MASTER

Algorithms for comparing moving complex shapes
higher-dimensional Fréchet distance

Ophelders, T.A.E.

Award date:
2014

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
Algorithms for Comparing Moving Complex Shapes

Higher-Dimensional Fréchet Distance

Master’s Thesis

Tim Ophelders

Supervisors  
Dr. Kevin Buchin  
Prof. Dr. Bettina Speckmann  

TU/e

Section  
Algorithms

Date  
Eindhoven, September 29, 2014
Abstract

We investigate methods for the computation of the similarity between moving shapes. We will focus on the comparison of moving curves, which are modeled as moving polylines. Moving polylines are in turn represented as quadrilateral meshes. We present methods for the comparison of such quadrilateral meshes under adaptations of the Fréchet distance. Our results show that computing the Fréchet distance between quadrilateral meshes is NP-hard even in various settings. We also consider three more restricted settings in which the Fréchet distance can be computed using polynomial time algorithms. The polynomial algorithms assume that the compared moving shapes move synchronously, such that part of their matching is known in advance.
Contents

1 Introduction 1
  1.1 Shapes .................................................. 2
    1.1.1 Quadrilateral Meshes .............................. 2
  1.2 Norms .................................................. 4
  1.3 Matchings and Applications ......................... 5
  1.4 Results ............................................... 9

2 Computation ............................................. 11
  2.1 Synchronous Identity .................................. 12
  2.2 Synchronous Constant ................................ 14
    2.2.1 Matching non-moving curves ..................... 14
    2.2.2 Matching moving curves ........................... 16
  2.3 Synchronous Dynamic ................................ 24
    2.3.1 Decision Problem .................................. 24
    2.3.2 Parametric search .................................. 27
    2.3.3 Alternative exact computation .................... 28

3 NP-Hardness ............................................. 29
  3.1 Example ................................................ 29
  3.2 Asynchronous Constant ............................... 31
  3.3 Asynchronous Dynamic ................................ 37
  3.4 Unrestricted Dynamic ................................ 37

4 Freespace Cell Visualizations ......................... 41

5 Conclusion ............................................. 45
  5.1 Future Work ........................................... 45
Chapter 1

Introduction

A great amount of geographic data has been collected over the past few years. Applications deliver satellite imagery to the public in the form of world and road maps. Similarly, data on road traffic is being collected in real time to allow rerouting trips, thereby speeding up traffic. Although these large scale examples collect a lot of time-varying data, small scale examples also exist. In fact, as motion tracking devices become cheaper, more and more people are tracking their own movement data when hiking, biking, running and even driving. In personal health, people have started self-monitoring their heart rate, sleep patterns and even metabolism.

Research is actively trying to extract patterns from such data in order to create a model for how the data changes over time. Accurate models can then be used to simulate such dynamics and predict changes in real world data. To validate the accuracy of these models, we need reasonable ways to compare them to real world behavior.

For simplicity, we assume data consist of a sequence of instances. An instance is the data collected at a given point in time, or timestamp, represented in some canonical form. A series of instances is called a trace, which intuitively captures the changes of the data over time. An instance may be equipped with a timestamp to capture information about the temporal relation between instances. This thesis covers the comparison of movement data for complex shapes, meaning that the traces will represent trajectories of shapes.

For comparing trajectories representing the movement of a single point, a popular similarity measure between two trajectories is the Fréchet distance, we define it together with some other comparison methods in Section 1.2. To compute the Fréchet distance between polylines of length n, an \( O(n^2 \log n) \) time algorithm \[\text{AG95}\] can be used. In different contexts, slight variations \[\text{BBW09, BBMS12, DHP13, BBMM14}\] of the Fréchet distance may be suitable. For example, we might wish to ignore the relative positions or orientations of the compared polylines \[\text{CM03}\], or we might prefer a more smooth shape \[\text{Rot07}\] to represent the movement of a point. Further extensions allow comparing polylines to embedded graphs \[\text{AERW03, CDG+11}\], whereas even further extensions to compare two embedded graphs are generally NP-hard due to the graph isomorphism problem.

In addition to one dimensional shapes, attempts have been made to compare pairs of surfaces. In many cases, this turns out to be NP-hard \[\text{God99, BBS10}\]. It was recently shown that the Fréchet distance between surfaces is upper computable \[\text{AB10}\], but it is unknown whether it is fully computable. Since moving curves resemble surfaces, we can deduce that comparing moving shapes is a challenging task.

Many practical moving shapes form trajectories that are more complex than points. Think of trajectories whose instances are shapes such as a curve, a polygon, an embedded graph or even an embedded simplicial complex. Practical examples include countries, forests, mountains,
buildings, but also road maps, rivers, spider webs, snakes, handwriting and more abstract objects like trajectories, family trees and social networks. When tracking a forest as it grows or shrinks, we can think of representing its outline as a polygon. The resulting movement of a forest is then represented as the trace of a polygon. Similarly, when tracking the movement of a slithering snake, it makes sense to represent the movement of its spine as the trace of a polyline. As a final example, the dynamics of a road map can be captured in embedded graphs.

Note that for snakes, we can collect additional structure for the polyline by keeping track of certain points on their skin. Because of this, we can track the movement of a single point on the snake’s spine. In this case, we say that snakes are equipped with a matching between consecutive instances. Forests do not naturally provide such a matching because it is harder to label points on the polygonal outline between consecutive instances of a growing forest. We should note that for some classes of shapes—including polylines and polygons—algorithms exist to calculate matchings between consecutive instances with sensible properties \cite{AC95}. Given the trace of a shape equipped with matchings between the consecutive instances, we can interpolate shapes between instances and treat time as a continuous dimension.

We present algorithms for moving curves, which can be represented as trajectories of polylines when an interpolation between consecutive instances is provided. We model moving curves as quadrilateral meshes as explained in Section 1.1. The methods we present for comparing those shapes are explained in Section 1.3 and our results are summarized in Section 1.4.

1.1 Shapes

A shape is a function \( P \rightarrow \mathbb{R}^n \) where \( P \) is some set referred to as the parameter space of the shape. Examples of shapes (see Figure 1.1) include curves \( [0, 1] \rightarrow \mathbb{R}^n \) and surfaces \( [0, 1]^2 \rightarrow \mathbb{R}^n \). We represent shapes this way because the structure of the parameter space is often important when comparing shapes. In these cases, simply representing a shape as a subset of \( \mathbb{R}^n \) would ignore important information.

![Figure 1.1: Examples of shapes.](image)

In this thesis, we restrict ourselves to the comparison of moving curves, which form a basis for many other complex moving shapes such as graphs. For practical reasons, curves are commonly represented as a polyline, which is a chain of line segments. We define a similar model for moving curves that is based on moving line segments. As we will see shortly, it is convenient to represent a moving line segment as a strip of quadrilaterals, and moving curves as a particular shape called a quadrilateral mesh.

1.1.1 Quadrilateral Meshes

In Figure 1.2a we linearly interpolate a single line segment between two instances at timestamps \( t = 0 \) and \( t = 1 \), forming a moving line segment. Any point of the moving line segment then has a trajectory indicated by the dashed lines. We represent the linear interpolation between such consecutive instances of a line segment as a quadrilateral, as illustrated in Figure 1.2b.
1.1. Shapes

A quadrilateral can be defined using its four corner-points in \( \mathbb{R}^n \). Let \( Q \) be a quadrilateral with corners \( q_1, q_2, q_3 \) and \( q_4 \) as in Equation (1.1). Denote the bilinear interpolations of its corners by \( Q : [0, 1]^2 \rightarrow \mathbb{R}^n \) as defined in Equation (1.2). As illustrated in Figure 1.2b, we say that points in the image of \( Q \) lie on \( Q \). Observe that interchanging parameters \( p \) and \( t \) of \( Q \) is equivalent to interchanging points \( q_2 \) and \( q_3 \), which results in an other quadrilateral.

\[
V_Q = \begin{pmatrix}
q_{1,1} & \cdots & q_{4,1} \\
\vdots & \ddots & \vdots \\
q_{1,n} & \cdots & q_{4,n}
\end{pmatrix} \in \mathbb{R}^{n \times 4}
\tag{1.1}
\]

\[
Q(p,t) = V_Q \begin{pmatrix}
(1-p)(1-t) \\
(1-p)t \\
p(1-t) \\
p t
\end{pmatrix} = (1-p)(1-t)q_1 + (1-p)tq_2 + ptq_3 + p(1-t)q_4
\tag{1.2}
\]

The trajectory of a line segment over more than two instances can then be constructed as a chain of quadrilaterals. Furthermore, the trajectory of a polyline can be constructed as a mesh of quadrilaterals, called a quadrilateral mesh (see Figure 1.3). We therefore use quadrilateral meshes to model moving curves. Before we define ways to compare them, we first look at how simpler shapes can be compared in Section 1.2.

Figure 1.3: From a sequence of polylines to a quadrilateral mesh.
1.2 Norms

In the past, methods have been developed to compare relatively simple shapes. Two static points can easily be compared using an appropriate distance metric, perhaps the most famous metric is the Euclidean distance, which is often suitable geographic applications. The Euclidean distance is also called the \( L^2 \)-norm and can be generalized to \( L^p \) norms. For comparing two points \( a \) and \( b : \mathbb{R}^n \), the \( L^p \)-norms are characterized by Equation (1.3). Even more general norms can be defined with a metric space \((M,d)\) where \( d : M \times M \rightarrow \mathbb{R} \) is a distance function. However, we shall often restrict ourselves to metric spaces of the form \((\mathbb{R}^n, L^p)\).

\[
L^p(a, b) = \|b - a\|_p = \left( \sum_{k=1}^{n} |b_k - a_k|^p \right)^{1/p}
\]

(1.3)

The comparison between moving points is not so simple and depending on the intended application, different methods may be desirable. Once we have settled on a distance metric between points, there are various ways in which we can extend the comparison to handle the trajectory given by a moving point.

The Hausdorff distance as defined in Equation (1.4) is often used to measure the similarity of two point sets \( A \) and \( B \). Because each shape can be interpreted as a point set, the Hausdorff distance can be used to compare any pair of shapes. Although each trajectory can be interpreted as a point set, doing so often ignores too much of the internal structure of the shape.

\[
d_H(A, B) = \max \left\{ \sup_{a \in A} \inf_{b \in B} \|b-a\|, \sup_{b \in B} \inf_{a \in A} \|b-a\| \right\}
\]

(1.4)

To illustrate why the Hausdorff distance is often an undesirable method for comparing trajectories, we give a concrete example of a better way to compare trajectories. Suppose one trajectory is sampled at 800 timestamps over the course of a day, and we want to compare this to a trajectory that is sampled at 40 timestamps over the course of six hours. The best way to compare those trajectories largely depends on the nature of the trajectories. To compare them we might select a portion of the larger trajectory and compare it to the smaller trajectory. Alternatively, we might speed up the larger trajectory so that it spans the same time interval. Once the timestamps of the trajectories are aligned, points on different trajectories at the same timestamp are assumed to have some meaningful distance between them.

Now we can define the similarity between the two trajectories as the distance between points of the trajectory at the timestamp that they are furthest apart. This measure already seems more accurate than the Hausdorff distance. For instance, if the trajectories represent the flight of two planes, then the Hausdorff distance might compare the landing of one plane to the take off of an other plane, which is often not a good measure.

We have explained a few ways of realigning the timestamps of trajectories to give a suitable way to compare trajectories. The Fréchet distance is a method for the comparison of curves that deals with such alignments in their full generality. We typically call these alignments reparameterizations of timestamps. Assume two curves \( A \) and \( B : [0, 1] \rightarrow \mathbb{R}^n \) are aligned using functions \( \mu \) and \( \nu : [0, 1] \rightarrow [0, 1] \), respectively. For instance if \( \mu(0.2) = 0.5 \), then \( A(\mu(0.2)) \) is the point halfway curve \( A \).
Given $\mu$ and $\nu$, a simple method is then to define the distance between the curves as in Equation (1.5), namely as the distance between the two aligned points that are furthest apart. If $\nu \circ \mu^{-1}$ is the identity function $id(t) = t$, we call the alignment an identity-alignment.

$$d_{\mu,\nu}(A, B) = \sup_{t \in [0, 1]} \|B(\nu(t)) - A(\mu(t))\|$$

(1.5)

Rather than a single alignment, it is often desirable to consider entire classes of alignments. Assuming that $\mu$ and $\nu$ range over the considered reparameterizations of the trajectories, the Fréchet distance as defined in Equation (1.6) takes the minimum over all allowed reparameterizations. The precise timestamps associated with the trajectories are generally ignored, and instead only the order of timestamps matters.

$$d_{FD}(A, B) = \inf_{\mu,\nu} d_{\mu,\nu}(A, B)$$

(1.6)

The Fréchet distance is often explained with an analogy characterized by a man walking his dog, where the two curves represent the paths along which the respective man and dog walk. Their Fréchet distance is then the minimal required length of the leash, such that the dog stays within leash-distance of the man for at least one allowed reparameterization. A commonly considered class of reparameterizations is that of continuous nondecreasing surjections, meaning that the man and dog continuously walk along their entire curves without backtracking.

We can extend the Fréchet distance from moving points to moving shapes with parameter space $P$ using Equation (1.7) instead of Equation (1.5). Here, we extended reparameterizations $\mu$ and $\nu$ to incorporate the parameter space of the shapes. This allows realigning the internal structure of the compared shapes as well as timestamps.

$$d_{\mu,\nu}(A, B) = \sup_{(p,t) \in P \times [0,1]} \|B(\nu(p,t)) - A(\mu(p,t))\|$$

(1.7)

Let us consider moving curves (or quadrilateral meshes) from this point onwards. To obtain a point on a moving curve, we need two parameters, namely the position along the curve, and timestamp along its trajectory. Therefore, a curve has a parameter space $P = [0,1]$, which yields $\mu : [0, 1]^2 \to [0, 1]^2$ and $\nu : [0, 1]^2 \to [0, 1]^2$ for moving curves, see Equation (1.8). The definition of the Fréchet distance now allows us to choose restrictions on reparameterizations $\mu$ and $\nu$ to make them suitable for our use. We describe nine such restrictions for moving curves in Section 1.3.

$$d_{\mu,\nu}(A, B) = \sup_{(p,t) \in [0,1]^2} \|B(\nu(p,t)) - A(\mu(p,t))\|$$

(1.8)

### 1.3 Matchings and Applications

To summarize the Fréchet distance, consider Figure 1.4, we are given two moving curves $A$ and $B : [0, 1]^2 \to \mathbb{R}^n$ and choose a suitable distance function $d : \mathbb{R}^n \to \mathbb{R}$ to compare points on $A$ and $B$. The two functions $\mu$ and $\nu : [0, 1]^2 \to [0, 1]^2$ define which points on $A$ should be compared to which points on $B$ using $\nu \circ \mu^{-1}$, we call this mapping between points of $A$ and $B$ a matching. With the mere restriction that reparameterizations $\mu$ and $\nu$ are surjective, we obtain the Hausdorff distance. Because this may be undesirable, we consider several more restrictive classes of matchings $\nu \circ \mu^{-1}$.
To capture the essence of classes of matchings, we can illustrate them visually as in Figure 1.5. Given such an image, one can find a point on $B$ corresponding to some point on $A$. Beware that not all matchings can be represented this way, namely when multiple points on $A$ map to the same point on $B$. However, these images will suffice to give an intuition for a certain class of reparameterizations, we will often draw $\nu \circ \mu^{-1}(id)$ for some representative matching $\nu \circ \mu^{-1}$ in that class.

In this thesis, both the time ($t$) and the position ($p$) parameters of moving curves will be subject to three types of reparameterizations, namely fixed, constant or dynamic. We use the keywords **Synchronous**, **Asynchronous** and **Unrestricted** to refer to classes of alignments of the time parameter and the keywords **Identity**, **Constant** and **Dynamic** to refer to classes of alignments of the position parameter. As a visual aid, representative matchings for the covered classes are illustrated in Table 1.1. We will define and explain the restrictions we define for those classes on the next page.

Throughout this thesis, we use functions $\alpha$, $\beta$, $\mu$, $\nu$, $\tau$ and $\sigma$ to denote continuous surjections. As a convention, $\alpha$ and $\beta$ reparametrize positions at a given timestamp and symmetrically $\tau$ and $\sigma$ reparameterize time. The combined reparameterizations (of entire moving curves) are denoted by $\mu$ and $\nu$. We use $\alpha$ and $\sigma$ within the definition of $\mu$, which is the reparameterization of moving curve $A$, and we use $\beta$, $\tau$ and $\nu$ for $B$. 
1.3. Matchings and Applications

**Synchronous Identity**

\[ \mu(p, t) = (p, t) \quad \nu(p, t) = (p, t) \]

The simplest class considers a single matching, namely the identity matching \((\nu \circ \mu^{-1})(p, t) = (p, t)\). Calculating the Fréchet distance under this class is useful if the desired alignment between the two moving shapes is known in advance. The Synchronous Identity Fréchet distance can be computed in \(O(pt)\) time.

**Synchronous Constant**

\[ \mu(p, t) = (\alpha(p), t) \quad \nu(p, t) = (\beta(p), t) \]

The Synchronous Constant class assumes that the alignment of timestamps is known in advance, but the alignment of positions along curves is not. The reparametrizations are of the form \(\mu(p, t) = (\alpha(p), t)\) and \(\nu(p, t) = (\beta(p), t)\), where \(\alpha\) and \(\beta\) are continuous nondecreasing surjections. This class can be used to compare shapes whose individual locations can be tracked, but the correspondence of positions between shapes is unknown.

**Synchronous Dynamic**

\[ \mu(p, t) = (\alpha(p, t), t) \quad \nu(p, t) = (\beta(p, t), t) \]

The Synchronous Dynamic class may be useful when the individual locations of a shape cannot be tracked. The relations between positions on consecutive instances of the shape are then unknown. Reparameterizations are of the form \(\mu(p, t) = (\alpha(p, t), t)\) and \(\nu(p, t) = (\beta(p, t), t)\), where \(\alpha, \beta : [0, 1] \times [0, 1] \rightarrow [0, 1]\) are continuous functions that form nondecreasing surjections for any fixed \(t\). Because we still assume the trajectory is represented as a quadrilateral mesh, the quadrilateral mesh defines how we interpolate between consecutive instances. This interpolation may not correspond to the matching used by this class and might therefore be of limited use in this specific setting. However, we believe this class may be useful in related settings. To accommodate these, a polynomial algorithm is presented.

**Asynchronous Identity**

\[ \mu(p, t) = (p, \sigma(t)) \quad \nu(p, t) = (p, \tau(t)) \]

This class is equivalent to the Synchronous Constant class after interchanging \(p\) and \(t\).

**Asynchronous Constant**

\[ \mu(p, t) = (\alpha(p), \sigma(t)) \quad \nu(p, t) = (\beta(p), \tau(t)) \]

The Asynchronous Constant class is an analogue of the Synchronous Constant class with the difference that the alignments of timestamps are not known in advance. Computing the Fréchet distance under this class turns out to be \(\text{NP-hard}\).

**Asynchronous Dynamic**

\[ \mu(p, t) = (\alpha(p, t), \sigma(t)) \quad \nu(p, t) = (\beta(p, t), \tau(t)) \]

The Asynchronous Dynamic class is an analogue of the Synchronous Dynamic class with the difference that the alignments of timestamps are not known in advance. Computing the Fréchet distance under this class turns out to be \(\text{NP-hard}\).

**Unrestricted Identity**

\[ \mu(p, t) = (p, \sigma(t)) \quad \nu(p, t) = (p, \tau(t)) \]

This class is equivalent to the Synchronous Dynamic class after interchanging \(p\) and \(t\).

**Unrestricted Constant**

\[ \mu(p, t) = (\alpha(p), \sigma(p, t)) \quad \nu(p, t) = (\beta(p), \tau(p, t)) \]

This class is equivalent to the Asynchronous Dynamic class after interchanging \(p\) and \(t\).

**Unrestricted Dynamic**

\[ \nu \circ \mu^{-1} \text{ is an orientation preserving homeomorphism.} \]

The Unrestricted Dynamic class can be used to compare quadrilateral meshes without looking enforcing any relation between the position and time parameters. The Fréchet distance under this class is useful for comparing surfaces, but turns out to be \(\text{NP-hard}\) to compute.
Table 1.1: Representative homeomorphisms and corresponding running times for various classes of reparameterizations of moving curves. We assume that the compared moving curves are quadrilateral meshes of size $p \times t$ where $p$ denotes position and $t$ denotes time.

<table>
<thead>
<tr>
<th>Position</th>
<th>Synchronous</th>
<th>Asynchronous</th>
<th>Unrestricted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>$O(pt)$</td>
<td>$O(t^2 \lambda_s(p) \log p \log t)^*,**$</td>
<td>$O(pt \log(p)t)^*$</td>
</tr>
<tr>
<td>Constant</td>
<td>$O(p^2 \lambda_s(t) \log p \log t)^{**}$</td>
<td>NP-hard</td>
<td>NP-hard$^*$</td>
</tr>
<tr>
<td>Dynamic</td>
<td>$O(p^4 t \log(pt))$</td>
<td>NP-hard</td>
<td>NP-hard</td>
</tr>
</tbody>
</table>

* Recall that moving curves are represented as quadrilateral meshes, which can be transposed to interchange parameters time $t$ and position $p$. This means that we can reuse the results in the transposed matrix after transposing the input shapes.

** The running time for the Synchronous Constant class depends on the underlying norm. For the Euclidean norm, we have $\lambda_2(n) = O(n)$, such that computation takes $O(p^2 t \log p \log t)$ time. More complex norms may require up to $O(p^2 t^2 \log p)$ time.
1.4 Results

We present algorithms for computing variants of the Fréchet distance between moving curves. Moving curves are the next step up from the existing variants of the Fréchet distance on moving points as defined in Section 1.2. These moving curves are represented as quadrilateral meshes as defined in Section 1.1. The variants we consider are the classes introduced in Section 1.3. In each class, \( \nu \circ \mu^{-1} \) will be restricted to orientation preserving homeomorphisms as in [AB10]. The matching \( \nu \circ \mu^{-1} \) may have further restrictions depending on the considered class of matchings. Table 1.1 gives a compact overview of our results.

We present polynomial time algorithms for computing the Fréchet distance for three classes of matchings between moving curves in Chapter 2. By symmetry, the Fréchet distance for two related classes can also be computed in polynomial time. The most restrictive class is that of Synchronous Identity matchings covered in Section 2.1. This class simply compares trajectories of two aligned moving curves of \( p \) line segments moving over \( t \) timestamps without realigning any points or timestamps. Their similarity can be computed in \( O(pt) \) time.

A slightly wider class is the Synchronous Constant class which allows positions along moving curves to be realigned, but does not allow realigning timestamps. In Section 2.1, we show how to compute an aggregate norm for this class. We also present methods to derive important properties for single line segments that move over \( t \) timestamps. The norm itself can be computed in \( O(t) \) time, and the important properties (called critical values) can be computed in \( O(t^2) \) or even \( O(t \log t) \) time for sufficiently simple underlying norms such as the Euclidean distance. The reason for deriving this aggregate norm is that we can use it directly in the original algorithms for the Fréchet distance. A chain of moving line segments can be used to form a moving curve, allowing the presented norm to be used directly in algorithms for existing variants of the Fréchet distance. Effectively, this extends existing algorithms from comparing curves to comparing moving curves.

Whereas the aggregate norm assumes that the alignments of positions along polylines are constant, the Synchronous Dynamic class covered in Section 2.3 allows alignments to change over time. An \( O(p^4t) \) time decision algorithm based on partitioning obstacles in a 3D space is presented, this leads to an \( O(p^4t \log(pt)) \) time algorithm to compute the exact Synchronous Dynamic Fréchet distance between two moving curves.

For more complex classes where the alignments of timestamps are not known in advance, we present reductions to prove the NP-hardness of approximating the Fréchet distance within a factor 1.5. Refer to Chapter 3 for these reductions. One of these reductions extends to known results on computing the Fréchet distance between triangular meshes. We think this construction is simpler than the existing constructions, and we believe this construction might lead to even stronger results.

Finally, in Chapter 4, we present visualisations of distance fields used in the comparison of quadrilateral meshes. The tool of Chapter 4 visualizes the internals of individual freespace-cells—the distances between points on two quadrilaterals—and has given fruitful insights in the behavior of those distance fields.
Chapter 2

Computation

We present algorithms for computing the Fréchet distance between moving curves under three classes of reparameterizations. The simplest class is the Synchronous Identity matching covered in Section 2.1. It consists of a single matching, namely the identity matching. If the moving curves form a quadrilateral mesh of $P \times T$ quadrilaterals, the Fréchet distance for their identity matching can be computed in $O(PT)$ time (assuming the distance between two single points can be computed in constant time). We also explain how to transform certain Asynchronous Constant matchings to Synchronous Identity matchings such that the Fréchet distance for these particular instances can be computed. Keep in mind that this computes the Fréchet distance for a predetermined Asynchronous Constant matching between quadrilateral meshes whereas finding an Asynchronous Constant matching minimizing the Fréchet distance is NP-hard as shown in Section 3.2.

A slightly wider class than the Synchronous Identity matching is the class of Asynchronous Identity matchings covered in Section 2.2. To compute the Fréchet distance under this class of matchings, we present methods to compute an aggregate convex norm that can be used directly in most algorithms computing the Fréchet distance between curves. This convex norm can be computed in $O(T)$ time when comparing moving curves of $P \times T$ quadrilaterals. This allows solving decision problems for the Synchronous Identity Fréchet distance in $O(P^2T)$ time. An exact computation also requires the computation of so called critical values. Critical values for the aggregate norm can be generally computed in $O(T^2)$ time, but some norms allow computation in $O(T \log T)$ time. The exact Synchronous Identity Fréchet distance can then generally be computed in $O(P^4T \log(PT))$ time. It is likely that a more efficient decision algorithm exists, and this could significantly speed up the exact computation. We think classes for moving curves wider than the Synchronous Dynamic class are either not meaningful, or do not admit polynomial algorithms. Hardness constructions for the remaining classes we regard as meaningful are given in Chapter 3.
2.1 Synchronous Identity

Suppose we know the alignments between all points of two shapes in advance, a common question is to find the distance between the two aligned points that are furthest apart. The Fréchet distance between the two shapes under the identity matching is then what we intend to compute. We call this distance the Synchronous Identity Fréchet distance and we show how to compute it for two moving curves (see Equation (2.1)). First we find the two aligned points that are furthest apart, computing the distance between those points is then trivial.

\[ d(A, B) = \sup_{p \in [0, P], t \in [0, T]} \| B(p, t) - A(p, t) \| \] (2.1)

For simplicity, assume that the two quadrilateral meshes \( A \) and \( B \) are moving curves both sampled at \( T + 1 \) timestamps with \( P + 1 \) vertices per timestamp. These vertices then form quadrilateral meshes of \( P \times T \) quadrilaterals for both \( A \) and \( B \). We denote the quadrilaterals on these meshes by \( A_{i,j} \) and \( B_{i,j} \) with \( i \in \{0, \ldots, P-1\} \) and \( j \in \{0, \ldots, T-1\} \) as illustrated in Figure 2.1.

\[ d(A_{i,j}, B_{i,j}) = \sup_{p \in [0, 1], t \in [0, 1]} \| B_{i,j}(p, t) - A_{i,j}(p, t) \| \] (2.3)

In the identity matching, each \( A_{i,j} \) is matched with \( B_{i,j} \) and we can therefore rewrite Equation (2.1) as Equation (2.2). That is, the Synchronous Identity Fréchet distance between \( A \) and \( B \) is determined by the pair of quadrilaterals \( A_{i,j} \) and \( B_{i,j} \) with the greatest Synchronous Identity Fréchet distance.

\[ d(A, B) = \sup_{i \in [0, \ldots, P-1], j \in [0, \ldots, T-1]} d(A_{i,j}, B_{i,j}) \] (2.2)

To compute the Synchronous Identity Fréchet distance between two quadrilaterals \( A_{i,j} \) and \( B_{i,j} \), we construct a new shape \( C_{i,j} \) such that \( C_{i,j}(p, t) = B_{i,j}(p, t) - A_{i,j}(p, t) \). It turns out that \( C_{i,j} \) is also a quadrilateral. Now we only need to find the point on \( C_{i,j} \) that is furthest from the origin because this point determines the Synchronous Identity Fréchet distance between \( A_{i,j} \) and \( B_{i,j} \) (see Equation (2.3)).

\[ d(A_{i,j}, B_{i,j}) = \sup_{p \in [0, 1], t \in [0, 1]} \| B_{i,j}(p, t) - A_{i,j}(p, t) \| = \sup_{p \in [0, 1], t \in [0, 1]} \| C_{i,j}(p, t) \| \] (2.3)
2.1. Synchronous Identity

Because all points on a quadrilateral lie within the convex hull of its four corner points, the point on $C_{i,j}$ with the greatest distance from the origin must be one of the corner points of $C_{i,j}$ for convex norms such as the $L^p$ norms. When computing the Synchronous Identity Fréchet distance between $A$ and $B$ it is therefore sufficient to check only the aligned vertices of $A$ and $B$ as in Equation (2.4). Therefore, the Synchronous Identity Fréchet distance can be computed using $O(PT)$ distance calculations.

$$d(A, B) = \sup_{p \in \{0, \ldots, P\}, t \in \{0, \ldots, T\}} \|B(p, t) - A(p, t)\| \tag{2.4}$$

**Non-equal dimensions** Now suppose that one of $A$ and $B$ has fewer than $T + 1$ timestamps or fewer than $P + 1$ positions, then we can introduce $O(PT)$ extra vertices on $A$ and $B$ to make them equal in size. Specifically, if quadrilateral mesh $A$ has a vertex at timestamp $t$ whereas $B$ does not, we can subdivide the row of $O(P)$ quadrilaterals on $B$ into two rows of $O(P)$ quadrilaterals, with vertices at the timestamp $t$. We are then left with a quadrilateral mesh with one less missing timestamp and $O(P)$ extra vertices. Repeat this process for all missing positions and timestamps of both $A$ and $B$. The resulting shapes then both have $O(PT)$ vertices. In fact, $A$ and $B$ will be quadrilateral meshes with equally many vertices, all of which are aligned.

**Non-identity alignments** We can reduce certain instances of Asynchronous Constant matchings to identity matchings. Consider the case where the matching of time is piecewise linear. An instance might map timestamps $[0, 1.5, 4]$ of $A$ to timestamps $[0, 2.5, 4]$ of $B$. The piecewise linear matching of timestamps between the two shapes can then be visualised by the solid black lines of Figure 2.2. Note that sampled timestamps might no longer map to sampled timestamps of the other quadrilateral mesh, as indicated by the dashed lines.

![Figure 2.2](image_url)

Figure 2.2: A piecewise linear matching of timestamps. (a) The matching drawn with the original time axis. (b) The matching drawn after warping the time axes.

We can reuse the procedure of introducing vertices at a certain timestamp to construct new quadrilateral meshes $A'$ and $B'$ that have the piecewise linear matching of time as their identity matching. For the example matching of Figure 2.2, the timestamps $t \in \{1, 2, 2.5, 3\}$ of $B$ no longer map to sampled timestamps of $A$, so to obtain $A'$ we introduce new vertices at the corresponding timestamps of $A$ as indicated by the dashed curves in Figure 2.3. Similarly, we obtain $B'$ by introducing new vertices at the necessary unsampled timestamps of $B$.

If the piecewise linear matching is defined by $j$ timestamps, then $A'$ and $B'$ will be sampled at at most $2(T + 1) + j = O(T + j)$ timestamps with respect to $A$ and $B$ which had at most $T + 1$ timestamps. Comparing $A'$ and $B'$ under the identity matching now gives the Fréchet distance of $A$ and $B$ under the desired matching of timestamps.
Similarly, positions can be realigned with piecewise linear matchings, yielding $A'$ and $B'$ both with a total of $O((P + k)(T + j))$ vertices if the realignment of positions is defined by $k$ positions. Then the quadrilateral meshes $A'$ and $B'$ can be compared under the Synchronous Identity Fréchet distance to find the distance between $A$ and $B$ under a specific realignment of timestamps and positions.

**Results** Assume the distance between two points can be computed in constant time. Then the Synchronous Identity Fréchet distance can be computed in $O(PT)$ time if the quadrilateral meshes have vertices on $O(T)$ timestamps and $O(P)$ positions. Moreover, the Fréchet distance for a fixed piecewise linear realignment of $j$ timestamps and $k$ positions can be computed in $O((P+k)(T+j))$ time.

### 2.2 Synchronous Constant

Recall that for the Synchronous Identity Fréchet distance, alignments between both timestamps and positions of the moving curves are known in advance. Therefore, only one matching had to be tested in the Synchronous Identity class. Suppose now that we know the alignment of timestamps between two moving curves in advance, but not the alignment of positions along those moving curves. Because the alignment between positions along the curves is not known in advance, the method of Section 2.1 cannot be used directly.

This problem gives rise to the Synchronous Constant class of matchings. Let reparameterizations $\alpha : [0,1] \to [0,1]$ and $\beta : [0,1] \to [0,1]$ constitute the matching between positions along the curves of $A$ and $B$. Then, the Synchronous Constant Fréchet distance as defined by Equation (2.5) is what we intend to compute. Note that we assume that the parameter space of the moving curves is $[0,1]^2$ in this equation. However, in subsequent equations we will often assume an isomorphic parameter space of the form $[0,P] \times [0,T]$. This will make it easier to reason about important parts, namely quadrilaterals, of the quadrilateral meshes separately.

$$d(A,B) = \inf_{\alpha,\beta : [0,1] \to [0,1]} \sup_{(p,t) \in [0,1]^2} \|B(\beta(p),t) - A(\alpha(p),t)\| \quad (2.5)$$

#### 2.2.1 Matching non-moving curves.

Before we show how to compute the Synchronous Constant Fréchet distance between moving curves, we first investigate an algorithm [AG95] used to compute the Fréchet distance for non-moving curves. The algorithm uses a parametric search on the decision problem, which determines...
whether a matching with distance at most $\varepsilon$ exists between curves $A$ and $B$. This decision problem can be solved in $O(P^2)$ time when curves $A$ and $B : [0, P] \rightarrow \mathbb{R}^n$ are polylines with $P$ segments.

The **freespace diagram** is a data structure used to solve the decision problem. This data structure represents the set of points on polylines $A$ and $B$ that are within distance $\varepsilon$ of each other, see Equation (2.6).

$$F_\varepsilon(A, B) = \{(a, b) \in [0, P] \times [0, P] \mid \|B(b) - A(a)\| \leq \varepsilon\} \quad (2.6)$$

For two curves of $p$ line segments and a given value of $\varepsilon$, this data structure can be used to determine in $O(p^2)$ time whether a matching with distance at most $\varepsilon$ exists. Freespace diagrams can be visualized as done in Figure 2.4b for the polylines of Figure 2.4a. The gray rectangle represents the product $\text{dom}(A) \times \text{dom}(B)$ of the parameter spaces of $A$ and $B$. The white areas in this rectangle is the freespace between $A$ and $B$, and this set consists of the points defined by Equation (2.6). A matching with distance at most $\varepsilon$ between polylines $A$ and $B$ exists iff there exists a path through the (white) freespace such that this path starts at $(0, 0)$ and ends at $(P, P)$. For the regular variant of the Fréchet distance, a requirement is that this path is monotone in both dimensions.

**Cells** of the freespace diagram are drawn separated by grid lines and represent pairs of line segments of $A \times B$. For convex norms, such cells contain a (possibly empty) convex freespace (white area), representing the set of points on the two line segments that are close to each other. Due to this convexity, the intersection of the freespace with one of the four sides bounding the cell is a single (possibly empty) interval, see Figure 2.5. We can decide whether a path through the freespace exists if we can compute these four intervals for each cell. The freespace is a function of $\varepsilon$, and the smallest value of $\varepsilon$ for which a path exists is the Fréchet distance. To avoid confusion, we denote by a **curve-freespace** the freespace arising from two polylines and a convex norm.
Chapter 2. Computation

2.2.2 Matching moving curves.

To decide whether two moving curves have a matching for a certain value of $\varepsilon$, we can extend the notion of freespace diagrams to higher dimensions. For simplicity, assume the quadrilateral meshes representing our moving curves are of equal size, consisting of $P \times T$ quadrilaterals. The resulting higher dimensional freespace defined by Equation (2.7) is then a subset of $\mathbb{R}^4$. This high dimension can be quite unwieldy, as is evidenced by the hardness proofs of Chapter 3.

For the Synchronous Constant matching, we know the alignments between timestamps of the two moving curves, which means that not the entire 4D freespace is used. Assume that the timestamps are matched using the identity matching and recall from Section 2.1 that any piecewise linear matching of timestamps can be transformed into an identity matching by subdividing the quadrilateral meshes. This identity alignment of timestamps allows us to simplify the freespace into three dimensions as defined in Equation (2.8).

$$
F^4_{\varepsilon}(A, B) = \left\{(a, b, s, t) \in [0, P]^2 \times [0, T]^2 \mid \|B(b, t) - A(a, s)\| \leq \varepsilon\right\} \quad (2.7)
$$

$$
F^3_{\varepsilon}(A, B) = \left\{(a, b, t) \in [0, P]^2 \times [0, T] \mid \|B(b, t) - A(a, t)\| \leq \varepsilon\right\} \quad (2.8)
$$

Whereas a matching between two curves is given by a path $\pi(p) = (\alpha(p), \beta(p))$ through the 2D freespace, a Synchronous Constant matching between moving curves is given by a surface $\pi(p, t) = (\alpha(p), \beta(p), t)$ through the 3D freespace. Note that $\alpha$ and $\beta$ are constant in the sense that they do not depend on $t$. Refer to Section 2.3 for the more general Synchronous Dynamic class where the matching between positions is allowed to change over time.

Analogous to the way cells in the curve-freespace diagrams of Section 2.2.1 represent the distances between the points on two line segments, the cells in the 3D freespace diagram correspond to the distances between points on two quadrilaterals. Note that because a moving curve at a certain timestamp is just a curve (polyline) of $P$ segments, the 3D freespace between moving curves gives a curve-freespace of $P \times P$ cells for a certain timestamp. We call this curve-freespace at a certain timestamp of the 3D freespace a snapshot.

Because positional matchings of the Synchronous Constant class are constant over time, we can further simplify the 3D freespace into a 2D freespace by taking its intersection over all snapshots as given by Equation (2.9). Call this 2D variant of the freespace the intersected freespace. Then, a Synchronous Constant matching with distance at most $\varepsilon$ exists iff there exists a monotone
path \( \pi \) from \((a, b) = (0, 0)\) to \((a, b) = (P, P)\) through the intersected \(\varepsilon\)-freespace.

\[
\mathcal{F}_{\varepsilon}^{2D}(A, B) = \bigcap_{t \in [0, T]} \{(a, b) \in [0, P]^2 | \left\| B(b, t) - A(a, t) \right\| \leq \varepsilon\} \tag{2.9}
\]

Recall that each cell of a snapshot is convex and that the intersected freespace is the intersection of snapshots. Then, because the intersection of convex sets is again convex, the cells of the intersected freespace are also convex. Moreover, this means that each side of a cell in the intersected freespace is intersected in a single (possibly empty) interval. We show how to compute these intervals; such that we can reuse the algorithm that decides whether a path through a curve-freespace exists to decide whether a path through the intersected freespace exists.

Observe that Equation \((2.9)\) requires us to compute the intersection of a possibly infinite number of snapshots. To overcome this problem, we show that for convex norms, an intersection of only \(O(T)\) snapshots yields the same intersected freespace. Specifically, we show that Equation \((2.10)\) where \(t\) only takes integer values is equivalent to Equation \((2.9)\) where \(t\) had a continuous range.

\[
\mathcal{F}_{\varepsilon}^{2D}(A, B) = \bigcap_{t \in \{0, 1, \ldots, T\}} \{(a, b) \in [0, P]^2 | \left\| B(b, t) - A(a, t) \right\| \leq \varepsilon\} \tag{2.10}
\]

**Quadrilaterals** Consider the case where both moving curves \(A\) and \(B\) are quadrilaterals instead of quadrilateral meshes; then their 3D freespace diagram consists of a single cell as illustrated in Figure 2.6. The dashed lines on the shown quadrilaterals display how a point on the moving curve moves over time. Assuming the two dashed lines are matched with each other, the solid line indicated in the freespace diagram must lie entirely in the freespace.

We prove that if two points \((a, b, 0)\) and \((a, b, 1)\) lie in the freespace between two quadrilaterals, then all points \((a, b, t)\) with \(t \in [0, 1]\) also lie in the freespace. This follows from Lemma 2.2.1 which shows that the 3D freespace between two quadrilaterals intersected with a line parallel to the \(t\)-axis is convex. It then follows that the intersected freespace between two quadrilaterals \(A\) and \(B\) can be simplified from Equation \((2.11)\) to Equation \((2.12)\). This means that the intersected freespace between two quadrilaterals can be computed as the intersection of two snapshots, rather than an infinite number of snapshots. We shall extend this idea from quadrilaterals to quadrilateral meshes.

![Figure 2.6: On the left, two quadrilaterals \(A\) and \(B\). On the right, a 3D freespace cell resulting from \(A\) and \(B\). The actual freespace is not shown, but is visualized for other quadrilaterals in Chapter 4.](image)
Lemma 2.2.1. For convex norms, the 3D freespace between two quadrilaterals $A$ and $B$ intersected with a line parallel to its $t$-axis is convex.

Proof. Consider $f(t) = A(a, t)$ and $g(t) = B(b, t)$, which represent the movement of single positions $(a$ and $b)$ on moving curves $A$ and $B$, respectively. Then, because $A$ and $B$ are quadrilaterals, $f$ and $g$ are affine transformations and their difference $g(t) - f(t)$ is again affine. Because the preimage of a convex norm ball under an affine map $(g(t) - f(t))$ is convex, the intersection of the freespace (between two quadrilaterals) with a line parallel to its $t$-axis forms a convex set. □

\[
quad F_{2D}^3(A, B) = \bigcap_{t \in [0,1]} \{(a, b) \in [0,1]^2 | \|B(b,t) - A(a,t)\| \leq \varepsilon\} \tag{2.11}
\]

\[
= \bigcap_{t \in [0,1]} \{(a, b) \in [0,1]^2 | \|B(b,t) - A(a,t)\| \leq \varepsilon\} \tag{2.12}
\]

Quadrilateral meshes Now consider two quadrilateral meshes $A$ and $B$, both with $P \times T$ quadrilaterals. As before, we denote their quadrilaterals by $A_p,t$ and $B_p,t$ with $p \in \{0,\ldots,P-1\}$ and $t \in \{0,\ldots,T-1\}$. The 3D freespace between the two quadrilateral meshes then consists of $P \times P \times T$ cells. Each (3D) cell then contains the freespace between two quadrilaterals and has the properties of Equation (2.12).

We use these properties to simplify the 3D freespace to the (2D) intersected freespace as follows. Let cell $(i,j,t)$ contain the freespace ($\subseteq [0,1]^3$) between the quadrilaterals $A_{i,t}$ and $B_{i,t}$. Its 2D intersected freespace ($\subseteq [0,1]^2$) can be computed using $quad F_{2D}^3(A_{i,t}, B_{i,t})$. For the quadrilateral meshes, cell $(i,j)$ of the 2D intersected freespace ($F_{2D}^3(A, B)$) is the intersection of the intersected freespace $quad F_{2D}^3(A_{i,t}, B_{j,t})$ of the $T$ pairs of quadrilaterals $(A_{i,t}, B_{j,t})$ with $t \in \{0,\ldots,T\}$. This means that we can compute each of the $P \times P$ cells of the 2D intersected freespace between $A$ and $B$ using $O(T)$ intersections, see Equation (2.13). In this equation, the $\{(i,j)\}$+-term is used to translate each cell $(i,j)$ of the 2D intersected freespace into place, namely from $[0,1] \times [0,1]$ to $[i,i+1] \times [j,j+1]$. We should state that Equation (2.13) simplifies further to the more concise Equation (2.10) which was mentioned before.

\[
F_{2D}^3(A, B) = \bigcup_{i,j \in \{0,\ldots,P-1\}} \left\{(i,j)\right\} + \bigcap_{t \in \{0,\ldots,T-1\}} quad F_{2D}^3(A_{i,t}, B_{j,t}) \tag{2.13}
\]

What follows is that each of the $P \times P$ cells in the intersected freespace intersect the four sides of a cell in a single (possibly empty) interval. This interval can be computed in $O(T)$ time; namely by taking the intersection of the $O(T)$ intervals resulting from the $O(T)$ pairs of corresponding quadrilaterals.

To give an idea of what this intersected freespace might look like, an intersected freespace diagram is shown in Figure 2.7. The white parts form the freespace between the complete meshes, the light gray parts were in the freespace of some (but not all) snapshots, and the darkest parts were in the freespace of none of the snapshots. The key observation is that the white freespace is still convex for each cell, and it therefore only intersects a side of a cell in a single interval. Moreover, for a given $\varepsilon$, each of the $O(P^2)$ intervals intersecting the side of a cell can be computed in $O(T)$ time.

Since each of these intervals can be computed in $O(T)$ time, and the freespace within each cell in convex, the existing algorithm for the decision problem between curves can be used to
solve the decision problem between quadrilateral meshes for a given value of $\varepsilon$ in $O(P^2T)$ time. As such, a binary search can be used to approximate the Synchronous Constant Fréchet distance. However, we can also compute it exactly in $O(P^2T^2 \log P)$ time using a parametric search.

![Figure 2.7: In white, $F^2_{\varepsilon}(A, B)$.](image)

**Parametric search** When computing the Fréchet distance between two (non-moving) curves of $P$ segments, there are $O(P^3)$ candidate values for $\varepsilon$, such that the actual Fréchet distance is one of these candidates. These candidates are called critical values and correspond to the values of $\varepsilon$ for which new paths might ‘appear’ in the directed topology given by the $\varepsilon$-freespace.

To make this intuitive, assume we gradually increase $\varepsilon$, starting from 0; then the freespace will continuously grow. A path from $(0,0)$ to $(P,P)$ might appear in three cases, namely at the minimal value of $\varepsilon$ for which:

(a) $(0,0) \in F_\varepsilon(A, B)$ or $(P,P) \in F_\varepsilon(A, B)$; or

(b) an interval of $F_\varepsilon(A, B)$ on the side of some cell $(i,j)$ becomes non-empty; or

(c) in $F_\varepsilon(A, B)$, the lower endpoint of an interval of cell $(i,j)$ aligns with the upper endpoint of an interval of some cell $(i,k)$ or $(k,j)$.

Then there are two critical values of type (a), $O(P^2)$ critical values of type (b), and $O(P^3)$ critical values of type (c). Suppose the values of $\varepsilon$ for which any of these events occur can be computed in $time_{\text{crit}}$ time and the decision problem as described before can be computed in $time_{\text{dec}} = O(P^2)$ time. A parametric search on the $O(P^2)$ critical values of type (a) and (b) to find a small interval in which the actual Fréchet distance lies. Within that interval a second parametric search on $O(P^2)$ critical values of type (c) can be used to find the exact Fréchet distance.

The two parametric searches take $O((P^2 time_{\text{crit}} + time_{\text{dec}}) \log P)$ time and yield the minimal value of $\varepsilon$ for which a matching exists through the freespace. Assuming $time_{\text{crit}} = O(1)$ and $time_{\text{dec}} = O(P^2)$, the Fréchet distance between two curves can therefore be computed in $O(P^2 \log P)$ time.

We can extend this method to compute the Synchronous Constant Fréchet distance between two quadrilateral meshes in $O(P^2T^2 \log P)$ time. Recall that the intersected freespace ($F^2_{\varepsilon}(A, B)$) has a structure similar to the freespace between two curves in that the cells of the intersected freespace are convex. This means that if we can also compute each of the critical values described
above for quadrilateral meshes, we can apply the same parametric search as the one for curves. For this, we show that each of the critical values can be computed in $O(T^2)$ time.

(a) First consider critical values of type (a), these are equal to $\sup_{t \in \{0, \ldots, T\}} \| B(b, t) - A(a, t) \|$ with $(a, b) = (0, 0)$ or $(a, b) = (P, P)$, depending on which of the two critical values we need to compute. Therefore, critical values of type (a) can be computed in $O(T)$ time.

(b) Now consider critical values of type (b), where an interval of $F^2_\varepsilon (A, B)$ on a side $l$ of some cell $(i, j)$ becomes nonempty. This interval is the intersection of the intervals on side $l$ of $T + 1$ snapshots of the 3D freespace. Such a critical value occurs when either the interval on some snapshot becomes nonempty, or the lower endpoint of the interval of some snapshot aligns with the upper endpoint of an other snapshot. Each of the $O(T)$ values of $\varepsilon$ where single snapshots become nonempty can be computed in the same way critical values of type (b) between curves were computed. Computing the maximum of these $O(T)$ values for a single interval then takes $O(T)$ time.

The case where the lower endpoint and upper endpoint of the intervals of two snapshots align is harder to compute. The lower endpoint we are interested is the lower endpoint of the interval on side $l$ for one of the $T + 1$ snapshots. Similarly, the upper endpoint we are interested is the upper endpoint on $l$ for one of the $T + 1$ snapshots. The critical value for two such intervals can be computed using the method for computing critical values of type (c) between curves, and is assumed to be a constant time operation. A naive algorithm computes all the $O(T^2)$ values of $\varepsilon$ for which endpoints of two such endpoints align.

The critical value for side $l$ of the intersected freespace can then be computed in $O(T^2)$ time by taking the maximum over the $O(T + T^2)$ computed values.

(c) Critical values of type (c) can be computed in a similar manner. We must find the value of $\varepsilon$ for which the lower endpoint of an interval of some cell aligns with the upper endpoint of an interval of some other cell. This is the same problem as before, with the exception that the $T + 1$ intervals defining the lower endpoint may lie on a different cell boundary than the $T + 1$ intervals defining the upper endpoint. The previous method did not rely on the assumption that endpoints were part of the same cell boundary, so the method can be reused. Thus, the critical values of type (c) can also be computed in $O(T^2)$ time by taking the maximum over the critical values of the $O(T^2)$ pairs of snapshots.

As a result, the time used to compute a critical value is $time_{crit} = O(T^2)$. Therefore, the exact Synchronous Constant Fréchet distance can be computed in $O((P^2 time_{crit} + time_{dec}) \log P) = O((P^2 T^2 + P^2) \log P) = O(P^2 T^2 \log P)$ time. It turns out that for some norms, critical values can even be computed in $O(T \log T)$ time, allowing us to compute the exact Synchronous Constant Fréchet distance in $O(P^2 T \log P \log T)$ time.

**Efficient computation of critical values** Observe that the limiting factor in the running time of $O((P^2 time_{crit} + time_{dec}) \log P) = O(P^2 T^2 \log P)$ is $time_{crit} = O(T^2)$. Because critical values of type (a) can be computed in $O(T)$ time, we are looking to reduce the $O(T^2)$ time used to compute critical values of type (b) and (c). A more efficient way to compute these critical values exists for many norms.

The intervals on the side of a cell of a snapshot are given by Equation (2.14) or Equation (2.15), depending on whether we are trying to compute a horizontal or vertical interval. For two quadrilaterals (which define the freespace cell), the terms $B(b, t) - A(a+p, t)$ and $B(b+p, t) - A(a, t)$ form affine interpolations with parameter $p$. Therefore we consider intervals of the general form of Equation (2.16), where $C_{a,b,t}(p)$ is an affine interpolation that can be substituted...
by $B(b, t) - A(a + p, t)$ or $B(b + p, t) - A(a, t)$ to find the original intervals. The respective lower and upper endpoints of these intervals are given by Equations (2.17) and (2.18), yielding $\infty$ and $-\infty$ if the interval is nonexistent for a given $\varepsilon$.

\begin{align*}
I_{a,b,t}^1(\varepsilon) &= \{ p \in [0, 1] \mid \| B(b, t) - A(a + p, t)\| \leq \varepsilon \} \\
I_{a,b,t}^2(\varepsilon) &= \{ p \in [0, 1] \mid \| B(b + p, t) - A(a, t)\| \leq \varepsilon \} \\
I_{a,b,t}(\varepsilon) &= \{ p \in [0, 1] \mid \| C_{a,b,t}(p)\| \leq \varepsilon \} \\
L_{a,b,t}(\varepsilon) &= \inf \{ p \in [0, 1] \mid \| C_{a,b,t}(p)\| \leq \varepsilon \} \\
U_{a,b,t}(\varepsilon) &= \sup \{ p \in [0, 1] \mid \| C_{a,b,t}(p)\| \leq \varepsilon \} \\
\inf \{ \varepsilon \mid \bigcap_{t \in [0, \ldots, T]} I_{a,b,t}(\varepsilon) \neq \emptyset \} &=\inf \{ \varepsilon \mid \max_{t \in [0, \ldots, T]} L_{a,b,t}(\varepsilon) = \min_{t \in [0, \ldots, T]} U_{a,b,t}(\varepsilon) \} \\
\text{(2.14)} & \quad \text{(2.15)} & \quad \text{(2.16)} & \quad \text{(2.17)} & \quad \text{(2.18)} & \quad \text{(2.19)} & \quad \text{(2.20)}
\end{align*}

For critical values of type (b), we need to compute the value of $\varepsilon$ for which the intersection of $O(T)$ such intervals becomes nonempty, see Equation (2.19) or equivalently Equation (2.20). The critical values of type (c) follow after a minor modification to Equation (2.20), namely by replacing $U_{a,b,t}$ with $U_{a+k,b,t}$ or $U_{a,b+k,t}$. The problem of finding these values of $\varepsilon$ can now be solved by finding the minimal point $(p, \varepsilon)$ with $p \in [0, 1]$ on the upper envelope of $O(T)$ convex functions, namely the inverses of $L_{a,b,t}$ and $U_{a,b,t}$.

An illustration for the upper envelope when computing critical values of type (b) is given in Figure 2.8a. The red lines correspond to $L_{a,b,t}$ and the blue lines correspond to $U_{a,b,t}$. Note that we have also encoded the interval $[0, 1]$ with a pair of red and blue lines. The white area that lies above all functions is called the upper envelope, and its intersection with a horizontal line at height $\varepsilon$ gives the interval that is open on the side of a cell in the intersected freespace. Thus, the value of $\varepsilon$ for the lowest point on the envelope is the critical value for a side of cell $(a, b)$.

Similarly, Figure 2.8b shows an upper envelope of $O(T)$ functions, but this time for critical values of type (c). The only difference is that the blue lines correspond to $U_{a+k,b,t}$ or $U_{a,b+k,t}$ instead of $U_{a,b,t}$. This means that the blue and red lines arising from a single snapshot do not necessarily align as was the case for critical values of type (b).

For the squared Euclidean distance, these functions are parabolas; and Buchin et al. [BBVL+13] show how to query the minimal point on a dynamic upper envelope can be computed in $O(T \log^2 T)$ time. Our problem is slightly simpler because our upper envelopes are not dynamic, allowing a slightly faster computation.

Recall that the algorithm for computing the Fréchet distance between two curves assumes that the minimum of such these convex functions (parabolas in the squared Euclidean case) can be computed in constant time (to find critical values of type (b)). Moreover, it assumes that intersections between two such functions can also be computed in constant time (to determine critical values of type (c)). We make the same assumptions, and additionally assume that two such functions intersect at most $s$ times.

Then the complexity (number of segments) of the upper envelope of $T$ such functions is bounded by $\lambda_s(T)$, where $\lambda_s(T)$ is the maximal length of a Davenport-Schinzel sequence of order $s$ for $T$ functions. The upper envelope can then be constructed in $O(\lambda_s(T) \log T)$ time [SA95]. The $\lambda_s(T)$ function is known to grow almost linearly\footnote{$\lambda_s(T) = O\left(2^{s(T)} \log \alpha(T)T\right)$ where $\alpha$ is the slow growing inverse Ackermann function.} in $T$. The minimum point on this upper envelope can be found after spending $O(\lambda_s(T) \log T)$ time to construct the envelope, and using a
Chapter 2. Computation

Figure 2.8

(a) A critical value of type (b).

(b) A critical value of type (c).

linear scan of $O(\lambda_s(T))$ time through this envelope to find the minimum, which is the critical value.

Because our upper envelope is defined by $O(T)$ functions, each critical value of type (b) and (c) can be computed in $\text{time}_{\text{crit}} = O(\lambda_s(T) \log T)$ time, assuming the functions intersect each other at most $s$ times. Hence, we can compute the exact Synchronous Constant Fréchet distance in $O(P^2\lambda_s(T) \log P \log T)$ time.

For the squared Euclidean distance covered in $\text{BBVL}^{13}$, each of the functions $L_{a,b,t}$ and $U_{a,b,t}$ are truncated parabolas and intersect at most twice, so $s = 2$ for the (squared) Euclidean norm, which yields $\lambda_2(T) = 2T - 1 = O(T)$. Thus, for the Euclidean norm, critical values can be computed in $\text{time}_{\text{crit}} = O(T \log T)$ time. This allows us to compute the exact Synchronous Constant Fréchet distance under the Euclidean norm in $O(P^2 \log P \log T)$ time.

For general $L^p$ norms, where $p$ is a positive even number, the intervals are defined by convex polynomials of degree $p$. Two such polynomials then intersect at most $s = p$ times, but these intersections (polynomial roots) are often harder to compute. However, if we can compute them in constant time, we can compute the critical values in $O(\lambda_p(T) \log T)$ time.

Results We have provided a way to compute critical values and intervals on freespace-cell boundaries for an aggregate convex norm, defined on quadrilateral meshes of size $1 \times T$. These properties can be used directly in the original algorithm for computing the Fréchet distance between curves. An intuitive way of viewing this approach is that instead of sequences of line $P$ segments, curves are now sequences of $P$ quadrilateral meshes of size $1 \times T$, which gives a quadrilateral mesh of size $P \times T$.

The intersection of freespace-cell boundaries under this aggregate norm can be computed in $O(T)$ time. Moreover, critical values of type (a) can be computed in $O(T)$ time, and critical values of type (b) and (c) can be computed in $O(\lambda_s(T) \log T)$ time, which is $O(T \log T)$ time for the Euclidean norm. This brings the running time for computing the Synchronous Constant Fréchet distance between quadrilateral meshes to $O(P^2 T^2 \log P)$ and in some cases even $O(P^2 T \log P \log T)$.

The aggregate norm can then be used in most existing algorithms that compute variants of the
2.2. Synchronous Constant

Fréchet distance between curves. Most of these algorithms require only that the freespace within a cell is convex, and that critical values of type (a), (b) and (c) can be computed. Effectively, this extends these variants on the Fréchet distance between curves to variants on the Synchronous Constant Fréchet distance between quadrilateral meshes. Example variants include the Weak Fréchet distance and the Fréchet distance between moving polygon boundaries.

We should make a few remarks regarding quadrilateral meshes of different size. To handle the case where meshes have some piecewise linear matching of timestamps, the input meshes can be transformed using the methods explained in Section 2.1. Moreover, when the meshes are of different size, say $P \times T$ and $Q \times T$, the original algorithm computes their Synchronous Constant Fréchet distance in $O(PQ\lambda_s(T) \log(PQ) \log T)$ time without any modifications. This means that when $Q$ is much smaller than $P$, we do not require that the meshes are transformed to be of equal size, because that might slow down computations.
2.3 Synchronous Dynamic

The widest class of matchings for which we think a polynomial algorithm exists is that of Synchronous Dynamic matchings. As opposed to the Synchronous Constant class where the matchings between positions stayed the same as time progressed, the matchings between positions in the Synchronous Dynamic class may change over time, with the restriction that this change is continuous. See Equation (2.21), where \( \alpha(p,t) \) and \( \beta(p,t) : [0,1]^2 \rightarrow [0,1] \) are continuous non-decreasing surjections for any fixed \( t \), and continuous functions for any fixed \( p \). Figure 2.9 may give an intuition of what these matchings look like in the 3D freespace. An other way of looking at this is that we need to find the minimal value of \( \varepsilon \) for which such a surface through the 3D freespace exists. We can decide whether such a surface exists for a given \( \varepsilon \) using the algorithm of Section 2.3.1.

\[
d(A,B) = \inf_{\alpha,\beta : [0,1]^2 \rightarrow [0,1]} \sup_{(p,t) \in [0,1]^2} \| B(\beta(p,t), t) - A(\alpha(p,t), t) \|
\]  

(2.21)

![Figure 2.9: Four different projections of the same Synchronous Dynamic matching.](image)

2.3.1 Decision Problem

Consider Figure 2.10 which shows snapshots of the 3D freespace between two quadrilateral meshes of size \( 4 \times 1 \). Assume there exists a Synchronous Dynamic matching between those meshes such that at \( t = 0 \) and \( t = 1 \), the positional matchings are given by the blue and red paths from \( (0,0) \) to \( (4,4) \), respectively. For these paths to be part of the same Synchronous Dynamic matching, there must exist some continuous surface through the 3D freespace between the two
snapshots, such that this surface connects the blue matching with the red matching. In any snapshot with $t$ between 0 and 1, this surface will look like a monotone path from $(0,0)$ to $(4,4)$, and must lie in the freespace.

![Figure 2.10](image)

Figure 2.10: (a)(b) In red and blue, matchings in two snapshots of a 3D freespace. (c) A surface between the two snapshots connecting the two matchings.

The decision algorithm for the Fréchet distance between non-moving curves transforms the 2D freespace to a structure called the reachable space, defined in Equation (2.22), such that a matching exists iff $(P,P) \in R$. One might think that a similar transformation can be performed on the 3D freespace for moving curves. First we formally define the analogue of monotone paths that applies for the Synchronous Dynamic Fréchet distance, namely Synchronous Dynamic surfaces as defined in Equation (2.23). Here, $\alpha$ and $\beta$ are continuous functions where $\alpha(p,t)$ and $\beta(p,t)$ are nondecreasing surjections for any fixed value of $t$. Then the analogue of the reachability $R$ is $S$ as defined in Equation (2.24), such that a matching exists iff $(P,P,T) \in S$. For non-moving curves, $R$ can easily be computed because if some point is known to lie in $S$, other points that lie in $S$ directly follow from the information captured in the freespace diagram. For moving curves, knowing that a certain point lies in $S$ does not always capture enough information to derive that other points lie in $S$ as well. Instead, knowledge about the surfaces leading to points in $S$ is required.

\[
R = \{(a,b) \mid \text{a monotone path from } (0,0) \text{ to } (a,b) \text{ lies in } F(A,B)\} \tag{2.22}
\]

\[
S = \{\{(\alpha(p,t),\beta(p,t),t \cdot T) \mid (p,t) \in [0,1]^2\} \mid \alpha,\beta : [0,1]^2 \to [0,P]\} \tag{2.23}
\]

\[
S = \{(a,b,t) \in S \mid ([0,a] \times [0,b] \times [0,t]) \cap S \subseteq F \text{ for some surface } S \in S\} \tag{2.24}
\]

To avoid the need to explicitly maintain information about such surfaces, we instead consider a more abstract problem. The 3D freespace can be interpreted as a cube with obstacles inside it, such that some Synchronous Dynamic surface avoids all obstacles. Any such surface effectively partitions the obstacles into two sets, namely those obstacles to the left of the surface and those to the right of it. Some of these obstacles are related to each other in the sense that if one lies to the left of such a surface, the other must lie to the left of that surface as well. We define this relation on obstacles in the freespace as $R$, such that $(x,y) \in R$ means that for any surface $S$, if obstacle $x$ lies to the right of $S$, then obstacle $y$ also lies to the right of $S$. This relation is reflexive and transitive, and therefore forms a preorder.
We can view each element of \( R \) as an edge in a graph whose vertices are obstacles as in Figure 2.11 which depicts a few elements of \( R \) as directed edges between obstacles in a 2D freespace. For cleanliness, not all elements that follow from transitivity have been drawn. Note that we have introduced two dummy obstacles, \( l \) and \( r \), representing the upper left and lower-right boundary of the freespace. The decision problem of whether a matching exists through the freespace is then equivalent to deciding whether a matching exists, such that \( l \) and \( r \) lie on opposite sides. Formally, we have that \((l,r) \in R\) and a matching through the freespace exists if \((r,l) \notin R\). The blue path in Figure 2.11 is a witness that \((r,l) \in R\) (due to transitivity), which proves the absence of a monotone path between \((0,0)\) and \((P,P)\).

\[\text{Figure 2.11: Four objects (and two dummy objects } l \text{ and } r\text{) in a fictional 2D freespace. An arrow from obstacle } x \text{ to } y \text{ indicates that if a monotone path goes over } x, \text{ it must also go over } y.\]

We have seen that for a subset of \( R \) whose transitive closure is \( R \), deciding whether \((r,l) \in R\) is equivalent to finding a path from \( r \) to \( l \) in this subset. If such a subset consists of \( n \) edges of \( R \), a graph traversal can then decide in \( O(n) \) time whether the obstacles can be partitioned into two sets, such that \( r \) and \( l \) are in different sets. This means that a matching in the freespace exists, separating the two sets, thereby solving the decision problem. The aim is now to find a small subset \( R' \subseteq R \), whose transitive closure is \( R \), such that the decision problem can be solved in \( O(\left| R' \right|) \) time.

**Derivation of \( R' \)** We consider the problem of finding a small set \( R' \) representing the structure of the 3D freespace between two quadrilateral meshes. For this, we need to define what our obstacles are. We say that an obstacle is a maximal connected set of points within a single cell of the freespace, with the points that lie in the freespace removed. For the 3D freespace, the number of obstacles is bounded by \( O(P^2T) \), namely because the number of obstacles within a single cell of the freespace is bounded by a constant due to its partial convexity.

If the boundary between two neighboring cells connects two obstacles \( x \) and \( y \), we can draw a bidirectional edge (\((x,y) \in R' \wedge (y,x) \in R'\)) between those obstacles because no surface is able to pass through them. We remark that in \( R' \) no edges between cells of disjoint timestamps \((t \text{ and } t + 2 + k)\) are required because all possible interactions will be captured by transitivity through intermediate cells of \((t + 1 \ldots t + 1 + k)\). Furthermore, if the top of obstacle \( x \) lies above and to the left of the bottom of a different obstacle \( y \) at the same timestamp, we can...
2.3. Synchronous Dynamic

require \((x, y) \in R'\) due to monotonicity of the matching. Analogously, if the left side of obstacle \(x\) lies above and to the left of the right side of a different obstacle \(y\) at the same timestamp, we can require \((x, y) \in R'\). Doing this exhaustively results in \(|R'| = O(P^4T)\) edges and we can then solve the decision problem in \(O(P^4T)\) time.

We note that this large number of edges is unnecessary in \(R'\) because some edges of \(R\) can be induced by transitivity. Each obstacle needs only \(O(P)\) incoming and outgoing edges in \(R'\). To see this, assume there exist edges \((x, y) \in R'\) and \((y, z) \in R'\), then edge \((x, z)\) is not necessary in \(R'\). We can conclude that obstacle \(x\) has edges to \(n\) other obstacles without edges between them in \(R\). These \(n\) obstacles must then lie to the top-right (or bottom-left) of each other, and this can only happen for \(n = O(P)\) obstacles due to the size of the freespace. So \(R'\) needs at most \(O(P^3T)\) edges and perhaps even fewer.

We therefore think it is possible to reduce the number of edges in \(R'\) by a significant amount, making use of the transitive closure induced by \(R'\). Given such a small set \(R'\), the decision problem can be solved in \(O(|R'|)\) time. This leads us to conjecture that \(R'\) can be computed in \(O(P^3T)\) time, such that the decision problem for the Synchronous Dynamic Fréchet distance can be solved in \(O(P^3T)\) time.

2.3.2 Parametric search

The parametric search to calculate the exact Synchronous Dynamic Fréchet distance involves similar critical values as those used in the original algorithm for the Fréchet distance between non-moving curves\[^{AG95}\]. Consider Figure 2.12, which labels the intervals relevant for a cell \((i, j, k)\) in the 3D freespace. Critical values of type (a), (b) and (c) cover the critical values for intervals \(a_{i,j,k}\) and \(b_{i,j,k}\).

\[
\begin{align*}
(a) \quad & t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \\
(b) \quad & t_{i+1,j,k} + 1, \quad t_{i+1,j,k} + 1, \quad t_{i+1,j,k} + 1, \quad t_{i+1,j,k} + 1, \\
(c) \quad & t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \\
(d) \quad & t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \\
(e) \quad & t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1, \quad t_{i,j,k} + 1,
\end{align*}
\]

Figure 2.12: Intervals on the border of cell \((i, j, k)\) of a 3D freespace.

However, there are two additional types, (d) and (e), of critical values that arise from passages that open when obstacles stop overlapping in the \(t\)-axis. Due to the partial convexity of freespace cells (Lemma 2.2.1), the remaining critical values can be derived from the intersection of the freespace with the cell boundaries. We get the following two types of critical values:

(d) \ interval \(t_{i,j,k}\) becomes nonempty; or

(e) \ the lower endpoint of interval \(t_{i,j,k}\) aligns with the upper endpoint of \(t_{l,m,k}\).

In total, we have \(O(T)\) critical values of type (a), \(O(P^2T)\) critical values of type (b) and (d), \(O(P^3T)\) critical values of type (c) and \(O(P^4T)\) critical values of type (e). To il-
lustrate why critical values of type (e) are necessary, consider Figure 2.13 where a saddle point of the 3D freespace passes through the boundary of a cell. In the left image, the lower endpoint of one interval lies just above the upper endpoint of another interval. In the right image (which has a slightly greater value of $\varepsilon$), the two intervals overlap and allow a surface to pass through the cell.

Assuming each of the $O(P^4T)$ critical value can be computed in constant time, a naive parametric search takes $O((P^4T\text{time}_{\text{crit}} + \text{time}_{\text{dec}}) \log(PT)) = O(P^4T \log(PT))$ time to compute the exact Synchronous Dynamic Fréchet distance between two quadrilateral meshes. This parametric search can be optimized further by first considering critical values of types (a), (b), (c) and (d). This leaves a relatively small number of critical values of type (e) to consider. However, the running time is currently majorized by the time $\text{time}_{\text{dec}} = O(P^4T)$ used to solve the decision problem.

2.3.3 Alternative exact computation

In the decision problem, the relation $R$ is defined by a fixed value of $\varepsilon$, name this relation $R_\varepsilon$. We can bypass a parametric search using the observation that $R_\varepsilon$ gets smaller as $\varepsilon$ grows. This means that we can transform relation to a weighted graph, where the weight of an edge is the maximal value of $\varepsilon$ for which the edge lies in $R_\varepsilon$. Then the exact Synchronous Dynamic Fréchet distance is the weight of the minimum-weight edge on the path from $r$ to $l$ maximizing this weight. This problem is also known as the widest path problem and can be solved in $O(\min(v \log v + e, e \log^* v))$ time [G18S] where $v$ and $e$ are the number of vertices and edges in the graph, respectively.

Again, the graph need not be complete, because we do not need all edges in the transitive closure. On the other hand, due to the large amount of critical values, it seems unlikely that this graph can be captured in a small number of edges. The widest path problem, also known as the bottleneck shortest path problem, is known to arise in other variants [HPRT14] of the Fréchet distance as well. It makes sense to first investigate this approach for the simpler problem of computing the Fréchet distance between non-moving curves, possibly allowing a computation in $O(P^2 \log^* P)$ time. This follows if the weighted graph for the original Fréchet distance requires only $O(P^2)$ edges, that can be computed in $O(P^2)$ time.
Chapter 3

NP-Hardness

We present NP-hardness constructions for the classes of reparameterizations where neither the matching of timestamps, nor the matching of positions is known in advance. These are the cases where the matchings between the two curves cannot be captured as surfaces through three-dimensional freespace, and instead require four dimensions.

We start with the simplest NP-hard case, namely that of Asynchronous Constant matchings (see Section 3.2) where the matchings of time, as well as position are constant. This construction is then extended in Section 3.3 to handle the case of Asynchronous Dynamic matchings where the positional matching may change over time. These two cases turn out to be NP-hard even if all vertices lie on a straight line (i.e. in $\mathbb{R}^1$).

Finally, we show how to extend the construction of Section 3.3 further to prove NP-hardness for the Unrestricted Dynamic class in Section 3.4. Here, matchings between quadrilateral meshes are unrestricted in both the temporal and positional dimension. In contrast to the previous constructions in $\mathbb{R}^1$, the construction for this class uses moving curves embedded in $\mathbb{R}^2$. Specifically, the vertices are positioned on three parallel lines in the plane instead of on a single line. Although the Asynchronous Constant and Asynchronous Dynamic class are NP-hard even if all vertices lie on a single line, it is unknown whether this is also true for the Unrestricted Dynamic class.

Because all of our quadrilaterals lie in the plane, this result extends to a known result on the Fréchet distance between triangulated surfaces. Although several NP-hardness constructions [God99, BBS10] for triangulated surfaces are known, we give an arguably simpler reduction.

3.1 Example

We show that computing the Fréchet distance between moving curves for the given classes of matchings is NP-hard using reductions from 3-SAT to the decision problem of the Fréchet distance of various reparameterization classes. Before we give any formal proofs, we illustrate the idea behind the constructions with an example.

Consider the moving curves $A$ and $B$, shown in Figures 3.1 and 3.2. In all of the considered reparameterization classes, they will have a small Fréchet distance if the disjunction $(X \lor \neg Y \lor \neg Z)$ can be satisfied. Suppose this disjunction is part of a 3CNF-formula with variables $W, X, Y, Z$. 
Figure 3.1: Moving curve $A$ embedded in $\mathbb{R}^1$ representing a single clause of three variables. The $p$- and $t$-axes are part of the parameter space and used merely for illustrative purposes. They display how points of the curve move over time. Points on the actual moving curve have only a single coordinate, namely the $y$-coordinate. In red we indicate the curve at an important timestamp $t = c_1$ and in green we indicate important positions $p \in \{\text{Synch}, W_A, X_A, Y_A, Z_A\}$ along the moving curve.

Figure 3.2: Moving curve $B$ embedded in $\mathbb{R}^1$ encoding a clause of three variables. It has a small Fréchet distance to moving curve $A$ iff the clause can be satisfied. In blue we indicate the curve at important timestamps $t \in \{X^1, Y^1, Z^1\}$ and in green we indicate important positions $p \in \{\text{Synch}, W, W, X, X, Y, Y, Z, Z\}$ along the moving curve.

Surface $A$ contains one ridge for each variable in the formula whereas surface $B$ contains two ridges per variable, representing the possible values (true or false) of the variable. We position the two surfaces such that ridges on surface $A$ are near the corresponding ridges on surface $B$. Then, in order to obtain a small Fréchet distance, the top of ridge $v_A$ has to be mapped to either ridge $v$ or ridge $\overline{v}$ of $B$ for each variable $v$.

For each clause in our formula, we pick a timestamp for which we place spikes on all ridges of $A$ (the row of spikes is marked as $c_1$ in Figure 3.1). The timestamp of each clause on $A$ must be matched with some timestamp on $B$. For each clause we provide three such timestamps on $B$ as candidates, one for each variable in the clause.

For clause $c_1 = (X \lor \overline{Y} \lor Z)$, these three candidates are marked with $X^1$, $Y^1$ and $Z^1$ in
Figure 3.2: Cross sections of \( B \) are shown in blue in Figure 3.3, the left curve corresponds to the timestamp of \( Y^1 \), and the right curve corresponds to the timestamp of \( Z^1 \). In the same figure, the ridges of \( A \) are shown in red, ridges are shown without spikes in the left picture, and with spikes in the right picture.

Figure 3.3: (a) Curve \( A \) (red) just before it crosses the clause and \( B \) (blue) as it crosses second row of spikes. All four red spikes can be matched to both adjacent blue spikes. (b) Curve \( A \) as it crosses the clause and \( B \) as it crosses its last row of spikes. The top of the rightmost red spike cannot be matched to the rightmost blue spike.

At the timestamp of some candidate \( v^1 \), surface \( B \) has spikes on all ridges except for \( v \). Symmetrically, at the timestamp of candidate \( \pi^1 \), spikes exist on all ridges of \( B \) except for \( v \). The chosen candidate timestamp of \( B \) must then be matched with the spikes of \( c_1 \) on \( A \). If candidate \( \pi^1 \) is matched with \( c_1 \), ridge \( v_A \) must be matched with \( \pi \) because ridge \( v \) is too far from the top of the spike of \( v_A \).

Additional clauses can be incorporated in \( A \) and \( B \) by extending the \( t \)-axis and replicating a similar structure on both moving curves. At a given time, both moving curves start with a moving point marked ‘Synch’ in the figures, this moving point is used to prevent the candidates of a clause of \( B \) from being matched to unrelated clauses of \( A \).

### 3.2 Asynchronous Constant

Consider the Asynchronous Constant class of reparameterizations where the matching between the two moving curves is defined by a constant matching of timestamps obtained by \( \sigma \) and \( \tau \) and a constant matching of positions obtained by \( \alpha \) and \( \beta \). The moving curves \( A(p,t) \) and \( B(p,t) \) are then reparameterized as \( A(\mu(p,t)) = A(\alpha(p), \sigma(t)) \) and \( B(\nu(p,t)) = B(\beta(p), \tau(t)) \). We present a construction that shows that approximating the Fréchet distance under this class of matchings within a factor of 1.5 is NP-hard.

**Construction**

Given a 3CNF-formula \( F \) of \( m \) clauses \( (c_1 \ldots c_m) \) and \( n \) variables \( (v_1 \ldots v_n) \), we construct surfaces \( A \) and \( B \) whose Fréchet distance is at most \( \delta \) iff \( F \) is satisfiable. Let \( A \) and \( B \) be quadrilateral meshes of \( (4n+2) \times (4m+1) \) and \( (6n+2) \times (8m+1) \) vertices, respectively. Let
the vertices of \( A(a \in \{0, \ldots, 4n+1\}, s \in \{0, \ldots, 4m\}) \) be defined by Equation (3.1) and those of \( B(b \in \{0, \ldots, 6n+1\}, t \in \{0, \ldots, 8m\}) \) by Equation (3.2).

\[
A(a, s) = \begin{cases} 
0 & \text{if } a = 0 \text{ and } s \mod 4 = 0 \text{ (synch)} \\
3\delta & \text{else if } a = 0 \text{ (synch)} \\
-2\delta & \text{else if } a - 2 \mod 4 = 0 \text{ (valley)} \\
0 & \text{else if } a - 2 \mod 4 \neq 2 \text{ (y = 0)} \\
2\delta & \text{else if } s \mod 4 = 2 \text{ (c\lceil s/4 \rceil)} \\
\frac{3}{2}\delta & \text{otherwise (v)} 
\end{cases}
\]

(3.1)

\[
B(b, t) = \begin{cases} 
0 & \text{if } b = 0 \text{ and } t \mod 8 = 0 \text{ (synch)} \\
3\delta & \text{else if } b = 0 \text{ (synch)} \\
-2\delta & \text{else if } b - 2 \mod 6 = 0 \text{ (valley)} \\
\text{lowRidge}(v_i, t) & \text{else if } b - 2 \mod 6 = 2 \text{ (πi)} \\
\text{lowRidge}(\overline{v_i}, t) & \text{else if } b - 2 \mod 6 = 4 \text{ (vi)} \\
0 & \text{otherwise (y = 0)} 
\end{cases}
\]

(3.2)

Where \( i = 1 + \lfloor (b - 2)/6 \rfloor \) is the index of the variable at position \( b \) and \( \text{lowRidge}(v_i, t) \) prevents spikes on the ridge of \( v_i \) where needed, see Equation (3.3).

\[
\text{lowRidge}(w, t) = \begin{cases} 
\frac{1}{2}\delta & \text{if } t \mod 8 \notin \{2, 4, 6\} \text{ (ridge)} \\
\frac{1}{2}\delta & \text{if } t \mod 8 = 2 \text{ and } c\lceil t/8 \rceil = (w \lor \cdot \cdot) \text{ (π}_2^{\lceil t/8 \rceil)} \\
\frac{1}{2}\delta & \text{if } t \mod 8 = 4 \text{ and } c\lceil t/8 \rceil = (\cdot \lor w \cdot) \text{ (π}_3^{\lceil t/8 \rceil)} \\
\frac{1}{2}\delta & \text{if } t \mod 8 = 6 \text{ and } c\lceil t/8 \rceil = (\cdot \lor \cdot \lor) \text{ (π}_3^{\lceil t/8 \rceil)} \\
\frac{1}{2}\delta & \text{otherwise (spike)} 
\end{cases}
\]

(3.3)

Surfaces \( A \) and \( B \) are then moving curves embedded in \( \mathbb{R}^1 \) and correspond to the examples of Figures 3.1 and 3.2.

**Formal Proof**

The proofs in the first part of this section (up until Lemma 3.2.7) only prove lower bounds on \( \epsilon \). It is safe to assume that the \( p \) and \( t \)-axes of the defined moving curves \( A \) and \( B \) are also spatial dimensions as hinted at by Figures 3.1 and 3.2. Effectively, this embeds \( A \) and \( B \) in \( \mathbb{R}^3 \) instead of \( \mathbb{R}^1 \). This assumption is valid for the proofs up until Lemma 3.2.7 and might make them more intuitive. Keep in mind that these proofs do not make any assumptions on the \( p \)- and \( t \)-axes. As such, they are also valid for the moving curves embedded in \( \mathbb{R}^1 \) as defined by Equations (3.1) and (3.2).

Let the parameter spaces of \( A \) and \( B \) be \( \text{dom}(A) = [0, 4n + 1] \times [0, 4m] \) and \( \text{dom}(B) = [0, 6n + 1] \times [0, 8m] \), in correspondence with the definitions of Equations (3.1) and (3.2). We will refer to various parts of the surfaces with the names given on the far right of these equations. The Asynchronous Constant Fréchet distance between \( A \) and \( B \) is at most \( \epsilon \) iff an Asynchronous Constant matching through the \( \epsilon \)-freespace between \( A \) and \( B \) exists. Then the Fréchet distance is at
most $\varepsilon$ iff orientation preserving homeomorphisms $\mu : [0,1]^2 \to \text{dom}(A)$ and $\nu : [0,1]^2 \to \text{dom}(B)$ exist satisfying Equation (3.4).

$$ (p,t) \in [0,1]^2 \Rightarrow \|B(\nu(p,t)) - A(\mu(p,t))\| \leq \varepsilon $$ (3.4)

Observe that any continuous path on surface $A$ must be matched with some continuous path on surface $B$ and vice-versa. Consider two straight paths $\pi_a$ and $\pi_s : [0,1] \to \text{dom}(A)$ on the parameter space of $A$ defined by $\pi_a(p) = ((4n + 1)p, 0)$ and $\pi_s(p) = (0, (4m)p)$. Similarly, we define two straight paths $\pi_b$ and $\pi_t$ on the parameter space of $B$ with $\pi_b(p) = ((6n + 1)p, 0)$ and $\pi_t(p) = (0, (8m)p)$. Because all considered homeomorphisms are surjective and orientation preserving, we know that $\pi_a$ is matched with $\pi_b$, and that $\pi_s$ is matched with $\pi_t$. For the example of Section 3.1, the matching between $\pi_a$ and $\pi_b$ for $\varepsilon = \delta$ lies in the freespace diagram of Figure 3.4a; if we assume that $p$ and $t$ are spatial dimensions; refer to Figure 3.5a for the case where we do not make this assumption.

![Figure 3.4](image-url)

Figure 3.4: For two freespace-diagrams of $A$ and $B$, a snapshot at the pair of timestamps $(s,t)$. Parameter $a$ on the vertical and $b$ on the horizontal axis. In (a) notice that there are $2^4$ paths (up to directed homotopy) from the bottom-left to the top-right corner, one for each assignment of variables. At the timestamps of (b) $A$ and $B$ have spikes on there ridges, except for the $X$ ridge of $B$. The paths where ridge $X$ is mapped to ridge $X_A$ are not in the freespace for this alignment of timestamps.

Let a valley be a maximal connected subset of a shape with negative $y$-coordinates, and a ridge be a maximal connected subset with positive $y$-coordinates. We show that ridge $v_A$ of $A$ is mapped to $v$ or $\bar{v}$ of $B$ in Corollary 3.2.3, for this we first show that this is the case for the curves at the first timestamps of $A$ and $B$ with Lemma 3.2.1 and Corollary 3.2.2.

**Lemma 3.2.1.** If the Fréchet distance between $A$ and $B$ is $\varepsilon < 1.5\delta$, then the $k$-th valley of $A \circ \pi_a$ is mapped to the $k$-th valley of $B \circ \pi_b$.

**Proof.** Consider the bottoms of the $n$ valleys of $A \circ \pi_a$. All these points have $y$-coordinates of $-2\delta$ and must be matched with some point on $B \circ \pi_b$ with $y \leq \varepsilon - 2\delta$. If $\varepsilon < 1.5\delta$, then those points have $y \leq -0.5\delta$ and therefore lie in valleys of $B$. Because valleys of $B$ are separated by points with $y \geq 0$, and we consider orientation preserving homeomorphisms, every valley bottom of $A \circ \pi_a$ is mapped to at most one valley of $B$. Similarly, the top of each ridge of $A \circ \pi_a$ has $y \geq 1.5\delta$ and must therefore be mapped to at most one ridge of $B$. Because all valleys of $A$ are separated by
ridges and there are equally many valleys on $B \circ \pi_b$, orientation preserving reparameterizations with $\varepsilon < 1.5\delta$ must map the $k$-th valley of $A \circ \pi_a$ to the $k$-th valley of $B \circ \pi_b$. \hfill \square

**Corollary 3.2.2.** If $\varepsilon < 1.5\delta$, the top of each ridge $v_A$ of $A \circ \pi_a$ is mapped to ridge $v$ or $\overline{v}$ of $B \circ \pi_b$.

**Proof.** Let $v = v_i$ be the $i$-th variable, then the top of ridge $v_A$ of $A \circ \pi_a$ must be mapped to one of the two ridges following the $i$-th valley of $B \circ \pi_b$ due to Lemma 3.2.1. The only ridges there are ridge $v_i$ and $\overline{v}$. \hfill \square

**Corollary 3.2.3.** For matchings with $\varepsilon < 1.5\delta$, the top of each ridge $v_A$ of $A$ is mapped to a single ridge $v$ or $\overline{v}$ of $B$.

**Proof.** Consider a path $\pi_b$ along the top of a ridge $v_A$ of $A$; specifically, $\pi_b(p) = (a, (4m)p)$ for some constant $a$ depending on the ridge. By Corollary 3.2.2, the start of this path is matched to ridge $v$ or $\overline{v}$ of $B$. Since ridges of $B$ are separated by paths with $y = 0$ and the image of $A \circ \pi_a$ has $y \geq 1.5\delta > \varepsilon$, the ridges of $A$ are mapped to a single ridge. Therefore, each ridge $v_A$ is mapped to a single ridge $v$ or $\overline{v}$ of $B$. \hfill \square

We can view the matching between $A \circ \pi_a$ and $B \circ \pi_t$ as an assignment of values to the $n$ variables. Let clauses in $F$ be of the form $c_j = (w^1_j \lor w^2_j \lor w^3_j)$, where each $w^i_j$ is of the form $v_i$ or $\overline{v}$. Because of their duality, we will also denote the timestamp of the corresponding spike peaks of $B$ by $w^i_j$. We refer to spikes on the ridges of $A$ by $(v_{iA}, c_j)$ with $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. Define a spike of $B$ to be a maximal connected subset of $B$ with $y > 0.5\delta$. In Lemma 3.2.7, we show that if $F$ is unsatisfiable, no Asynchronous Constant matching with $\varepsilon \leq 1.5\delta$ exists.

Consider the ‘Synch’-parts of both moving curves, in a way similar to Lemma 3.2.1, we can show that their $j$-th bumps are matched (see Lemma 3.2.4).

**Lemma 3.2.4.** If $\varepsilon < 1.5\delta$, the ‘Synch’-parts of $A$ and $B$ match up; i.e. the top of the $j$-th ridge of $A \circ \pi_a$ is mapped to the $j$-th ridge of $B \circ \pi_t$.

**Proof.** All ridges of $A \circ \pi_a$ and $B \circ \pi_t$ have their peak at $y = 3$. This peak can only be mapped to points with $y \geq 3\delta - \varepsilon > 1.5\delta$. Because ridges of $B$ are separated by points with $y \leq 0$, and we consider orientation preserving homeomorphisms, every peak of $A \circ \pi_a$ is mapped to at most one ridge of $B$. Since there are equally many ridges on $B \circ \pi_t$, orientation preserving reparameterizations with $\varepsilon < 1.5\delta$ must map the $j$-th ridge of $A \circ \pi_a$ to the $j$-th ridge of $B \circ \pi_t$. \hfill \square

**Lemma 3.2.5.** If $\varepsilon < 1.5\delta$, the top of spike $(v_i, c_j)$ on $A$’s ridge $v_{iA}$ at timestamp $c_j$ maps to a spike on $B$’s ridge $v_i$ or $\overline{v}$.\hfill \square

**Proof.** Because the spike lies on a ridge, by Corollary 3.2.3, it maps to ridge $v_i$ or $\overline{v}$ of $B$. The top of a spike of $A$ has $y = 2\delta$, so it must map to a point with $y \geq 2\delta - \varepsilon > 0.5\delta$. It must therefore map to a spike on ridge $v_i$ or $\overline{v}$ of $B$.

**Corollary 3.2.6.** For the Asynchronous Constant and Asynchronous Dynamic classes, if $\varepsilon < 1.5\delta$, timestamp $c_j$ on $A$ maps to a timestamp of $B$ with spikes whose peak is at timestamp $w^1_j$, $w^2_j$ or $w^3_j$.

**Proof.** For the Asynchronous classes, if $(a, s) = (0, c_j)$ of $A$ is mapped to $(b, t)$ on $B$, then the path $\pi_{a}(p) = ((4n + 1)p, c_j)$ on $A$ is mapped to some path $((6n + 1)p, t)$ on $B$. By Lemma 3.2.4, the $j$-th ridges of the ‘Synch’-parts are matched, timestamp $c_j$ of $A$ cannot be matched with any spike whose peak has timestamp $w^k_j$, $w^2_j$ or $w^3_j$ unless $j = k$. Because of Lemma 3.2.5, timestamp $c_j$ on $A$ then maps to the timestamp of the spikes of $B$ with a peak at timestamp $w^1_j$, $w^2_j$ or $w^3_j$. \hfill \square
Lemma 3.2.7. Given that $F$ is unsatisfiable, no Asynchronous Constant or Asynchronous Dynamic matching satisfies Equation \((3.4)\) for $\varepsilon < 1.5\delta$.

Proof. Suppose some matching $\nu \circ \mu^{-1}$ exists satisfying Equation \((3.4)\) for $\varepsilon < 1.5\delta$, we show that we can then give a valid assignment $S$ satisfying $F$. An assignment is valid unless it contains both $v$ and $v'$ for some variable $v$. Consider the assignment $S = \{w^j_i \mid (\nu \circ \mu^{-1})(0, c_j) = (0, t) \wedge w^j_i - 1 < t < w^j_i + 1\}$. We show that $S$ is valid and satisfies $F$. It is easy to see that $S$ satisfies $F$ because of Corollary 3.2.6, which states that $w^*_j$, $w^*_j$ or $w^*_j$ is in $S$ for each clause $c_j = (w^*_j \lor w^*_j \lor w^*_j)$.

To show that $S$ is valid, suppose that $S$ is invalid. Then there exist $v = w^j_i \in S$ and $v' = w'^j_i \in S$ for some variable $v$. From Lemma 3.2.5 it follows that if $v = w^j_i \in S$, then ridge $v_A$ of $A$ is mapped to ridge $v$ of $B$ whereas if $v' \in S$, this ridge is mapped to ridge $v'$ of $B$. According to Corollary 3.2.3, ridge $v_A$ can only be mapped to a single ridge of $B$, so $v$ and $v'$ cannot both be in $S$, so $S$ is valid. Therefore no Asynchronous Constant matching with $\varepsilon < 1.5\delta$ exists unless $F$ is satisfiable.

Before we prove the converse, we stress that we assume that $p$ and $t$ are not spatial dimensions. Recall that this does not invalidate any of the previous lemmas in this section because they only restrict the $y$-coordinates. Now the surfaces formed by the moving curves $A$ and $B$ are embedded entirely in $\mathbb{R}^1$. A consequence is that the result holds for any of the $L^p$-norms in any dimension $\mathbb{R}^n$ where $n$ can be as small as 1. Although the freespace of $\varepsilon = \delta$ gets bigger if we remove dimensions $p$ and $t$ (see Figure 3.5), no additional paths open with respect to Figure 3.4 (which assumed axes $p$ and $t$ were spatial).

![Figure 3.5: Snapshots of the freespace diagram where $A$ and $B$ are embedded in $\mathbb{R}^1$. Because backtracking is not allowed, diagrams 3.5a and 3.5b are equivalent to Figures 3.4a and 3.4b embedded in $\mathbb{R}^2$.](image)

Observe that for each ridge (or valley) on $A$ or $B$, its peak (or bottom) has a preimage of the form $\{p\} \times T$ for some $p$, where $T = [0, 4m]$ for $A$ and $T = [0, 8m]$ for $B$. The entire ridge (or valley) including the boundary points with $y = 0$ then has a preimage of the form $[p - 1, p + 1] \times T$. Because this pattern is so common in our construction, we define some notation to match entire ridges, valleys and spike rows etc. instead of just their peaks and bottoms in Definition 3.2.8.

Let $S$ be an assignment of variables, then we define a matching of timestamps and a matching of positions in Matchings 1 and 2 which together form a Asynchronous Constant matching. In Lemma 3.2.9 we show that this matching satisfies Equation \((3.4)\) for $\varepsilon = \delta$ if $S$ satisfies $F$. 
Definition 3.2.8 (Extended matching). Define an extended matching to match the surroundings of positions $a$ and $b$ along two curves with $\text{ext}_{\alpha,\beta}(a, b) \Rightarrow \forall \alpha \in (-1, 1) \alpha(p) = a + \alpha \Rightarrow \beta(p) = b + \alpha$.

**Matching 1** $(\alpha \circ \beta^{-1})$. We define $\alpha$ and $\beta$ which can be thought of as choosing ridges corresponding to the values of the variables. Match the $k$-th valley of $B$ with $\text{ext}_{\alpha,\beta}(2 + 4k, 2 + 6k)$ for $k \in \mathbb{N}$. Similarly, match the ‘Synch’-parts of $A$ and $B$ with $\text{ext}_{\alpha,\beta}(0, 0)$. We still need to match the ridges of $B$ to $A$. If $v_i \in S$, use $\text{ext}_{\alpha,\beta}(4i + 4, 6i + 6)$ to match ridge $v_i$ to $v_{iA}$, otherwise map the entire ridge $v_i$ to the base of $v_{iA}$ with $(\beta(p) \in [6i + 6 - 1, 6i + 6 + 1] \Rightarrow \alpha(p) = 4i + 4 + 1)$. Conversely, if $v_i \in S$, use $\text{ext}_{\alpha,\beta}(4i + 4, 6i + 4)$ to match ridge $\overline{p}$ to $v_{iA}$, otherwise map the entire ridge $\overline{p}$ to the base of $v_{iA}$ with $(\beta(p) \in [6i + 4 - 1, 6i + 4 + 1] \Rightarrow \alpha(p) = 4i + 4 - 1)$. Now $\alpha \circ \beta^{-1}$ constitutes a surjective orientation preserving matching between the positions along curves of $B$ and $A$.

**Matching 2** $(\tau \circ \sigma^{-1})$. We define the matching of timestamps with $\sigma$ and $\tau$, this can be thought of as assigning a witness to each clause. Because $S$ satisfies $F$, each clause $c_j = (w^j_1 \lor w^j_2 \lor w^j_3)$ has $w^j_1$, $w^j_2$ or $w^j_3$ in $S$, so choose such $w^j_i \in S$. Let $t_j = 2 + 2l + 8j$ be the timestamp of $B$ corresponding to the peaks of the spike row labeled $w^j_i$. Match the spike row of $B$ between timestamps $t_j - 1$ and $t_j + 1$ with the $j$-th spike row of $A$ using $\text{ext}_{\sigma,\tau}(c_j, t_j)$ where $c_j = 2 + 4j$. We can map both of the remaining two spike rows $(w^j_1, w^j_2)$ of $B$ to timestamp $c_j - 1$ of $A$ if $l' < l$, and to $c_j + 1$ if $l' > j$. Finally, the timestamps in between clauses can be trivially matched using $\text{ext}_{\sigma,\tau}(4j, 8j)$ for $j \in \mathbb{N}$. Now $\tau \circ \sigma^{-1}$ constitutes a surjective orientation preserving matching between the timestamps of $A$ and $B$.

**Lemma 3.2.9.** Given $F$ and an assignment $S$ of variables satisfying it, an Asynchronous Constant matching exists satisfying Equation (3.4) for $\varepsilon = \delta$.

**Proof.** Consider the matching defined by $\mu(p, t) = (\alpha(p), \sigma(t))$ and $\nu(p, t) = (\beta(p), \tau(t))$ where $\alpha \circ \beta^{-1}$ is given by Matching 1 and $\tau \circ \sigma^{-1}$ is given by Matching 2. Now $\mu$ and $\nu$ form an Asynchronous Constant matching and we show that this matching satisfies Equation (3.4) for $\varepsilon = \delta$. Evidently, all valleys, as well as the ‘Synch’-parts coincide with their matching. Note that we have defined matchings such that $\nu \circ \mu - 1$ is injective. Therefore we only have to verify that all points on ridges of $B$ are matched to some point on $A$ within distance $\delta$. All ridges $v_i \notin S$ and $\overline{v_i} \notin S$ of $B$ have $0 \leq y \leq \delta$ and are mapped to $y = 0$ on $A$, which is within distance $\delta$. Consider the remaining ridges $v_i \in S$ and $\overline{v_i} \in S$ of $B$. If the top of such a ridge $r$ is in $S$ with the top of $v_{iA}$ within distance $\delta$, the entire ridge can be matched within distance $\delta$ due to the extended matchings, whose bottleneck is the peak of the ridge. We show that the top of $r$ is matched with the top of $v_{iA}$ of $A$ within distance $\delta$. The top of ridge $r$ of $B$ has $y \geq \frac{1}{2} \delta$ and the top of $v_{iA}$ has $y = \frac{1}{2}$ except where spikes occur on $v_{iA}$. We can therefore ignore any part of the top that is not matched with a spike of $v_{iA}$. Due to extended matchings, we now only need to verify that the timestamps of ridge $r$ have $y = \delta$ if they are mapped to the top of a spike (with $y = 2\delta$) on $v_{iA}$. The $m$ timestamps of $B$ that are mapped to the top of spikes of $v_{iA}$ are given by $t_j$ in $\tau \circ \sigma^{-1}(t_j) = c_j = 2 + 4j$ for $j \in \{1, \ldots, m\}$. We know that $t_j = 2 + 2l + 8j$ for some $l \in \{1, 2, 3\}$ and that $w^j_1 \in S$ satisfies clause $(w^j_1 \lor w^j_2 \lor w^j_3)$. The point corresponding to $t_j$ on $B$ is given by Equations (3.2) and (3.3) at coordinate $(r, t_j) = (6i + 6, 2l + 8j)$ if $r = v_i \in S$ and $(r, t_j) = (6i + 4, 2l + 8j)$ if $r = \overline{v_i} \in S$. This point has $y = \delta$ unless either $r = v_i \in S$ and $w^j_1 = \overline{v_i} \in S$, or $r = \overline{v_i} \in S$ and $w^j_1 = v_i \in S$. Because a valid assignment $S$ cannot contain both $v_i$ and $\overline{v_i}$, this point must have $y = \delta$. Therefore this matching satisfies Equation (3.4) for $\varepsilon = \delta$. \qed
3.3. Asynchronous Dynamic

To conclude the Asynchronous Constant case, we briefly summarize the NP-hardness result in Theorem 3.2.10.

**Theorem 3.2.10.** The Asynchronous Constant Fréchet distance cannot be approximated in polynomial time within a factor $1.5$ unless $P=NP$, even if all vertices lie in $\mathbb{R}^1$.

**Proof.** By Lemmas 3.2.7 and 3.2.9, a 3CNF-formula $F$ can be transformed into quadrilateral meshes $A$ and $B$ embedded in $\mathbb{R}^1$ in polynomial time, such that the Asynchronous Constant Fréchet distance is at most $c$ if $F$ is satisfiable and $1.5c$ otherwise. It follows that computing the Asynchronous Dynamic Fréchet distance is NP-hard. \hfill \blacksquare

### 3.3 Asynchronous Dynamic

For the Asynchronous Dynamic class of reparameterizations, the matching between the two moving curves is defined by a constant matching of timestamps obtained by $\sigma$ and $\tau$ and a dynamic matching of positions obtained by $\alpha$ and $\beta$. The moving curves $A(p,t)$ and $B(p,t)$ are then reparameterized as $A(\mu(p,t)) = A(\alpha(p,\sigma(t)),\sigma(t))$ and $B(\nu(p,t)) = B(\beta(p,\tau(t)),\tau(t))$. We can easily extend the result from Section 3.2 to the Asynchronous Dynamic case.

**Theorem 3.3.1.** The Asynchronous Dynamic Fréchet distance cannot be approximated in polynomial time within a factor $1.5$ unless $P=NP$, even if all vertices lie in $\mathbb{R}^1$.

**Proof.** Recall that Lemma 3.2.7 also holds for Asynchronous Dynamic matchings. Because any Asynchronous Constant is also an Asynchronous Dynamic matching, Lemma 3.2.9 also holds for Asynchronous Dynamic matchings. Therefore, a 3CNF-formula $F$ can be transformed into quadrilateral meshes $A$ and $B$ embedded in $\mathbb{R}^1$ in polynomial time, such that the Asynchronous Dynamic Fréchet distance is at most $c$ if $F$ is satisfiable and at least $1.5c$ otherwise. \hfill \blacksquare

### 3.4 Unrestricted Dynamic

Consider the Unrestricted Dynamic class of reparameterizations. The allowed matchings between the two moving curves then range over all orientation preserving homeomorphisms. As opposed to the Asynchronous Constant and Asynchronous Dynamic classes, it is unknown whether the Unrestricted Dynamic case is NP-hard in $\mathbb{R}^1$. Instead, we will modify the surfaces used in the construction of Section 3.3 to show that the problem is NP-hard for moving curves embedded in $\mathbb{R}^2$.

Note that Lemma 3.2.7 does not hold for Unrestricted Dynamic matchings because Corollary 3.2.6 assumes constant temporal matchings. We therefore come up with a different way of making sure a clause of $A$ does not match to spikes on $B$ with different timestamps, i.e. we want to restrict the matching of a clause to a single spike row of $B$. Assume for now that we are using the $L^\infty$ norm which gives $\|(x,y)\| = \max(|x|,|y|)$. We will only sketch the construction because of its resemblance with the proofs for the Asynchronous Constant and Asynchronous Dynamic cases.

Just like we used $y$-coordinates to separate variables and ridges, we can use $x$-coordinates to separate spike rows, call these separation mechanisms $x$Ridges (see Figures 3.6 and 3.7). Let the ‘top’ of $x$Ridges of $A$ have $x = \frac{1}{2}\delta$, the ‘top’ of $x$Ridges of $B$ have $x = \frac{1}{3}\delta$, and the base of each $x$Ridge have $x = 0$, just like the original ridges had for $y$-coordinates. We build an $x$Ridge
Figure 3.6: The \(x\)-coordinates for the moving curve \(A\) of the example of Section 3.1. We overlay the important \(y\)-coordinates of Figure 3.1 in red and green.

Figure 3.7: The \(x\)-coordinates for the moving curve \(B\) of the example of Section 3.1. We overlay the important \(y\)-coordinates of Figure 3.2 in blue and green.

on top of each spike row to make sure spike rows lie on different xRidges. We make sure clauses stay separated by connecting the xRidges to the ‘Synch’-parts that are already separated by Lemma 3.2.4. Now the \(j\)-th xRidge (with a peak at timestamp \(c_j\)) of \(A\) must be mapped to one of the three xRidges of \(B\) corresponding to the \(j\)-th clause. As a consequence, we can prove Lemma 3.4.1 as a generalization of Corollary 3.2.6. Now there exists a satisfying assignment to \(F\) if the new surface which can be matched with \(\varepsilon < 1.5\delta\) under the \(L^\infty\) norm by analogy with Lemma 3.2.7. Conversely Matching 1 and Matching 2 still constitute an Asynchronous Constant matching with \(\varepsilon \leq \delta\) under the \(L^\infty\) norm given an assignment \(S\) satisfying \(F\).

**Lemma 3.4.1.** If \(\varepsilon < 1.5\delta\), timestamp \(c_j\) on \(A\) maps to a single xRidge of \(B\) whose peak is at timestamp \(w^1_j, w^2_j\) or \(w^3_j\).

**Proof.** Then the xRidge on timestamp \(c_j\) of \(A\) has its peak at \(x \geq 1.5\delta\), which maps to a single xRidge of \(B\) because their boundaries have \(x = 0\). Timestamp \(c_j\) of \(A\) cannot be matched with any spike whose peak has timestamp \(w^1_k, w^2_k\) or \(w^3_k\) unless \(j = k\) due to the matching of the ‘Synch’-parts. Therefore, timestamp \(c_j\) on \(A\) maps to a single xRidge of \(B\) with a peak at timestamp \(w^1_j, w^2_j\) or \(w^3_j\). \(\square\)
Theorem 3.4.2. The Unrestricted Dynamic Fréchet distance under the $L^\infty$ norm cannot be approximated in polynomial time within a factor 1.5 unless $P=NP$, even if all vertices lie on three parallel lines in $\mathbb{R}^2$.

Proof. Because any Unrestricted Dynamic is also an Asynchronous Dynamic matching, a version of Lemma 3.2.9 that takes into account the $x$-coordinates also holds for Unrestricted Dynamic matchings. Therefore, a 3CNF-formula $F$ can be transformed into quadrilateral meshes $A$ and $B$ embedded in $\mathbb{R}^2$ in polynomial time, such that the Unrestricted Dynamic Fréchet distance is at most $c$ if $F$ is satisfiable and at least $1.5c$ otherwise.

By Theorem 3.4.2 the bound of 1.5 on the approximation factor still holds for $L^\infty$ norm (and the $L^0$ norm if we rotate the shapes by 45 degrees), similar bounds exist for other $L^p$ norms. Moreover, our construction uses only vertices on three parallel lines in the plane ($\mathbb{R}^2$). Because all quadrilaterals lie in the plane, these can be reproduced with two triangles, so our result extends to the Fréchet distance between triangulated surfaces under orientation preserving homeomorphisms. Although it was already known that the problem was NP-hard for triangulated surfaces under Unrestricted Dynamic matchings, our construction is possibly easier to understand.
Chapter 4

Freespace Cell Visualizations

In Sections 2.2 and 2.3, we presented algorithms to decide whether particular kinds of surfaces through a 3D freespace between quadrilateral meshes exist. The decision algorithms are used to compute or approximate the Synchronous Constant and Synchronous Dynamic Fréchet distance between two quadrilateral meshes. The 3D freespace is a subset of $\mathbb{R}^3$ defined as a function of two quadrilateral meshes $A$ and $B$ and a real number $\varepsilon$. This function is defined by Equation (4.1), where $P \times T$ represents the number of quadrilaterals defining $A$, as well as $B$.

$$F_{3D}^{\varepsilon}(A, B) = \{(a, b, t) \in [0, P] \times [0, T] \mid \|B(b, t) - A(a, t)\| \leq \varepsilon\}$$

(4.1)

Although the 3D freespace for two quadrilateral meshes can be quite complex, it is possible to construct it using simpler sets. These simpler sets are the 3D freespace between two quadrilaterals and are defined by Equation (4.2), where $A$ and $B$ are single quadrilaterals instead of quadrilateral meshes. To gain an intuition for the shape of these sets, we have visualized this set for several pairs of quadrilaterals.

$$F_{\varepsilon}^{3D}(A, B) = \{(a, b, t) \in [0, 1]^3 \mid \|B(b, t) - A(a, t)\| \leq \varepsilon\}$$

(4.2)

Figure 4.1: The complement $[0, 1]^3 \setminus F_{\varepsilon}^{3D}(A, B)$ of the 3D freespace for three values of $\varepsilon$.

The empty space of Figure 4.1 represents the freespace between two quadrilaterals for three (increasing from left to right) values of $\varepsilon$. Images in this section use the Euclidean distance as norm for the freespace.
Chapter 4. Freespace Cell Visualizations

The points of the visualised sets that lie on the boundary of the unit cube are drawn in gray, whereas the points in the interior are colored. To give an additional sense of direction, we have colored points on the boundary of the freespace according to the gradient at that point. Specifically, the alignment of the normal vector at a point with each of the three axes \((a, b\text{ and } t)\) determines the respective red, green and blue components of that point’s color. In the illustrations of this section, the \(t\)-axis always faces either up or down.

Light grid-lines in the directions of the \(a\)- and \(b\)-axes are projected onto the visualized sets. Dark contour lines are drawn for several values of \(t\), for convex norms, these contour lines will be convex. For the Euclidean distance, the contour lines form ellipses as can be seen in Figure 4.2. The empty space of Figure 4.2 is the freespace between the same quadrilaterals as those of Figure 4.1 but from a different angle and for different values of \(\varepsilon\).

![Figure 4.2: The contour lines form ellipses.](image)

As can be seen in the previous figures, the freespace between two quadrilaterals is not necessarily convex. In fact, the freespace between two quadrilaterals may even consist of multiple disconnected components, as shown on the next page. Whereas Figures 4.1 and 4.2 display the points outside the freespace, Figure 4.3 displays the points within the freespace. Figures 4.3a and 4.3b both show the 3D freespace for five values of \(\varepsilon\) (increasing from top to bottom), but for different pairs of quadrilaterals. It can be seen that for small \(\varepsilon\), the freespace may even consist of three disconnected sets.

Observe that the intersection of the 3D freespace with any vertical line (aligned with the \(t\) axis) is convex, as was proven in Lemma 2.2.1. Although the freespace between two quadrilaterals is still quite complex, it turns out that our algorithms need not compute the interior of these cells, and the intersections with the boundaries of the unit cube suffice.
Figure 4.3: Disconnected freespace.
Chapter 5

Conclusion

We have provided polynomial time algorithms for computing the Fréchet distance for three classes of matchings between quadrilateral meshes, which represent moving curves. By symmetry, the Fréchet distance for two related classes can also be computed in polynomial time. The most restrictive class (of Synchronous Identity matchings) simply compares trajectories of two aligned moving curves of \( p \) line segments moving over \( t \) timestamps without realigning any points or timestamps in \( O(pt) \) time.

A slightly wider class is the Synchronous Constant class which allows positions along moving curves to be realigned, but does not allow realigning timestamps. For this class we have shown how to compute an aggregate norm and derived related properties for single line segments that move over \( t \) timestamps. The norm itself can be computed in \( O(t^2) \) time, and critical values can be computed in \( O(t \log t) \) time for sufficiently simple underlying norms such as the Euclidean distance. A chain of moving line segments can then be used to form a moving curve, allowing the presented norm to be used directly in algorithms for existing variants of the Fréchet distance. Effectively, this extends existing algorithms from comparing curves to comparing moving curves.

We should note that the aggregate norm applies to a rather restrictive class of matchings, and more complex models of comparison can be required. Whereas the aggregate norm assumes that the alignments of positions along polylines are constant, the Synchronous Dynamic class allows alignments to change over time. An \( O(p^4t) \) time decision algorithm based on partitioning obstacles in a 3D space is presented, this leads to an \( O(p^4t \log(pt)) \) time algorithm to compute the exact Synchronous Dynamic Fréchet distance between two moving curves.

For more complex classes where the alignments of timestamps are not known in advance, we have presented reductions to prove the NP-hardness of approximating the Fréchet distance within a factor 1.5. One of these reductions extends to known results on computing the Fréchet distance between triangular meshes. We think our construction is simpler than the previous constructions, and we believe this construction might lead to even stronger results.

5.1 Future Work

The decision algorithm used to compute the Synchronous Dynamic Fréchet distance between two quadrilateral meshes of size \( p \times t \) has a running time of \( O(p^4t) \). We have shown that intermediary steps of this algorithm may construct a graph with \( O(p^4t) \) edges where only \( O(p^3t) \) edges are required. We therefore believe that more sophisticated algorithms can efficiently construct a smaller graph and perhaps solve this problem in \( O(p^3t) \) time.
A related problem exists where instead of a quadrilateral mesh, a sequence of polylines has no predetermined interpolation scheme between consecutive polylines. We believe this problem is worth investigating because it may give a more practical similarity measure.

As an alternative to a parametric search, we have mentioned an approach based on widest paths, or bottleneck shortest paths in weighted graphs. Although such a weighted graph might be inefficient to compute due to the large amount of critical values, it might be possible to prune many critical values while computing this graph. Studying this approach might lead to faster exact algorithms for many variants of the Fréchet distance.

Finally, we have provided NP-hardness constructions for three classes of matchings between quadrilateral meshes. Two of these classes turned out to be NP-hard even if the meshes were embedded in $\mathbb{R}^1$. The third reduction instead uses an embedding in $\mathbb{R}^2$ and we wonder if this reduction extends to embeddings in $\mathbb{R}^1$. 


Bibliography


