connected induced subgraphs of size three instead. This introduces a degree of indeterminacy into the graph: for any triple in our specification, we know that either two or three of the edges between the listed vertices are present. The primary question is then whether the original graph can be reconstructed from such a list of triples. While it is clear that the answer in general is no (a path of three vertices and a triangle give the same single triple), we determined that even for graphs of arbitrary size, different graphs may give the same set of triples. While we could not yet give a full classification of all the graphs for which the reconstruction is ambiguous, we did show that trees, 2-connected outerplanar graphs, triangulated planar graphs, and graphs with girth at least five can be uniquely reconstructed if they are sufficiently large. We also briefly explored an extension of our specification by triples to a specification by k-tuples, showing that trees can still be uniquely reconstructed if they have size at least $2k - 1$.

6.1 Future outlook

There is much still to be explored on the topic of geometric similarity measures. On a fundamental level, we need a deeper understanding of the structure of solutions of

more complicated similarity measures like the earth mover's distance, and how to compute them efficiently. Much work has been done on relatively simple similarity measures like the Hausdorff distance. Its simplicity means that it has seen wide adoption in many applications, even though it can often be a rather poor measure of similarity. The Fréchet distance is generally a better measure of similarity, but can only be applied to more restricted settings. Clearly, there is a need for similarity measures that better capture the likeness of objects, but are still applicable in a wide range of settings, and efficiently computable.

The earth mover's distance seems like a prime candidate: it is a very natural notion of distance between a wide array of objects, and is sensitive to even small changes in the input. It has already been widely applied in many different domains, but unless the situation can be modelled by weighted points, it is always approximated numerically, without any guarantee of approximation bounds. A great step forward would be to find an exact algorithm that can compute it for objects other than weighted points. Efficiency is currently a secondary concern, as to our knowledge, no algorithm is known at all.

With respect to abstract morphing, there are many possible avenues for future research. We have investigated the problem from the perspective of the Hausdorff distance; an obvious choice would study the situation for a different metric. For instance, we could consider the case where our input are curves, and we want to interpolate between them with respect to the Fréchet distance. We trivially get an interpolated curve from the reparameterisations of the input curves: for an interpolation variable $\gamma \in [0, 1]$, simply take the curve given by $A(\alpha(t)) + \gamma(B(\beta(t)) - A(\alpha(t)))$. Similarly, computing the earth mover's distance gives us a mapping of mass from A to B, which lets us compute a natural middle shape by moving each point on A partway along the line segment to the point on B it is mapped to. The interesting questions would be what properties such a middle has, depending on the input shapes, or if it can be calculated directly from the input shapes, instead of via first computing the reparameterisations or transport plan. For the case of the earth mover's distance, another interesting question is how to handle shapes with different areas. For other metrics, like the area of symmetric difference, it is not clear how to compute a good middle shape. One option is to simply "flood fill" the parts of each shape containing the symmetric difference over time, with the flood filling from B to A being reversed in time, but this may not give appealing results.

Another aspect of the problem is the degree to which the results are visually convincing. Practical morphing methods are generally more controlled than the ones presented in our work. It would be interesting both to analyse existing "good" morphs from a theoretical perspective, to see if the resulting shapes can be bounded in terms of e.g. the Hausdorff distance to the shapes being morphed between, but also to see if our abstract methods can be made more practical. For instance, we could consider extensions of our Hausdorff morphing methods to allow for semantics to be maintained throughout the morph. For example, if both shapes have a part

representing a head, then the intermediate shapes should also have a head. This would be challenging, as this is directly at odds with our Hausdorff requirements, so we may need to redefine our framework to only consider the Hausdorff distance between the semantic features.

Finally, an obvious open problem with our model of indeterminacy in graphs is to give a full characterisation of all ambiguous sets of triples. Our extension to k-tuples could also be investigated further: we have shown that trees of sufficient size can be reconstructed, but does the same hold for other classes of graphs? Furthermore, the problem of determining whether a set of k-tuples is ambiguous in polynomial time is still open for $k > 3$; our method using a 2-SAT formula does not directly generalise to larger tuples. Many other models for indeterminacy could also be studied. For instance, can we reconstruct more graphs if we also specify for each triple or k-tuple how many edges were present in the original graph? We could also consider a model where not all tuples necessarily have the same size. This would allow us to make any set of tuples unambiguous by including some number of positive and negative 2-tuples, specifying the existence or non-existence some of the edges explicitly. What is the maximum number of 2-tuples needed to make a set of triples unambiguous, relative to the size of the set of triples? Are there any sets of k-tuples that can be made unambiguous by including k'-tuples for $2 < k' < k$? There is also an algorithmic question here: for a given set of triples (or k-tuples), find the minimum number of edges and non-edges (or k'-tuples) that need to be specified to make the reconstruction unambiguous. Finally, it would be interesting to find applications that are naturally modelled by this type of indeterminacy.

Bibliography

[1] M. Abrahamsen, J. Erickson, I. Kostitsyna, M. Löffler, T. Miltzow, J. Urhausen, J. L. Vermeulen and G. Viglietta. Chasing Puppies: Mobile Beacon Routing on Closed Curves. *Proceedings of the 37th International Symposium on Computational Geometry*, pages 5:1–5:19, 2021.

[2] P. K. Agarwal, K. Fox, D. Panigrahi, K. R. Varadarajan and A. Xiao. Faster Algorithms for the Geometric Transportation Problem. *Proceedings of the 33rd International Symposium on Computational Geometry*, pages 7:1–7:16, 2017.

[3] P. K. Agarwal, J. Pach and M. Sharir. State of the Union (of Geometric Objects). *Surveys on Discrete and Computational Geometry*, pages 9–48, 2008.

[4] M. Ahmed and C. Wenk. Constructing Street Networks from GPS Trajectories. *Proceedings of the 20th Annual European Symposium on Algorithms*, pages 60–71, 2012.

[5] H.-K. Ahn, S.-W. Cheng, H. J. Kweon and J. Yon. Overlap of convex polytopes under rigid motion. *Computational Geometry*, 47(1):15–24, 2014.

[6] O. Aichholzer, F. Aurenhammer, D. Alberts and B. Gärtner. A Novel Type of Skeleton for Polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[7] A. B. Albu, T. Beugeling and D. Laurendeau. A Morphology-Based Approach for Interslice Interpolation of Anatomical Slices from Volumetric Images. *IEEE Transactions on Biomedical Engineering*, 55(8):2022–2038, 2008.

[8] H. Alt, B. Behrends and J. Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3):251–265, 1995.

[9] H. Alt, P. Braß, M. Godau, C. Knauer and C. Wenk. Computing the Hausdorff Distance of Geometric Patterns and Shapes. *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, pages 65–76, 2003.

[10] H. Alt and M. Buchin. Can We Compute the Similarity between Surfaces? *Discrete & Computational Geometry*, 43(1):78–99, 2010.

[11] H. Alt, A. Efrat, G. Rote and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003.

[12] H. Alt, U. Fuchs, G. Rote and G. Weber. Matching Convex Shapes with Respect to the Symmetric Difference. *Algorithmica*, 21(1):89–103, 1998.

[13] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1):75–91, 1995.

[14] H. Alt and L. J. Guibas. Discrete Geometric Shapes: Matching, Interpolation, and Approximation. *Handbook of Computational Geometry*, pages 121–153, 2000.

[15] H. Alt and L. Scharf. Computing the Hausdorff distance between curved objects. *International Journal of Computational Geometry & Applications*, 18(4):307–320, 2008.

[16] N. Aspert, D. Santa-Cruz and T. Ebrahimi. Mesh: Measuring errors between surfaces using the Hausdorff distance. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 705–708, 2002.

[17] B. Aspvall, M. F. Plass and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[18] M. J. Atallah. A linear time algorithm for the Hausdorff distance between convex polygons. *Information Processing Letters*, 17(4):207–209, 1983.

[19] F. Aurenhammer, R. Klein and D.-T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013.

[20] G. Barequet, M. T. Goodrich, A. Levi-Steiner and D. Steiner. Contour interpolation by straight skeletons. *Graphical Models*, 66(4):245–260, 2004.

[21] G. Barequet and M. Sharir. Piecewise-Linear Interpolation between Polygonal Slices. *Computer Vision and Image Understanding*, 63(2):251–272, 1996.

[22] G. Barequet and A. Vaxman. Nonlinear interpolation between slices. *International Journal of Shape Modeling*, 14(1):39–60, 2008.

[23] G. Barequet and A. Vaxman. Reconstruction of Multi-Label Domains from Partial Planar Cross-Sections. *Computer Graphics Forum*, 28(5):1327–1337, 2009.

[24] M. de Berg, O. Cheong, O. Devillers, M. van Kreveld and M. Teillaud. Computing the Maximum Overlap of Two Convex Polygons under Translations. *Theory of Computing Systems*, 31(5):613–628, 1998.

[25] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd edition. Springer, 2008.

[26] D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 359–370, 1994.

[27] J.-D. Boissonnat. Shape Reconstruction from Planar Cross Sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1–29, 1988.

[28] J. A. Bondy and R. L. Hemminger. Graph reconstruction – A survey. *Journal of Graph Theory*, 1(3):227–268, 1977.

[29] Q. W. Bouts, I. Kostitsyna, M. van Kreveld, W. Meulemans, W. Sonke and K. Verbeek. Mapping Polygons to the Grid with Small Hausdorff and Fréchet Distance. *Proceedings of the 24th Annual European Symposium on Algorithms*, pages 22:1–22:16, 2016.

[30] U. Brandes and S. Cornelsen. Phylogenetic graph models beyond trees. *Discrete Applied Mathematics*, 157(10):2361–2369, 2009.

[31] K. Buchin, M. Buchin, W. Meulemans and B. Speckmann. Locally correct Fréchet matchings. *Computational Geometry*, 76:1–18, 2019.

[32] K. Buchin, M. Buchin and A. Schulz. Fréchet Distance of Surfaces: Some Simple Hard Cases. *Proceedings of the 18th Annual European Symposium on Algorithms*, pages 63–74, 2010.

[33] K. Buchin, E. W. Chambers, T. Ophelders and B. Speckmann. Fréchet Isotopies to Monotone Curves. *Proceedings of the 33rd European Workshop on Computational Geometry*, pages 41–44, 2017.

[34] K. Buchin, A. Nusser and S. Wong. Computing Continuous Dynamic Time Warping of Time Series in Polynomial Time. *Proceedings of the 38th International Symposium on Computational Geometry*, pages 22:1–22:16, 2022.

[35] S. Cabello, P. Giannopoulos, C. Knauer and G. Rote. Matching point sets with respect to the Earth Mover's Distance. *Computational Geometry*, 39(2):118–133, 2008.

[36] E. W. Chambers, É. Colin de Verdière, J. Erickson, S. Lazard, F. Lazarus and S. Thite. Homotopic Fréchet distance between curves or, walking your dog in the woods in polynomial time. *Computational Geometry*, 43(3):295–311, 2010.

[37] E. W. Chambers, D. Letscher, T. Ju and L. Liu. Isotopic Fréchet Distance. *Proceedings of the 23rd Canadian Conference on Computational Geometry*, 2011.

[38] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg and D. Kravets. Geometric pattern matching under Euclidean motion. *Computational Geometry*, 7(1):113–124, 1997.

[39] P. Cignoni, C. Rocchini and R. Scopigno. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms*. 4th edition. MIT Press, 2022.

[41] D. Dadush and S. Huiberts. A Friendly Smoothed Analysis of the Simplex Method. *SIAM Journal on Computing*, 49(5):STOC18-449–STOC18-499, 2020.

[42] T. K. Dey, J. Wang and Y. Wang. Graph Reconstruction by Discrete Morse Theory. *Proceedings of the 34th International Symposium on Computational Geometry*, pages 31:1–31:15, 2018.

[43] M.-P. Dubuisson and A. K. Jain. A Modified Hausdorff Distance for Object Matching. *Proceedings of the 12th IEEE International Conference on Pattern Recognition*, pages 566–568, 1994.

[44] A. Dumitrescu and G. Rote. On the Fréchet distance of a set of curves. *Proceedings of the 16th Canadian Conference on Computational Geometry*, pages 162–165, 2004.

[45] H. Edelsbrunner, L. Guibas, J. Pach, R. Pollack, R. Seidel and M. Sharir. Arrangements of curves in the plane–topology, combinatorics, and algorithms. *Theoretical Computer Science*, 92(2):319–336, 1992.

[46] A. Efrat, A. Itai and M. J. Katz. Geometry Helps in Bottleneck Matching and Related Problems. *Algorithmica*, 31(1):1–28, 2001.

[47] S. Even, A. Itai and A. Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

[48] T. Feder. Network flow and 2-satisfiability. *Algorithmica*, 11(3):291–319, 1994.

[49] S. Fortune. A Sweepline Algorithm for Voronoi Diagrams. *Algorithmica*, 2(1):153–174, 1987.

[50] K. Fox and J. Lu. A near-linear time approximation scheme for geometric transportation with arbitrary supplies and spread. *Journal of Computational Geometry*, 13(1):204–225, 2022.

[51] L. C. van der Gaag, R. Kuijper, Y. M. van Geffen and J. L. Vermeulen. Towards Uncertainty Analysis of Bayesian Networks. *Proceedings of the 25th Benelux Conference on Artificial Intelligence*, pages 223–230, 2013.

[52] D. Geiß, R. Klein, R. Penninger and G. Rote. Optimally solving a transportation problem using Voronoi diagrams. *Computational Geometry*, 46(8):1009–1016, 2013.

[53] F. de Goes, K. Breeden, V. Ostromoukhov and M. Desbrun. Blue Noise through Optimal Transport. *ACM Transactions on Graphics*, 31(6):171:1–171:11, 2012.

[54] F. de Goes, D. Cohen-Steiner, P. Alliez and M. Desbrun. An Optimal Transport Approach to Robust Reconstruction and Simplification of 2D Shapes. *Computer Graphics Forum*, 30(5):1593–1602, 2011.

[55] D. Goldfarb and S. Liu. An $O(n^3L)$ primal interior point algorithm for convex quadratic programming. *Mathematical Programming*, 49(1):325–340, 1990.

[56] C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75, 2001.

[57] K. Grauman and T. Darrell. Fast Contour Matching Using Approximate Earth Mover's Distance. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 220–227, 2004.

[58] D. Halperin and M. Sharir. Arrangements. *Handbook of Discrete and Computational Geometry*, 3rd edition, pages 723–762, 2017.

[59] S. Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, 2011.

[60] S. Har-Peled and B. Raichel. The Fréchet Distance Revisited and Extended. *ACM Transactions on Algorithms*, 10(1):1–22, 2014.

[61] R. M. Haralick, S. R. Sternberg and X. Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):532–550, 1987.

[62] E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkte. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923.

[63] I. van der Hoog, M. van de Kerkhof, M. van Kreveld, M. Löffler, F. Staals, J. Urhausen and J. L. Vermeulen. Mapping Multiple Regions to the Grid with Bounded Hausdorff Distance. *Proceedings of the 17th Algorithms and Data Structures Symposium*, pages 627–640, 2021.

[64] I. van der Hoog, E. Khramtcova and M. Löffler. Dynamic Smooth Compressed Quadtrees. *Proceedings of the 34th International Symposium on Computational Geometry*, pages 45:1–45:15, 2018.

[65] D. P. Huttenlocher, K. Kedem and M. Sharir. The Upper Envelope of Voronoi Surfaces and Its Applications. *Discrete & Computational Geometry*, 9(3):267–291, 1993.

[66] P. Jungeblut, L. Kleist and T. Miltzow. The Complexity of the Hausdorff Distance. *Proceedings of the 38th International Symposium on Computational Geometry*, pages 48:1–48:17, 2022.

[67] O. van Kaick, H. Zhang, G. Hamarneh and D. Cohen-Or. A Survey on Shape Correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[68] S. Kannan, C. Mathieu and H. Zhou. Graph Reconstruction and Verification. *ACM Transactions on Algorithms*, 14(4):1–30, 2018.

[69] L. V. Kantorovich. On the translocation of masses. *Doklady Akademii Nauk SSSR*, 37:199–201, 1942.

[70] V. Kassiano, A. Gounaris, A. N. Papadopoulos and K. Tsichlas. Mining Uncertain Graphs: An Overview. *Algorithmic Aspects of Cloud Computing*, pages 87–116, 2017.

[71] K. Kedem, R. Livne, J. Pach and M. Sharir. On the Union of Jordan Regions and Collision-Free Translational Motion Amidst Polygonal Obstacles. *Discrete & Computational Geometry*, 1(1):59–71, 1986.

[72] V. Keikha, M. van de Kerkhof, M. van Kreveld, I. Kostitsyna, M. Löffler, F. Staals, J. Urhausen, J. L. Vermeulen and L. Wiratma. Convex Partial Transversals of Planar Regions. *Proceedings of the 29th International Symposium on Algorithms and Computation*, pages 52:1–52:12, 2018.

[73] A. B. Khesin, A. Nikolov and D. Paramonov. Preconditioning for the Geometric Transportation Problem. *Journal of Computational Geometry*, 11(2):234–259, 2021.

[74] J. Kleinberg and É. Tardos. *Algorithm Design*. Addison-Wesley, 2005.

[75] C. Knauer, M. Löffler, M. Scherfenberg and T. Wolle. The directed Hausdorff distance between imprecise point sets. *Theoretical Computer Science*, 412(32):4173–4186, 2011.

[76] D. Knuth. *The Art of Computer Programming, Volumes 1–4B*. Addison-Wesley, 2022.

[77] L. de Kogel, M. van Kreveld and J. L. Vermeulen. Abstract Morphing Using the Hausdorff Distance and Voronoi Diagrams. *Proceedings of the 30th Annual European Symposium on Algorithms*, pages 74:1–74:16, 2022.

[78] E. Koutsoupias, C. H. Papadimitriou and M. Sideri. On the Optimal Bisection of a Polygon. *ORSA Journal on Computing*, 4(4):435–438, 1992.

[79] M. K. Kozlov, S. P. Tarasov and L. G. Khachiyan. The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5):223–228, 1980.

[80] M. van Kreveld, T. Miltzow, T. Ophelders, W. Sonke and J. L. Vermeulen. Between shapes, using the Hausdorff distance. *Computational Geometry*, 100:101817, 2022.

[81] M. van Kreveld, F. Staals, A. Vaxman and J. L. Vermeulen. Approximating the Earth Mover's Distance between sets of points and line segments. *Proceedings of the 35th European Workshop on Computational Geometry*, pages 24:1–24:6, 2019.

[82] J. Lauri. The Reconstruction Problem. *Handbook of Graph Theory*, pages 79–98, 2003.

[83] H. Lavenant, S. Claici, E. Chien and J. Solomon. Dynamical Optimal Transport on Discrete Surfaces. *ACM Transactions on Graphics*, 36(6):250:1–250:16, 2018.

[84] Z. Liu, L. Zhou, H. Leung and H. P.-H. Shum. High-quality compatible triangulations and their application in interactive animation. *Computers & Graphics*, 76:60–72, 2018.

[85] A. Maheshwari, J.-R. Sack and C. Scheffer. Approximating the integral Fréchet distance. *Computational Geometry*, 70:13–30, 2018.

[86] F. Mémoli. Spectral Gromov-Wasserstein Distances for Shape Matching. *Proceedings of the 12th IEEE International Conference on Computer Vision Workshops*, pages 256–263, 2009.

[87] Q. Mérigot. A Multiscale Approach to Optimal Transport. *Computer Graphics Forum*, 30(5):1583–1592, 2011.

[88] Q. Mérigot, J. Meyron and B. Thibert. An Algorithm for Optimal Transport between a Simplex Soup and a Point Cloud. *SIAM Journal on Imaging Sciences*, 11(2):1363–1389, 2018.

[89] J. S. B. Mitchell, D. M. Mount and C. H. Papadimitriou. The Discrete Geodesic Problem. *SIAM Journal on Computing*, 16(4):647–668, 1987.

[90] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, :666–705, 1781.

[91] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part II: Convex quadratic programming. *Mathematical Programming*, 44(1):43–66, 1989.

[92] J. N. Mordeson and C.-S. Peng. Operations on Fuzzy Graphs. *Information Sciences*, 79(3):159–170, 1994.

[93] E. Mossel and N. Ross. Shotgun assembly of labeled graphs. *IEEE Transactions on Network Science and Engineering*, 6(2):145–157, 2017.

[94] S. Ramachandran. Graph Reconstruction - Some New Developments. *AKCE International Journal of Graphs and Combinatorics*, 1(1):51–61, 2004.

[95] A. Rosenfeld. Fuzzy Graphs. *Fuzzy Sets and their Applications to Cognitive and Decision Processes*, pages 77–95, 1975.

[96] G. Rote. Lexicographic Fréchet matchings. *Proceedings of the 30th European Workshop on Computational Geometry*, 2014.

[97] Y. Rubner, C. Tomasi and L. J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[98] C. Schulz, A. Nocaj, J. Goertler, O. Deussen, U. Brandes and D. Weiskopf. Probabilistic Graph Layout for Uncertain Network Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):531–540, 2017.

[99] R. Sharathkumar and P. K. Agarwal. A Near-Linear Time ε-Approximation Algorithm for Geometric Bipartite Matching. *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 385–394, 2012.

[100] Y. Shiono, T. Kirishima, Y. Ueda and K. Tsuchida. Drawing Algorithm for Fuzzy Graphs Using the Partition Tree. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 16(5):641–652, 2012.

[101] J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du and L. Guibas. Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains. *ACM Transactions on Graphics*, 34(4):66:1–66:11, 2015.

[102] F. Staals, J. Urhausen and J. L. Vermeulen. Querying the Hausdorff Distance of a Line Segment. *Proceedings of the 38th European Workshop on Computational Geometry*, pages 60:1–60:8, 2022.

[103] Z. Su, Y. Wang, R. Shi, W. Zeng, J. Sun, F. Luo and X. Gu. Optimal Mass Transport for Shape Matching and Comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2246–2259, 2015.

[104] J. L. Vermeulen, A. Hillebrand and R. Geraerts. A Comparative Study of k-Nearest Neighbour Techniques in Crowd Simulation. *Computer Animation and Virtual Worlds*, 28(3):e1775, 2017.

[105] J. L. Vermeulen, A. Hillebrand and R. Geraerts. Annotating Traversable Gaps in Walkable Environments. *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pages 3045–3052, 2018.

[106] A. Vigneron. Geometric Optimization and Sums of Algebraic Functions. *ACM Transactions on Algorithms*, 10(1):1–20, 2014.

[107] C. Villani. *Optimal Transport: Old and New*. Springer Verlag, 2008.

[108] Y. Yongzhi. The Reconstruction Conjecture is True if All 2-Connected Graphs are Reconstructible. *Journal of Graph Theory*, 12(2):237–243, 1988.

Nederlandse samenvatting

Computers en automatisering vervullen een steeds belangrijkere rol in onze samenleving. Nieuw vergaarde kennis en inzicht uit onderzoek binnen de informatica is een van de drijvende krachten achter deze groeiende rol, en de theoretische informatica vormt de basis van dit vakgebied. In de theoretische informatica houden we ons bezig met wat het betekent als we zeggen dat een probleem door een computer opgelost kan worden, en proberen we te bepalen welke problemen dit zijn. We ontwikkelen algoritmes en datastructuren waarmee we deze problemen op de beste manier kunnen oplossen. 'Beste' kan dan refereren aan veel verschillende relevante aspecten, zoals snelheid, precisie of betrouwbaarheid. Veel werk is gericht op het wiskundig bewijzen dat een probleem door een computer opgelost kan worden, of op het bewijzen dat een specifiek algoritme een gegeven probleem optimaal oplost.

In de computationele geometrie bestuderen we deze vraagstukken voor problemen met een geometrisch aspect. In plaats van ons alleen bezig te houden met abstracte zaken als getallen en vergelijkingen, focussen we ons op de gevallen waarbij de objecten die we bestuderen geometrisch zijn. Het aantal geometrische problemen dat we mogelijk willen oplossen is bijna eindeloos. Sommige problemen zijn vrij abstract, zoals het volgende: gegeven een verzameling cirkels, bepaal of er een punt bestaat dat binnen alle cirkels ligt. Andere problemen hebben voor de hand liggende praktische toepassingen, zoals: bereken het kortste pad tussen twee gegeven locaties in een omgeving met obstakels.

In de bovengenoemde problemen is het relatief makkelijk om te definiëren wat een correcte oplossing is. Dit is echter niet het geval voor alle computationele vraagstukken. Neem bijvoorbeeld de vraag wat in een gegeven schaakpositie de beste zet is. Of de vraag om aan de hand van een 3D-model van een boom nieuwe 3D-modellen van dezelfde boomsoort te genereren. In zulke gevallen is het niet altijd mogelijk om een optimale oplossing te berekenen, en is het vaak niet eens duidelijk wat 'optimaal' betekent.

We kunnen meerdere technieken toepassen om dit soort complexere problemen toch op te laten lossen door een computer. We zouden bijvoorbeeld een AI kunnen trainen om goede schaakzetten te maken, of een genetisch algoritme kunnen gebruiken om een procedure te vinden die nieuwe boommodellen kan genereren.

Voor zo'n soort aanpak is het vaak nodig om te kunnen meten hoe goed een gegeven oplossing is. Schaakprogramma's hebben bijvoorbeeld vaak een evaluatiefunctie die probeert in te schatten hoe gunstig een bepaalde bordpositie is voor de speler. Deze evaluatiefunctie kan dan gebruikt worden om efficiënter naar goede zetten te zoeken. Voor het genereren van boommodellen zouden we een functie kunnen definiëren die de gelijkenis met het voorbeeldmodel meet, en die dan gebruikt kan worden om het evolutionaire proces te sturen.

In dit proefschrift onderzoeken we zulke maten van gelijkenis voor geometrische objecten, en kijken we ook naar hoe deze gebruikt kunnen worden in bepaalde toepassingen. In Hoofdstuk 2 kijken we naar de *earth mover's distance*, een afstandsmaat die veel gebruikt wordt om discrete data, zoals foto's, te vergelijken. Deze afstandsmaat kan informeel als volgt worden uitgelegd. Stel dat je een aantal hopen aarde hebt liggen, en een aantal gaten in de grond van exact dezelfde totale omvang. De gaten in de grond moeten gevuld worden met de stapels aarde, maar er zijn bepaalde kosten verbonden aan het verplaatsen van de aarde: deze kosten zijn gelijk aan het product van de massa van de aarde die verplaatst wordt en de afstand waarover deze verplaatst wordt. De earth mover's distance tussen de stapels aarde en de gaten is gelijk aan de totale kosten van de goedkoopste manier om de gaten te vullen.

Deze beschrijving kunnen we formeel maken zodanig dat de afstandsmaat goed gedefinieerd is voor massadistributies van verscheidene objecten, zoals punten, polygonen en 3D-objecten. Bestaande algoritmes om de afstand te berekenen vallen echter uiteen in twee categorieën. Aan de ene kant is er veel bekend over het geval waarbij de afstand tussen twee verzamelingen (gewogen) punten wordt uitgerekend. Deze variant kan efficiënt exact uitgerekend worden, of nog efficiënter worden benaderd. Aan de andere kant wordt het probleem in toepassingen, zoals het interpoleren tussen 3D-objecten, vaak numeriek opgelost. Hierbij kan over het algemeen wel gegarandeerd worden dat de oplossing convergeert naar de juiste oplossing, maar kunnen geen garanties gegeven worden over hoe goed de uiteindelijke benadering is.

Onze bijdrage is het eerste approximatie-algoritme voor de earth mover's distance voor situaties waarbij minstens een van de twee verzamelingen andere objecten dan punten bevat. Wij geven algoritmes die, in polynomiale tijd, de earth mover's distance tussen een verzameling gewogen punten en een verzameling lijnsegmenten, driehoeken of hoger-dimensionale simplices benaderen tot een factor $1 + \varepsilon$, waar ε een door de gebruiker gekozen kleine constante is. Ook geven we algoritmes die, in polynomiale tijd, de earth mover's distance tussen twee verzamelingen lijnsegmenten, driehoeken of hoger-dimensionale simplices benaderen tot een factor $1 + \varepsilon$ plus een kleine additieve term.

In de Hoofdstukken 3 en 4 kijken we naar hoe we kunnen interpoleren tussen twee vormen. Interpolatie tussen twee vormen wordt ook wel *morphing* genoemd, en heeft toepassingen in bijvoorbeeld medische beeldvorming en animatie. Wij bekijken morphing specifiek in de context van de Hausdorff-afstand. Dat is een simpele afstandsmaat die iets kan zeggen over de mate waarin twee objecten op

elkaar lijken. Deze afstandsmaat kijkt voor alle punten op een van de twee objecten naar het dichtstbijzijnde punt op het andere object, en is gelijk aan het maximum van al deze afstanden.

In Hoofdstuk 3 definiëren we een morphing-methode die, gegeven twee invoervormen, als eigenschap heeft dat hij op tijdstip $\alpha \in [0, 1]$ altijd Hausdorff-afstand α tot de eerste invoervorm heeft, en Hausdorff-afstand $1 - \alpha$ tot de tweede invoervorm. We bewijzen enkele belangrijke eigenschappen die deze morphing-methode heeft. Zo is de vorm op ieder tijdstip de maximale vorm met de genoemde Hausdorff-afstanden, en is hij 1-Lipschitz continu. Ook analyseren we structurele eigenschappen, zoals de verbondenheid en de beschrijvingscomplexiteit van de tussenvormen. We bestuderen ook een uitbreiding naar een situatie met meer dan twee invoervormen, waarbij we laten zien dat het niet altijd mogelijk is om een vorm te vinden die een kleinere Hausdorff-afstand dan 1 heeft tot alle invoervormen. Wel geven we algoritmes die de kleinst haalbare afstand kunnen berekenen.

De Hausdorff morph uit Hoofdstuk 3 heeft mooie eigenschappen, maar in de praktijk heeft hij de neiging om halverwege een vorm te geven die op geen van de twee invoervormen lijkt. In Hoofdstuk 4 geven we een alternatieve morph die veel van de goede eigenschappen behoudt, en er in de praktijk subjectief beter uitziet. Dit onderbouwen we door een experimentele vergelijking van de twee morphs op basis van oppervlakte, omtrek, en het aantal componenten en gaten van de tussenvormen. De alternatieve morph definiëren we als volgt: op tijdstip $\alpha \in [0, 1]$ bewegen we alle punten op de eerste invoervorm een factor α naar het dichtstbijzijnde punt op de tweede invoervorm toe. Alle punten op de tweede bewegen analoog, maar met een factor $1 - \alpha$. De vereniging van al deze punten vormt de tussenvorm op tijdstip α. Deze morph noemen we de *Voronoi morph*, omdat hij met behulp van Voronoi-diagrammen berekend kan worden. We testten ook een aangepaste versie van deze morph, *mixed morph* genaamd, die een veel voorkomend probleem met de Voronoi morph probeert op te lossen. Onze experimenten laten zien dat de mixed morph het beste presteert op de eigenschappen die we getest hebben.

Als laatste behandelen we in Hoofdstuk 5 een ongerelateerd onderwerp dat te maken heeft met graafreconstructie. We kijken specifiek naar een informatiemodel waarin we de structuur van een graaf niet expliciet gegeven krijgen. In plaats daarvan krijgen we alleen een lijst van alle combinaties van drie knopen in de graaf die een verbonden geïnduceerde subgraaf vormen. We noemen deze combinaties van drie knopen *triples*. De vraag is nu of de oorspronkelijke graaf op basis van deze informatie gereconstrueerd kan worden. We bewijzen dat dit in het algemeen niet mogelijk is: er zijn meerdere oneindige families van paren van verschillende grafen die dezelfde set triples opleveren. Tegelijkertijd laten we zien dat voor sommige klassen grafen de reconstructie wel altijd uniek is als de graaf groot genoeg is. Ook geven we een simpel algoritme gebaseerd op 2-SAT dat een graaf efficiënt reconstrueert op basis van een verzameling triples. Als laatste bestuderen we kort een generalisatie naar verbonden subgrafen van meer dan drie knopen.

Curriculum vitae

Jordi Vermeulen was born on May 15, 1991, in Amsterdam. He completed his secondary education in a bilingual programme at the Berlage Lyceum in Amsterdam in 2009. After attending Amsterdam University College for one year, and working for the City of Amsterdam, he completed his computer science bachelor in the *Game Technology* track (cum laude, with honours) at Utrecht University in 2014. In 2017, he earned his master's degree (cum laude) in the *Game and Media Technology* track at the same university with his thesis 'Bridging gaps in walkable environments'. He started as a PhD student in the Geometric Computing group of the Department of Information and Computing Sciences of Utrecht University in the same year. During his time as a PhD student, he was a TA for a variety of courses, was a co-lecturer for the *Game Programming* bachelor's course, and co-developed and co-lectured the new master's course *AI for Game Technology*.

Published papers not presented in this thesis

- L. C. van der Gaag, R. Kuijper, Y. M. van Geffen and J. L. Vermeulen. Towards Uncertainty Analysis of Bayesian Networks. *Proceedings of the 25th Benelux Conference on Artificial Intelligence*, pages 223–230, 2013.

- J. L. Vermeulen, A. Hillebrand and R. Geraerts. A Comparative Study of k-Nearest Neighbour Techniques in Crowd Simulation. *Computer Animation and Virtual Worlds*, 28(3):e1775, 2017.

- J. L. Vermeulen, A. Hillebrand and R. Geraerts. Annotating Traversable Gaps in Walkable Environments. *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pages 3045–3052, 2018.

- V. Keikha, M. van de Kerkhof, M. van Kreveld, I. Kostitsyna, M. Löffler, F. Staals, J. Urhausen, J. L. Vermeulen and L. Wiratma. Convex Partial Transversals of Planar Regions. *Proceedings of the 29th International Symposium on Algorithms and Computation*, pages 52:1–52:12, 2018.

- I. van der Hoog, M. van de Kerkhof, M. van Kreveld, M. Löffler, F. Staals, J. Urhausen and J. L. Vermeulen. Mapping Multiple Regions to the Grid with Bounded Hausdorff Distance. *Proceedings of the 17th Algorithms and Data Structures Symposium*, pages 627–640, 2021.
- M. Abrahamsen, J. Erickson, I. Kostitsyna, M. Löffler, T. Miltzow, J. Urhausen, J. L. Vermeulen and G. Viglietta. Chasing Puppies: Mobile Beacon Routing on Closed Curves. *Proceedings of the 37th International Symposium on Computational Geometry*, pages 5:1–5:19, 2021.
- F. Staals, J. Urhausen and J. L. Vermeulen. Querying the Hausdorff Distance of a Line Segment. *Proceedings of the 38th European Workshop on Computational Geometry*, pages 60:1–60:8, 2022.

Acknowledgements

First and foremost, I want to thank my advisors for their great support over the last five years. Doing research together was a lot of fun, and I appreciate that you were always willing to help with any issues I was having. I learned a great deal about research, writing, teaching and maths from you. I also really enjoyed our informal chats on a wide variety of topics, such as music, travel and art.

Next, I am very thankful to Ivor and Jérôme for sharing an office with me and making it a nice place to work and hang out. I had a tremendous amount of fun talking, doing research, and sharing some hobbies with you. Ivor, thank you for rekindling my old interest in Warhammer and miniature painting. Jérôme, thank you for getting me back to playing Age of Empires, and for joining me on many bouldering sessions.

I also want to thank all the other people in the Geometric Computing group for just generally being a great bunch of people to work with. I really enjoyed doing research, playing board games and attending conferences with all of you. I would like to thank Maarten in particular for organising so many board game evenings. In addition I would like to thank Bettina, together with Marc, for organising many joint events with the people from Eindhoven, especially the AGA workshops. They were really one of the highlights of each year in which I was able to attend.

One of the things I enjoy the most is teaching, and in that context I want to thank Remco and Till for teaching courses with me multiple times. I really enjoyed teaching Game Programming, and I want to thank Remco for all the effort he put into this course, and especially for taking care of most of the organisational aspects. Till, I had a lot of fun developing a new course with you, and I really appreciate all the work and ideas you put into it. I'm quite proud of the course we were able to produce together!

Finally, and most importantly, I want to thank my family for their constant support and encouragement. There is great comfort in knowing that there are people who are always there for you, regardless of the situation. I also want to thank Ben for all the amazing holidays we've shared together. I hope that there will be many more to come in the future!