

# Matching Planar Maps

Helmut Alt\*

Alon Efrat<sup>†</sup>

Günter Rote\*

Carola Wenk<sup>†</sup>

## 1 Introduction

Patterns consisting of line segments occur in many applications of a geometric nature, like computer vision, geographic information systems, CAGD, etc. In many cases the problem occurs to determine whether some given pattern  $H$  is equal to or similar to some part of a larger pattern  $G$ . Here, we consider the cases that  $H$  is a polygonal curve and  $G$  is a geometric graph, or that both  $H$  and  $G$  are geometric graphs. We give both feasible distance measures as well as efficient algorithms for computing these distances. In this extended abstract we only sketch the results; more details can be found in [1]. These distance measures are generalizations of the well known Fréchet distance for curves, which has been investigated before in [2].

**DEFINITION 1.1.** (FRÉCHET DISTANCE) *Let  $f : I = [l_I, r_I] \rightarrow \mathbb{R}^2$ ,  $g : J = [l_J, r_J] \rightarrow \mathbb{R}^2$  be two planar curves, and let  $\|\cdot\|$  denote the Euclidean norm. Then the Fréchet distance  $\delta_F(f, g)$  is defined as*

$$\delta_F(f, g) := \inf_{\substack{\alpha: [0,1] \rightarrow I \\ \beta: [0,1] \rightarrow J}} \max_{t \in [0,1]} \|(f(\alpha(t)), g(\beta(t)))\|.$$

where  $\alpha$  and  $\beta$  range over continuous and non-decreasing reparametrizations with  $\alpha(0) = l_I$ ,  $\alpha(1) = r_I$ ,  $\beta(0) = l_J$ ,  $\beta(1) = r_J$  only.

## 2 Matching a Curve in a Graph

As a first task we consider a given polygonal curve  $\alpha : [0, p] \rightarrow \mathbb{R}^2$ , and an undirected connected planar graph  $G = (V, E)$  with a given straight-line embedding in  $\mathbb{R}^2$ ,  $|V| = q$ . We wish to find a path  $\pi$  in  $G$ , which we identify with the corresponding polygonal curve, such that the Fréchet distance  $\delta_F(\alpha, \pi)$  is minimized.<sup>1</sup> An application is to locate the path a person travelled, while recording a sequence of *GPS positions*, in a given roadmap. This is a prerequisite for incrementally constructing roadmaps from such GPS curves, which is

especially interesting for roads such as hiking trails in a forest which are not visible on aerial pictures. We attack the minimization problem by first solving the decision problem for which we fix  $\varepsilon > 0$  and wish to find a path (if it exists) in  $G$  such that the Fréchet distance is at most  $\varepsilon$ . Afterwards we apply parametric search, similar as in [2], to eventually solve the minimization problem. Our algorithm solves the decision problem in  $O(pq \log q)$  time and  $O(pq)$  space. Parametric search adds another factor of  $\log(pq)$  to the runtime. Hence, our algorithm is only a logarithmic factor slower than the algorithm for computing the Fréchet distance between two curves, although the task seems to be much more challenging at first sight.

**DEFINITION 2.1.** ([2]) *Let  $f : I \rightarrow \mathbb{R}^2$ ,  $g : J \rightarrow \mathbb{R}^2$  be two curves;  $I, J \subseteq \mathbb{R}$ . The set  $F_\varepsilon(f, g) := \{(s, t) \in I \times J \mid \|f(s) - g(t)\| \leq \varepsilon\}$  denotes the free space of  $f$  and  $g$ . We call the partition of  $I \times J$  into regions belonging or not belonging to  $F_\varepsilon(f, g)$  the free space diagram  $FD_\varepsilon(f, g)$ .*

We call points in  $F_\varepsilon$  *white* or *feasible* and points in  $FD_\varepsilon \setminus F_\varepsilon$  *black* or *infeasible*. In [2] it has been shown that  $\delta_F(f, g) \leq \varepsilon$  if and only if there exists a curve within  $F_\varepsilon(f, g)$  from its lower left to its upper right endpoint, which is monotone in both coordinates. We call such a curve *feasible*.

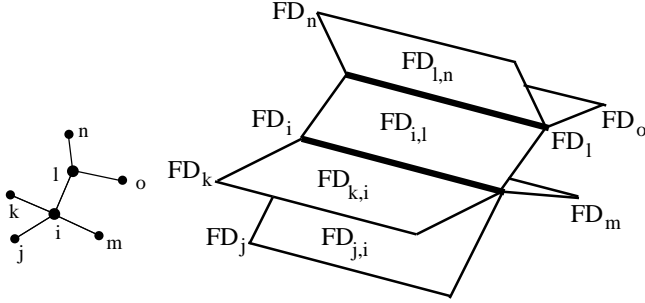
Let  $s_{i,j} : [0, 1] \rightarrow \mathbb{R}^2$  for all  $(i, j) \in E$  be the line-segment embedding of  $(i, j)$ . For every edge  $(i, j) \in E$  consider the free space diagram  $FD_{i,j}$  between  $\alpha$  and  $s_{i,j}$ , which is a subdivision of  $[0, p] \times [0, 1]$ .  $FD_{i,j}$  consists of a row of  $p$  cells, each of which corresponds to a line segment of  $\alpha$ . For a vertex  $j \in V$  let  $FD_j$  be the one-dimensional free space diagram between  $\alpha$  and  $v_j$ . Furthermore, let  $\mathbf{L}_j$  be the left endpoint and  $\mathbf{R}_j$  be the right endpoint of  $FD_j$ .

For each  $i \in V$  the free space diagrams  $FD_{i,j}$  and  $FD_{j,i}$  for all  $j \in \text{Adj}(i)$  have the one-dimensional free space diagram  $FD_i$  in common - as the bottom of  $FD_{i,j}$  and as the top of  $FD_{j,i}$ . Thus we can imagine to glue together the two-dimensional free space diagrams at the one-dimensional free space they have in common, according to the adjacency information of  $G$ . Like this we obtain a rather complex topological structure which we call the *free space surface* for  $G$  and  $\alpha$ .

\*Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Germany. {alt, rote}@inf.fu-berlin.de

<sup>†</sup>University of Arizona, Computer Science Dept., 1040 E 4th Street, Tucson, AZ 85721. {alon, carolaw}@cs.arizona.edu

<sup>1</sup>Note that this definition allows a path  $\pi$  in  $G$  to travel the same edges in  $G$  multiple times.



We apply an approach related to [2] to our more general setting: We search for a feasible path in the free space surface. This path has to start at some white left corner  $\mathbf{L}_k$  and has to end at some white right corner  $\mathbf{R}_j$ , for two vertices  $j, k \in V$ . Any path  $\pi$  in  $G$  selects a sequence of free space diagrams in the free space surface, whose concatenation yields  $FD_\varepsilon(\alpha, \pi)$ .

For every vertex  $j \in V$  let  $\mathcal{R}(j)$  be the set of all points  $u \in F_j$  for which there exists a  $k \in V$  and a path  $\pi$  from  $k$  to  $j$  in  $G$  such that there is a monotone feasible path from  $\mathbf{L}_k$  to  $u$  in  $F_\varepsilon(\alpha, \pi)$ . We call points in  $\mathcal{R}(j)$  *reachable*. We thus know that there is a path  $\pi$  in  $G$  with  $\delta_F(\alpha, \pi) \leq \varepsilon$  iff there is a vertex  $j \in V$  such that  $\mathbf{R}_j \in \mathcal{R}(j)$ .

In a preprocessing stage we first compute all one-dimensional free space diagrams  $FD_i$  for all  $i \in V$ . Further, for all  $(i, j) \in E$  we compute for each white interval  $I$  of  $FD_i$  the leftmost point and the rightmost point on  $FD_j$  which can be reached from some point in  $I$  by a monotone feasible path in  $FD_{i,j}$ .<sup>2</sup> In the dynamic programming stage we decide whether there exists a feasible monotone path in the free space surface by computing parts of  $\mathcal{R}(j)$ , for all  $j \in V$ . Conceptually we sweep all  $FD_{i,j}$  at once with a vertical sweep line from left to right. Let  $0 \leq x \leq p$  denote the position of the sweep line. For each  $i \in V$  we store a set  $C_i \subseteq \mathcal{R}(i) \subseteq F_i$  of white points, which we compute in a dynamic programming manner. Throughout the algorithm we maintain the invariant that  $C_i$  consists of all reachable points  $u \in \mathcal{R}(i) \subseteq FD_i$ , such that  $u \geq x$ , and for which the last segment of their associated feasible monotone path crosses or ends at the sweep line. We maintain a priority queue  $Q$  of the first white intervals of each  $C_j$  – these are white intervals of  $FD_i$  which are known to be reachable. The priority of an interval is its left endpoint. We process all intervals in  $Q$  and update all  $C_j$ , until we either find a  $j \in V$  such that  $\mathbf{R}_j \in C_j$ , or until  $Q$  is empty. In the latter case there is no path  $\pi$  in  $G$  with  $\delta_F(\alpha, \pi) \leq \varepsilon$ . In the first case we know that there exists such a path, and we reconstruct it using a backtracking variant.

<sup>2</sup>Similar reachability pointers have been used in [2] for attacking the case of closed curves.

Hence, our algorithm is a mixture of a sweep (since we are sweeping with a sweep line), dynamic programming (on the  $C_i$  we incrementally build up), and Dijkstra's algorithm for shortest paths (since we are computing paths using a priority queue to augment the path in a similar fashion to Dijkstra). In the more detailed version of this paper [1] we consider some variants of the original problem, one of which is a time-space tradeoff which also applies to the computation of the Fréchet distance for two polygonal curves.

### 3 Graph-to-Graph Distance

In this section we generalize the Fréchet distance to two embedded, connected graphs  $H = (V_H, E_H)$  and  $G = (V_G, E_G)$  with straight edges. Identifying  $H$  and  $G$  with the points lying on their edges we call a mapping  $f : [0, 1] \rightarrow H$  which is continuous and surjective, a *traversal* of  $H$ . If  $f$  is not surjective we call it a *partial traversal*. The *traversal distance* from  $H$  to  $G$  is defined as

$$\delta_T(H, G) = \inf_{f, g} \max_{t \in [0, 1]} \|f(t) - g(t)\|$$

where  $f$  ranges over all traversals of  $H$  and  $g$  over all partial traversals of  $G$ . If  $H$  and  $G$  are polygonal chains this definition corresponds to the weak Fréchet-distance, see [2]. Observe that the traversal distance is not a generalization of the Fréchet distance between a curve and a graph as defined in Section 2.

We show that for given  $\varepsilon > 0$  we can decide whether  $\delta_T(H, G) \leq \varepsilon$  in time  $O(pq \log pq)$ , where  $p = |E_G|$  and  $q = |E_H|$ . Using parametric search the traversal distance can be computed in time  $O(pq \log^2 pq)$ .

For given  $\varepsilon \geq 0$  we consider the free space surface of  $H$  and  $G$ . For edges  $e \in E_H$  and  $f \in E_G$ , if a feasible path  $\pi$  traverses some cell  $C_{e,f}$  of the surface, then let  $I_{\pi, e, f}$  be the set of those points on  $e$  that are traversed by the corresponding motion on the graphs.  $e$  is called *satisfied* by  $\pi$  if the union  $\bigcup_{f \in E_G} I_{\pi, e, f} = e$ . It means that all points of  $e$  are eventually traversed by the motion on the graphs corresponding to  $\pi$ . Therefore we have to search for a feasible path  $\pi$  that satisfies all edges  $f \in E_H$ . We can construct such a path  $\pi$  by traversing the free space surface from edge to edge, and making sure at each edge to visit the leftmost and the rightmost point of the freespace; see [1] for details.

### References

- [1] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. In *accepted to 14th ACM-SIAM Sympos. Discrete Algorithms*, 2003.
- [2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.