FREIE UNIVERSITÄT
BERLIN

Computing the Hausdorff Distance of Geometric Patterns and Shapes

Helmut Alt, Peter Braß, Michael Godau,
Christian Knauer, and Carola Wenk

B 01-07
November 2001

FACHBEREICH MATHEMATIK UND INFORMATIK
SERIE B · INFORMATIK

Computing the Hausdorff Distance of Geometric Patterns and Shapes

Helmut Alt* Peter Braß*† Michael Godau* Christian Knauer*
Carola Wenk*‡

Abstract

A very natural distance measure for comparing shapes and patterns is the Hausdorff distance. In this article we develop algorithms for computing the Hausdorff distance in a very general case where geometric objects are represented by finite collections of k-dimensional simplices in d-dimensional space. The runtime of our algorithms are for constant dimension polynomial in the size of the input. In addition, we will present more efficient algorithms for special cases like sets of points, or line segments, or triangulated surfaces in three dimensions.

1 Introduction

In application areas like computer vision or pattern recognition it is often necessary to compare shapes and patterns and to have a numerical value describing their similarity or dissimilarity. Mostly, these geometric objects are compact subsets of $\mathbb{R}^2$ or $\mathbb{R}^3$, or in some cases of higher dimensional space. The most natural distance measure for such objects P and Q apparently is the *Hausdorff distance* where for each point on one object we consider the closest point on the other one and then maximize over all these values. More formally, the Hausdorff distance between P and Q is defined as

$$\delta(P,Q) = \max(\tilde{\delta}(P,Q), \tilde{\delta}(Q,P))$$

where

$$\tilde{\delta}(A,B) = \max_{x\in A}\min_{y\in B}||x-y||$$

is the *directed Hausdorff distance* from A to B. We will assume throughout this paper that the underlying metric $||x-y||$ is the Euclidean metric, but many of our considerations are valid for other commonly used metrics, as well. The directed Hausdorff distance is interesting on its own because $\tilde{\delta}(P,Q)$ is a measure of similarity between P and some *part* of Q.

First results on computing the Hausdorff distance between two convex polygons in $\mathbb{R}^2$ were obtained in [7] and for two finite sets of points or line segments in [5]. Other previous research is concerned with *matching* shapes under certain allowable motions minimizing the Hausdorff

*Free University of Berlin, Takustr. 9, 14195 Berlin, Germany; {name}@*inf.fu-berlin.de*

†Part of this research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant Br 1465/5-1 (Heisenberg scholarship)

‡Part of this research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant Al 253/4-3

distance or *simplifying* shapes within a certain tolerance with respect to the Hausdorff distance. For a survey see [6].

In many cases patterns and shapes are modeled as finite sets of points, finite sets of line segments representing for example, curves or two-dimensional shapes, or triangulated surfaces in three dimensions. Here, we will unify all these examples by considering two finite sets P and Q consisting of n and m k-dimensional simplices in d-dimensional space. The Hausdorff distance is defined by considering P and Q as the union of the sets of all points lying in the individual simplices. We give algorithms for this general situation in section 3. Most of the ideas and results of this section appeared originally in the Ph.D. dissertation of Michael Godau [12].

Besides giving algorithms for the general setting we will, in section 4, consider more special cases like point patterns in arbitrary dimensions and sets of triangles in three dimensions and find more efficient algorithms for these instances.

2 Points and Line Segments in Two Dimensions

For two finite sets of points P and Q in arbitrary dimension the straightforward algorithm, calculating all pairwise distances and maximizing, has runtime $O(nm)$. An improved algorithm will be given in Section 4.1. In two dimensions we obtain an asymptotically faster algorithm by first constructing the Voronoi diagram $V(P)$ of P. Then we perform a line sweep on $V(P)$ and Q finding for each point in Q its nearest neighbor in P. The maximum over all distances of these pairs gives the directed Hausdorff distance $\tilde{\delta}(Q, P)$. $\tilde{\delta}(P, Q)$ can be determined analogously and altogether we have an algorithm of runtime $O((m + n)\log(m + n))$.

A generalization of this approach to sets of line segments in two dimensions can be found in [5]. For the sake of completeness we will present the main ideas of this algorithm here. We assume that the line segments are pairwise *not properly intersecting* which means that any pair of segments either does not intersect or has an intersection point that does not lie in the interior of both segments. Observe that, for example, simple polygons or polygonal chains fulfill this condition. The algorithm is based on the following lemma.

Lemma 2.1. *Let P and Q be finite sets of line segments in $\mathbb{R}^2$. Then the directed Hausdorff distance $\tilde{\delta}(P, Q)$ is attained either at an endpoint of a line segment in P or at an intersection point of a line segment in P with an edge of the Voronoi diagram $V(Q)$.*

To see this, observe that when moving along a line segment e in P within a Voronoi cell of a line segment f in Q the distance to f is bitonic, i.e., first monotone decreasing and then monotone increasing. Consequently, the distance is maximal either at an endpoint of e or at a point where e leaves the Voronoi cell of f.

Lemma 2.1 gives $O(nm)$ candidate points where the Hausdorff distance can be attained. Their number can be cut down to linear by the following observation:

If we move along a Voronoi edge s bounding the Voronoi cell of some line segment f in Q the distance to f is again a bitonic function. Consequently, if several line segments of P are intersecting s the maximum distance of an intersection point to f is attained at one of the two extreme intersection points on s. Therefore, for each Voronoi edge of Q we only have to consider at most two intersection points.

These linearly many candidates are then determined by two line sweeps over P and $V(Q)$, one from left to right where each Voronoi edge s of Q is deleted as soon as its leftmost intersection point with an edge in P is found. By a second sweep from right to left all rightmost intersection points are found. In an analogous manner the directed Hausdorff distance from Q to P is found.

Summarizing, we have

Theorem 2.2. *[5] The Hausdorff distance between two sets $P, Q \subset \mathbb{R}^2$ consisting of n and m points or pairwise not properly intersecting line segments can be found in time $O((n+m)\log(n+m))$*

3 The General Case

In this section we will consider the general case of computing the Hausdorff distance between certain geometric objects of arbitrary dimension k in d-dimensional space. When analyzing the algorithms we will assume that k and d are constants.

More precisely, let P, Q be two sets of n and m k-dimensional simplices in $\mathbb{R}^d$. In order to determine the directed Hausdorff distance $\tilde{\delta}(P, Q)$ we assign to each simplex T in Q a distance function d_T: $\mathbb{R}^d \to \mathbb{R}$
$d_T(x) = \min_{y \in T} ||x - y||^2$ for all $x \in \mathbb{R}^d$. (For reasons of simplicity we will work with the square of the Euclidean distance rather than the Euclidean distance itself.) Then the squared distance of any $x \in \mathbb{R}^d$ to Q is given by $g(x) = \min\{d_T(x) | T \text{ simplex of } Q\}$, i.e., by the *lower envelope* of the m functions d_T. We obtain the directed Hausdorff distance $\tilde{\delta}(P, Q)$ as the square root of the *maximum of the lower envelope* g over all points in P.

In order to find this maximum, we will consider each simplex S of P separately and decompose each function d_T into algebraic pieces. The following lemma, which is a generalization of the so-called *0-dimensionality lemma* in [12] and might be of independent interest, gives a necessary condition for the maximum of the lower envelope of a set of convex functions on a compact convex set.

Lemma 3.1. *Let K be a convex, compact subset of $\mathbb{R}^q$, F a set of m continuous convex functions from K to $\mathbb{R}$, and $g : K \to \mathbb{R}$ their lower envelope. Assume that $c = \max_{x \in K} g(x)$ is not attained at the boundary of K, and let $a \in \mathrm{Int}(K)$ be the smallest point with respect to the lexicographic order with $g(a) = c$. Then $m \geq q + 1$ and:*

a) $f(a) = c$ for at least $q + 1$ functions $f \in F$.

b) If all functions in F are twice continuously differentiable then there is a neighborhood U of a so that these $q + 1$ functions coincide in no point in U other than a.

Proof. The maximum of g must be attained somewhere in K because of the compactness of K. So if it does not occur at the boundary it must occur in the interior. Since $\{x \in K | g(x) = c\}$ is a compact subset of $\mathrm{Int}(K)$ its minimal element a with respect to the lexicographic order exists . Let $P_a \in \mathbb{R}^{q+1}$ be the point (a, c).

Let $M = \{f \in F | f(a) = c\}$, then $M \neq \emptyset$ and $f(a) > c$ for all $f \notin M$. Consider the *graphs* of the functions $f \in F$, i.e., the sets $G_f = \{(x, f(x)) | x \in K\}$. Because of the convexity of the functions in F, there exists for each $f \in M$ a hyperplane H_f that is tangent to G_f in P_a. More precisely, if H_f^+ denotes the upper halfspace determined by H_f, then $P_a \in H_f$ and $G_f \subset H_f^+$, i.e., $H_f(x) \leq f(x)$ for all $x \in K$, where in the latter inequality we identified H_f with the affine function whose graph is H_f.

We now claim

Claim 3.2. $\bigcap_{f\in M} H_f = \{P_a\}$.

Assume otherwise. Then there exists a straight line $l \subset \bigcap_{f\in M} H_f$ with $P_a \in l$. l lies below all $G_f, f \in F$. Consider the projection l' of l to K and consider l as an affine function from l' to $\mathbb{R}$. Then $a \in l'$, $l(a) = f(a) = c$ for all $f \in M$, and $l(x) \leq f(x)$ for all $x \in l', f \in F$, i.e., $l(x) \leq g(x)$ for all $x \in l'$. Consider some convex neighborhood $V \subset K$ of a and the line segment $s = V \cap l'$. Now, there are two possibilities:

1. If l is a constant function then $l(x) = l(a) = c$ for all $x \in s$. Since $l(x) \leq g(x)$ and c is the maximal value of g, $g(x) = c$ for all $x \in s$. Since a lies in the interior of s this contradicts the lexicographic minimality of a.

2. If l is not constant then $l(x) > c$ for all $x \in s$ on one side of a. Since $l(x) \leq g(x)$ this contradicts the maximality of c.

So in any case, we get a contradiction which proves claim 3.2.

claim 3.2 states that the hyperplanes $H_f, f \in M$ intersect in one point. Therefore, it must be $|M| \geq q+1$ which proves part a) of the lemma.

In order to prove part b) select a subset $M' \subset M$ of $q+1$ elements with $\bigcap_{f\in M'} H_f = \{P_a\}$. Then, considering each H_f as a function from $\mathbb{R}^q$ to $\mathbb{R}$, we have

Claim 3.3. *There is a constant $\alpha > 0$ such that for all $x \in \mathbb{R}^q$ there are functions $f_1, f_2 \in M'$ with $H_{f_1}(x) - H_{f_2}(x) \geq \alpha\|x-a\|$.*

To see this consider the sphere $C = \{x \in \mathbb{R}^q \mid \|x-a\| = 1\}$ and let $\alpha = \min_{x\in C} \max_{f_1,f_2\in M'} H_{f_1}(x) - H_{f_2}(x)$ which exists because of the compactness of C. Also $\alpha > 0$ since $\bigcap_{f\in M'} H_f = P_a$. For arbitrary $x \in \mathbb{R}^q, x \neq a$ let x' be the point on C on the straight line through a and x, i.e., $x' = a + (x-a)/\|x-a\|$. Then there are $f_1, f_2 \in M'$ with $h(x') = H_{f_1}(x') - H_{f_2}(x') \geq \alpha$. Since h is an affine function with $h(a) = 0$ we have $h(x) = h(x')\|x-a\| \geq \alpha\|x-a\|$, which proves claim 3.3.

On the other hand, we have

Claim 3.4. *There is a constant $\beta > 0$ such that for all $x \in K$*

$$H_f(x) \leq f(x) \leq H_f(x) + \beta\|x-a\|^2.$$

The first inequality was explained before, the second one is obtained by the Taylor-expansion of f about a, since the second derivatives of f are bounded in the compact set K.

To finish the proof of Lemma 3.1 consider the functions f_1, f_2 from claim 3.3 for $x = a+\varepsilon$, where $\varepsilon \in \mathbb{R}^q$ is sufficiently short such that $a+\varepsilon \in K$. Then, using claims 3.3 and 3.4 we get:

$$\begin{aligned} f_1(a+\varepsilon) - f_2(a+\varepsilon) &\geq H_{f_1}(a+\varepsilon) - H_{f_2}(a+\varepsilon) - \beta\|\varepsilon\|^2 \\ &\geq \alpha\|\varepsilon\| - \beta\|\varepsilon\|^2 \end{aligned}$$

The latter expression is greater then 0 for sufficiently small $\|\varepsilon\| > 0$ which shows that f_1 and f_2 do not coincide in some neighborhood of a except in a itself. $\square$

Lemma 3.1 can be applied to the problem of computing the Hausdorff-distance between P and Q in order to obtain a finite number of candidates where the directed Hausdorff distance $\tilde{\delta}(P,Q)$ can occur.

First let us have a closer look at the n distance functions determined by the simplices T in Q. For a face T' of T let $A_{T'}$ be the affine space spanned by T' and $f_{T'} : \mathbb{R}^d \to \mathbb{R}$ the function that assigns to $x \in \mathbb{R}^d$ its squared distance to $A_{T'}$. Observe that each $f_{T'}$ is a quadratic and, hence, a convex and twice continuously differentiable function. Also, for the restriction of the distance function d_T to $V_{T'}$ where $V_{T'}$ is the Voronoi cell of T' within the Voronoi diagram of all sites of T (see Figure 1) we have $d_T|V_{T'} = f_{T'}|V_{T'}$, so d_T is a convex, piecewise quadratic function with $2^k - 1$ pieces determined by all Voronoi cells.

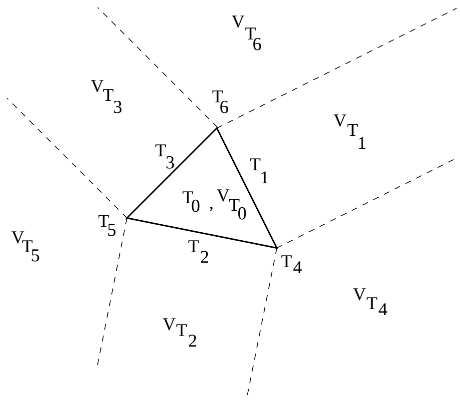


Figure 1: Voronoi cells of a triangle $T_0 \subset \mathbb{R}^2$ and its faces.

Lemma 3.5. *Let S be a k-dimensional simplex in P. Then the directed Hausdorff-distance $\tilde{\delta}(S,Q)$ occurs at some point a on some face S' of S of dimension $k' \leq k$ where $k'+1$ distance functions $f_{T'}(x)$ for faces T' of simplices in Q have the same value. Furthermore the point a is isolated in the sense that in S' there is a neighborhood U of a so that for no point in U other than a these $k'+1$ functions have the same value.*

Proof. Let S' be a face of S of minimal dimension k' that contains a point a where the directed Hausdorff distance $\tilde{\delta}(S,Q)$ is attained, i.e., $\tilde{\delta}(S,Q) = \min_{x \in Q} \|a - x\|$. Observe that a cannot lie at the boundary of S'.

Let us identify the affine span $A_{S'}$ with $\mathbb{R}^{k'}$, consider S' as a compact subset of $\mathbb{R}^{k'}$, and consider all distance functions d_T and $f_{T'}$ as restricted to $A_{S'}$, i.e., $\mathbb{R}^{k'}$. For $x \in A_{S'}$ the distance $\tilde{\delta}(\{x\},Q)$ of x to Q is given by the lower envelope $g(x)$ of the functions $d_T(x)$ for all simplices T of Q.

Assume that $a \in S'$ is the point where g attains its maximum that is minimal with respect to the lexicographic order. Since all d_T are convex we can apply Lemma 3.1 a) to this setting and, therefore, $k'+1$ distance functions d_T for simplices T in Q have the same value $c = g(a)$ in a.

Therefore, also $k'+1$ distance functions $f_{T'}$ for faces T' of simplices in Q have the value c in a. Let $F = \{f_{T'} | T' \text{ face of a simplex in } Q, f_{T'}(a) = c\}$. Then, we have:

Claim 3.6. *There is a neighborhood V of a so that the lower envelope h of the functions in F restricted to V has a maximum in a and a is the lexicographically smallest point with that property in V.*

To see this, consider the complement of F, i.e., the set $\bar{F} = \{f_{T'} | T' \text{ face of a simplex in } Q, f_{T'}(a) \neq g(a)\}$. Since all functions involved are continuous, there must be a neighborhood V of a so that $f(x) \neq g(x)$ for all $x \in V$ and all $f \in \bar{F}$. On the other hand for any $x \in \mathbb{R}^{k'}$ there is a face T' of a simplex in Q with $g(x) = f_{T'}(x)$. Consequently, for any $x \in V$ there is an $f \in F$ with $g(x) = f(x)$. Therefore in V, the lower envelope of the functions in F does not exceed g, i.e., $h(x) \leq g(x)$ for all $x \in V$. On the other hand, $h(a) = g(a)$ so a is a point in V where the lower envelope h attains its maximum, because g does. There cannot be any point b in V lexicographically smaller than a with this property, since we would have $g(b) \geq h(b) = h(a) = g(a)$, i.e., $g(b) = g(a)$ which would contradict the lexicographic minimality of a. This proves claim 3.6.

Consider a convex compact subset $K \subset V$ that contains a in its interior and apply Lemma 3.1 b) to the functions in F restricted to K. This proves Lemma 3.5. □

Finally, in order to compute $\tilde{\delta}(P, Q)$, we will consider all simplices in P and their faces separately which makes $n(2^k - 1)$ simplices S of dimension k or lower.

Lemma 3.5 gives a straightforward algorithm for computing $\tilde{\delta}(S, Q)$:

1. For each selection $T_1, \ldots, T_{k'+1}$ of faces of simplices of Q, where k' is the dimension of S:

   1a. solve the system of equations consisting of k' quadratic equations

   $$f_{T_1}(x) = f_{T_2}(x), \ldots, f_{T_{k'}}(x) = f_{T_{k'+1}}(x)$$

   and $d - k'$ linear equations defining the affine span A_S.

   1b. For each isolated solution x of this system test whether

   a) $x \in S$
   b) $x \in V_{T_i}$ for $i = 1, \ldots, d + 1$
   c) $f_T(x) \geq f_{T_1}(x)$ for all faces T of simplices in Q with $x \in V_T$

2. For all x which passed the test in step 1b consider $f_{T_1}(x)$ and return the maximum of all these values as $\tilde{\delta}(S, Q)$.

This procedure is carried out for each face S of each simplex in P and the maximum value obtained is $\tilde{\delta}(P, Q)$. Likewise, $\tilde{\delta}(Q, P)$ and, thus, $\delta(P, Q)$ can be determined.

For the analysis of the runtime we observe that there are $n(2^k - 1) = O(n)$ faces of simplices in P. For each of them, at most $\binom{m(2^k-1)}{k+1} = O(m^{k+1})$ selections of faces are made in step 1. For each selection we spend constant time in step 1a and time $O(m)$ in step 1b, which gives a total runtime of $O(nm^{k+2})$ for computing $\tilde{\delta}(P, Q)$ and $O(nm^{k+2} + n^{k+2}m)$ for computing $\delta(P, Q)$.

This result is summarized in the following theorem where, as in the theorems thereafter, we only give the runtime for computing the directed Hausdorff distance. Note that if this runtime is $T(n, m)$ then the Hausdorff distance itself can be computed in time $O(T(n, m) + T(m, n))$.

Theorem 3.7. *Given two sets $P, Q \subseteq \mathbb{R}^d$ of n and m k-dimensional simplices, we can compute $\tilde{\delta}(P, Q)$ in $O(nm^{k+2})$ time.*

Besides this straightforward approach, we also can use results obtained in the theory of lower envelopes. Essentially, Lemma 3.5 says that the maximum of the lower envelope is attained at a *vertex* of the lower envelope of the functions restricted to some face of the polytope S. Fortunately, vertices of lower envelopes can be determined efficiently. In fact, Agarwal et al. [1, 14] give a Las Vegas algorithm for computing the vertices of the lower envelope of m partially defined algebraic functions of constant degree in $O(m^{d+\varepsilon})$ time. Since in our case we have to do this for all $O(n)$ faces of simplices in P, we obtain

Theorem 3.8. *Given two sets $P, Q \subseteq \mathbb{R}^d$ of n and m k-dimensional simplices, we can compute $\tilde{\delta}(P,Q)$ in $O(nm^{k+\varepsilon})$ randomized expected time, for any $\varepsilon > 0$.*

4 Efficient Algorithms for Special Problems

4.1 Point Patterns

The Voronoi based approach of section 2 for finite sets of points does not yield efficient algorithms in higher dimensions, because the complexity of constructing the Voronoi cells is too large. So up to now only the trivial $O(nm)$ algorithm for computing all pairwise distances $\mathrm{d}(x,y)$ was known. In this section we will develop more efficient algorithms for the case $d \geq 3$ which are randomized in the Las Vegas sense. Let us briefly outline our basic approach: First we develop an algorithm to compute $\tilde{\delta}(P,Q)$, that performs well, when n is large compared to m (Lemma 4.1), and then we describe another approach that is better when m is larger than n (Lemma 4.2). Finally we combine the two approaches in Theorem 4.3.

Lemma 4.1. *We can compute $\tilde{\delta}(P,Q)$ in $O(n^{1-1/\lfloor\frac{d+1}{2}\rfloor} m \log n + n \log n)$ randomized expected time.*

Proof. We first describe a randomized algorithm that computes $\tilde{\delta}(P,Q)$ in $O(m^{\lfloor\frac{d+1}{2}\rfloor} + n \log m)$ expected time, which we speed up with a simple batching technique afterwards. First observe that

$$||x-y|| \leq \delta \iff \langle y,y\rangle - 2\langle x,y\rangle \leq \delta^2 - \langle x,x\rangle.$$

For fixed $y \in Q$, the d-variate function $h_y(x) := \langle y,y\rangle - 2\langle x,y\rangle$ is linear, and its graph is a hyperplane in $\mathbb{R}^{d+1}$. Thus

$$\begin{aligned}\tilde{\delta}(P,Q) \leq \delta &\iff \forall x \in P\ \exists y \in Q : ||x-y|| \leq \delta \\ &\iff \forall x \in P : \min_{y \in Q} h_y(x) \leq \delta^2 - \langle x,x\rangle.\end{aligned}$$

So if we define the lower envelope $\Lambda_Q(x) := \min_{y\in Q} h_y(x)$, we can conclude that

$$(\tilde{\delta}(P,Q))^2 = \max_{x \in P}(\Lambda_Q(x) + \langle x,x\rangle).$$

With a randomized algorithm from [13] we can preprocess the m hyperplanes h_y in expected $O(m^{\lfloor(d+1)/2\rfloor})$ time into a data structure that supports $O(\log m)$ time ray-shooting queries to their lower envelope. After that preprocessing step we can evaluate the function Λ_Q in $O(\log m)$ time by performing a vertical ray-shooting query. The overall runtime of this method is $O(m^{\lfloor(d+1)/2\rfloor} + n \log m)$. We can gain a significant speedup with a simple batching technique. To this end we distinguish two cases:

$n \le m^{\lfloor (d+1)/2 \rfloor}$:
We partition Q into $g = \lceil m/n^{1/\lfloor (d+1)/2 \rfloor} \rceil$ groups Q^i of size at most $m' := n^{1/\lfloor (d+1)/2 \rfloor} \le m$ each. For each group we run the algorithm we just described, keeping track of the maximum value of $\Lambda_{Q^i}(x) + \langle x, x \rangle$ for each point $x \in P$. The total time spent is $O(g(m'^{\lfloor (d+1)/2 \rfloor} + n \log m')) = O(gn \log n) = O(n^{1-1/\lfloor (d+1)/2 \rfloor} m \log n)$.

$m^{\lfloor (d+1)/2 \rfloor} \le n$:
In that case the runtime is $O(n \log m) = O(n \log n)$.

□

The second algorithm we describe performs well when m is large compared to n.

Lemma 4.2. *We can compute $\tilde{\delta}(P, Q)$ in $O(nm^{1-1/\lfloor (d+1)/2 \rfloor} \log m + m \log m)$ randomized expected time.*

Proof. We first describe a randomized algorithm that computes $\tilde{\delta}(P, Q)$ in $O(n^{\lfloor (d+1)/2 \rfloor} \log m + m \log n)$ expected time, which we speed up with a simple batching technique afterwards. For each point $x \in P$ the algorithm (recursively) computes one closest point $y_x \in Q$. We proceed as follows:

1. Randomly, under uniform distribution, choose a subset $Q' \subseteq Q$ of size $m/2$.
2. For each point $x \in P$ recursively compute one closest point $y'_x \in Q'$. Set $\delta_x = ||x - y'_x||$.
3. Compute the set $Q'' = \{y \in Q - Q' \mid \exists x \in P : ||x - y|| \le \delta_x\}$, i.e., the set of all points in $Q - Q'$ that are closer to x than y'_x is, for some $x \in P$.
4. For each $x \in P$ compute a closest point $y''_x \in Q''$ by brute force in $O(|Q''|)$ time. Set y_x to either y'_x or y''_x, depending on which of them is closer.

The correctness of this algorithm is obvious. In order to analyze the complexity, we have to describe how we compute the set Q'' in step 3. The approach we take is similar to the one in the previous section. First observe that

$$\exists x \in P : ||x - y|| \le \delta_x \iff \exists x \in P : \langle x, x \rangle - 2\langle x, y \rangle - \delta_x^2 \le \langle y, y \rangle.$$

For fixed $x \in P$, the d-variate function $h_x(y) := \langle x, x \rangle - 2\langle x, y \rangle - \delta_x^2$ is linear, and its graph is a hyperplane in $\mathbb{R}^{d+1}$. Thus

$$\exists x \in P : ||x - y|| \le \delta_x \iff \min_{x \in P} h_x(y) \le \langle y, y \rangle.$$

So if we define the lower envelope $\Lambda_P(y) := \min_{x \in P} h_x$, it follows that

$$y \in Q'' \iff \Lambda_P(y) - \langle y, y \rangle \le 0.$$

We can preprocess the n hyperplanes h_x in expected $O(n^{\lfloor (d+1)/2 \rfloor})$ time into a data structure that supports $O(\log n)$ time ray-shooting queries to their lower envelope with a randomized algorithm from [13]. After that preprocessing step we can evaluate the function Λ_P in $O(\log n)$ time by performing a vertical ray-shooting query. The overall runtime of this method is $O(n^{\lfloor (d+1)/2 \rfloor} +$

$m \log n)$. We can conclude that the expected runtime of the algorithm, $T(n,m)$, obeys the following recursion:

$$T(n,m) \leq T(n,m/2) + O(n^{\lfloor (d+1)/2 \rfloor} + m \log n) + nE(|Q''|). \tag{1}$$

Now we prove that the expected value $E(|Q''|) \leq n$. To see this, fix some point $x \in P$, and let K_x denote the random variable that counts the number of points in Q that are closer to x than y'_x is. By linearity of expectation we have $E(|Q''|) \leq nE(K_x)$. To estimate $E(K_x)$, we consider Q to be ordered in increasing distance to x, i.e., $Q = (y_1, \ldots, y_m)$ with $||x - y_i|| \leq ||x - y_{i+1}||$ for $1 \leq i < m$. The probability that $y_i \in Q'$ is $1/2$, and the probability that y_i is the first element in the sequence Q that lies in Q' is $1/2^{i+1}$. If y_i is the first element in this sequence that lies in Q', then $y_j \in Q''$ for all $j < i$, so $E(K_x) \leq \sum_{i \geq 0} i/2^{i+1} = 1$.

Since $d \geq 3$, recursion (1) now reads as follows:

$$T(n,m) \leq T(n,m/2) + O(n^{\lfloor (d+1)/2 \rfloor} + m \log n),$$

so the expected runtime of the algorithm is $O(n^{\lfloor (d+1)/2 \rfloor} \log m + m \log n)$.

Applying the same batching technique as in the proof of Lemma 4.1 with n and m exchanged we obtain the bound claimed. $\square$

Now we combine the two approaches from Lemma 4.1 and Lemma 4.2 to compute $\tilde{\delta}(P,Q)$: For $m \leq n$ we use the algorithm of Lemma 4.1, whereas for $n \leq m$ we use the algorithm of Lemma 4.2.

Summarizing, we have

Theorem 4.3. *Given two point sets $P, Q \subseteq \mathbb{R}^d$ of n, and m points, respectively, we can compute $\tilde{\delta}(P,Q)$ in $O((n + m + (nm)^{1-\frac{1}{2\lfloor \frac{d+1}{2} \rfloor}}) \log(n+m))$ randomized expected time.*

4.2 Triangles in Three Dimensions

Surfaces in three dimensions are in many cases modeled in a triangulated form. We will give algorithms that compute the Hausdorff distance not only for triangulated surfaces but, more generally, forsets of triangles in $\mathbb{R}^3$. For the remainder, P and Q are sets of n and m triangles. Notice, that Theorem 3.8 gives a bound of $O(nm^{2+\varepsilon})$ for computing the directed Hausdorff distance $\tilde{\delta}(P,Q)$. By a different approach using linesweep on the affine spans of the triangles involved and parametric search we can obtain a bound of $O(nm^2)$ times polylogarithmic factors. Here, we will present a third approach, which is asymptotically faster than cubic in the input size. However we need to require that within one set the triangles do not intersect properly, i.e., any two distinct triangles in the set do not intersect in their relative interiors.

Let us assume, that $\delta > 0$ is fixed and let P^0 denote the set of vertices of P. We have that $\tilde{\delta}(P,Q) \leq \delta$ iff for each point of P there is a point of Q within distance at most δ. Therefore it is reasonable to look at the set of all points of distance at most δ to Q; in the following $\mathsf{nh}_\delta(Q) = \{x \in \mathbb{R}^3 \mid \mathrm{d}(x,Q) \leq \delta\}$ denotes the *δ-neighborhood* of Q, and $\mathsf{bd}_\delta(Q) = \{x \in \mathbb{R}^3 \mid \mathrm{d}(x,Q) = \delta\}$ denotes the *boundary of the δ-neighborhood* of Q. Our results are based on the following simple observation that is subsumed in Lemma 4.4 below: The directed Hausdorff distance from P to Q is at most δ iff all vertices of P are contained in the δ-neighborhood of Q, and none of the triangles in P intersects the boundary $\mathsf{bd}_\delta(Q)$.

Lemma 4.4. *Let and P, Q be compact sets in $\mathbb{R}^3$, and $\delta > 0$. Then*

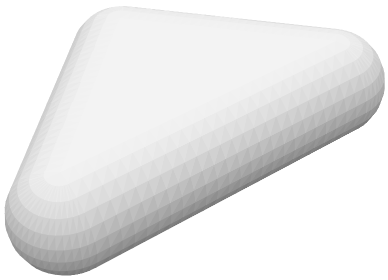
$$\tilde{\delta}(P,Q) < \delta \iff P^0 \subset \mathsf{nh}_\delta(Q) \text{ and } P \cap \mathsf{bd}_\delta(Q) = \emptyset.$$

So we are left with the task of verifying whether $P^0 \subset \mathsf{nh}_\delta(Q)$ ('inclusion property'), and $P \cap \mathsf{bd}_\delta(Q) = \emptyset$ ('intersection property').

The inclusion property can easily be checked in $O(nm)$ steps by computing the distance of each vertex in P^0 to each triangle in Q in $O(1)$ time. Note that this method also identifies the triangles $\Delta \in P$ that contain a vertex outside of $\mathsf{nh}_\delta(Q)$.

Lemma 4.5. *We can decide whether $P^0 \subset \mathsf{nh}_\delta(Q)$ in $O(nm)$ time.*

In the following we will describe an efficient algorithm to verify the intersection property.



For a triangle Δ, the set $\mathsf{nh}_\delta(\Delta)$ (called a *kreplach* in [4]) is the convex hull of three copies of a δ-ball centered at the vertices of Δ; it is the (non-disjoint) union of three balls of radius δ around the vertices of Δ, three cylinders of radius δ around the edges of Δ, and a triangular prism of height 2δ containing Δ in its center.

As usual, we mean by the complexity of $\mathsf{bd}_\delta(Q)$ its number of vertices, edges, and 2-faces. By a result of Agarwal and Sharir [3, 4], the boundary $\mathsf{bd}_\delta(Q)$ has complexity $O(m^{2+\varepsilon})$, and can be computed in $O(m^{2+\varepsilon})$ randomized expected time for any $\varepsilon > 0$; the algorithm computes a description of $\mathsf{bd}_\delta(Q)$ where each 2-face is partitioned into semialgebraic[1] surface patches of constant description complexity. Each of these surface patches is contained in one spherical, cylindrical, or triangular portion of $\mathsf{bd}_\delta(\Delta)$ for some $\Delta \in Q$ (the same Δ that contains the corresponding 2-face), and is bounded by at most four arcs. Each arc in turn is part of the intersection of the portion of the boundary that contains the patch with either a plane, or an δ-sphere, or an δ-cylinder. A polynomial expression defining a patch is formed by the conjunction of five atomic expressions of degree at most two: one polynomial equation describing the portion of $\mathsf{bd}_\delta(\Delta)$ that contains the patch (i.e., a cylinder, a sphere, or a plane), and at most four polynomial inequalities defining the arcs (again these are equations describing a cylinder, a sphere, or a plane).

In order to verify the intersection property we need a method to detect intersections between the triangles in P and the surface patches of bd_δ. We will apply a standard approach suggested in [9] and [8], and transform this problem to a semialgebraic point-location problem.

Lemma 4.6. *Let Ω be a set of k semialgebraic sets of constant description complexity in $\mathbb{R}^3$. For any $\varepsilon > 0$ we can build a data structure of size $O(k^{16+\varepsilon})$ in $O(k^{16+\varepsilon})$ randomized expected time, such that for any query triangle Δ we can decide in $O(k^\varepsilon)$ time whether Δ intersects Ω.*

[1] A set $S \subseteq \mathbb{R}^d$ is called *semialgebraic* if it satisfies a *polynomial expression*. A polynomial expression is any finite Boolean combination of *atomic polynomial expressions*, which are of the form $P(\mathbf{x}) \leq 0$, where $P \in \mathbb{R}[x_1, \ldots, x_d]$ is a d-variate polynomial.

Proof. Let $\Delta(p_1, p_2, p_3; x)$ be a polynomial expression that defines a triangle Δ depending on its three vertices p_1, p_2, and p_3, i.e., $\Delta = \{x \in \mathbb{R}^3 \mid \Delta(p_1, p_2, p_3; x) \text{ holds}\}$; we can form Δ as the conjunction of three linear inequalities, and one linear equation. Let $\Gamma(q; x)$ be a polynomial expression that defines a set $\Gamma \in \Omega$, depending on a sequence of real parameters q, i.e., $\Gamma = \{x \in \mathbb{R}^3 \mid \Gamma(q; x) \text{ holds}\}$. For some fixed Γ, consider the set $C_\Gamma = \{(p_1, p_2, p_3) \in \mathbb{R}^9 \mid (\exists x : \Delta(p_1, p_2, p_3; x) \wedge \Gamma(q; x)) \text{ holds}\}$. If we look at $\mathbb{R}^9$ as the configuration space of the set of all triangles in 3-space, then C_Γ is the set of (the parameters of) all triangles that intersect Γ. By quantifier elimination [11] we can find a polynomial expression $C_\Gamma(q; p_1, p_2, p_3)$ that defines C_Γ; therefore this set is semialgebraic, too.

Let F denote the set of $O(k)$ many polynomials that appear in the atomic polynomial expressions forming the expressions C_Γ. With an algorithm from [8] we can compute a point-location data structure of size $O(k^{16+\varepsilon})$ in $O(k^{16+\varepsilon})$ time for the varieties defined by F. Since the signs of all polynomials in F, and therefore the validity of each polynomial expression C_Γ is constant for each cell of the decomposition of $\mathbb{R}^9$ induced by these varieties, the claim follows. ☐

Lemma 4.7. *For any $\varepsilon > 0$ we can decide whether $P \cap \mathsf{bd}_\delta(Q) = \emptyset$ in $O(nm^\varepsilon + m^{2+\varepsilon} n^{15/16+\varepsilon})$ randomized expected time.*

Proof. In a first step we compute a description of $\mathsf{bd}_\delta(Q)$ with the algorithm of Agarwal/Sharir. This can be done in $O(m^{2+\varepsilon})$ time, and yields a set of $O(m^{2+\varepsilon})$ semialgebraic surface patches of constant description complexity that partition the boundary of $\mathsf{nh}_\delta(Q)$. Now we distinguish two cases:

$m^{32} \le n$: We run the algorithm of Lemma 4.6 to build a data structure of size $O(m^{32+\varepsilon})$ in $O(m^{32+\varepsilon})$ time that supports triangle intersection queries to $\mathsf{bd}_\delta(Q)$ in $O(m^\varepsilon)$ time, and then we query this data structure with all triangles in P to test for intersections in $O(nm^\varepsilon)$ steps. The total time spent is $O(m^{32+\varepsilon} + nm^\varepsilon) = O(nm^\varepsilon)$.

$n \le m^{32}$: We partition $\mathsf{bd}_\delta(Q)$ into $g = \lceil m^{2+\varepsilon}/n^{1/16} \rceil$ groups of $k = n^{1/16} \le m^2$ surface patches each. For each group, we run the algorithm of Lemma 4.6 to build a data structure of size $O(k^{16+\varepsilon})$ in $O(k^{16+\varepsilon})$ time that supports triangle intersection queries in $O(k^\varepsilon)$ time, and then we query this data structure with all triangles in P to test for intersections in $O(nk^\varepsilon)$ steps. The total time spent is $O(g(k^{16+\varepsilon} + nk^\varepsilon)) = O(gn^{1+\varepsilon/16}) = O(m^{2+\varepsilon} n^{15/16+\varepsilon})$.

Note that this algorithm can also determine the triangles $\Delta \in P$ that intersect $\mathsf{bd}_\delta(Q)$. ☐

Putting Lemma 4.5 and Lemma 4.7 together, we obtain

Lemma 4.8. *For any $\varepsilon > 0$ we can compute the set $X = \{\Delta \in P \mid \tilde{\delta}(\Delta, Q) > \delta\}$ in $O(nm + m^{2+\varepsilon} n^{15/16+\varepsilon})$ randomized expected time.*

With the well known Clarkson/Shor technique, c.f. [10], we can easily turn the algorithm for the decision problem into a randomized procedure that actually computes the minimal distance, and obtain

Theorem 4.9. *Given two sets $P, Q \subseteq \mathbb{R}^3$ of n and m triangles with the property that no two triangles in Q intersect in their interior, we can compute $\tilde{\delta}(P, Q)$ in $O(nm \log n + m^{2+\varepsilon} n^{15/16+\varepsilon})$ randomized expected time, for any $\varepsilon > 0$.*

Proof. We follow a strategy similar to that proposed in [2]. Initially we set $\delta = 0$ and $X = P$. Then we repeat the following steps until X becomes empty: Choose a random triangle $\Delta \in X$, and compute $\delta' = \tilde{\delta}(\Delta, Q)$ in $O(m^{2+\varepsilon})$ time according to Theorem 3.8. Set δ to $\max(\delta, \delta')$. Now compute the set $X' = \{\Delta \in X \mid \tilde{\delta}(\Delta, Q) > \delta\}$ in $O(nm + m^{2+\varepsilon} n^{15/16+\varepsilon})$ time with the algorithm from Lemma 4.8. Finally set X to X'.

Obviously the last value of δ will be $\tilde{\delta}(P, Q)$. As is shown in [10], the expected number of iterations is $O(\log n)$, and therefore the expected time to compute $\tilde{\delta}(P, Q)$ with this algorithm is $O(nm \log n + m^{2+\varepsilon} n^{15/16+\varepsilon})$. □

References

[1] P. K. Agarwal, B. Aronov, and M. Sharir. Computing envelopes in four dimensions with applications. *SIAM J. Comput.*, 26:1714–1732, 1997.

[2] P. K. Agarwal and M. Sharir. Efficient randomized algorithms for some geometric optimization problems. *Discrete Comput. Geom.*, 16:317–337, 1996.

[3] P. K. Agarwal and M. Sharir. Motion planning of a ball amid polyhedral obstacles in three dimensions. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 21–30, 1999.

[4] P. K. Agarwal and M. Sharir. Pipes, cigars, and kreplach: The union of Minkowski sums in three dimensions. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 143–153, 1999.

[5] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. *Ann. Math. Artif. Intell.*, 13:251–266, 1995.

[6] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[7] M. J. Atallah. A linear time algorithm for the Hausdorff distance between convex polygons. *Inform. Process. Lett.*, 17:207–209, 1983.

[8] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoret. Comput. Sci.*, 84:77–105, 1991.

[9] B. Chazelle and M. Sharir. An algorithm for generalized point location and its application. *J. Symbolic Comput.*, 10:281–309, 1990.

[10] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[11] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes Comput. Sci.*, pages 134–183. Springer-Verlag, 1975.

[12] M. Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions.* PhD thesis, Department of Computer Science, Freie Universitaet Berlin, 1999.

[13] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10(2):215–232, 1993.

[14] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.