Pattern Matching for Sets of Segments*

Alon Efrat† Piotr Indyk‡ Suresh Venkatasubramanian§

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 $O(n^3 \text{polylog} n)$ in the general case, and run in time $O(n^2 \text{polylog} n)$ for the case when all segments are horizontal. In addition, we show that when restricted to translations that are only vertical, an ε-approximation to Hausdorff distance between two sets of horizontal segments can be computed in time roughly $O(n^{3/2} \text{polylog} n)$. These algorithms are significant improvements over the general algorithm of Chew et al. that required 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 present a simple approximation algorithm for the Fréchet distance and its generalization that runs in time $O(n^2 \text{polylog} n)$.

1 Introduction

Traditionally, geometric pattern matching employs as a measure of similarity the (directed) 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 ignore the directionality 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 axis-align 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, 9, 28]. We discuss each measure in detail below.

*A perliminary version of this paper appeared in [16]

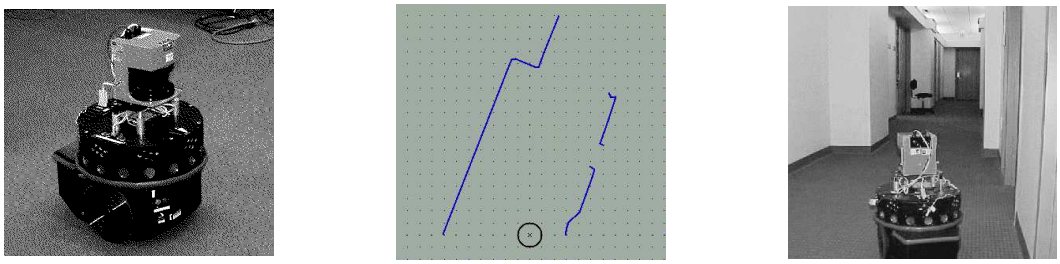
†Computer Science Department, The University of Arizona **Email:** *alon@cs.arizona.edu.* Work supported in part by a Rothschild Fellowship and by DARPA contract DAAE07-98-C-L027

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

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

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.



(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

In one such mapping project at the Stanford Robotics Laboratory[1] the robot ALON SAYS: name of robot and scanner 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 visited. We make the reasonable assumption that walls of buildings 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 $A = \{a_1 \ldots a_n\}$ and $B = \{b_1, \ldots b_n\}$ be two sets of axis-aligned line segments in the plane, and let $\varepsilon > 0$ be a given parameter. A point p of a horizontal (resp. vertical) segment $b \in B$ is *covered* if there is a point of a horizontal (resp. vertical) segment $a \in A$ whose distance from p is at most ε, where the distance is measured using the ℓ_∞ norm. A point on a horizontal (resp. vertical) segment can be covered only by a point of another horizontal (resp. vertical) segment. Let $w(A, B)$ denote the ALON SAYS: Do we use this notation somewhere else? collection of sub-segments of B consisting of covered points. Let $Cov_\varepsilon(A, B)$ 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 that maximizes $Cov_\varepsilon(t) = Cov_\varepsilon(A, t + B)$. Here ε is a parameter specified by the user based on the physical model. To the best of our knowledge, this similarity measure is novel. ALON SAYS: Check if the web address stile ←— ←—

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

alive 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, 14]. We also show 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 [26], and demonstrates that our algorithms, much like the algorithms of [12, 13], are able to avoid having to examine each cell of combinatorially different translations.

In Section 4 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 [7], ALON SAYS: And if it is vertical indicating that finding a sub-quadratic algorithm may be hard. However, we present an ε-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 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 $P : [0,1] \to \mathbb{R}^2$. The Fréchet distance between two curves P and Q, $d_F(P,Q)$ is defined as:

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

where α, β range over continuous increasing functions from $[0,1] \to [0,1]$.

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 $O(pq \log pq)$, where p and q are the number of segments in the two polygonal chains. In his Ph.D thesis [18], 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 of intrinsic dimension $d \geq 2$ is NP-hard.

Although the Fréchet distance is a natural measure for curve similarity, its applicability has been limited by the fact that no algorithms exists 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 [27]. 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 $O(n^5 \text{polylog} n)$ (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 5 we present the first algorithm for finding a translation of a polygonal chain, so that its Fréchet distance to another polygonal chain is $\leq \varepsilon$, for a given parameter ε, or determine that

no such translation exists. The algorithm is based on a reduction to a dynamic graph reachability problem; its running time is $O(n^{10}\text{polylog}n)$.

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 $O(n^4\text{polylog}n)$.

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 $O(n^2\text{poly}(\log n, 1/\beta))$.

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 $\varepsilon > 0$ be a given parameter. We first consider the case that the sets A and B consists of both horizontal and vertical segments.

ALON SAYS: Say that we use some ideas from kinetic data structures ... ←—

2.1 Coverage with axis-parallel segments

The main result of this subsection is the following theorem:

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

For the proof of this theorem, we need several lemmas and definitions. 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.

Let $A^h \subseteq A$ (resp. $B^h \subseteq B$) be the set of horizontal segments of A (resp. B) and let $A^v \subseteq A$ (resp. $B^v \subseteq B$) be the set of vertical segments of B. Note that $Cov_\varepsilon(A, t+B) = Cov_\varepsilon(A^h, t+B^h) + Cov_\varepsilon(A^v, t+B^v)$. Let s be a non-vertical segment. We define the function $s(x) : \mathbb{R} \to \mathbb{R}$ as follows: For every $x \notin X(s)$, $s(x) = 0$, and for $x' \in X(s)$, $s(x')$ is the value of the y-coordinate of the intersection point of s and the vertical line $x = x'$. To emphasize that s defines a function, we refer to s as a *graph-segment* or *gsegment* for short. Informally speaking, our interset in gsegments results from the fact the the maximal obtained along all tranlsations which lie on a horizontal line in the translation plane can be described as a sum of a set of gsegment, as we desctibe below.

Lemma 2.1. *Let* $P = \{(x_1, y_1), \dots (x_m, y_m)\}$ *be a point set. We can construct in time* $O(m \log^2 m)$ *a data structure for* P *such that for a query gsegment* s*, the point* $(x_k, y_k) \in P$ *maximizing the set* $\{s(x_i) + y_i \mid x_i \in X(s)$ *and* $1 \leq i \leq m\}$ *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 $O(\log n)$, as it is a standard linear programming query in a convex polygon. To answer the query in the case that $X(P) \not\subseteq X(s)$, we construct a balanced binary search tree $\Psi(P)$ on the set $\{x_1 \dots x_m\}$. For each node $\mu \in \Psi(P)$ 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 $\Psi(P)$. Once a query gsegment s is given, we find a set of $O(\log |P|)$ nodes μ of $\Psi(P)$ with the property that $X_\mu \subseteq X(s)$, and $X_{\text{parent}(\mu)} \not\subseteq X(s)$. We perform the desired maximization query on C_μ. The time required is clearly as claimed. □

For a set $\mathcal{S} = \{s_1 \dots s_m\}$ of non-vertical gsegments in $\mathbb{R}^2$, let $sum_{\mathcal{S}}(x) = \Sigma_{i=1}^{m} s_i(x)$, and let $max\_sum(\mathcal{S}) = \max_{x\in\mathbb{R}} sum_{\mathcal{S}}(x)$.

Lemma 2.2. *Let $\mathcal{S}(\tau) = \{(a_1(\tau), b_1(\tau)), (a_2(\tau), b_2(\tau)), \dots, (a_m(\tau), b_m(\tau))\}$ be a set of gsegments whose location is a function of a parameter τ, where*

$$a_i(\tau) = (a_i^0.x,\ a_i^0.y + \alpha_i\tau) \quad \textit{and} \quad b_i(\tau) = (b_i^0.x,\ b_i^0.y + \beta_i\tau)\ ,$$

where $a_i^0.x, a_i^0.y, b_i^0.x, b_i^0.y, \alpha_i, \beta_i$ are given constants. Thinking about τ as a time parameter, the endpoints of the segments move vertically at constant velocities. Then we can construct a data structure $\mathcal{Q}(\mathcal{S}(\cdot))$, so that given a time τ_i and a query point q on the x-axis, we can find $\{sum_{S(\tau_i)}(q) \mid x \in X(e_i)\}$ time $O(\log^2 m)$.

Proof. We construct a segment tree $\mathcal{T}'$ on the x-projections of the segments of $\mathcal{S}(\tau)$ (note that their projections do not change in time). With each node μ of $\mathcal{T}'$ we maintain the interval I_μ on the x-axis assosiated with μ, and the seubset $S(\tau) \subseteq \mathcal{S}(\tau)$ stored with μ. Let ℓ_L and ℓ_R be the left and right verticall llines passing through the endpoints x_L and x_R of I_μ. We can express the y-coordination of the intersection point of the ith segment e of S_μ with ℓ_L and ℓ_R (resp.) by $a^L_{\mu,i}\tau + b^L_{\mu,i}$ and $a^R_{\mu,i}\tau + b^R_{\mu,i}$ (for $i = 1, \dots |S_\mu|$), when $a^L_{\mu,i}$, $b^L_{\mu,i}$, $a^R_{\mu,i} b^R_{\mu,i}$ are appropriate constants. Let x be a point on I_μ, and let $\alpha = (x - x_L)/(x_R - x_L)$. Hence at time τ,

$$e(x) = \alpha\left(a^L_{\mu,i}\tau + b^L_{\mu,i}\right) + (1-\alpha)\left(a^R_{\mu,i}\tau + b^R_{\mu,i}\right)$$

Defining

$$A^L_\mu = \Sigma_1^{|S_\mu(\tau)|} a^L_{\mu,i},\ \ B^L_\mu = \Sigma_1^{|S_\mu(\tau)|} b^L_{\mu,i},\ \ A^R_\mu = \Sigma_1^{|S_\mu(\tau)|} a^R_{\mu,i}\ ,\ A^R_\mu = \Sigma_1^{|S_\mu(\tau)|} a^R_{\mu,i}\ ,$$

We have that

$$max_{S_\mu(\tau)}(x) \ =\ \alpha\left(\tau A^L_\mu + B^L_\mu\right)\ +\ (1-\alpha)\left(\tau A^R_\mu + B^R_\mu\right) \tag{2.1}$$

We store $A^L_\mu, B^L_\mu, A^R_\mu, A^R_\mu$ with each node μ. Once a point x and a time τ are given, we find the $O(\log m)$ nodes μ of $\mathcal{T}'$ for which $x \in I_\mu$ (the is at most one such node at each level of $\mathcal{T}'$). For each, we evaluate the expression 2.1, and sum the results. Clealy this can be done in $O(\log m)$, and the construction of $\mathcal{T}'$ can be done in $O(m \log m)$. □

Lemma 2.3. *Let $\mathcal{S}(\tau) = \{(a_1(\tau), b_1(\tau)), (a_2(\tau), b_2(\tau)), \dots, (a_m(\tau), b_m(\tau))\}$ as in Lemma 2.2. We can construct a data structure $\mathcal{D}(\mathcal{S}(\cdot))$, so that given a query time τ and a query gsegments $e(x)$, we can find $\max\{e(x) + sum_{S(\tau)}(x) \mid x \in X(e)\}$, so that the total time needed to answer k queries is $O((m+k)\log^2 m)$, provided that the time τ of each query is no larger than the time of the previous query.* ALON SAYS: can we construct it with time moving upward ←

Proof. We first explain how to construct the data strucure for a fixed time τ_1. Before each query is posed to the data structure, we modify the data structure according to the time τ_i of that query.

Using a simple divde-and-conqure technique we construct in time $O(m \log m)$ the graph of the function $sum_{S(\tau_1)}(x)$. This is a piecewise linear graph. Let $V(S(\tau_1))$ denote the set of vertices of this graph. Note that every such vertex has the same x-coordinate as an endpoint of one of the gsegments of $\mathcal{S}(\tau_1)$, thus $|V(\mathcal{S}(\tau_1))| = O(m)$.

Clearly for every gsegment e the maximum of the set $\{e(x) + sum_{S(\tau_1)}(x) \mid x \in X(e)\}$ at the x-projection of either a vertex of $V(S(\tau_1))$, or an endpoint of $e(x)$. We use the data structure of Lemma 2.2 to find the value at the endpoint of $e(x)$. In order to handle the former case, we construct the data structure $\Psi(V(S(\tau_1)))$ of Lemma 2.1. This enables answering a query (for time τ_1) in time $O(\log^2 m)$.

Once the next query is submitted (with a smaller time $\tau_2 < \tau_1$), we need to efficiently modify $\Psi(V(S(\tau)))$ to create $\Psi(V(S(\tau_2)))$. We decrease τ graduatly, while keeping track of the changes the data structure goes through. Note that as τ decreses from τ_1 to τ_2, the vertices of the graph $\max_{S(\tau)}(x)$ move vertically downward at a constnt speed, as the speed of each of them is the sum of $\leq m$ values which changes linearly. We keep track of the changes that each convex hull C_μ stored at a node of the tree of $\Psi(V(S(\tau)))$, goes through.

It is well known that the convex hull of such a set of k points moving vertically at constant speeds can go through $O(k)$ combinatorial changes. These changes can be tracked in a total of $O(k \log k)$ time by a simple divide-and-conqure algorithm, by splitting the vertices into two equal-cardinality subsets to the left and right of a vertical line, maintain recursively the convex hull of each subsets, and show that the common tangents to these hulls can goes through $O(k)$ combinatorial changes which are trivial to tackle.

Thus this is the time needed to be spent on maintaing C_μ, as we decrease τ from τ_1 to τ_m. Since the total sizes of convex hulls in $\Psi(V(S(\tau)))$ is $O(m \log^2 m)$, we need $O(m \log^2 m)$ time to maintain $\Psi(V(S(\tau)))$ as τ decreases from the first to the last query. □

Lemma 2.4. *Let $\mathcal{S}(\tau)$ be as in Lemma 2.3. Then we can maintain $\max\_sum(\mathcal{S}(\tau))$ under gsegment insertions or deletions in amortized time $O(\sqrt{m'} \log^2 m')$ per operation. In addition, we can maintain $\max\_sum$ under a* time-decrease *step ($\tau \leftarrow \tau - \Delta$) in $O(\sqrt{m'} \log m')$ time per update. Here m' is the maximum betweem m and the total number of operations done on the set.*

Proof. A deletion of a gsegment e is resolved by adding the negation of e, so $m' = |\mathcal{S}(\tau)|$, and we direct our attention to the insertion of gsegments. We partition $\mathcal{S}(\tau)$ into $\mathcal{S}_1(\tau)$ and $\mathcal{S}_2(\tau)$. The set $\mathcal{S}_1(\tau)$ contains at most $\sqrt{m'}$ of the gsegments of $\mathcal{S}(\tau)$. We define $\mathcal{S}_2(\tau) = \mathcal{S}(\tau) \setminus \mathcal{S}_1(\tau)$. Each time a gsegment is inserted into $\mathcal{S}(\tau)$, it is inserted into $\mathcal{S}_1(\tau)$. Once the cardinality of $\mathcal{S}_1(\tau)$ exceeds $\sqrt{m'}$, we set $\mathcal{S}_2(\tau)$ to be $\mathcal{S}(\tau)$, empty $\mathcal{S}_1(\tau)$, and construct the data structure $\mathcal{D}(S_2(\tau))$ of Lemma 2.3 for the vertices of the graph of $sum_{\mathcal{S}_2(\tau)}(\cdot)$.

In order to maintain $\max\_sum_{\mathcal{S}(\tau)}(\cdot)$, we do the following. Once a gsegment is inserted into $\mathcal{S}_1$, we explicitly compute the graph of $sum_{\mathcal{S}_1(\tau)}(\cdot)$ which is piecewise linear of complexity $O(\sqrt{m'})$. With each gsegment e of this graph (not to be confused with the segments of $\mathcal{S}(\tau)$) we perform a query in $\mathcal{D}(\mathcal{S}_1(\tau))$. The maximum obtained is $\max\_sum_{\mathcal{S}(\tau)}$. This operation is doable ALON SAYS: give ref to the lemma ←
in time $O(\sqrt{m'} \log^2 m')$. Decreasing τ is obtained as in Lemma 2.3. Hence the lemma holds. □

ALON SAYS: find ref to the number of changes in a convex hull ←

We next turn our attention to conclude the proof of of the first theorem of this section.

Proof. (of Theorem 2.1) The algorithm consists of sweeping the translation plane from top to bottom, using a horizontal sweeping line $\ell(y)$. We maintain a set $\mathcal{S}(y)$ of gsegments, initially empty, and maintain its maximum $\max\_sum(\mathcal{S}(y))$ using the data structure $\mathcal{D}(\mathcal{S}(y))$ of Lemma 2.3. As shown later, the maximum value obtained is equal to $\max_t Cov_\varepsilon(A, t + B)$.

Let D_ε be a square of edge-length 2ε and consider the Minkovski sum

$$D_\varepsilon \oplus A^h = \{d + a \mid d \in D_\varepsilon \text{ and } a \text{ is a point of a segment of } A^h\}.$$

Note that $D_\varepsilon \oplus A^h$ can be expressed as the union of n rectangles, all of height 2ε, so the boundaries of each two intersect in at most two points, and by [22] the complexity of their union is $O(n)$. We impose a horizontal decomposition on $D_\varepsilon \oplus A^h$ to obtain a set of $O(n)$ rectangles $R^h = \{\gamma_1, \gamma_2 \dots\}$ and a vertical decomposition on $D_\varepsilon \oplus A^v$ to obtain a set of $O(n)$ rectangles $R^v = \{\rho_1, \rho_2 \dots\}$. There are two types of events that we handle in the line sweep process, called *horizontal segment event* and *vertical segment*

event. Upon encountering each such event, say for $\ell(y_0)$, we modify the data structures (described below) and compute $\max_{t\in\ell(y_0)} \mathit{Cov}_\varepsilon(A, t+B)$. The events are computed in the preprocessing stage, and stored in the line-sweep queue. The events are described as follows:

ALON SAYS: Do we need to take special care of the union of A had no inner boundaries in the H decomposition ⟵

Horizontal segment events. For every $b_i \in B^h$ and rectangle $\gamma_j \in R^h$ we create a set of events, as follows: Assume

$$\gamma_j = ((c,d),(c+\Delta_x,d),(c,d+\Delta_y),(c+\Delta_x,d+\Delta_y)) ,$$

and the segment $b_i = ((a,b),(a+\delta,b)) \in B$. Assume that $\delta \le \Delta_x$. The case that $\delta > \Delta_x$ is treated analogously.

The first event at which the couple b_i, γ_j are involved happens when $y = d+\Delta_y - b$ (i.e. when the $(x,y)+b_i$ is aligned with the upper edge of γ_j). Upon this event, we insert the following gsegments into $\mathcal{S}(y)$.

- $r_1(y) = ((c-a-\delta,0),(c-a,\delta))$. The x-span of this gsegment corresponds to all translations on $\ell(y)$ for which $(x,y)+b_i$ intersects γ_j, but the left endpoint of b_i is outside γ_j.
- $r_2(y) = ((c-a,\delta),(c+\Delta_x-a-\delta,\delta))$. The x-span of this gsegment corresponds to all translations on $\ell(y)$ for which $(x,y)+b_i$ is fully contained in γ_j.
- $r_3(y) = ((c+\Delta_x-a-\delta,\delta),\ (c+\Delta_x-a,\delta))$. The x-span of this gsegment corresponds to all translations on $\ell(y)$ for which $(x,y)+b_i$ intersects γ_j, but the right endpoint of b_i is outside γ_j.

The second event at which the couple b_i, γ_j are involved happens when $y = d-b$ (i.e. when $(x,y)+b_i$ is aligned with the lower edge of γ_j.) Upon this event, we delete $r_1(y), r_2(y), r_3(y)$ from $\mathcal{S}(y)$. Note that for every $y \in [d+\Delta_y-b, d-b]$ the function $\mathit{sum}_{\{r_1(y),r_2(y),r_3(y)\}}(x)$ equals the length of the portion of $(x,y)+b_i$ inside γ_j.

Vertical segment events. For every $b_i \in B^v$ and rectangle $\rho_j \in R^v$ we create a set of events, as follows: Assume that

$$\rho_j = ((c,d),(c+\Delta_x,d),(c,d+\Delta_y),(c+\Delta_x,d+\Delta_y)) ,$$

and $b_i = ((a,b),(a,b+\delta))$. Again assume that $0 < \delta \le \Delta_y$ (the other case is treated analogously). The pair b_i, ρ_j is involved in a few events.

- Once $y = d+\Delta_y-b$ (i.e. the lower endpoint of $(x,y)+b_j$ is aligned with the upper edge of ρ_j we insert the gsegment $s_1(y) = (c-a, y'-y),\ (c+\Delta_x-a, y'-y))$ into $\mathcal{S}(y)$, where y' is the current value of y. Note that $s(y)$ is a moving horizontal gsegment.
- Once $y = d+\Delta_y-\delta-b$ (i.e. the upper point of $(x,y)+b_i$ is aligned with the upper edge of ρ_j) we delete $s_1(y)$ from $\mathcal{S}(y)$ and insert $s_2(y') = ((c-a,\delta),(c+\Delta_x-a,\delta))$. This is a static horizontal gsegment.
- Once $y = d-b-\delta$ (i.e. the lower point of $(x,y)+b_i$ is aligned with the lower edge of ρ_j) we delete $s_2(y)$ from $\mathcal{S}(y)$, and insert the gsegment $s_3(y) = ((c-a,\delta+d-b+y),\ (c+\Delta_x-a,\delta+d-b+y))$.

- Once $y = d - b - \delta$ (i.e. the upper point of $(x, y) + b_i$ is aligned with the lower edge of ρ_j) we delete $s_3(y)$ from $\mathcal{S}(y)$.

Note that $s_1(y), s_2(y), s_3(y)$ represent, each in its y-span, the function which is the length of the portion of b_j inside ρ_j.

Observe that for any given y, $sum_{\mathcal{S}(y)}(x)$ represents the total length of the portion of the segments of $(x, y) + B$ which is inside $D_\varepsilon + A$, i.e. $Cov_\varepsilon(A, (x, y) + B)$. Since the maximum value of these functions must be obtained at one of the events listed above, and at each such event we check thi maximum, the correctness of the algorithm follows.

Time analysis: Overall, we add and delete four (moving) gsegments for each pair (b_i, γ_j) (for $b_i \in B^h, \gamma_j \in R^h$) or a pair (b_i, ρ_i) (for $b_i \in B^v, \gamma_j \in R^v$), thus a total of $O(n^2)$ events. We compute $max\_sum(\mathcal{S}(y))$, each is doable in time $O(\sqrt{n}^2 \log^2 n)$, for each of these events, hence the overall running time of the algorithm is $O(n^3 \log^2 n)$. It is not hard to show that this bound also bounds the time needed for constructing the data structures. ALON SAYS: running time of constructing the DS This ⟵ conclude the proof of Theorem 2.1.

□

2.2 Maximum coverage for horizontal segments

We present a faster algorithm for the case that all segments are horizontal. This is a line-sweep algorithm influenced somwhow ALON SAYS: english by the Chew-Kedem [12] and Chew *et al.* [13] algorithms ⟵ for computing the Hausdorff distance under translation between point-sets in the plane, under the L_∞ norm. This time, we sweep the plane from left to right, using a vertical line sweep. Let $D_\varepsilon \oplus A^h$ and $R^h = \{\gamma_1, \gamma_2 \dots\}$ be as in Section 2.1. For every horizontal segment $b_i \in B$ and $\gamma_j \in R^h$ let β_{ij} be the rectangles (in the translation plane) of all translations in which b_i intersects γ_j. Let $\mathcal{E}$ denote the set of the vertical edges of all rectangles β_{ij}.

Let $\mathcal{T}$ be a segment tree constructed on the projections of the segments of $\mathcal{E}$ on the y-axis. During the algorithm we sweep the translation plane *TP* using a vertical sweep line ℓ. Once ℓ meets an edge $e \in \mathcal{E}$, we insert e into $\mathcal{T}$. No edge is deleted.

Let μ be a node of $\mathcal{T}$. Let I_μ be the y-span corresponds to μ, and let $S_\mu \subseteq \mathcal{E}$ denote the edges of $\mathcal{E}$ corresponding to μ, i.e. the edges of $\mathcal{E}$ whose y-span contains I_μ but not $I_{parent(\mu)}$. Let $\mathcal{T}_\mu$ denote the subtree rooted at μ. For $x \in \mathbb{R}$, set $S_\mu(x) \subseteq \mathcal{S}_\mu$ denote the segment of S_μ whose x-span is $\leq x$, and let

$$\mathcal{L}_\mu(x) = \{(b_i, \gamma_j) \mid b_i \in B, y \in I_\mu, \text{ and an edge of } \beta_{ij} \text{ is stored at } S_{\mu'}(x),\ \mu' \in \mathcal{T}_\mu\}$$

In other words $\mathcal{L}_\mu(x)$ contains all pairs (b_i, γ_j) such that $(x', y) + b_i$ intersect γ_j for some $x' \leq x, y \in I_\mu$. We define the *maximal coverage associated with a node* μ at x_0, denoted by $Cov_\mu(x_0)$ as the maximal total length of segments

$$\{(x_0, y) + b_i \cap \gamma_j \mid (b_i, \gamma_j) \in \mathcal{L}_\mu(x_0)\}$$

where the maximum is taken over all translations $(x_0, y), y \in I_\mu$. Let $\pi^*_\mu(x)$ denote that path in $\mathcal{T}$ from μ to the leaf that contains the thasnations that maximize maximum coverage assosiated with μ at x. Thus for example, then $\pi^*_{root(\mathcal{T})}(x)$ is the path from $root(\mathcal{T})$ to the leaf μ' such that $y^* \in I_{\mu'}$, where $Cov_\varepsilon(A, (x, y^*) + B) = \max_y\{Cov_\varepsilon(A, (x, y) + B)\}$.

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.

- Pos_μ: the number of edges of $\mathcal{E}$ currently in $S_\mu(x)$ resulting from the right (resp. left) endpoint of a segment $b \in B^h$ meeting a left (resp. right) vertical edge of some rectangle γ_j. We call such an event a *Positive event*.
- Neg_μ: the number of edges of $\mathcal{E}$ currently in $S_\mu(x)$ resulting from the left (resp. right) endpoint of a segment $b_\in B$ meeting a left (resp. right) vertical edge of some rectangle γ_j. We call such an event a *Negative event*. Observe that $Pos_\mu - Neg_\mu$

$$|\{(b_i, \gamma_j) \in \mathcal{L}_\mu(x) \mid \gamma_j \text{ contains the right endpoint of } b_i, \text{ but not the left endpoint}\}| -$$
$$|\{(b_i, \gamma_j) \in \mathcal{L}_\mu(x) \mid \gamma_j \text{ contains the left endpoint of } b_i, \text{ but not the right endpoint}\}|$$

- x_last_μ — the last x at which we inserted an edge into the subtree of μ.
- $Max\_Tot\_at\_event_\mu$. We will show in Lemma 2.5 that this parameter stores $Cov_\mu(x\_last_\mu)$.
- $MaxMul_\mu$ — a multiplicative factor specifying the rate of increase of the coverage as the horizontal distance increases. That is, if x_1 and x_2 are two close points in $\mathbb{R}$, then the difference in the coverage $Cov_\mu(x_2) - Cov_\mu(x_1) = (x_2 - x_1) * MaxMul_\mu$. In other words

$$MaxMul_\mu = sum_{\mu' \in \pi^*_\mu}(Pos_{\mu'} - Neg_{\mu'})$$

Handling an event.

During the algorithm we encounter two types of events. An **edge event** happens when the line sweep hits a vertical edge of $\mathcal{E}$. A **dominance event** happens at time x and node μ if

$$MaxMul_{left(\mu)} * (x - x\_last_{left(\mu)}) + Max\_Tot\_at\_event_{left(\mu)} =$$
$$MaxMul_{right(\mu)} * (x - x\_last_{right(\mu)}) + Max\_Tot\_at\_event_{right(\mu)}$$

This event occurs at x' if the translation (x, y) that maximizes $\{Cov(A, (x, y) - B) | y \in I_\mu\}$ is in $I_{left(\mu)}$ for x slightly smaller than x', and occurs at $I_{right(\mu)}$ for x slightly larger, or vice versa.

The x-coordinate of this event is computed and inserted into the queue of the line sweep, once the values of the fields of $left(\mu)$ or $right(\mu)$, or the fields of any of their descendants are modified. We explain next how we handle each such event.

We will show in Lemma 2.5 that the following claim is an invariant of the algorithm: For every $x \in \mathbb{R}, \mu \in \mathcal{T}$, $Cov_\mu(x) = Max\_Tot\_at\_event_\mu + (x - x\_last) * MaxMul_\mu$. We use the following function to maintain this invariant:

Function `UpdateNode`(μ)
$Max\_Tot\_at\_event_\mu = Max\_Tot\_at\_event_\mu + (x - x\_last_\mu) * Max\_Tot\_at\_event_\mu$
$x\_last_\mu = x$

Let $\delta > 0$ be an infinitely small constant
If $Max\_Tot\_at\_event_{left(\mu)} + MaxMul_{left(\mu)} * (x + \delta - x\_last_{left(\mu)}) >$
  $Max\_Tot\_at\_event_{right(\mu)} + MaxMul_{right(\mu)} * (x + \delta - x\_last_{right(\mu)})$
Then
  $MaxMul_\mu = Pos_\mu - Neg_\mu + MaxMul_{left(\mu)}$
Else
  $MaxMul_\mu = Pos_\mu - Neg_\mu + MaxMul_{right(\mu)}$

If μ is not the root of $\mathcal{T}$
  Then `UpdateNode`(parent(μ)).

Handling edge events at node μ. Let x be the current x-value of the line sweep. Once ℓ hits a new edge $s \in \mathcal{E}$, 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, and perform `UpdateNode(`μ`)`.
Handling dominance events at node μ Once an dominate event occur at a node μ, we call `UpdateNode(`μ`)`.

Lemma 2.5. *The invariant*

$$Cov_\mu(x) = Max\_Tot\_at\_event_\mu + (x - x\_last_\mu) * MaxMul_\mu \quad \textit{for every } x \in \mathbb{R}, \mu \in \mathcal{T}$$

holds at any stage of the algorithm.

Proof. The proof is by a double induction on the height of a node μ and the sequence of events in which μ was updated. Assume first that μ is a leaf. Assume that x_1 is an event at which the fields of μ are updated, and that no event happened between x_1 and $x_2 > x_1$. x_2 is not necessarily an event. Also assume that in x_1 the invariant holds.

Let y_0 be a point in I_μ (since μ is a leaf, the choice of y_0 is not relevant). Let $(b_i\, \gamma_j) \in \mathcal{L}_\mu(x_2)$. Then $|(x_2, y_0) + b_i \cap \gamma_j| - |(x_1, y_0) + b_i \cap \gamma_j|$ is $x_2 - x_1$, $x_1 - x_2$ or 0, according to whether γ_j contains only the right endpoint of $(x_2, y_0) + b_i$, only its left endpoint of b_i, or neither or both endpoints. In the first case, (b_i, γ_j) contributes 1 to Pos_μ. In the second case, (b_i, γ_j) contributes 1 to Pos_μ and 2 to Neg_μ, and in the last case, (b_i, γ_j) contributes the same amount to Pos_μ and Neg_μ. Summing over all pairs $(b_i, \gamma_j) \in \mathcal{L}_\mu(x_2)$, we obtain that $Cov_\mu(x_2) - Max\_Tot\_at\_event(x_1)$ equals $(Pos_\mu - Neg_\mu)(x_2 - x_1)$, from which the claim follows.

Next assume that μ is an internal node. Consider the pairs $(b_i, \gamma_j) \in \mathcal{L}_\mu(x_2)$ for which the vertical edges of β_{ij} are stored in S_μ. The contribution of these pairs to $Cov_\mu(x_2)$ is counted as in the case of a leaf node μ. Moreover, one can show that the $MaxMul$ at x_2 equals the $sum_{\mu'} Pos_{\mu'} - Neg_{\mu'}$ where the sum is taken over all nodes μ' on the path leaving from μ to the leaf containing the translations that maximizes $Cov_\mu(x_2)$. This is because of the dominance events mechanism that guarantees that $MaxMul_\mu$ would take into account the contribution from the child μ' of μ from which $Cov_{\mu'}$ is larger. □

Theorem 2.2. *Let A and B be two sets of horizontal segments. Then we can find in $O(n^2 \log^2 n)$ a translation t at which $Cov_\varepsilon(A, t + B)$ is maximal.*

Proof. The number of edge events is clearly $O(n^2)$. Each edge event occurring at a node $\mu \in \mathcal{T}$ can cause $O(\log n)$ dominance events, one at each of the ancestor nodes of μ, thus there are $O(n^2 \log n)$ dominance events. Each event is handled at $O(\log n)$ time. Thus the bound of the running time follows.

To find the optimal translation, we monitor the maximal value of $Max\_Tot\_at\_event_{root(\mathcal{T})}$. Clearly the maximal coverage must be obtained at an edge event, and Lemma 2.5 guarantees that the maximum coverage must be equal to $Max\_Tot\_at\_event_{root(\mathcal{T})}$ at this event. □

3 A lower bound

Rucklidge [26] 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 translation plane (TP) of all translations t for which $h(A, t + B) \leq \varepsilon$ is in the worst case $\Omega(n^4)$, where $h(A, B)$ is the one way Hausdorff distance from A to B. This bound is tight, since the number of intersection points created by n^2 rectangles in the plane is $O(n^4)$. We next show that the $\Omega(n^4)$—bound holds also in the case that the segments are horizontal. That

is, we show a construction of sets A and B of n horizontal segments each, such that the combinatorial complexity of the regions of all translations t for which $h(A, t + B) \leq \varepsilon$ is $\Omega(n^4)$.

Assume for the construction that $\varepsilon = 1/2$. The first component in the construction (see Figure 2) is the set B_1' consisting of $2n$ points, which are

$$\{(i, 1/2 + i/n^2) \text{ and } (i, -1/2 + i/n^2 - \delta), \text{ for } i =, 1 \ldots n\}$$

where δ is a small enough parameter. Thus the pair $(i, 1/2 - i/n)^+$ and $(i, -1/2 - i/n - \delta)^+$ forms two very close vertically aligned squares, where the gap between them is of unit height and width δ, and located at distance i/n below the y-axis. We add the segment B_1'', which is the long horizontal segment between the points $(-n, 0)$ and $(0, 0)$ and the segment B_1''' between $(n, 0)$ and $(2n, 0)$. Let $B_1 = B_1' \cup B_1'' \cup B_1'''$.

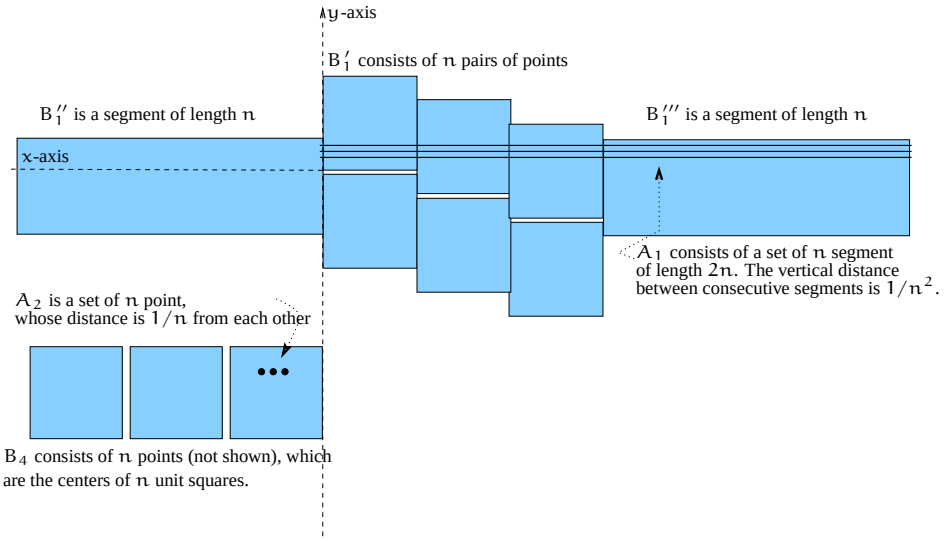


Figure 2: The lower bound construction for $n = 3$. The set B is not shown explicitly — only B^+ is shown.

The set A_1 consists of a n horizontal segments of length $2n$, whose vertical distance is $\frac{1}{2n}$. The left endpoint of all of them is on the y-axis, and the middle one is on the x-axis. By shifting them vertically, each segment in turn is not completely covered at some time, when it passes between the gaps between one of the pairs of B_1. In all other cases, all the segments are completely covered. The region in TP corresponds to all translations t for which $h(A_1, t + B_1) \leq 1$ consists of $\Omega(n^2)$ horizontal strips, each of length n.

The set B_2 consists of the n points $(-(1 + 1/n^2)i, -5)$ (for $i = 1 \ldots n$). Thus B_2^+ creates n unit squares along the line $y = -5$, with a gap of $1/n^2$ between them. The set A_1 consist of n points along the horizontal line $(-1/2n, -5)$ (for $i = 1 \ldots n$). Observe that A_1 fits completely into each of the squares of B_2^+. However, by sliding A_1 horizontally, along $y = -5$ or anywhere at distance ≤ 1 from h, each of the points of A_1 "falls" at some stage into each of the gaps between each of the squares of B_2^+, The region $S_2 = \{t \mid h(A_2, t + B_2) \leq 1\}$ consists of $\Omega(n^2)$ vertical strips in TP, each of height 2. Letting $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$, the region $S = \{t \mid h(A, t + B) \leq 1\}$ is merely the intersection of S_1 and S_2, which is clearly of complexity $\Omega(n^4)$, thus proving our claim.

4 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 $\rho^* = \min_t h(A, t + b)$ 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 $\{1 \dots M\}$.

Theorem 4.1. *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 $h(A, t+B) \le (1+\varepsilon)\rho^*$ in time $O(n^{3/2} \max(poly(\log M, \log n, 1/\varepsilon)))$.*

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

Definition 4.1. (Interval matching): *given two sequences $t = t[1] \dots t[n]$ and $p = p[1] \dots p[m]$, such that $p[i] \in [M]$ and $t[i]$ is a union of disjoint intervals $\{a_i^1 \dots b_i^1\} \cup \{a_i^2 \dots b_i^2\} \dots$ 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 [10, 21], 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 $O(s^{3/2}\text{polylogs})$.
- 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 $s' = O(s \text{ polylogs})$.

These three observations yield the proof of Theorem 4.1. In the remainder of this section, we sketch proofs of the above observations.

The interval matching problem. Our method follows the approach of [1, 25] and [6].

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 $x_1 \dots x_k$ by one symbol $\{x_1 \dots x_k\}$, 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 $O(nM'\text{polylog}n)$ 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 $p'[i] = \cup_{i':i' \bmod P = i} p[i']$ and $t'[i] = \cup_{i':i' \bmod P = i} t[i']$. It can be shown (using similar ideas as in [10]) 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.

5 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:

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

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 d_F is a metric and $d_{\tilde{F}}$ is a pseudo-metric, i.e., it fulfills almost all properties of a metric, but two different curves can have a weak Fréchet distance 0.

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[2]. 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:

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

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

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

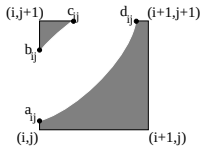


Figure 3: A single cell in the free space

$p_1 = (x_1, y_1), p_2 = (x_2, y_2)$, $p_1 \leq p_2$ if $x_1 \leq x_2$ and $y_1 \leq y_2$. 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 $d_F(P, Q) \leq \varepsilon$. 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 $d_{\tilde{F}}(P, Q) \leq \varepsilon$. 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 $\min_t d_F(P, Q + t) \leq \varepsilon$.

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[3] $ut(r) = k$.

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 5.1. *For a point* x_{ij}*, the function* $x_{ij}(t)$ *is given by a second degree polynomial* $p(x_{ij}, t_x, t_y) = 0$*, where* (t_x, t_y) *are the coordinates of* t.

5.1 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. ALON SAYS: what is ν ?? ←

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

Let

$$V = \bigcup_{i,j} \{v^a_{ij}, v^b_{ij}, v^c_{ij}, v^d_{ij}\}$$

and

$$T = \bigcup_{\substack{i,j,k \\ i<k\le p}} \{t^a_{ijk}, t^b_{ijk}\} \ \cup \ \bigcup_{\substack{i,j,k \\ j<k\le q}} \{t^c_{ijk}, t^d_{ijk}\}$$

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

$$V^2_{ij} = \{v^c_{ij}, v^d_{ij}\} \cup \bigcup_{l<j\le ut(c_{il})} t^c_{ilj} \cup \bigcup_{l<j\le ut(d_{lj})} t^d_{ilj} \ .$$

im 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 5.1. *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 follow from Proposition 5.1. 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.

Lemma 5.1. $|\Gamma| = O(p^2q^2(p+q)^4)$.

Lemma 5.2. *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 5.1 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 5.2. *There exists a translation* t *such that* $d_F(P, Q + t) \leq \varepsilon$, *if and only if 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 $rt(a_{ij}) > k$. If this is the case, then the event results in setting $rt(a_{ij}) = k$, 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[4].

Conversely, it can be seen that any transition from one cell of the arrangement to another corresponds to a critical event. □

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 5.2 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 5.2), and thus the cost of this transition can be distributed among these cells. Hence, the total number of updates is given by Lemma 5.1.

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 [11, 8]) 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 5.3. *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).*

[4]Note that since the regions corresponding to this set of edges are nested (by Lemma 5.2), 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.

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[24], 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 5.1. There are $O(r^2 n^2 \log^2 r)$ primitive super-cells, each intersected by $O(n^3/r)$ regions. Consider a single primitive super-cell i. We apply Lemma 5.3 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 [15], and thus $\min(N^\omega, Mk) = Mk = n^6/r$ for all $r = \Omega(1)$. The cost of processing i is therefore $n^6/r + n^6 U_i/r^2$. 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 $O(n^{10}\text{polylog}n)$.

Theorem 5.3. *Given two polygonal chains $P, Q, |P| = p, |Q| = q$, and $\varepsilon > 0$, we can check whether $\min_{t \in TP} d_F(P, Q + t) \leq \varepsilon$ in time $O(n^{10} polylog n)$.*

Remark: Recently, Alt, Knauer and Wenk [5] were able to obtain an algorithm that solves our problem is $O(n^8 \text{polylog}n)$, significantly improving our result.

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 [18], 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 [20].

Theorem 5.4. *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 [19]) for $d_F(P, Q)$ under translations can be obtained from the following observation:

Lemma 5.4. *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 $O(n^2 \text{poly}(\log n, 1/\beta))$.

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 (1987), 1039–51.

[2] P. K. Agarwal, M. Sharir and S. Toledo, Applications of parametric searching in geometric optimization, *J. Algorithms*, 17 (1994), 292–318.

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

[4] H. Alt, J. Blömer, M. Godau, and H. Wagener. Approximation of convex polygons, *Proc. 17th International Colloquium on Automata, Languages and Programming*, LNCS Vol. 443, 1990, 703–716.

[5] H. Alt, C. Knauer and C. Wenk, Matching polygonal curves with respect to the Frchet distance, *Proc. 18th Int. Symp. Theoretical Aspects of Computer Science (STACS)* 2001, 63–74.

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

[7] G. Barequet and S. Har-Peled, Some Variants of Polygon Containment and Minimum Hausdorff Distance under Translation are 3sum-Hard, *Proceedings thAnnual ACM-SIAM Symposium on Discrete Algorithms*1999.

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

[9] 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*, 25–32. Vanderbilt University Press, 1998.

[10] D. Cardoze, L. Schulman, Pattern Matching for Spatia l Point Sets, *Proceedings 39thAnnual IEEE Symposium on Foundations of Computer Science*, 1998.

[11] B. Chazelle. Cutting hyperplanes for divide-and-conquer, *Discrete and Computational Geometry*9 (1993) 145–158.

[12] L.P. Chew and K. Kedem, Improvements on geometric pattern matching problems, *Proceedings 3rd Scand. Workshop on Algorithms Theory*, LNCS Vol. #621, 1992, 318–325.

[13] L.P. Chew, D. Dor, A. Efrat, and K. Kedem, Geometric Pattern Matching in d-Dimensional Space, *Proceedings of the 3rd European Symposium on Algorithms (ESA)* LNCS Vol. #979, 1995, 264–279. Also in *Discrete and Computational Geometry* 21, (1999) 257–274

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

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

[16] A. Efrat, P. Indyk and S. Venkatasubramanian. Pattern Matching for Sets of Segments. *Proceedings 12thAnnual ACM-SIAM Symposium on Discrete Algorithms*, 2001, 295–304.

[17] M. Fréchet, Sur quelques points du calcul fonctionnel, *Rendiconti del Circolo Mathematico di Palermo* 22 (1906), 1–74.

[18] M. 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.

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

[20] J. Holm, K. Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge and biconnectivity, *Proceedings 30thAnnual ACM Symposium on Theory of Computing*, 1998, 79–89.

[21] P. Indyk, R. Motwani, S. Venkatasubramanian, Geometric Matching Under Noise: Combinatorial Bounds and Algorithms, *Proceedings 10thAnnual ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[22] K. Kedem, R. Livne, J. Pach, M. Sharir, On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles, *Discrete and Computational Geometry*, 1 (1986), 59–71.

[23] S. Khanna, R. Motwani, and R. Wilson. On certificates and lookahead in dynamic graph problems. *Algorithmica*, 21 (1998), 377–394.

[24] V. King. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs *Proceedings 40thAnnual IEEE Symposium on Foundations of Computer Science*, 1999.

[25] S. R. Kosaraju, Efficient string matching. manuscript, 1987.

[26] W. Rucklidge, Lower Bounds for the Complexity of the Hausdorff Distance, *Proceedings 5thCandian Conf. Computational Geometry* 1993, 145–150.

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

[28] A. Winzen and H. Niemann. Matching and fusing 3D-polygonal approximations for model generation. In *Proc. IEEE International Conference on Image Processing* (1994), Vol. 1 228–232.