Pattern Matching for Sets of Segments*

Alon Efrat† Piotr Indyk‡ Suresh Venkatasubramanian§

October 18, 2000

Abstract

In this paper we present algorithms for a number of problems in geometric pattern matching where the input consist of a collections of segments in the plane. Our work consists of two main parts. In the first, we address problems and measures that relate to collections of orthogonal line segments in the plane. Such collections arise naturally from problems in mapping buildings and robot exploration.

We propose a new measure of segment similarity called a *coverage measure*, and present efficient algorithms for maximising this measure between sets of axis-parallel segments under translations. Our algorithms run in time $O(n^3 polylogn)$ in the general case, and run in time $O(n^2 polylogn)$ for the case when all segments are horizontal. In addition, we show that when restricted to translations that are only vertical, the Hausdorff distance between two sets of horizontal segments can be computed in time roughly $O(n^{3/2} polylog\, n)$. These algorithms form significant improvements over the general algorithm of Chew et al. that takes time $O(n^4 \log^2 n)$.

In the second part of this paper we address the problem of matching polygonal chains. We study the well known Fréchet distance, and present the first algorithm for computing the Fréchet distance under general translations. Our methods also yield algorithms for computing a generalization of the Fréchet distance, and we also present a simple approximation algorithm for the Fréchet distance that runs in time $O(n^2 polylogn)$.

1 Introduction

Traditionally, geometric pattern matching employs as a measure of similarity the Hausdorff distance h(A,B) defined as $h(A, B) = \max_{p\in A} \min_{q\in B} d(p, q)$ for two point sets A and B. However, when the patterns to be matched are line segments or curves (instead of points), this measure is less than satisfactory. It has been observed that measures like the Hausdorff measure that are defined on point sets are ill-suited as measures of curve similarity, because they destroy the continuity inherent in continuous curves.

This paper addresses problems in geometric pattern matching where the inputs are sets of line segments. Our work consists of two main parts; in the first part we consider the problem of matching (under translation) segments that are axis-parallel (i.e either horizontal or vertical), and in the second we consider the problem of matching polygonal chains under translation. We study two different measures in this context; the first is a novel measure called the *coverage measure*, which captures the similarity between orthogonal segments that may partially overlap with one another. The other is the well known Fréchet distance, first proposed by Maurice Fréchet in 1906 as a measure of distance between distributions, which has often been referred to as a natural measure of curve similarity [3, 6, 23]. We discuss each measure in detail below.

1.1 Mapping and orthogonality The motivation for considering instances of pattern matching where the input line segments are orthogonal comes from the domain of *mapping*, in which a robot is required to map the underlying structure of a building by moving inside the building, and "sensing" or "studying" its environment.

In one such mapping project at the Stanford Robotics Laboratory[1] the robot is equipped with a laser range finder which supplies the distance from the robot to its nearest neighbor in a dense set of directions in a horizontal plane. We call the resulting distances map a *picture*. Figure 1.1(a) shows the robot used at Stanford for this purpose.

During the mapping process, the robot must merge into a single map the series of pictures that it captures from different locations in the building. Since the dead reckoning of the robot is not very accurate, it cannot rely solely on its motion to decide how the pictures are placed together. Thus, we need a matching process that can align (by using overlapping regions) the different pictures taken from different points of the same environment. In addition, we need to determine whether the robot has returned to a point already

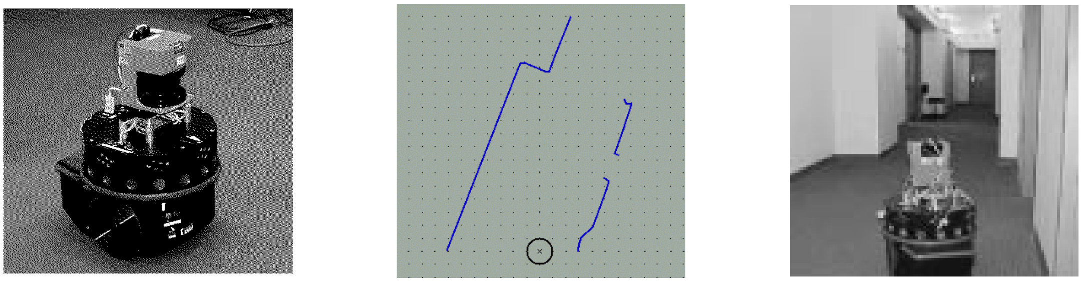
*A full version of this paper can be found at http://graphics.stanford.edu/~alon/papers/seg_match.ps.gz

†Computer Science Department, Stanford University, **Email:** *alon@cs.stanford.edu*. Work supported in part by a Rothschild Fellowship and by DARPA contract DAAE07-98-C-L027

‡Computer Science Department, Stanford University. **Email:** *indyk@cs.stanford.edu*.

§AT&T Labs – Research. **Email:** *suresh@research.att.com*.

[1] The interested reader can find more information at the URL `http://underdog.stanford.edu`



(a) (b) (c)

Figure 1: Left: The robot, and the laser range finder installed on it. Middle: Typical "picture" obtained by the robot of a corridor (after segmentation). Right: The corridor itself

visited. We make the reasonable assumption that buildings walls are almost always either orthogonal or parallel to each other, and that these walls are frequently by far the most dominant objects in the pictured. This is especially significant in the case that the robot is inside a corridor, where there is a lack of detail needed for good registration. In some cases most of the picture consists merely of two walls with a small number of other segments. See Figure 1.1(b),(c) for a typical picture and the real region that the laser range finder senses.

This application suggests the study of matching sets of horizontal and vertical segments. Observe that we may restrict ourself to alignments under *translation*, as it is easy to find the correct rotation for matching sets of orthogonal segments. Formally, let $A = \{a_1 \dots a_n\}$ and $B = \{b_1, \dots b_n\}$ be two sets of orthogonal line segments in the plane, and let ε be a given parameter. A point p of a horizontal (resp. vertical) segment $a \in A$ is *covered* if there is a point of a horizontal (resp. vertical) segment $b \in B$ whose distance from p is $\leq \varepsilon$, where the distance is measured using the ℓ_∞ norm. Let $w(A, B)$ denote the collection of sub-segments of A consisting of covered points. Let $Cov(A, B)$ be the total length of the segments of $w(A, B)$. The *maximum coverage problem* is to find a translation t^* in the translation plane (TP) that maximizes $Cov(t) = Cov(t + A, B)$. To the best of our knowledge, this measure is novel.

The coverage measure is especially relevant in the case of long segments e.g. inside a corridor, when we might be interested in partially matching portions of long segments to portions of other segments.

Our Results In Section 2 we present an algorithm that solves the Coverage problem between sets of axis-parallel segments in time $O(n^3 \log^2 n)$ and the Coverage problem between horizontal segments in time $O(n^2 \log n)$. Note that the known algorithms for matching arbitrary sets of line segments are much slower. For example, the best known algorithm for finding a translation that minimizes the Hausdorff Distance between two sets of n segments in the plane runs in time $O(n^4 \log^2 n)$ [2, 9]. We also show that the that the combinatorial complexity of the Hausdorff matching between segments is $\Omega(n^4)$, even if all segments are *horizontal*. This strengthens the bounds shown by Rucklidge [21], and demonstrates that our algorithms, much like the algorithms of [11, 10], are able to avoid having to examine each cell of $\mathcal{F}$ individually. Note that all our results extend to the case when segments are *weighted* and the coverage is now a weighted sum of interval lengths.

In Section 3 we consider the related problem of matching horizontal segments under vertical translations (under the Hausdorff measure). It has been observed that if horizontal translations are allowed, then this problem is 3SUM-hard [5], indicating that finding a sub-quadratic algorithm may be hard. However, we present an approximation algorithm running in time $O(n^{3/2} \max\{\log^c M, \log^c n, 1/\varepsilon^c)\})$, for some fixed constant c, which is sub-quadratic in most cases. Here, M denotes the ratio of the diameter to the closest pair of points in the sets of segments (where pairs of points must lie on different segments).

1.2 The Fréchet distance

In the second part of the paper, we consider measures for matching polygonal chains under the Fréchet distance. Let us define a curve as a continuous mapping $P : [a, a'] \to \mathbb{R}^2$. The Fréchet distance between two curves P and Q, $d_F(P, Q)$ is defined as:

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

where α, β range over continuous increasing functions from $[0, 1] \to [a, a']$ and $[0, 1] \to [b, b']$ respectively.

Alt and Godau proposed the first algorithm for computing the Fréchet distance between two polygonal chains (with no transformations). Their method is elegant and simple, and runs in time $O(pq)$, where p

and q are the number of segments in the two polygonal chains. In his Ph.D thesis [14]. Michael Godau presents an extensive study of the complexity of computing the Fréchet distance. He shows that computing the Fréchet distance between two simplicial objects is NP-hard, for any dimension $d \geq 3$.

Although the Fréchet distance is a natural measure for curve similarity, its applicability has been limited by the fact that no algorithms exist to minimise the Fréchet distance between curves under various transformation groups. Prior to our work, the only result on computing the Fréchet distance under transformations was presented by Venkatasubramanian [22]. He computes $\min_{t \in TP_x} d_F(P, Q+t) \leq \varepsilon$, where TP_x is the set of translations along a fixed direction, in time $O(n^5 polylog n)$ (where $n = p + q$). In fact, our methods can be viewed as a generalization of his methods and can be used to solve his problem in the same time bound.

Our Results In Section 4 we present the first algorithm for computing the Fréchet distance between two polygonal chains minimized under translations[2]. The algorithm is based on a reduction to a dynamic graph reachability problem; its running time is $O(n^{10} polylog n)$.

If we drop the restriction that the functions α, β must be increasing, we obtain a measure that we call the *weak* Fréchet distance, denoted by $d_{\tilde{F}}$. Our methods can be used to decide whether $\min_{t \in TP} d_{\tilde{F}}(P, Q+t) \leq \varepsilon$; in this case, the underlying graph is undirected, yielding an algorithm that runs in time $O(n^4 polylog n)$.

With the exact algorithms being rather expensive, it is natural to ask whether approximations can be obtained efficiently. A simple observation shows that we can obtain an (ε, β)-approximation to the Fréchet distance under translations in time $O(n^2 poly(\log n, 1/\beta))$.

2 Algorithms for maximum Coverage

Let $A = \{a_1 \dots a_n\}$ and $B = \{b_1, \dots b_n\}$ be two sets of axis-parallel line segments in the plane, and let ε be a given parameter. Recall the coverage measure $Cov(A, B)$ as defined in the introduction.

2.1 Coverage with axis-parallel segments We first consider the case that the sets A and B consists of both horizontal and vertical segments. Let A^h (resp. B^h) be a set of n horizontal segments and let A^v (resp. B^v) be a set of n vertical segments. Let ε be a given parameter. Let $A = A^h \cup A^v$ and let $B = B^h \cup B^v$. Let $Cov(t+A, B) = Cov(t+A^h, B^h) + Cov(t+A^v, B^v)$. Let $\mathcal{S} = \{s_1 \dots s_m\}$ be a set of non-vertical segments in $\mathbb{R}^2$. For each segment $s_i \in \mathcal{S}$ we define the functions $s_i(x) \to \mathbb{R}$ as follows: For every $x \in \mathbb{R}$, $s_i(x)$ is the y-coordinate of the intersection point of s and the vertical line passing through x, if such an intersection point exists. We set $s_i(x)$ to be 0 otherwise. Let $sum_{\mathcal{S}}(x) = \Sigma_{i=1}^{m} s_i(x)$, and let $\max(sum_{\mathcal{S}}(\cdot)) = \max_{x \in \mathbb{R}} sum_{\mathcal{S}}(x)$. Furthermore, let $T = T(\tau)$ be a subset of $\mathcal{S}$ consisting of horizontal segments that can move vertically at constant speed i.e the y-coordinates of the endpoints of each $s_i \in T$ are given by $y = a_i \tau + b_i$.

Lemma 2.1. (Proof in Appendix A) *Given a set of non-vertical segments S with a subset T of horizontal moving segments, we can maintain $\max(sum_{\mathcal{S}}(\cdot))$ under segment insertions or deletions in amortized time $O(\sqrt{|\mathcal{S}|})$ per operation. In addition, we can maintain $\max(sum_{\mathcal{S}}(\cdot))$ under a* time-decreasing *step ($\tau \leftarrow \tau - \Delta$) in $O(1)$ time.*

Theorem 2.1. *We can find a translation t that maximizes $Cov(t + A, B)$ in time $O(n^3 \log^2 n)$, where $n = |A| + |B|$*

Proof. The proposed algorithm is a line-sweep algorithm, with the sweep line moving from top to bottom. For a segment $b_i \in B$ let b_i^+ denote the rectangle consisting of all points whose $\ell_i nfty$ distance from b_i is at most ε. Let B^+ denote the union $\bigcup_{i=1}^{n} b_i^+$. Note that any two rectangles b_i^+, b_j^+ intersect in at most two points, so by [18] the complexity of the boundary of B^+ is $O(n)$. Consider $E = \{p_1 \dots p_{2n}\}$, the set of the $2n$ endpoints of the segments of A. Define the *layer* $L_i = B^+ - p_i$, which is the region in the TP of all translations t that shift p_i into B^+ i.e $t + p_i \in B^+$. Let $\mathcal{B}^h$ (resp. $\mathcal{B}^v$) be the collection of layers created by the horizontal (resp. vertical) segments of A. As the line sweep traverses the translation plane from top to bottom, we encounter events where ℓ intersects a horizontal boundary segment of either $\mathcal{B}^h$ or $\mathcal{B}^v$.

Horizontal Boundaries Of $\mathcal{B}^h$: Let $Cov(x) : \mathbb{R} \to \mathbb{R}$ be the value of $Cov(t + A^h, B^h)$, where t is the point on ℓ vertically above x. Consider the contribution to $Cov(t + A^h, B^h)$ from the interaction between the segments $a \in A^h, b \in B^h$. This contribution to the function consists of a piecewise linear function, consists of five segments: It is zero for value of x which are very far from the regions of interaction between a and b, it is a constant that equals the minimum of the length of a and b when x is near the region of intersection, and it consists of two segments of slopes are 1 and -1, connecting these segments. These segments exist for all instances of the line sweep where its horizontal distance to the boundary of the rectangle of B_j corrsponds to a_i is $\leq \varepsilon$. There are $O(n^2)$ update operations, and

[2] Actually, we solve the decision version of the problem: For a given ε, determine whether $\min_{t \in TP} d_F(P, Q + t) \leq \varepsilon$.

each update can be processed in $O(n \log^2 n)$ time from Lemma 2.1.

Horizontal Boundaries Of $\mathcal{B}^v$. For two vertical segments $a_i \in A, b_j \in B$, let $\mathcal{T}_{ij}$ be the set of translations for which the horizontal distance from a_i to b_j is at most ε. Assume w.l.o.g that $|a_i| > |b_j|$. Let UP_{ij} denote all translations t for which the upper endpoint of a_i is covered by $t + b_j$, (i.e. its distance from some point of $t + b_j$ is at most ε) but the lower endpoint of a_i is not covered. Similarly, let $DOWN_{ij}$ denote all translations t for which the lower endpoint of a_i is covered by $t + b_j$, but the upper endpoint of a_i is not covered and let MID_{ij} denote all translations t for which both endpoints of a_i are covered.

Thus $Cov(t+a_i, b_j)$ is zero when $t \notin UP_{ij} \cup MID_{ij} \cup DOWN_{ij}$, $Cov(t+a_i, b_j)$ is a constant when $t \in MID_{ij}$, and it is a decreasing (resp. increasing) linear function that depends only on the y-coordinate of t when $t \in UP_{ij}$ (resp. $t \in DOWN_{ij}$). Therefore, we can represent the contribution of a_i and b_j to $Cov(a_i, t + b_j)$ by a horizontal segment $u_{ij}(\tau)$ of length 2ε that starts at $y = 0$ and moves upwards with constant velocity as the line sweep intersects $DOWN_{ij}$. It remains constant at a maximum height as ℓ passes thru MID_{ij} and moves downwards to 0 as ℓ passes through UP_{ij}.

This suggests the following operations on the data structures, using Lemma 2.1. Consider the rectangle b_j of the vertical decompostion of L_i, (which corresponds to translations for which a_i is in the vicinity of b_j). We divide b_j into three rectangles $b_{ij,UP}$, $b_{ij,MID}$ and $b_{ij,DOWN}$, which are the intersection regions of b_j and UP_{ij}, MID_{ij} and $DOWN_{ij}$. As the linesweep hits the upper boundary of a rectangle $b_{ij,UP}$, we insert the moving segment $u_{ij}(\tau)$ into $T(\tau)$. When ℓ reaches the upper boundary of $b_{ij,MID}$ we insert a horizontal moving segment $u'_{ij}(\tau)$ chosen such that that $u_{ij}(\tau) + u'_{ij}(\tau)$ equals Max_{ij}. This is done in order to avoid deleting or changing $u_{ij}(\tau)$. When ℓ reaches the upper boundary of $b_{ij,DOWN}$, we insert into $T(\tau)$ the segment $u''_{ij}(\tau)$ which is also decreases linearly as τ decreases, and is choosen such that $u(\tau)_{ij} + u'(\tau)_{ij} + u''_{ij}(\tau)$ equals $Cov(a_i, t + b_j)$ at this translation t, $t \in DOWN_{ij}$. Overall, we add three (moving) segments for each rectangles of L_i, and since the number of these rectangles is $O(n^2)$, it follows that the overall running time of the algorithm is $O(n^3 \log^2 n)$. Note also that at each update, we decrease the current "time" τ; this is a constant time operation per update.

2.2 Maximum coverage for horizontal segments

This is a line-sweep algorithm reminiscient of the Chew-Kedem [11] and Chew *et al.* [10] algorithms for computing the similarity between point-sets in the plane, under the ℓ_∞ norm. As in Section 2.1, we define layers L_i for each endpoint p_i of segments in A. Construct a horizontal decomposition of L_i, breaking it into a collection $\mathcal{B}_i = \{\beta_{i1}\, \beta_{i2} \ldots\}$ of $O(n)$ interior-disjoint rectangles.

Let $\mathcal{S}$ denote the set of *vertical* segments on the boundaries of the layers L_i (for $i = 1 \ldots 2n$). Let $\mathcal{T}$ be a segment tree constructed on the segments of $\mathcal{S}$. During the algorithm we sweep the translation plane TP using a vertical sweep line ℓ. Once ℓ meets a segment $e \in \mathcal{S}$, we insert e into $\mathcal{T}$. No segment is deleted.

Let μ be a node of $\mathcal{T}$. Let I_μ be the horizontal infinite strip whose y-span is the interval of μ and let $S_\mu \subseteq \mathcal{S}$ denote the segments on or to the left of ℓ which correspond to μ i.e. the segments whose y-span contains I_μ but not $I_{father(\mu)}$. We maintain the following fields with each node μ of $\mathcal{T}$. All of these are set to zero at the beginning of the algorithm:

- $last_\mu$: the last x event at which a segment was inserted into S_μ.
- Pos_μ: the number of segments in S_μ resulting from the right (resp. left) endpoint of a segment $a \in A$ meeting a left (resp. right) vertical segment of some layer. We call such an event a *Positive event*
- Neg_μ: the number of segments in S_μ resulting from the left (resp. right) endpoint of a segment $a \in A$ meeting a left (resp. right) vertical segment of some layer. We call such an event a *Negative event.*
- w_μ: The maximal coverage obtained by segments stored at S_μ itself.
- Cov_μ: The maximal coverage obtained by events of "segments" stored at the descendants nodes of μ *including* μ itself.

Performing an insertion: Once ℓ hits a new segment $s \in \mathcal{S}$, we first find all nodes μ for which $s \in S_\mu$ as in a standard segment tree. Next, for each such node μ, we increase either Pos_μ or Neg_μ by one, according to the type of s. Next we add to w_μ the quantity $(Pos_\mu - Neg_\mu)d$, where d is the horizontal distance from the previous insertion event into S_μ, (stored at $last_\mu$) till the current position of the ℓ. We update Cov_μ for each μ in bottom-up fashion, namely: $Cov_\mu = \max\{Cov_{left(\mu)}, Cov_{right(\mu)}\} + w_\mu$. Each insertion can be performed in $O(\log n)$ time, so the overall running time of the algorithm is $O(n^2 \log n)$. When the algorithm terminates, we report a translation t_{output} that corresponds to the maximum value of $Cov_{root(\mathcal{T})}$ obtained by the algorithm.

Remark: The algorithm can easily be modified to handle the *weighted* case, where each segment has a weight, and the contribution to the coverage of a segment is the length of the covered portions times the weight of the segment. This is useful when some segments are more important than others.

THEOREM 2.2. *Let $t^* \in TP$ be the leftmost translation that maximises $Cov(t+A, B)$. Then when the line-sweep passes through t^*, $(t^* + A, B) = Cov_{root(\mathcal{T})}$.*

Proof. We first make the following observation. Consider the infinite horizontal ray r emerging from t^* to the left. Let $x_1 \ldots x_l$ be the x-coordinates of the events encountered along this ray, ordered from left to right. Let Pos_i (resp. Neg_i) be defined as the number of positive intersection points of r to the left of x_i, with boundaries of layers that corresponds to positive (resp. negative) events, as described above. Clearly

(2.) $Cov(t^*A, B) = \Sigma_{i=1}^{l}(Pos_i - Neg_i)(x_i - x_{i-1})$

On the other hand, the sum of the right hand side of Equation 2. equals the sum of the fields w_μ, taken over all nodes μ of the segment tree on the path from the root to the leaf node containing t^*, at the instance when the line sweep intersects t^*. This follows from the fact that each event x_i is also an event in one of the nodes μ along this path. Therefore this sum equals $Cov_{root(\mathcal{T})}$, since the sum of the fields w_μ along every path from the root to a leaf equals $Cov(t+A, B)$ at any translation t stored at that leaf, and t^* by our assumption is maximal.

2.3 A lower bound Rucklidge [21] showed that given a parameter ε and two families A and B of segments in the plane, the combinatorial complexity of the regions in the translations plane (TP) of all translations t for which $h(t + A, B) \leq \varepsilon$ is in the worst case $\Omega(n^4)$, where $h(A, B)$ is the one way Hausdorff distance from A to B. It turns out that the $\Omega(n^4)$ bound holds even in the case that all segments are *horizontal* (the proof is deferred to a full version of the paper). This yields:

THEOREM 2.3. *The region of all translations t for which $Cov(A, t + B)$ is maximal has combinatorial complexity $\Omega(n^4)$.*

3 Matching Horizontal Segments Under Vertical Translation

In this section we describe a sub-quadratic algorithm for the Hausdorff matching between sets A and B of horizontal segment, when translations are restricted to the vertical direction.

Let $\rho^* = \min_t h(t + A, b)$ where t varies over all vertical translations, and $h(\cdot,\cdot)$ is the one-way Hausdorff distance. Let M denote the ratio of the diameter to the closest pair of segments in $A \cup B$. Further, let $[M]$ denote the set of integers $\{1 \ldots M\}$.

THEOREM 3.1. *Let A and B be two set of horizontal segments, and let $\varepsilon < 1$ be a given parameter. Then we can find a vertical translation t for which $h(t + A, B) \leq (1 + \varepsilon)\rho^*$ in time $O(n^{3/2} poly(\log M, \log n, 1/\varepsilon))$.*

We first relate our problem to a problem in string matching:

DEFINITION 3.1. **(Interval matching):** *given two sequences $t = t[1] \ldots t[n]$ and $p = p[1] \ldots p[m]$, such that $p[i] \in [M]$ and $t[i]$ is a union of disjoint intervals $\{a_i^1 \ldots b_i^1\} \cup \{a_i^2 \ldots b_i^2\} \ldots$ with endpoints in $[M]$, find all translations j such that $p[j] \in t[i + j]$ for all i. The* size *of the input to this problem is defined as $s = sum_i |t[i]| + m$.*

We also define the *sparse* interval matching problem, in which both $p[i]$ and $t[i]$ are allowed to be equal to a special empty set symbol $\emptyset$, which matches any other symbol or set. The size s in this case is defined as $sum_i |t[i]|$ plus the number of non-empty pattern symbols. Using standard discretization techniques [7, 17], we can show that the problem of $(1 + \varepsilon)$-approximating the minimum Hausdorff distance between two sets of n horizontal intervals with coordinates from $[M]$ under vertical motion can be reduced to solving an instance of sparse interval matching with size $s = O(n)$.

Having thus reduced the problem of matching segments to an instance of sparse interval matching, we show that:

- The (non-sparse) interval matching problem can be solved in time $O(s^{3/2} polylog s)$.
- The same holds even if the pattern is allowed to consists of unions of intervals.
- The sparse interval matching problem of size s can be reduced to $O(\log M)$ non-sparse interval matching problems, each of size $s' = O(s\ polylog s)$.

These three observations yield the proof of Theorem 3.1. In the remainder of this section, we sketch proofs of the above observations.

The interval matching problem. Our method follows the approach of [1, 20] and [4]; therefore, we sketch the algorithm here, omitting detailed proofs of correctness.

Firstly, we observe that the universe size M can be reduced to $O(s)$, by sorting the coordinates of the points/interval endpoints and replacing them by their rank, which clearly does not change the solution. Then we reduce the universe further to $M' = O(\sqrt{s})$ by merging some coordinates, i.e. replacing several coordinates

$x_1 \ldots x_k$ by one symbol $\{x_1 \ldots x_k\}$, in the following way. Each coordinate (say x) which occurs more than $\sqrt{s}$ times in t or p is replaced by a singleton set $\{x\}$ (clearly, there are at most $O(\sqrt{s})$ such coordinates). By removing those coordinates, the interval $[M]$ is split into at most $O(\sqrt{s})$ intervals. We partition each interval into smaller intervals, such that the sum of all occurrences of all coordinates in each interval is $O(\sqrt{s})$. Clearly, the total number of intervals obtained in this way is $\sqrt{s}$. Finally, we replace all coordinates in an interval by one (new) symbol from $[M']$ where $M' = O(\sqrt{s})$. By replacing each coordinate x in p and t by the number of a set to which x belongs, we obtain a "coarse representation" of the input, which we denote by p' and t'.

In the next phase, we solve the interval matching problem for p' and t' in time $\tilde{O}(nM')$ using a Fast Fourier Transform-based algorithm (see the above references for details). Thus we exclude all translations j for which there is i such that $p[i]$ is not included in the *approximation* of $t[i+j]$. However, it could be still true that $p[i] \notin t[i+j]$ while $p'[i] \in t'[i+j]$. Fortunately, the total number of such pairs (i,j) is bounded by the number of new symbols (i.e. M') times the number of pairs of all occurrences of any two (old) symbols corresponding to a given new symbol (i.e. $O(\sqrt{s}^2)$). This gives a total of $O(s^{3/2})$ pairs to check. Each check can be done in $O(\log n)$ time, since we can build a data structure over each set of intervals $t[i]$ which enables fast membership query. Therefore, the total time need for this phase of the algorithm is $\tilde{O}(s^{3/2})$, which is also a bound for the total running time.

The generalization to the case where $p[i]$ is a union of intervals follows in essentially the same way, so we skip the description here.

The sparse-to-non-sparse reduction. The idea here is to map the input sequences to sequences of length P, where P is a random prime number from the range $\{c_1 s \log M \ldots c_2 s \log M\}$ for some constants c_1, c_2. The new sequences p' and t' are defined as $p'[i] = \cup_{i': i' mod P = i}\, p[i']$ and $t'[i] = \cup_{i': i' mod P = i}\, t[i']$. It can be shown (using similar ideas as in [7]) that if a translation j *does not* result in a match between p and t, it will remain a mismatch between p' and t' with constant probability. Therefore, all possible mismatches will be detected with high probability by performing $O(\log M)$ mappings modulo a random prime.

4 Computing The Fréchet Distance Under Translation

In this section, we present algorithms for computing the Fréchet distance between two polygonal chains. Recall that the Fréchet distance between two curves P and Q, $d_F(P,Q)$ is defined as:

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

where α, β range over continuous increasing functions from $[0,1] \to [a,a']$ and $[0,1] \to [b,b']$ respectively.

Dropping the restriction that α, β are increasing functions yields a measure we call the *weak* Fréchet distance, denoted by $d_{\tilde{F}}$. It can be easily seen that both d_F and $d_{\tilde{F}}$ are metrics.

Let the curves P and Q be length-parameterized by r, s. In other words, $P = P(r), Q = Q(s)$, where $0 \le r, s \le 1$. For any fixed ε, let $F_\varepsilon(P,Q)$, the *free space*, be defined as

$$F_\varepsilon(P,Q) = \{(r,s) \mid \|P(r) - Q(s)\| \le \varepsilon\}$$

where $\|\cdot\|$ is the underlying norm[3]. The free space captures the space of parameterizations that achieve a Fréchet distance of at most ε. In the sequel we will denote the free space by F_ε when the parameters P and Q are clear from the context.

Let a polygonal chain $P : [0,n] \to \mathbb{R}^2$ be a curve such that for each $i \in \{0,\ldots,n-1\}$, $P_{|[i,i+1]}$ is affine i.e $P(i+\lambda) = (1-\lambda)P(i) + \lambda P(i+1), 0 \le \lambda \le 1$. For such a chain P, denote $|P| = n$. Let P_i denote the segment $P_{|[i,i+1]}$. For two polygonal chains P, Q where $|P| = p, |Q| = q$, and a fixed ε, the free space $F_\varepsilon \subseteq [0,p] \times [0,q]$ is given (as before) by:

$$F_\varepsilon(P,Q) = \{(r,s) \mid \|P(r) - Q(s)\| \le \varepsilon\}$$

Let $F_\varepsilon^{ij} = F_\varepsilon \cap (P_i \times Q_j)$. Observe that $F_\varepsilon^{ij} = F_\varepsilon(P_i, Q_j)$. It can be seen [3] that F_ε^{ij} is the affine inverse of a unit ball with respect to the underlying norm. Consequently, F_ε^{ij} is convex.

Consider the points of intersection of a single cell $C_{ij} = F_\varepsilon^{ij}$ with the line segment from (i,j) to $(i,j+1)$. Since C_{ij} is convex, there are at most two such points, which we denote as a_{ij}, b_{ij}, where a_{ij} is below b_{ij}. Similarly, let c_{ij} and d_{ij} be the points of intersection of C_{ij} with the line segment from (i,j) to $(i+1,j)$, where c_{ij} is to the left of d_{ij}. We define an order on the points as follows: For any two points $p_1 = (x_1,y_1), p_2 = (x_2,y_2)$, $p_1 \le p_2$ if $x_1 \le x_2$ and $y_1 \le y_2$. Let an (x,y)-monotone path be a path that is increasing in both x and y coordinates. Alt and Godau [3] observed that the existence of a (x,y)-monotone path in F_ε from $(0,0)$ to (p,q) is a necessary and sufficient condition for $d_F(P,Q) \le \varepsilon$. A similar property holds for $d_{\tilde{F}}$; namely, the existence of *any* non-self-intersecting path in F_ε from $(0,0)$ to (p,q) implies that $d_{\tilde{F}}(P,Q) \le \varepsilon$.

[3] In this section, we will consider the l_2 norm unless otherwise specified.

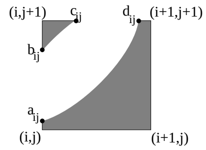


Figure 2: A single cell in the free space

Denote the property "(p, q) is reachable from $(0, 0)$" as property $\mathcal{P}$ (similarly define $\tilde{\mathcal{P}}$).

We wish to solve the decision problem for the Fréchet distance between P and Q minimised over translations i.e given ε, check whether $\min_t d_F(P, Q + t) \le \varepsilon$.

The configuration space A *critical event* is one that can change the truth value of $\mathcal{P}$. Each such event is one of the following two types: (1) The intersection points $a_{ij}, b_{ij}, c_{ij}, d_{ij}$ appear (or disappear). (2) For two cells C_{ij} and $C_{kj}, k > i$, a_{ij} and a_{kj} (or b_{kj}) change their relative vertical ordering. Analogously, for two cells C_{ij} and $C_{ik}, k > j$ the points c_{ij} and c_{ik} (or d_{ik}) change their relative horizontal ordering.

Type 2 events correspond to the creation or deletion of *tunnels*. For any point r in the space $[0, p] \times [j, j+1]$, let k be the *rightmost* interval such that r projected onto the interval $[a_{kj}, b_{kj}]$ lies between the endpoints of the interval. We define $rt\ (r) = k$. For any point $r \in [i, i+1] \times [0, q]$, let k be the *topmost* interval such that r projected onto the interval $[c_{ik}, d_{ik}]$ lies between the endpoints of the interval. We define[4] $ut(r) = k$.

As Q translates, each of the $x_{ij}, x \in \{a, b, c, d\}$ can be represented as a function $x_{ij}(t) : \mathbb{R}^2 \to [0, 1]$.

PROPOSITION 4.1. *For a point x_{ij}, the function $x_{ij}(t)$ is a second degree polynomial in the coordinates of t.*

From free space to a graph Our algorithm for computing $d_F(P, Q)$ is based on a reduction of the problem to a directed graph reachability problem. Intuitively, we can think of a monotone path in the free space as a path in a directed graph (actually a DAG). The advantage of this approach is that we can exploit known methods for maintaining graph properties dynamically in an efficient manner. Thus, as we traverse the space of translations, we need not recompute the free space at each critical event.

Let $V = \bigcup_{i,j} \{v^a_{ij}, v^b_{ij}, v^c_{ij}, v^d_{ij}\}$ and $T = \bigcup_{i,j,i<k\le p} \{t^a_{ijk}, t^b_{ijk}\} \cup \bigcup_{i,j,j<k\le q} \{t^c_{ijk}, t^d_{ijk}\}$ where $0 \le i \le p$ and $0 \le j \le q$. The vertices in $V \cup T$ are associated with points of the free space. More precisely, vertex v^x_{ij} is associated with the point x_{ij} (where x is one of $\{a, b, c, d\}$). Vertex t^x_{ijk} is associated with the projection of point x_{ij} onto the interval $[a_{kj}, b_{kj}]$ ($x \in \{a, b\}$), and vertex t^y_{ijk} is associated with the projection of point y_{ij} onto the interval $[c_{ik}, d_{ik}]$ ($y \in \{c, d\}$). We define $f(v) = p$, where p is the point associated with vertex v.

Let $V^1_{ij} = \{v^a_{ij}, v^b_{ij}\} \cup \bigcup_{l<i\le rt(a_{lj})} t^a_{lji} \cup \bigcup_{l<i\le rt(b_{lj})} t^b_{lji}$ and $V^2_{ij} = \{v^c_{ij}, v^d_{ij}\} \cup \bigcup_{l<j\le ut(c_{il})} t^c_{ilj} \cup \bigcup_{l<j\le ut(d_{lj})} t^d_{ilj}$. V^1_{ij} denotes the set of vertices associated with points on the line segment from (i, j) to $(i, j+1)$. Similarly, V^2_{ij} denotes the set of vertices associated with points on the line segment from (i, j) to $(i+1, j)$. In addition, V^1_{ij} and V^2_{ij} contain vertices associated with points whose *tunnels* cross the cell C_{ij}.

We now describe the construction of the edge set for each (i, j). Firstly, set $E^1_{ij} = \{(v, v^b_{ij}) \mid v \in V^1_{ij}\}$ and set $E^2_{ij} = \{(v, v^d_{ij}) \mid v \in V^2_{ij}\}$ For each $v \in V^1_{ij}$, let $n(v) = \arg\min_{v' \in V^1_{i+1,j}, f(v') \ge v} f(v')$ Similarly, for each $v \in V^2_{ij}$, let $n(v)$ denote the vertex in $V^2_{i,j+1}$ having the same property. Let $E^3_{ij} = \{(v, n(v)) \mid v \in V^1_{ij} \cup V^2_{ij}\}$. Finally, set $E^4_{ij} = \{(v^b_{ij}, v^c_{i,j+1}), (v^d_{ij}, v^a_{i+1,j})\}$. Now, we set $E_{ij} = E^1_{ij} \cup E^2_{ij} \cup E^3_{ij} \cup E^4_{ij}$.

Let $E = \bigcup_{i,j} E_{ij}$. This yields the directed graph $G = (V \cup T, E)$. Note that $|V \cup T| = O(pq(p+q))$ and $|E| = O(pq(p+q))$. Also, it is easy to see that for any edge $(u, v) \in E$, the straight line from $f(u)$ to $f(v)$ is an (x, y)-monotone path. We first note that reachability in the graph G is equivalent to path construction in F_ε.

THEOREM 4.1. *An (x, y)-monotone path from $(0, 0)$ to (p, q) exists in F_ε iff v^b_{pq} is reachable from v^a_{00} and $f(v^a_{00}) = (0, 0), f(v^b_{pq}) = (p, q)$.*

For every edge $e \in E$, let $\gamma(e) \subseteq \mathbb{R}^2$ be the set of translations t such that in the graph G constructed from the free space $F_\varepsilon(P, Q + t)$, the edge e is present. Let Γ be the arrangement of all the $\gamma(e)$. We first establish a bound on the complexity of Γ.

The following three propositions (which we state without proof), follow from Proposition 4.1. Roughly speaking, with each edge e we can associate a boolean combination of predicates $P_1, P_2, \ldots, P_k$, where each predicate compares some constant degree polynomial to zero. (i.e the regions are semi-algebraic sets).

- For any region $\gamma(e)$, the boundaries consist of segments of curves described by constant degree polynomials.
- For an edge $e \in E_{ij} - T \times T$, the region $\gamma(e)$ is a constant number of simple regions of constant description complexity.
- For an edge of the form $(t^x_{ijk}, t^x_{ijk+1}), x \in \{a, b, c, d\}$, the region $\gamma(e)$ consists of a set of simple regions of total description complexity k.

[4] The term *rt* denotes a *right tunnel*; *ut* denotes an *upper tunnel*.

LEMMA 4.2. *$|\Gamma| = O(p^2q^2(p+q)^4)$.*

LEMMA 4.3. *Let $\gamma_k = \gamma((t^x_{ijk}, t^x_{ij,k+1}))$, where $x \in \{a,b,c,d\}$. Then for all l such that $i \leq l < k$, $\gamma_k \subseteq \gamma_l$.*

Theorem 4.1 indicates that the graph property that we need to maintain is the reachability of v^b_{pq} from v^a_{00}. The algorithm is now as follows: Fix a traversal of the arrangement of regions. Check reachability at the starting cell. Each time an edge is crossed in the traversal, it corresponds to the deletion (and insertion) of edges in the graph, which we use to update the graph and check for reachability. Stop whenever the above property holds, returning YES, else return NO.

THEOREM 4.2. *Iff there exists a translation t such that $d_F(P, Q+t) \leq \varepsilon$, the above algorithm will terminate with a YES.*

Proof. Consider a type 1 critical event, where the interval a_{ij}, b_{ij} is created. This interval corresponds to the edge (v^a_{ij}, v^b_{ij}). Hence, this event corresponds to entering the region associated with the above edge. Similar arguments hold for other type 1 critical events.

Suppose we have a type 2 critical event, where the point a_{kj} rises above a_{ij} (in their relative vertical ordering). Note that this event does not change the reachability of (p,q) in the free space unless $\mathrm{rt}(a_{ij}) > k$. If this is the case, then the event results in setting $\mathrm{rt}(a_{ij}) = k$, implying that all edges of the form $(t^a_{ijl}, t^a_{ij,l+1}), l \geq k$ are deleted, which corresponds to leaving the regions corresponding to this set of edges[5].

Conversely, it can be seen that any transition from one cell of the arrangement to another corresponds to a critical event. We defer the details to a full version of the paper.

It now remains to analyse the complexity of the above algorithm. A transition between cells yields $O(1)$ updates, except in the case described in Theorem 4.2 above, where a transition occurs across the boundary of region $r((t^a_{ij,l-1}, t^a_{ijl}))$ into the region $r((t^a_{ij,k-1}, t^a_{ijk}))$, causing $\Theta(l-k)$ updates. However, note that in this event, it must be the case that all the regions $r((t^a_{ij,m}, t^a_{ij,m+1})), k \leq m < l-1$ intersect at this transition point (from Lemma 4.3), and thus the cost of this transition can be distributed among these cells. Hence, the total number of updates is given by Lemma 4.2.

[5]Note that since the regions corresponding to this set of edges are nested (by Lemma 4.3), such a transition is indeed possible. In fact, the existence of such a critical point implies that all of these regions intersect in at least one point that is also contained in $r((t^a_{ij,k-1}, t^a_{ijk}))$. The critical event can be interpreted as the result of the translation across this point.

To determine reachability, we must now traverse the arrangement. For ease of notation, we will assume that $p = \Theta(q)$ and set $n = p+q$. The arrangement consists of $O(n^3)$ regions, each described by $O(n)$ curves of constant description complexity. Let us fix r (we will specify the value of r later). It can be shown (using the theory of cuttings [8, 13]) that we can compute a subset $\mathcal{R}$ of the regions of size $O(r \log r)$ with the property that if we compute the vertical decomposition of each *super-cell* in the arrangement of $\mathcal{R}$, each of the resulting *primitive super-cells* (of constant complexity) is intersected by $O(n^3/r)$ regions.

LEMMA 4.4. *Given a graph $G = (V,E), |V| = N, |E| = M$, designated nodes $s,t \in V$, and a set of k edges $E' \subset E$, s-t reachability in G can be maintained over edge insertions and deletions from E' in total time $O(\min(N^\omega, Mk) + k^2U)$, where U is the number of such updates (ω is the exponent for matrix multiplication).*

Proof. Let V' be the set of endpoints of edges in E'. We compute the graph $G' = (V'' = V' \cup \{s,t\}, E'')$, where $(u,v) \in E''$ if there is a directed path from u to v in G. Note that $|V''| \leq 2k$. The computation of this graph can be done by performing a full transitive closure on G that takes time $O(n^\omega)$. Alternatively, we can perform $O(k)$ depth-first searches (one from each vertex in V'') to construct G'.

Now, to process updates, we update the graph using a standard dynamic update procedure that takes time $O(k^2 \log k)$ time (amortized) per update[19], yielding the result.

The algorithm now proceeds as follows: Each primitive super-cell has a set of edges associated with it (one for each region that intersects it). We use the above lemma to perform an efficient dynamic reachability test for each cell of the original arrangement in this primitive super-cell. When we move to the next primitive super-cell, we recompute the induced graph and repeat the process.

We now compute the value of r. The total number of cells in the arrangement is $O(n^8)$ by Lemma 4.2. There are $O(r^2n^2\log^2 r)$ primitive super-cells, each intersected by $O(n^3/r)$ regions. Consider a single primitive super-cell i. We apply Lemma 4.4 with $N = M = O(n^3)$, $k = O(n^3/r)$, and $U = U_i$, where U_i is the number of cells in i. The current value of ω is approximately 2.376 [12], and thus $\min(N^\omega, Mk) = Mk = n^6/r$ for all $r = \Omega(1)$. The cost of processing i is therefore $n^6/r + n^6U_i/r^2$. Summing over all primitive super-cells, and replacing ΣU_i by $O(n^8)$, we obtain the overall running time of the algorithm to be $O(n^8 r \log^2 r + n^{14}/r^2)$. Balancing, we obtain an overall running time of $O(n^{10} polylog n)$.

THEOREM 4.3. *Given two polygonal chains $P, Q, |P| = p, |Q| = q$, and $\varepsilon > 0$, we can check if $d_F(P,Q) \leq \varepsilon$ in time $O(n^{10} polylog\, n)$.*

The weak Fréchet distance As described earlier, the weak Fréchet distance (denoted by $d_{\tilde{F}}$) relaxes the constraint that the parametrizations employed must be monotone. Note that for any two curves P, Q, the following inequality is true: $d_H(P,Q) \leq d_{\tilde{F}}(P,Q) \leq d_F(P,Q)$ Also, by the result of Godau [14], all three measures collapse to one if both curves are convex. The above inequality is significant because it suggests that the weak Fréchet distance may serve as a relaxed curve matching measure with possibly more tractable algorithms.

As it turns out, this is indeed the case. Our techniques from the previous algorithm apply here as well, with two key differences. Firstly, since the paths need not be monotone, we no longer need the concept of a tunnel, thus reducing the number of critical events that need to be examined to $O(pq)$. Secondly, the underlying graph is now undirected, and there are efficient procedures for maintaining connectivity in an undirected graph [16]. We defer details to a full version of the paper, and summarize the result as:

THEOREM 4.4. *Given two polygonal chains $P, Q, |P| = p, |Q| = q$, and $\varepsilon > 0$, we can check if $\min_t d_F(P, Q+t) \leq \varepsilon$ in time $O(n^4 polylog\, n)$, where $n = O(p+q)$.*

An approximation scheme An (ε, β)-approximation (defined by Heffernan and Schirra [15]) for $d_F(P,Q)$ under translations can be obtained from the following observation:

LEMMA 4.5. *Given polygonal chains P, Q, let t be the translation that maps the first point of Q to the first point of P. Then $d_F(P, Q+t) \leq 2d^*$, where $d^* = \min_{translations\ t} d_F(P, Q+t)$.*

Applying the standard discretization trick in a ball of radius d^* around the first point of P, we obtain an (ε, β)-approximation for any $\beta > 0$. Note that this scheme is very efficient, running in time $O(n^2 poly(\log n, 1/\beta))$.

Acknowledgements

We would like to thank Helmut Alt, Julien Basch, Mikkel Thorup, Carola Wenk and Li Zhang for fruitful discussion. We also thank Héctor H. González-Baños and Eric Mao for supplying some of the pictures in this paper.

References

[1] K. Abrahamson. Generalized string matching. *SIAM Journal on Computing*, 16:1039–1051, 1987.

[2] Pankaj K. Agarwal, Micha Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. *J. Algorithms*, 17:292–318, 1994.

[3] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International J. of Computational Geometry and Applications*, 5:75–91, 1995.

[4] A. Amir and M. Farach. Efficient 2-dimensional approximate matching of half-rectangular figures. *Information and Computation*, 118:1–11, 1995.

[5] G. Barequet and S. Har-Peled. Some variants of polygon containment and minimum hausdorff distance under translation are 3sum-hard. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 862–864, 1999.

[6] P. Bogacki and S. Weinstein. Generalized fréchet distance between curves. In M. Daehlen, T. Lyche, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces II*, pages 25–32. Vanderbilt University Press, 1998.

[7] D. Cardoze and L. Schulman. Pattern matching for spatial point sets. In *Proc. 39th Annual Symposium on the Foundations of Computer Science*, pages 156–165. IEEE, November 1998.

[8] Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[9] L. Chew, M.T. Goodrich, D.P. Huttenlocher, K. Kedem, J.M. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. *Comput. Geom. Theory and Appl.*, 7:113–124, 1997.

[10] L. P. Chew, D. Dor, A. Efrat, and K. Kedem. Geometric pattern matching in d-dimensional space. In *Proc. 2nd Annual European Symposium on Algorithms*, volume 979 of *Lecture Notes Comput. Sci.*, pages 264–279. Springer-Verlag, 1995.

[11] L. P. Chew and K. Kedem. Improvements on geometric pattern matching problems. In *Proc. 3rd Scandinavian Workshop on Algorithm Theory*, volume 621 of *Lecture Notes Comput. Sci.*, pages 318–325. Springer-Verlag, 1992.

[12] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:1–6, 1990.

[13] M. de Berg and O. Schwarzkopf. Cuttings and applications. *Internat. J. Comput. Geom. Appl.*, 5:343–355, 1995.

[14] Michael Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions.* PhD thesis, Department Mathematik u. Informatik, Freie Universitt Berlin, December 1998.

[15] P. J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. *Computational Geometry: Theory and Applications*, 4(3):137–156, 1994.

[16] J. Holm, K. Lichtenberg, and M. Thorup. Polylogarithmic deterministic fully-dynamic algorithms for

connectivity, minimum spanning tree, 2-edge and bi-connectivity. In *Proc. 30th ACM Symposium on Theory of Computing*, pages 79–89. ACM, 1998.

[17] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *Proc. 10th Annual SIAM-ACM Symposium on Discrete Algorithms*, 1999.

[18] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[19] V. King. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In *Proc. 40th IEEE Symposium on Foundations of Computer Science*. IEEE, October 1999.

[20] S. R. Kosaraju. Efficient string matching. Manuscript, 1987.

[21] W.T. Rucklidge. Lower bounds for the complexity of the Hausdorff distance. In *Proc. 5th Canadian Conference on Computational Geometry*, pages 145–150, 1993.

[22] S. Venkatasubramanian. *Geometric Shape Matching and Drug Design*. PhD thesis, Department of Computer Science, Stanford University, August 1999.

[23] A. Winzen and H. Niemann. Matching and fusing 3D-polygonal approximations for model generation. In *Proc. IEEE International Conference on Image Processing*, volume 1, pages 228–232, Austin, Texas, 1994.

A Proof of Lemma 2.1

DEFINITION A.1. *For a geometric object R let $X(R)$, the x-span of R, denote the interval of the x-axis between the leftmost and the rightmost point of R', where R' is the orthogonal projection of R on the x-axis.*

CLAIM A.2. *Let $P = \{(x_1, y_1), \ldots (x_m, y_m)\}$ be a point set. We can construct in time $O(m \log^2 m)$ a data structure for P such that for query segment s, the point (x_k, y_k) maximizing the y-value of the set $\{s(x_i) + y_i \mid x_i \in X(s), 1 \leq i \leq m\}$ can be found in time $O(\log^2 m)$.*

Proof. If $X(P) \subseteq X(s)$, then (x_k, y_k) is clearly a vertex of the convex hull of P, and once the convex hull is computed, we can find (x_k, y_k) in time $O(\log n)$. To answer the query in the case that $X(P)$ is not contained in $X(s)$, we construct a sorted balanced binary tree $\Psi = \Psi(P)$ on the set $\{x_1 \ldots x_m\}$. For each node $\mu \in \Psi$ let P_μ denote the points in the subtree of μ, and let X_μ denote the x-span of P_μ. We construct C_μ, the convex hull of P_μ, for each node μ of Ψ. Once a query segment s is given, we find a set U of $O(\log |P|)$ nodes of Ψ with the property that for each node $\mu \in U$, X_μ is contained in $X(s)$, and in addition, each $(x_i, y_i) \in P$ for which $x_i \in X(s)$ appears in exactly one of the sets P_μ, for $\mu \in U$. We perform the query suggested by the previous claim on C_μ for each $\mu \in U$.

Based on Claim A.2, we describe the data structure as follows. Let $m = |\mathcal{S}|$. First observe that the maximum must be obtained at an endpoint of a segment of $\mathcal{S}$. We partition $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$. The set $\mathcal{S}_2$ contains at least $m - \sqrt{m}$ of the segment of $\mathcal{S}$. It is updated after $\sqrt{m}$ insertions or deletion operations into/from $\mathcal{S}$ Once it is updated, we explicitly compute the function $sum_{\mathcal{S}_1}(\cdot)$, and construct the data structure $\Psi = \Psi_{\mathcal{S}_1}$ of Claim A.2 for the vertices of the graph of $sum_{\mathcal{S}_1}(\cdot)$. As easily observed, the complexity of the graph of $sum_{\mathcal{S}_1}(\cdot)$ is $O(m)$, since a vertex of this function occurs only at endpoint of a segment of $\mathcal{S}_1$, thus the time needed to constuct $\Psi = \Psi_{\mathcal{S}_1}$. The set $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$ has cardinality $\leq \sqrt{m}$. Each time a segment is inserted (resp. deleted) into/from $\mathcal{S}$, it is inserted (resp. deleted) into/from $\mathcal{S}_1$. Once the size of $\mathcal{S}_1$ exceeds $\sqrt{m}$, we set $\mathcal{S}_1$ to be $\mathcal{S}$, construct Ψ, and empty $\mathcal{S}_2$.

In order to maintain the maximum $\max(sum_{\mathcal{S}}(\cdot))$, we do the following. Once a segment is inserted or deleted into $\mathcal{S}_1$, we explicitly compute (the graph of) $sum_{\mathcal{S}}(\cdot)$ which is piecewise linear of complexity $O(\sqrt{m})$. With each segment e of this graph (not to be confused with the segments of $\mathcal{S}$) we perform a query in $\Psi_{\mathcal{S}_1}$. The maximum obtained is is $\max(sum_{\mathcal{S}}(\cdot))$.

Next we describe the modifications of the data structure needed in the case where (some of) the segments of $\mathcal{S}$ move vertially in a constant speed with the time parameter τ. Let $X' = \{x_1 \ldots x_m\}$ denote the x-coordinates of the endpoints of the segments of $\mathcal{S}$. They are not time dependent. Let $y(x, \tau)$ denote the y-value of the sum function at the coordination x at time τ. Clearly as long as no insertions or deletions are taken place in $\mathcal{S}$, $y(x, \tau)$ moves (vertically) at a constant velocity. It is well known fact that the convex hull of such a set of points can go through $O(m)$ combinatorial changes, which we can compute in time $O(m \log m)$. This suggest the following modification to the data structure of $\mathcal{T}$ as follows. As before, each node μ is associated as before with the convex hull $C_\mu = C_\mu(t)$, but now these convex hulls might change in time. However, as argued, the total number of changes they go through is only $O(m \log^2 m)$. The query process remains the same.