Fréchet View – A Tool for Exploring Fréchet Distance Algorithms

Peter Schäfer
FernUniversität in Hagen, Germany
peter.schaefer1@studium.fernuni-hagen.de

Abstract

The **Fréchet-distance** is a similarity measure for geometric shapes. Alt and Godau presented the first algorithm for computing the Fréchet-distance and introduced a key concept, the **free-space diagram**. Since then, numerous variants of the Fréchet-distance have been studied.

We present here an interactive, graphical tool for exploring some Fréchet-distance algorithms. Given two curves, users can experiment with the free-space diagram and compute the Fréchet-distance. The Fréchet-distance can be computed for two important classes of shapes: for polygonal curves in the plane, and for simple polygonal surfaces.

Finally, we demonstrate an implementation of a very recent concept, the ***k*-Fréchet-distance**.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Fréchet distance, free-space diagram, polygonal curves, simple polygons

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.66

Category Multimedia Exposition

Supplement Material `https://hrimfaxi.bitbucket.io/fv`

- Binaries (for Windows, Linux, macOS), sample data and source code
- Brief user guide
- Video presentation

Acknowledgements This program was developed as part of my master thesis at the chair of Prof. Dr. André Schulz, supervised by Dr. Lena Schlipf.

1 The Fréchet-distance

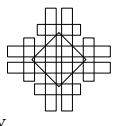
The Fréchet-distance is a metric for describing similarity between two continuous shapes. The need for comparing shapes arises in many fields of applications, like computer vision, handwriting recognition, proteine matching, map construction, analysis of movement data, and more. The Fréchet-distance is similar to the Hausdorff-metric, but it better captures the notion of similarity. It is defined as the minimum bottleneck-cost over all possible homeomorphisms.

▶ **Definition 1.** *Let P and Q be two given curves in a metric space S. A reparametrization α of $[0,1]$ is a continuous, non-decreasing surjection $\alpha : [0,1] \mapsto [0,1]$.*

*The **Fréchet-distance** between P and Q is the infimum over all reparametrizations α and β of the maximum over all $t \in [0,1]$ of the distance between $P(\alpha(t))$ and $Q(\beta(t))$:*

$$\delta_F(P,Q) \quad = \quad \inf_{\alpha,\beta} \max_{t\in[0,1]} \quad || \, P(\alpha(t)) - Q(\beta(t)) \, || \, ,$$

where $||\cdot||$ denotes the distance function of S. In the following sections we will always assume the Euclidean norm in $\mathbb{R}^2$, and we will only consider polygonal curves and shapes.



An intuitive explanation of the Fréchet-distance is this: imagine a man walking his dog on a leash. The man walks on one curve, while the dog follows the other curve. Both the man and the dog must move forward. The *weak* Fréchet-distance would allow for backward movement, also. We will come back to it in section 3.

- the *decision problem* asks: is it possible to walk the curves with a leash of given length ε?
- the *optimization problem* asks for the shortest possible leash.

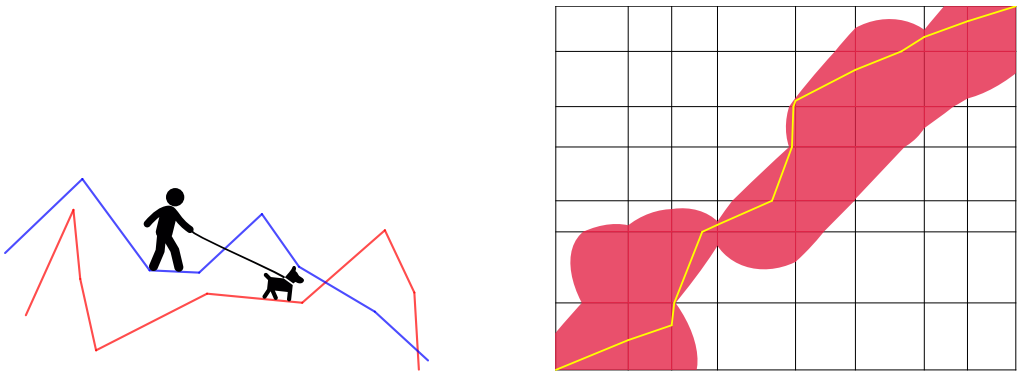


Figure 1 Two curves, and a dog on a leash. Right: the corresponding free-space diagram with a feasible path.

Alt and Godau [2] presented an algorithm for computing the Fréchet-distance for polygonal curves in the plane. The key concept of this algorithm is the **free-space diagram**: one curve is mapped to the x-axis of the diagram, the second curve is mapped to the y-axis, spanning a two-dimensional grid, or configuration space.

The **free-space** represents all pairs of points whose distance is less than or equal to ε. In Figure 1 the free-space is indicated by the shaded area (which is composed of numerous ellipse sections).

We have $d_F(P, Q) \leq \varepsilon$, if there is a monotone path in the free-space diagram starting at the lower left corner $(0, 0)$ and ending in the upper right corner (p, q) (where p and q denote the number of vertices in the curves). We call that a **feasible path**.

For *closed* curves, the left and right edges of the free-space diagram coincide. Just imagine the diagram to wrap around at the egdes, creating a cylinder (or a torus, though the latter is not relevant for our algorithms).[1] The starting point of a feasible path may be located anywhere on the x-axis. Alt and Godau's [2] algorithm finds such a path in $O(pq \log pq)$ time. We can thus solve the decision problem.

The optimization problem is solved by applying the algorithm repeatedly on a set of critical values, i. e. values of ε for which the structure of the free-space diagram changes significantly. **Fréchet View** is a useful tool for examining the structue of free-space diagrams.

2 The Fréchet-distance for Simple Polygons

The Fréchet-distance easily extends to higher dimensions. As it turns out, computation becomes significantly harder, already for two-dimensional surfaces. Buchin et al. [3] presented a polynomial-time algorithm for a natural subset of surfaces, namely **simple polygons** (i. e.

[1] In the literature this is often referred to as a *double*-free-space diagram

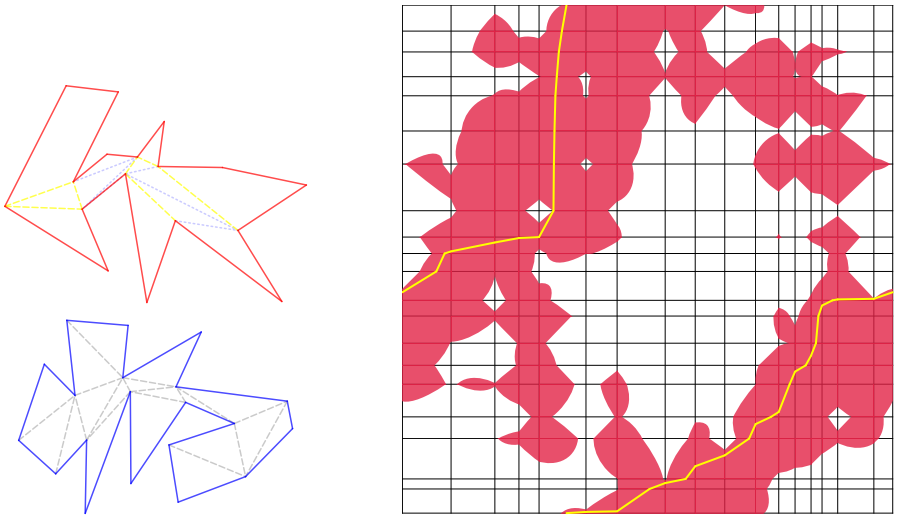


Figure 2 Triangulated polygons and their free-space diagram.

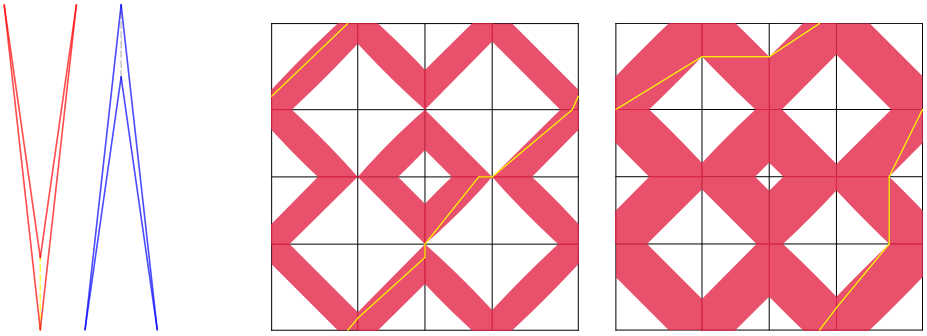


Figure 3 The Fréchet-distance of the polygonal surfaces differs from the Fréchet-distance of their boundary curves.

polygons without holes and self-intersection). Basically, their algorithm is able to break down the problem to a *convex decomposition* and *triangulation* of the polygons (see Figure 2). It is sufficient to consider certain diagonals, which are mapped to *shortest paths* in the other polygon, with additional conditions applying. To the best of our knowledge, ours is the first practical implementation of this algorithm.

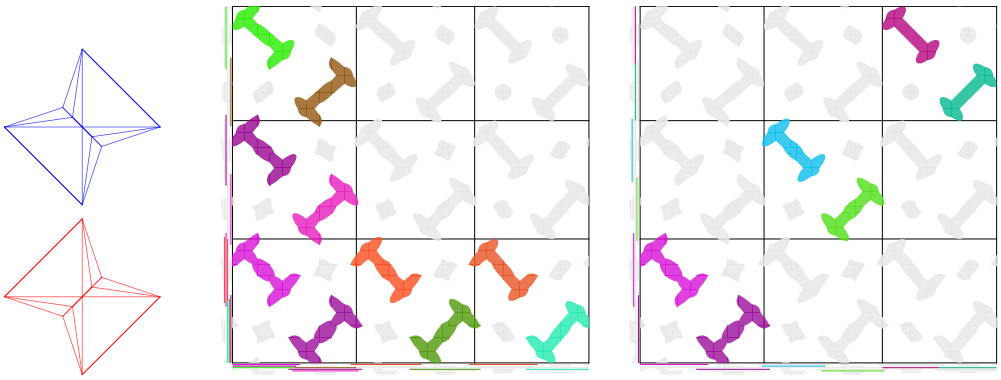
In many cases, the Fréchet-distance for simple polygons is just identical to the Fréchet-distance of their boundary curves. Figure 3 shows an example where they actually differ.

3 The k-Fréchet-distance

Buchin and Ryvkin [4, 1] recently introduced a new concept, the k-Fréchet-distance. It represents a connecting bridge between the Hausdorff-metric and the well-known *weak* Fréchet-distance. Roughly speaking, the input curves are cut into parts and each part is mapped using the weak Fréchet-distance. With the weak Fréchet-distance, reparametrizations are not required to be monotone; man and dog are allowed to move backwards.

The question is: can we find a mapping with at most k parts? This adds a strong combinatorial aspect to the problem. In fact, the problem was shown to be NP-complete by Buchin and Ryvkin [4, 1].

We present implementations of two algorithms: an exponential-time "*brute-force*" exact algorithm, and a *greedy* approximation algorithm.



(a) Input curves. **(b)** Result of the *greedy* algorithm. **(c)** Optimal result.

Figure 4 k-Fréchet-distance: greedy vs. optimal results. The approximation factor can be shown to approach **2**.

Many questions regarding the k-Fréchet-distance are still open to research. For one, the exact approximation factor of the greedy algorithm was not yet known. With the help of Fréchet View we were able to construct curves whose approximation ratio comes arbitrarily close to 2 (see Figure 4).

4 Fréchet View – the Implementation

Fréchet View is a tool for exploring various aspects of the Fréchet-distance algorithms. It expects two polygonal curves as input data. These curves are stored in vector graphics files (in SVG, or IPE format) and can be edited with any usual vector graphics editor, like InkScape, or IPE. In addition, we provide an implementation to assemble curves from straightforward JavaScript code.

Both curves and the correspoding free-space diagram are displayed on screen. The user can manipulate the parameter ε and watch the changes to the free-space diagram. Whenever a feasible path is found, it is drawn within the free-space diagram. Moving the mouse pointer over the path displays the mapped sections on the input curves (and vice versa).

Graphics produced by Fréchet View can be stored to disk. For example, all figures in this document have been created using Fréchet View.

The program is written in C++, making use of the Computational Geometry Algorithms Library [5], and a number of development frameworks like Qt, Boost, TBB, and more. It can be run from the command line with additional options for multi-core and GPGPU-support. However, these features are not part of our presentation.

References

1 Hugo A. Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet distance revisited and extended. In *35th European Workshop on Computational Geometry*, 2019. URL: `http://www.eurocg2019.uu.nl/papers/41.pdf`.

2 Helmut Alt and Michael Godau. Computing the Fréchet Distance between two Polygonal Curves. *International Journal of Computational Geometry and Applications*, 5(1-2):75–91, 1995. `doi:10.1142/S0218195995000064`.

3 Kevin Buchin, Maike Buchin, and Carola Wenk. Computing the Fréchet Distance Between Simple Polygons in Polynomial Time. In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, SCG '06, pages 80–87, 2006. doi:10.1145/1137856.1137870.

4 Maike Buchin and Leonie Ryvkin. The k-Fréchet distance of polygonal curves. In *34th European Workshop on Computational Geometry*, 2018. URL: https://conference.imp.fu-berlin.de/eurocg18/download/paper_43.pdf.

5 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.12 edition, 2018. URL: https://doc.cgal.org/4.12/Manual/packages.html.