Hausdorff Distance under Translation for Points and Balls*

Pankaj K. Agarwal† Sariel Har-Peled‡ Micha Sharir§ Yusu Wang†

August 5, 2005

Abstract

We study the shape matching problem under the Hausdorff distance and its variants. In the first part of the paper, we consider two sets $\mathcal{A}, \mathcal{B}$ of balls in $\mathbb{R}^d$, $d = 2, 3$, and wish to find a translation t that minimizes the Hausdorff distance between $\mathcal{A} + t$, the set of all balls in $\mathcal{A}$ shifted by t, and $\mathcal{B}$. We consider several variants of this problem. First, we extend the notion of Hausdorff distance from sets of points to sets of balls, so that each ball has to be matched with the nearest ball in the other set. We also consider the problem in the standard setting, by computing the Hausdorff distance between the unions of the two sets (as point sets). Second, we consider either all possible translations t (as is the standard approach), or consider only translations that keep the balls of $\mathcal{A} + t$ disjoint from those of $\mathcal{B}$. We propose several exact and approximation algorithms for these problems. In the second part of the paper, we note that the Hausdorff distance is sensitive to outliers, and thus consider two more robust variants—the root-mean-square (rms) and the summed Hausdorff distance. We propose efficient approximation algorithms for computing the minimum rms and the minimum summed Hausdorff distances under translation, between two point sets in $\mathbb{R}^d$. In order to obtain a fast algorithm for the summed Hausdorff distance, we propose a deterministic efficient dynamic data structure for maintaining an ε-approximation of the 1-median of a set of points in $\mathbb{R}^d$, under insertions and deletions.

*P.A. is supported by by NSF grants EIA-98-70724, EIA-99-72879, ITR-333-1050, CCR-97-32787, and CCR-00-86013, and by a grant from the U.S.-Israeli Binational Science Foundation (jointly with M.S.). S.H. is supported by NSF CAREER award CCR-01-32901. M.S. is also supported by NSF Grants CCR-97-32101 and CCR-00-98246, by a grant from the Israel Science Fund (for a Center of Excellence in Geometric Computing), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. Y.W. is supported by NSF grants ITR-333-1050 and CCR-02-04118.
A prelimary version of this paper appeared in *Proc. 19th Annual Sympos. Comput. Geom.*, 2003, pp. 282–291.

†Department of Computer Science, Duke University, Durham, NC 27708-0129, U.S.A. Email: pankaj, wys@cs.duke.edu.

‡Department of Computer Science, University of Illinois, Urbana, IL 61801, U.S.A. Email: sariel@cs.uiuc.edu.

§School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA. Email: michas@post.tau.ac.il.

1 Introduction

The problem of shape matching in two and three dimensions arises in a variety of applications, including computer graphics, computer vision, pattern recognition, computer aided design, and molecular biology [6, 15, 23]. For example, proteins with similar shapes are likely to have similar functionalities, therefore classifying proteins (or their fragments) based on their shapes is an important problem in computational biology. Similarly, the proclivity of two proteins binding with each other also depends on their shapes, so shape matching is central to the so-called *docking* problem in molecular biology [15].

Informally, the shape-matching problem can be described as follows: Given a distance measure between two sets of objects in $\mathbb{R}^2$ or $\mathbb{R}^3$, determine a transformation, from an allowed set, that minimizes the distance between the sets. In many applications, the allowed transformations are all possible rigid motions. However, in certain applications there are constraints on the allowed transformations. For example, in matching the pieces of a jigsaw puzzle, it is important that no two pieces overlap each other in their matched positions. Another example is the aforementioned docking problem, where two molecules bind together to form a compound, and, clearly, at this docking position the molecules should occupy disjoint portions of space [15]. Moreover, because of efficiency considerations, one sometimes restricts further the set of allowed transformations, most typically to translations only.

Several distance measures between objects have been proposed, varying with the kind of input objects and the application. One common distance measure is the *Hausdorff distance* [6], originally proposed for point sets. In this paper we adopt this measure, extend it to sets of non-point objects (mainly, disks and balls), and apply it to several variants of the shape matching problem, with and without constraints on the allowed transformations. In many applications (e.g., molecular biology), shapes can be approximated by a finite union of balls [7], which is therefore the main type of input assumed in the first part of this paper.

1.1 Problem statement

Let $\mathcal{A}$ and $\mathcal{B}$ be two (possibly infinite) sets of geometric objects (e.g., points, balls, simplices) in $\mathbb{R}^d$, and let $d : \mathcal{A} \times \mathcal{B} \to \mathbb{R}$ be a distance function between objects in $\mathcal{A}$ and in $\mathcal{B}$. For $a \in \mathcal{A}$, we define $d(a, \mathcal{B}) = \inf_{b \in \mathcal{B}} d(a, b)$. Similarly, we define $d(b, \mathcal{A}) = \inf_{a \in \mathcal{A}} d(a, b)$, for $b \in \mathcal{B}$. The *directional Hausdorff distance* between $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$h(\mathcal{A}, \mathcal{B}) = \sup_{a \in \mathcal{A}} d(a, \mathcal{B}),$$

and the *Hausdorff distance* between $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$H(\mathcal{A}, \mathcal{B}) = \max\{h(\mathcal{A}, \mathcal{B}), h(\mathcal{B}, \mathcal{A})\}.$$

(It is important to note that in this definition each object in $\mathcal{A}$ or in $\mathcal{B}$ is considered as a single entity, and not as the set of its points.) In order to measure similarity between $\mathcal{A}$ and $\mathcal{B}$, we compute the minimum value of the Hausdorff distance over all translates of $\mathcal{A}$

within a given set $T \subseteq \mathbb{R}^d$ of allowed translation vectors. Namely, we define

$$\sigma(\mathcal{A}, \mathcal{B}; T) = \inf_{t \in T} H(\mathcal{A} + t, \mathcal{B}),$$

where $\mathcal{A} + t = \{a + t \mid a \in \mathcal{A}\}$. In our applications, T will either be the entire $\mathbb{R}^d$ or the set of *collision-free* translates of $\mathcal{A}$ at which none of its objects intersects any object of $\mathcal{B}$. The collision-free matching between objects is useful for applications (such as those mentioned above) in which the goal is to locate a transformation where the collective shape of one set of objects best complements that of the other set. We will use $\sigma(\mathcal{A}, \mathcal{B})$ to denote $\sigma(\mathcal{A}, \mathcal{B}; \mathbb{R}^d)$.

As already mentioned, our definition of (directional) Hausdorff distance is slightly different from the one typically used in the literature [6], in which one considers the two unions $\cup\mathcal{A}$, $\cup\mathcal{B}$ as two (possibly infinite) point sets, and computes the standard Hausdorff distance

$$H(\cup\mathcal{A}, \cup\mathcal{B}) = \max\{h(\cup\mathcal{A}, \cup\mathcal{B}), h(\cup\mathcal{B}, \cup\mathcal{A})\},$$

where

$$h(\cup\mathcal{A}, \cup\mathcal{B}) = \sup_{p \in \cup\mathcal{A}} d(p, \cup\mathcal{B}) = \sup_{p \in \cup\mathcal{A}} \inf_{q \in \cup\mathcal{B}} d(p, q).$$

We will denote $\cup\mathcal{A}$ (resp., $\cup\mathcal{B}$) as $U_{\mathcal{A}}$ (resp., $U_{\mathcal{B}}$), and use the notation $h_U(\mathcal{A}, \mathcal{B})$ to denote $h(U_{\mathcal{A}}, U_{\mathcal{B}})$. Analogous meanings hold for the notations $H_U(\mathcal{A}, \mathcal{B})$ and $\sigma_U(\mathcal{A}, \mathcal{B}; T)$.

A drawback of the directional Hausdorff distance (and thus of the Hausdorff distance) is its sensitivity to outliers in the given data. One possible approach to circumvent this problem is to use "partial matching" [12], but then one has to determine how many (and which) of the objects in $\mathcal{A}$ should be matched to $\mathcal{B}$. Another possible approach is to use the *root-mean-square* (rms, for brevity) Hausdorff distance between $\mathcal{A}$ and $\mathcal{B}$, defined by

$$\begin{aligned} h_R(\mathcal{A}, \mathcal{B}) &= \left(\frac{\int_{\mathcal{A}} d^2(a, \mathcal{B}) da}{\int_{\mathcal{A}} da} \right)^{1/2} \quad \text{and} \\ H_R(\mathcal{A}, \mathcal{B}) &= \max\{h_R(\mathcal{A}, \mathcal{B}), h_R(\mathcal{B}, \mathcal{A})\}, \end{aligned}$$

with an appropriate definition of integration (usually, summation over a finite set or the Lebesgue integration over infinite point sets). Define $\sigma_R(\mathcal{A}, \mathcal{B}; T) = \inf_{t \in T} H_R(\mathcal{A} + t, \mathcal{B})$. Finally, as in [19], we define the *summed Hausdorff distance* to be

$$h_S(\mathcal{A}, \mathcal{B}) = \frac{\int_{\mathcal{A}} d(a, \mathcal{B}) da}{\int_{\mathcal{A}} da},$$

and similarly define H_S and σ_S. Informally, $h(\mathcal{A}, \mathcal{B})$ can be regarded as an L_∞-distance over the sets of objects $\mathcal{A}$ and $\mathcal{B}$. The two new definitions replace L_∞ by L_2 and L_1, respectively.

1.2 Previous results

It is beyond the scope of this paper to discuss all the results on shape matching. We refer the reader to [6, 15, 23] and references therein for a sample of known results. Here we summarize known results on shape matching using the Hausdorff distance measure.

Most of the early work on computing Hausdorff distance focused on finite point sets. Let $\mathcal{A}$ and $\mathcal{B}$ be two families of m and n points, respectively, in $\mathbb{R}^d$. In the plane, $H(\mathcal{A}, \mathcal{B})$ can be computed in $O((m+n)\log mn)$ time using Voronoi diagrams [4]. In $\mathbb{R}^3$, it can be computed in time $O((m+n)^{4/3+\varepsilon})$, where $\varepsilon > 0$ is an arbitrarily small constant, using the data structure of Agarwal and Matoušek [1]. Huttenlocher *et al.* [18] showed that $\sigma(\mathcal{A}, \mathcal{B})$ can be computed in time $O(mn(m+n)\alpha(mn)\log mn)$ in $\mathbb{R}^2$, and in time $O((mn)^2(m+n)^{1+\varepsilon})$ in $\mathbb{R}^3$, for any $\varepsilon > 0$. Chew *et al.* [12] presented an $O((m+n)^{\lceil 3d/2\rceil+1}\log^3 mn)$-time algorithm to compute $\sigma(\mathcal{A}, \mathcal{B})$ in $\mathbb{R}^d$ for any $d \geq 2$. The minimum Hausdorff distance between $\mathcal{A}$ and $\mathcal{B}$ under rigid motion in $\mathbb{R}^2$ can be computed in $O((m+n)^6 \log mn)$ time [17].

Faster approximation algorithms to compute $\sigma(\mathcal{A}, \mathcal{B})$ were first proposed by Goodrich *et al.* [14]. Aichholzer *et al.* proposed a framework of approximation algorithms using *reference points* [3]. In $\mathbb{R}^2$, their algorithm approximates the optimal Hausdorff distance within a constant factor, in $O((m+n)\log mn)$ time over all translations, and in $O(mn\log(mn)\log^* mn)$ time over rigid motions. The reference point approach can be extended to higher dimensions. However, it neither approximates the directional Hausdorff distance over a set of transformations, nor can it cope with the partial-matching problem.

Indyk *et al.* [19] study the partial matching problem, i.e., given a query $r > 0$, compute a rigid transform τ so that the number of points $p \in \mathcal{A}$ for which $d(\tau(p), \mathcal{B}) \leq r$ is maximized. They present algorithms for ε-approximating the maximum-size partial matching over the set of rigid motions in $O(mn\Delta/(r\varepsilon^2)\,\mathrm{polylog}(\frac{n\Delta}{\varepsilon r}))$ time in $\mathbb{R}^2$, and in $O(mn\Delta^3/(r^3\varepsilon^3)\,\mathrm{polylog}(\frac{n\Delta}{\varepsilon r}))$ time in $\mathbb{R}^3$, where Δ is the maximum of the spreads of the two point sets.[1] Their algorithm can be extended to approximate the minimum summed Hausdorff distance over rigid motions. Similar results were independently achieved in [11] using a different technique.

Algorithms for computing $H_U(\mathcal{A}, \mathcal{B})$ and $\sigma_U(\mathcal{A}, \mathcal{B})$, where $\mathcal{A}$ and $\mathcal{B}$ are sets of segments in the plane, or sets of simplices in higher dimensions are presented in [2, 4, 5]. Atallah [10] presents an algorithm for computing $H_U(\mathcal{A}, \mathcal{B})$ for two convex polygons in $\mathbb{R}^2$. Agarwal *et al.* [2] provide an algorithm for computing $\sigma_U(\mathcal{A}, \mathcal{B})$, where $\mathcal{A}$ and $\mathcal{B}$ are two sets of m and n segments in $\mathbb{R}^2$, respectively, in time $O((mn)^2\log^3 mn)$. If rigid motions are allowed, the minimum Hausdorff distance between two sets of points in the plane MICHA SAYS: Right? ⟵ can be computed in time $O((mn)^3\log^2(mn))$ (Chew *et al.* [13]). Aichholzer *et al.* [3] present algorithms for approximating the minimum Hausdorff distance under different families of transformations for sets of points or of segments in $\mathbb{R}^2$, and for sets of triangles in $\mathbb{R}^3$, using reference points. Other than that, little is known about computing $\sigma_U(\mathcal{A}, \mathcal{B})$ or $\sigma(\mathcal{A}, \mathcal{B})$ where $\mathcal{A}$ and $\mathcal{B}$ are sets of simplices or other geometric shapes in higher dimensions.

[1] The spread of a set of points is the ratio of its diameter to the closest-pair distance.

1.3 Our results

In this paper, we develop efficient algorithms for computing $\sigma(\mathcal{A}, \mathcal{B}; T)$ and $\sigma_U(\mathcal{A}, \mathcal{B}; T)$ for sets of balls, and for approximating $\sigma_R(\mathcal{A}, \mathcal{B}), \sigma_S(\mathcal{A}, \mathcal{B})$ for sets of points in $\mathbb{R}^d$. Consequently, the paper consists of three parts, where the first two deal with the two variants of Hausdorff distances for balls, and the third part studies the rms and summed Hausdorff-distance problems for point sets.

Let $D(c, r)$ denote the ball in $\mathbb{R}^d$ of radius r centered at c. Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$ be two families of balls in $\mathbb{R}^d$, where $A_i = D(a_i, \rho_i)$ and $B_j = D(b_j, r_j)$, for each i and j. Let $\mathcal{F}$ be the set of all translation vectors $t \in \mathbb{R}^d$ so that no ball of $\mathcal{A} + t$ intersects any ball of $\mathcal{B}$. We note, though, that balls of the same family can intersect each other, as is typically the case, e.g., in modeling molecules as collections of balls.

Section 2 considers the problem of computing the Hausdorff distance between two sets $\mathcal{A}$ and $\mathcal{B}$ of balls under the collision-free constraint, where the distance between two disjoint balls $A_i \in \mathcal{A}$ and $B_j \in \mathcal{B}$ is defined as $d(A_i, B_j) = d(a_i, b_j) - \rho_i - r_j$. We can regard this distance as an additively weighted Euclidean distance between the centers of A_i and B_j, and it is a common way of measuring distance between atoms in molecular biology [15]. In Section 2 we describe algorithms for computing $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in two and three dimensions. The running time is $O(mn(m+n)\log^4 mn)$ in $\mathbb{R}^2$, and $O(m^2n^2(m+n)\log^4 mn)$ in $\mathbb{R}^3$. The approach can be extended to solve the (collision-free) partial-matching problem under this variant of Hausdorff distance in the same asymptotic time complexity.

Section 3 considers the problem of computing $\sigma_U(\mathcal{A}, \mathcal{B})$ and $\sigma_U(\mathcal{A}, \mathcal{B}; \mathcal{F})$, i.e., of computing the Hausdorff distance between the union of $\mathcal{A}$ and the union of $\mathcal{B}$, minimized over all translates of $\mathcal{A}$ in $\mathbb{R}^d$ or in $\mathcal{F}$. We first describe an $O(mn(m+n)\log^4 mn)$-time algorithm for computing $\sigma_U(\mathcal{A}, \mathcal{B})$ and $\sigma_U(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in $\mathbb{R}^2$, which relies on several geometric properties of the *medial axis* of the union of disks. MICHA SAYS: Add [REFS] A straightforward extension of our algorithm to $\mathbb{R}^3$ is harder to analyze, and does not yield efficient bounds on its running time, mainly because little is known about the complexity of the medial axis of the union of balls in $\mathbb{R}^3$ [**?**]. We therefore consider approximation algorithms. In particular, given a parameter $\varepsilon > 0$, we compute a translation t, in time $O(((m+n)/\varepsilon^2)\log^3 mn)$ in $\mathbb{R}^2$ and in time $O(((m^2+n^2)/\varepsilon^3)\log^2 mn)$ in $\mathbb{R}^3$, such that $H_U(\mathcal{A}+t, \mathcal{B}) \leq (1+\varepsilon)\sigma_U(\mathcal{A}, \mathcal{B})$. We also present a "pseudo-approximation" algorithm for computing $\sigma_U(\mathcal{A}, \mathcal{B}; \mathcal{F})$: Given an $\varepsilon > 0$, the algorithm computes a region $X \subseteq \mathbb{R}^d$ that serves as an ε-approximation of $\mathcal{F}$ (in a sense defined formally in Section 3). It then returns a placement $t \in X$ such that

$$H_U(\mathcal{A} + t, \mathcal{B}; X) \leq (1+\varepsilon)\sigma_U(\mathcal{A}, \mathcal{B}; X),$$

in time $O(((m^2+n^2)/\varepsilon^3)\log^2 mn)$ in $\mathbb{R}^3$. This variant of approximation makes sense in applications where the data is noisy and shallow penetrations between objects are allowed, as is the case in the docking problem [15].

Finally, we turn, in Section 4, to the two variants of rms and summed Hausdorff distances. Given two sets of points $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^d$ of size m and n, respectively, Section 4 describes an $O((mn/\varepsilon^d)\log(mn/\varepsilon))$-time algorithm for computing an ε-approximation of

$\sigma_R(\mathcal{A}, \mathcal{B})$.[2] It also provides a data structure so that, for a query vector $t \in \mathbb{R}^d$, an ε-approximation of $H_R(\mathcal{A} + t, \mathcal{B})$ can be computed in $O(\log(mn/\varepsilon))$ time. In fact, we solve a more general problem, which is interesting in its own right. Given a family $P_1, \ldots, P_l$ of point sets in $\mathbb{R}^d$, with a total of N points, we construct a decomposition of $\mathbb{R}^d$ into $O(N/\varepsilon^d)$ cells, which is an ε-approximation of each of the Voronoi diagrams of $P_1, \ldots, P_l$, in the sense defined in [9, 16]. Moreover, given a semigroup operation $+$, we can preprocess this decomposition in $O((N/\varepsilon^d)\log(N/\varepsilon))$ time, so that for a query point q, an ε-approximation of $\sum_{i=1}^{l} d^2(q, P_i)$ can be computed in $O(\log(N/\varepsilon))$ time. We also extend the approach to obtain an algorithm that ε-approximates $\sigma_S(\mathcal{A}, \mathcal{B})$ in $O((mn/\varepsilon^{2d})\,\mathrm{polylog}(mn, 1/\varepsilon))$ time. This result relies on an efficient dynamic data structure, which we propose, for maintaining an ε-approximation of the 1-median of a point set in $\mathbb{R}^d$, under insertion and deletion of points.

2 Collision-Free Hausdorff Distance between Sets of Balls

Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$ be two sets of balls in $\mathbb{R}^d$, $d = 2, 3$. For two disjoint balls $A_i = D(a_i, \rho_i) \in \mathcal{A}$ and $B_j = D(b_j, r_j) \in \mathcal{B}$, we define

$$d(A_i, B_j) = d(a_i, b_j) - \rho_i - r_j,$$

namely, the (minimum) distance between A_i and B_j as point sets. Let $\mathcal{F}$ be the set of placements t of $\mathcal{A}$ such that no ball in $\mathcal{A} + t$ intersects any ball of $\mathcal{B}$. In this section, we describe an exact algorithm for computing $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$, and show that it can be extended to partial matching.

2.1 Computing $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in $\mathbb{R}^2$ and $\mathbb{R}^3$

As is common in geometric optimization, we first present an algorithm for the *decision problem*, namely, given a parameter $\delta > 0$, we wish to determine whether $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F}) \le \delta$. We then use the parametric-searching technique [2, 21] to compute $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$. Given $\delta > 0$, for $1 \le i \le m$, let $V_i \subseteq \mathbb{R}^d$ be the set of vectors $t \in \mathbb{R}^d$ such that

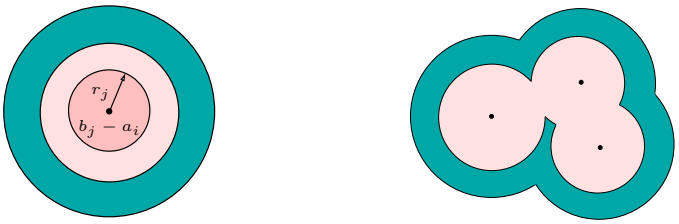
(V) $0 < \min_{1 \le j \le n} d(A_i + t, B_j) \le \delta$.

(In particular, $A_i + t$ does not intersect the interior of any $B_j \in \mathcal{B}$.)

Let $D_{ij}^- = D(b_j - a_i, \rho_i + r_j)$ and $D_{ij}^+ = D(b_j - a_i, \rho_i + r_j + \delta)$. Then $U_i^+ = \bigcup_{j \le n} D_{ij}^+$ is the set of vectors that satisfy $\min_{1 \le j \le n} d(A_i + t, B_j) \le \delta$, and the interior of $U_i^- = \bigcup_{j \le n} D_{ij}^-$ violates $0 < \min_{1 \le j \le n} d(A_i + t, B_j)$. Hence, $V_i = \mathrm{cl}(U_i^+ \setminus U_i^-)$. Let

$$V(\mathcal{A}, \mathcal{B}) = \bigcap_{1 \le i \le m} V_i = \mathrm{cl}\big(\big(\bigcap_i U_i^+\big) \setminus \big(\bigcup_i U_i^-\big)\big).$$

[2] Indyk *et al.* [19] outline an approximation algorithm for computing $\sigma_R(\mathcal{A}, \mathcal{B})$ without providing any details. We believe that if we work out the details of their algorithm, the running time of our algorithm is better. Moreover, our algorithm is more direct.



(a) (b)

Figure 1: (a) Inner, middle and outer disks are $B_j - a_i$, D_{ij}^-, and D_{ij}^+, respectively; (b) an example of V_i (dark region), which is the difference between U_i^+ (the whole union) and U_i^- (inner light region).

See Figure 1 for an illustration. By definition, $V(\mathcal{A}, \mathcal{B}) \subseteq \mathcal{F}$ is the set of vectors $t \in \mathcal{F}$ such that $h(\mathcal{A} + t, \mathcal{B}) \leq \delta$. Similarly, we define

$$V(\mathcal{B}, \mathcal{A}) \subseteq \mathcal{F} = \{t \in \mathcal{F} \mid h(\mathcal{B}, \mathcal{A} + t) \leq \delta\}.$$

Thus $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F}) \leq \delta$ if and only if $V(\mathcal{A}, \mathcal{B}) \cap V(\mathcal{B}, \mathcal{A}) \neq \emptyset$.

Lemma 2.1 *The combinatorial complexity of $V(\mathcal{A}, \mathcal{B})$ in $\mathbb{R}^2$ is $O(m^2 n)$.*

Proof: If an edge of $\partial V(\mathcal{A}, \mathcal{B})$ is not adjacent to any vertex, then it is the entire circle bounding a disk of D_{ij}^+ or D_{ij}^-. There are $O(mn)$ such disks, so it suffices to bound the number of vertices in $V(\mathcal{A}, \mathcal{B})$.

Let v be a vertex of $V(\mathcal{A}, \mathcal{B})$; v is either a vertex of V_i, for some $1 \leq i \leq m$, or an intersection point of an edge in V_i and an edge in V_k, for some $1 \leq i \neq k \leq m$. In the latter case,

$$v \in V_i \cap V_k = (U_i^+ \cap U_k^+) \setminus (U_i^- \cup U_k^-).$$

In other words, a vertex of $V(\mathcal{A}, \mathcal{B})$ is a vertex of $U_i^+ \cap U_k^+$, $U_i^+ \setminus U_k^-$, $U_k^+ \setminus U_i^-$, or $U_i^- \cup U_k^-$, for $1 \leq i, k \leq m$. Observe that a vertex of $U_i^+ \cap U_k^+$ (resp., of $U_i^+ \setminus U_k^-$) that lies on both ∂U_i^+ and ∂U_k^+ (resp., ∂U_k^-) is also a vertex of $U_i^+ \cup U_k^+$ (resp., $U_i^+ \cup U_k^-$). Therefore, every vertex in $V(\mathcal{A}, \mathcal{B})$ is a vertex of $U_i^+ \cup U_k^+$, $U_i^+ \cup U_k^-$, $U_k^+ \cup U_i^-$, or $U_i^- \cup U_k^-$, for some $1 \leq i, k \leq m$. Since each U_i^+, U_i^- is the union of a set of n disks, each of $U_i^- \cup U_k^+$, $U_i^+ \cup U_k^-$, $U_k^+ \cup U_i^-$, $U_i^- \cup U_k^-$ is the union of a set of $2n$ disks and thus has $O(n)$ vertices [20]. Hence, $V(\mathcal{A}, \mathcal{B})$ has $O(m^2 n)$ vertices. ∎

Lemma 2.2 *The combinatorial complexity of $V(\mathcal{A}, \mathcal{B})$ in $\mathbb{R}^3$ is $O(m^3 n^2)$.*

Proof: The number of faces or edges of $V(\mathcal{A}, \mathcal{B})$ that do not contain any vertex is $O(n^2 m^2)$ since they are defined by at most two balls in a family of $2mn$ balls. We therefore focus on the number of vertices in $V(\mathcal{A}, \mathcal{B})$. As in the proof of Lemma 2.1, any vertex $V(\mathcal{A}, \mathcal{B})$ satisfies:

$$v \in V_i \cap V_j \cap V_k = (U_i^+ \cap U_j^+ \cap U_k^+) \setminus (U_i^- \cup U_j^- \cup U_k^-),$$

for some $1 \le i \le j \le k \le m$. Again, such a vertex is also a vertex of $X_i \cup X_j \cup X_k$, where X_i is U_i^+ or U_i^-, and similarly for X_j, X_k. Since the union of r balls in $\mathbb{R}^3$ has $O(r^2)$ vertices, $X_i \cup X_j \cup X_k$ has $O(n^2)$ vertices, thereby implying that $V(\mathcal{A}, \mathcal{B})$ has $O(m^3 n^2)$ vertices. ■

Similarly, we can prove that the complexity of $V(\mathcal{B}, \mathcal{A})$ is $O(n^2 m)$ in $\mathbb{R}^2$ and $O(n^3 m^2)$ in $\mathbb{R}^3$. Extending the preceding arguments a little, we obtain the following.

Lemma 2.3 *$V(\mathcal{A}, \mathcal{B}) \cap V(\mathcal{B}, \mathcal{A})$ has a combinatorial complexity of $O(mn(m+n))$ in $\mathbb{R}^2$, and $O(m^2 n^2 (m+n))$ in $\mathbb{R}^3$.*

Remark. The above argument in fact bounds the complexity of the arrangement of $\mathcal{V} = \{V_1, V_2, \ldots, V_m\}$. For example, in $\mathbb{R}^2$, any intersection point of ∂V_i and ∂V_k lies on the boundary of $\partial(V_i \cap V_k)$, and we have argued that $V_i \cap V_k$ has $O(n)$ vertices. Hence, the entire arrangement has $O(m^2 n)$ vertices in $\mathbb{R}^2$.

MICHA SAYS: More space here. ←

We exploit a divide-and-conquer approach, combined with a plane-sweep, to compute $V(\mathcal{A}, \mathcal{B})$, $V(\mathcal{B}, \mathcal{A})$, and their intersections in $\mathbb{R}^2$. For example, to compute $V(\mathcal{A}, \mathcal{B})$, we compute $V' = \bigcap_{i=1}^{n/2} V_i$ and $V'' = \bigcap_{i=n/2+1}^{n} V_i$ recursively, and merge $V(\mathcal{A}, \mathcal{B}) = V' \cap V''$ by a plane-sweep method. The overall running time is $O((m+n)mn \log mn)$.

To decide whether $V(\mathcal{A}, \mathcal{B}) \cap V(\mathcal{B}, \mathcal{A}) = \emptyset$ in $\mathbb{R}^3$, it suffices to check whether

$$\Gamma_D = V(\mathcal{A}, \mathcal{B}) \cap V(\mathcal{B}, \mathcal{A}) \cap \partial D$$

is empty for all balls $D \in \{D_{ij}^-, D_{ij}^+ \mid 1 \le i \le m, 1 \le j \le n\}$. Using the fact that the various D_{ij}^-, D_{ij}^+ meet any ∂D in a collection of spherical caps, we can compute Γ_D in time $O(mn(m+n)\log mn)$, by the same divide-and-conquer approach as computing $V(\mathcal{A}, \mathcal{B}) \cap V(\mathcal{B}, \mathcal{A})$ in $\mathbb{R}^2$. Therefore we can determine in $O(m^2 n^2 (m+n)\log mn)$ time whether $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F}) \le \delta$ in $\mathbb{R}^3$.

Finally, the optimization problem can be solved by the parametric search technique [2]. In order to apply the parametric search technique, we need a parallel version of the above procedure. However, this divide-and-conquer paradigm uses plane-sweep during the conque stage, which is not easy to parallelize. Instead, we use the algorithm of [2] to compute the union/intersection of two planar or spherical regions. It yields an overall parallel algorithm for determining whether $V(\mathcal{A}, \mathcal{B}) \cap V(\mathcal{B}, \mathcal{A})$ is empty in $O(\log^2 mn)$ time using $O(mn(m+n)\log mn)$ processors in $\mathbb{R}^2$, and $O(m^2 n^2 (m+n)\log mn)$ processors in $\mathbb{R}^3$. The standard technique of parametric searching then implies the following result.

Theorem 2.4 *Given two sets $\mathcal{A}$ and $\mathcal{B}$ of m and n disks (or balls), we can compute $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in time $O(mn(m+n)\log^4 mn)$ in $\mathbb{R}^2$, and in time $O(m^2 n^2 (m+n)\log^4 mn)$ in $\mathbb{R}^3$.*

2.2 Partial matching

Extending the definition of partial matching in [19], we define the partial collision-free Hausdorff distance problem as follows.

Given an integer k, let $h_k(\mathcal{A}, \mathcal{B})$ denote the k^{th} largest value in the set $\{d(a, \mathcal{B}) \mid a \in \mathcal{A}\}$; note that $h(\mathcal{A}, \mathcal{B}) = h_1(\mathcal{A}, \mathcal{B})$. We define $h_k(\mathcal{B}, \mathcal{A})$ in a fully symmetric manner, and then define $H_k(\mathcal{A}, \mathcal{B})$, $\sigma_k(\mathcal{A}, \mathcal{B}; T)$ as above. The preceding algorithm can be extended to compute $\sigma_k(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in the same asymptotic time complexity. We briefly illustrate the two-dimensional case. Let $\mathcal{V} = \{V_1, V_2, \ldots, V_m\}$ be as defined above, and let $\Xi(\mathcal{V})$ be the arrangement of $\mathcal{V}$. For each cell $\Delta \in \Xi(\mathcal{V})$, let $\chi(\Delta)$ be the number of V_i's that fully contain Δ. Note that for any point t in a cell Δ with $\chi(\Delta) > (m-k)$, $h_k(\mathcal{A}+t, \mathcal{B}) \leq \delta$, and vice versa. Hence, we compute $\Xi(\mathcal{V})$ and $\chi(\Delta)$ for each cell $\Delta \in \Xi(\mathcal{V})$, and then discard all the cells Δ for which $\chi(\Delta) \leq (m - k)$. The remaining cells form the set $T_1 = \{t \mid h_k(\mathcal{A} + t, \mathcal{B}) \leq \delta\}$. By the Remark following Lemma 2.2, Ξ has $O(m^2 n)$ vertices, and it can be computed in $O(m^2 n \log mn)$ time. Therefore, T_1 can be computed in $O(m^2 n \log mn)$ time. Similarly, we can compute $T_2 = \{t \mid h_k(\mathcal{B}, \mathcal{A} + t) \leq \delta\}$ in $O(mn^2 \log mn)$ time, and we can determine in $O(mn(m + n) \log mn)$ time whether $T_1 \cap T_2 \neq \emptyset$. Similar arguments can solve the partial matching problem in $\mathbb{R}^3$, by computing the sets T_1, T_2, and by checking for their intersection along the boundary of each of the balls D_{ij}^+, D_{ij}^-. MICHA SAYS: How do we test for intersection in three dimensions? Putting everything together, we obtain the following. ←

Theorem 2.5 *Let $\mathcal{A}$ and $\mathcal{B}$ be two families of m and n balls, respectively, and let $k \geq 0$ be an integer, we can compute $\sigma_k(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in $O(mn(m + n) \log^4 mn)$ time in $\mathbb{R}^2$, and in $O(m^2 n^2 (m + n) \log^4 mn)$ time in $\mathbb{R}^3$.*

3 Hausdorff Distance between Unions of Balls

In Section 3.1 we describe an algorithm for computing $\sigma_U(\mathcal{A}, \mathcal{B})$ in $\mathbb{R}^2$. The same approach can be extended to compute $\sigma_U(\mathcal{A}, \mathcal{B}; \mathcal{F})$ within the same asymptotic time complexity. In Section 3.2, we present approximation algorithms for the same problem in $\mathbb{R}^2$ and $\mathbb{R}^3$.

3.1 The exact 2D algorithm

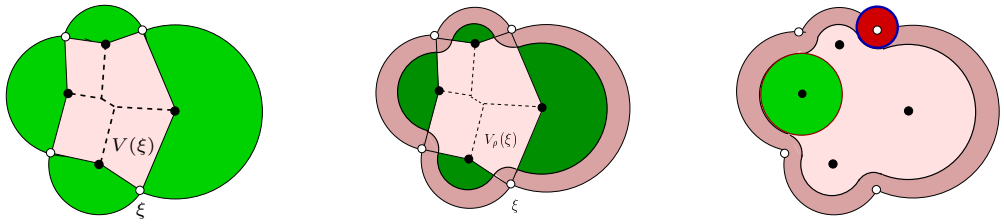
Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$ be two sets of disks in the plane. Write, as above, $A_i = D(a_i, \rho_i)$, for $i = 1, \ldots, m$, and $B_j = D(b_j, r_j)$, for $j = 1, \ldots, n$. Let $U_{\mathcal{A}}$ (resp., $U_{\mathcal{B}}$) be the union of the disks in $\mathcal{A}$ (resp., $\mathcal{B}$). As in Section 2, we focus on the decision problem for a given distance parameter $\delta > 0$.

For any point p, we have

$$\begin{aligned} d(p, U_{\mathcal{B}}) &= \min_{q \in U_{\mathcal{B}}} d(p, q) = \min_{1 \leq j \leq n} d(p, B_j) \\ &= \min_{1 \leq j \leq n} \max\{d(p, b_j) - r_j, 0\}. \end{aligned}$$

This value is greater than δ if and only if

$$\min_{1 \leq j \leq n} \Big(d(p, b_j) - (r_j + \delta) \Big) > 0.$$



(a) (b) (c)

Figure 2: (a) The medial axis (dotted segments) of the union of four disks centered at the solid points: The Voronoi diagram of the boundary decomposes the union into 8 cells; (b) shrinking by ρ the Voronoi cell $V(\xi)$ of each boundary element ξ of the union; (c) The boundary of the lighter-colored disk contains a convex arc, and the boundary of the darker-colored disk contains a concave arc.

In other words, $h_U(\mathcal{A}+t,\mathcal{B}) > \delta$ if and only if there exists a point $p \in U_\mathcal{A}$ such that $p+t \notin U_\mathcal{B}(\delta) = \bigcup_{j=1}^n B_j(\delta)$, where $B_j(\delta) = D(b_j, r_j+\delta)$ is the disk B_j expanded by δ.

Let

$$T_1 = \{t \mid h_U(\mathcal{A}+t,\mathcal{B}) \le \delta\};$$

T_1 is the set of all translations t such that $U_\mathcal{A}+t \subseteq U_\mathcal{B}(\delta)$. Our decision procedure computes the set T_1 and the analogously defined set

$$T_2 = \{t \mid h_U(\mathcal{B},\mathcal{A}+t) \le \delta\},$$

and then tests whether $T_1 \cap T_2 \neq \emptyset$. To understand the structure of T_1, we first study the case in which $\mathcal{A}$ consists of just one disk A, with center a and radius ρ. For simplicity of notation, we denote $U_\mathcal{B}(\delta)$ temporarily by U. Let Q denote the set of vertices of ∂U, and Γ the set of (relatively open) edges of ∂U; we have $|Q| \le |\Gamma| \le 6n-12$ [20].

Consider the Voronoi diagram $\mathrm{Vor}(Q \cup \Gamma)$ of the boundary features of U, clipped to within U. This is a decomposition of U into cells, so that, for each $\xi \in Q \cup \Gamma$, the cell $V(\xi)$ of ξ is the set of points $x \in U$ such that $d(x,\xi) \le d(x,\xi')$, for all $\xi' \in Q \cup \Gamma$. The diagram is closely related to the medial axis of ∂U. See Figure 2(a). For each $\gamma \in \Gamma$, let $W(\gamma)$ denote the circular sector spanned by γ within the disk $B_j(\delta)$ whose boundary is γ, and let $U' = U \setminus \bigcup_{\gamma\in\Gamma} W(\gamma)$. The diagram has the following structure. (A variant of the following lemma was observed in [7].)

Lemma 3.1 *(a) For each $\gamma \in \Gamma$, we have $V(\gamma) = W(\gamma)$.*
(b) For each $\xi \in Q$, we have $V(\xi) = U' \cap V'(\xi)$, where $V'(\xi)$ is the Voronoi cell of ξ in the Voronoi diagram $\mathrm{Vor}(Q)$ of Q. Moreover, $V(\xi)$ is a convex polygon.

MICHA SAYS: No proof? ←

Lemma 3.1 implies that $\mathrm{Vor}(Q \cup \Gamma)$ yields a convex decomposition of U of linear size. The medial axis of ∂U consists of all the edges of those cells $V'(\xi)$, for $\xi \in Q$, that are also edges of $\mathrm{Vor}(Q \cup \Gamma)$ (the dashed cells of Figure 2(a)).

Returning to the study of the structure of T_1, we have, by definition, $A + t \subseteq U$ if and only if $d(a+t, \xi) \geq \rho$, where ξ is the feature of $Q \cup \Gamma$ whose cell contains $a + t$. This implies that the set $T_1(A)$ of all translations t of A for which $A + t \subseteq U$ is given by

$$T_1(A) = \left(\bigcup_{\xi \in Q \cup \Gamma} V_\rho(\xi) \right) - a,$$

where

$$V_\rho(\xi) = \{x \in V(\xi) \mid d(x, \xi) \geq \rho\}.$$

For $\gamma \in \Gamma$, $V_\rho(\gamma)$ is the sector obtained from $W(\gamma)$ by shrinking it by distance ρ towards its center. For $\xi \in Q$, $V_\rho(\xi) = V(\xi) \setminus D(\xi, \rho)$. See Figure 2(b) for an illustration.

Now return to the original case in which $\mathcal{A}$ consists of m disks; we obtain

$$T_1 = \bigcap_{i=1}^{m} T_1(A_i) = \bigcap_{i=1}^{m} \bigcup_{\xi \in Q \cup \Gamma} (V_{\rho_i}(\xi) - a_i).$$

Note that each $T_1(A_i)$ is bounded by $O(n)$ circular arcs, some of which are *convex* (those bounding shrunk sectors), and some are *concave* (those bounding shrunk Voronoi cells of vertices). Convex arcs are bounded by disks $D(b_k - a_i, r_k + \delta - \rho_i)$, for some $1 \leq k \leq n$, while concave arcs are bounded by disks $D(\xi - a_i, \rho_i)$ for $\xi \in Q$. Furthermore, since $T_1(A_i)$ is obtained by removing all points $x \in U$ such that the nearest distance from x to ∂U is smaller than ρ_i, we have that: (i) $D(b_k - a_i, r_k + \delta - \rho_i) \subseteq T_1(A_i)$; and (ii) $D(\xi - a_i, \rho_i) \cap (T_1(A_i) \setminus \partial(T_1(A_i)) = \emptyset$. See Figure 2 (c) for an illustration.

Lemma 3.2 *For any pair of disks $A_i, A_j \in \mathcal{A}$, the complexity of $T_1(A_i) \cap T_1(A_j)$ is $O(n)$.*

Proof: Clearly, $T_1(A_i) \cap T_1(A_j)$ is bounded by circular arcs, whose endpoints are either vertices of $T_1(A_i)$ or $T_1(A_j)$, or intersection points between an arc of $\partial T_1(A_i)$ and an arc of $\partial T_1(A_j)$. It suffices to estimate the number of vertices of the latter kind.

Consider the set $\mathcal{B}'_{ij}$ of the $2n + 2|Q|$ disks

$$\{D(b_k - a_i, r_k + \delta - \rho_i), D(b_k - a_j, r_k + \delta - \rho_j)\}_{1 \leq k \leq n} \bigcup \{D(\xi - a_i, \rho_i), D(\xi - a_j, \rho_j)\}_{\xi \in Q}.$$

We claim that any intersection point between two arcs, one from $\partial T_1(A_i)$ and one from $\partial T_1(A_j)$, lies on $\partial(\cup \mathcal{B}'_{ij})$. Indeed, assume that x is such an intersection point that does not lie on $\partial(\cup \mathcal{B}'_{ij})$. Then it has to lie in the interior of $\cup \mathcal{B}'_{ij}$. That is, there is a disk $D \in \mathcal{B}'_{ij}$ that contains x. There are two possibilities for the choice of D.

(i) $D = D(b_k - a_i, r_k + \delta - \rho_i)$ (resp., $D = D(b_k - a_j, r_k + \delta - \rho_j)$), for some $1 \leq k \leq n$. The boundary of such a disk contains some convex arc on $\partial T_1(A_i)$ (resp., $\partial T_1(A_j)$), and $D \subseteq T_1(A_i)$ (resp., $D \subseteq T_1(A_j)$). As such, x cannot appear on the boundary of $\partial T_1(A_i)$ (resp., $\partial T_1(A_j)$), contrary to assumption.

(ii) $D = D(\xi - a_i, \rho_i)$ (resp., $D = D(\xi - a_j, \rho_j)$), for some $\xi \in Q$. Recall that Q is the set of vertices on the boundary of ∂U. Therefore, by definition, $A_i + x$ (resp., $A_j + x$) contains ξ in its interior, so it cannot be fully contained in U, implying that $x \notin T_1(A_i)$ (resp., $x \notin T_1(A_j)$), again a contradiction.

These contradictions imply the claim. It then follows, using the bound of [20], that the number of intersections under consideration is at most $6 \cdot (2n + 2|Q|) - 12 = O(n)$. ■

Each vertex of T_1 is also a vertex of some $T_1(A_i) \cap T_1(A_j)$. Applying the preceding lemma to all the $O(m^2)$ pairs A_i, A_j, we obtain the following.

Lemma 3.3 *The complexity of T_1 is $O(m^2 n)$, and it can be computed in $O(m^2 n \log mn)$ time.*

Similarly, the set T_2 has complexity $O(mn^2)$ and can be computed in time $O(mn^2 \log mn)$. Finally, we can determine whether $T_1 \cap T_2 \neq \emptyset$, by plane sweep, in time $O(mn(m+n) \log mn)$. Using parametric search, as in [2], $\sigma_U(\mathcal{A}, \mathcal{B})$ can be computed in $O(mn(m+n) \log^4 mn)$ time.

To compute $\sigma_U(\mathcal{A}, \mathcal{B}; \mathcal{F})$, we follow the same approach as computing $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in the preceding section. Specifically, we need to modify the definition of T_1 and of T_2, to require also that no disk of $\mathcal{A}+t$ intersect any disk of $\mathcal{B}$. This amounts, in the case of T_1, to redefine each $T_1(A)$ to consist of all $t \in \mathbb{R}^2$ such that $A + t \subseteq U$ and $(A + t) \cap U_{\mathcal{B}} = \emptyset$. The latter is equivalent to requiring that $t \notin U_{\mathcal{B}}(\rho) - a$. Hence

$$T_1 = \bigcap_{i=1}^{m} T_1(A_i) = \bigcap_{i=1}^{m} \bigcup_{\xi \in Q \cup \Gamma} (V_{\rho_i}(\xi) - a_i) \,. \setminus \bigcup_{i=1}^{m} (U_{\mathcal{B}}(\rho_i) - a_i).$$

It is now easy to modify the arguments in the proof of Lemma **??**, to conclude that the complexity of T_1 (and, symmetrically of T_2) remains asymptotically the same, which then implies the following result.

Theorem 3.4 *Given two families $\mathcal{A}$ and $\mathcal{B}$ of m and n disks in $\mathbb{R}^2$, we can compute both $\sigma_U(\mathcal{A}, \mathcal{B})$ and $\sigma_U(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in time $O(mn(m+n) \log^4 mn)$.*

3.2 Approximation algorithms

No good bounds are known for the complexity of the Voronoi diagram of the boundary of the union of n balls in $\mathbb{R}^3$, or, more precisely, for the complexity of the portion of the diagram inside the union [7]. The best known bound is $O(n^4)$. Hence, a naïve extension of the preceding exact algorithm to $\mathbb{R}^3$ yields an algorithm whose running time is hard to calibrate, and only rather weak upper bounds can be derived. We therefore resort to approximation algorithms.

Approximating $\sigma_U(\mathcal{A},\mathcal{B})$ in $\mathbb{R}^2$ and $\mathbb{R}^3$. Given a parameter $\varepsilon > 0$, we wish to compute a translation t of $\mathcal{A}$ such that $H_U(\mathcal{A}+t,\mathcal{B}) \leq (1+\varepsilon)\sigma_U(\mathcal{A},\mathcal{B})$, i.e., $H_U(\mathcal{A}+t,\mathcal{B})$ is an ε-approximation of $\sigma_U(\mathcal{A},\mathcal{B})$. Our approximation algorithm for $\sigma_U(\mathcal{A},\mathcal{B})$ follows the same approach as the one used in [3, 4]. That is, let $r(\mathcal{A})$ (resp., $r(\mathcal{B})$) denote the point with smallest coordinates, called the *reference point*, of the axis-parallel bounding box of $U_\mathcal{A}$ (resp., $U_\mathcal{B}$). Set $\tau = r(\mathcal{B}) - r(\mathcal{A})$. It is shown in [4] that in $\mathbb{R}^d$, $H_U(\mathcal{A}+\tau,\mathcal{B}) \leq (1+\sqrt{d})\sigma_U(\mathcal{A},\mathcal{B})$, and that the optimal translation lies in MICHA SAYS: Complete! Computing τ takes $O(m+n)$ time. We compute $H_U(\mathcal{A}+\tau,\mathcal{B})$ using the parametric search technique [2], which is based on the following simple implementation of the decision procedure: ⟵

Fix a parameter $\delta > 0$, and put $U_\mathcal{A}(\delta) = \bigcup_i D(a_i, \rho_i + \delta)$ and $U_\mathcal{B}(\delta) = \bigcup_j D(b_j, r_j + \delta)$. We observe that $H_U(\mathcal{A}+t,\mathcal{B}) \leq \delta$ if and only if $U_\mathcal{A} + t \subseteq U_\mathcal{B}(\delta)$ and $U_\mathcal{B} \subseteq U_\mathcal{A}(\delta) + t$. To test whether $U_\mathcal{A} + t \subseteq U_\mathcal{B}(\delta)$, we compute $(U_\mathcal{A} + t) \cup U_\mathcal{B}(\delta)$, the union of the balls in $\mathcal{A} + t$ and of the δ-expanded balls in $\mathcal{B}$, and check whether any ball of $\mathcal{A}$ appears on its boundary. If not, then $U_\mathcal{A} + t \subseteq U_\mathcal{B}(\delta)$. Similarly, we test whether $U_\mathcal{B} \subseteq U_\mathcal{A}(\delta) + t$. The total time spent is proportional to the time needed to compute the union of $m + n$ balls, which is $O((m+n)\log(m+n))$ in $\mathbb{R}^2$, and $O((m+n)^2)$ in $\mathbb{R}^3$. MICHA SAYS: Add [REF]? ⟵

In order to compute an ε-approximation of $\sigma_U(\mathcal{A},\mathcal{B})$ from this constant-factor approximation, we use the standard trick MICHA SAYS: Add [REF] of placing a grid of cell size $\frac{\varepsilon}{1+\sqrt{d}} \cdot H_U(\mathcal{A}+\tau,\mathcal{B})$ in the neighborhood of τ, and returning the smallest $H_U(\mathcal{A}+t,B)$, where t ranges over the grid points. We thus obtain the following result. ⟵

Theorem 3.5 *Given two sets of balls, $\mathcal{A}$ and $\mathcal{B}$, of size m and n, respectively, and $\varepsilon > 0$, an ε-approximation of $\sigma_U(\mathcal{A},\mathcal{B})$ can be computed in $O(((m+n)/\varepsilon^2)\log^3 mn)$ time in $\mathbb{R}^2$, and in $O(((m^2+n^2)/\varepsilon^3)\log^2 mn)$ time in $\mathbb{R}^3$.*

MICHA SAYS: I dont understand where the log's came from in the last theorem. Please add a sentence or two. ⟵

Pseudo-approximation for $\sigma_U(\mathcal{A},\mathcal{B};\mathcal{F})$. Currently, we do not have an efficient algorithm to ε-approximate $\sigma_U(\mathcal{A},\mathcal{B};\mathcal{F})$ in $\mathbb{R}^3$. Instead, we present a "pseudo-approximation" algorithm, in the following sense.

The set $\mathcal{K} = U_\mathcal{B} \oplus (-U_\mathcal{A})$, where $\oplus$ denotes the Minkowski sum, is the set of all placements of $\mathcal{A}$ at which $U_\mathcal{A}$ intersects $U_\mathcal{B}$; we have $\mathcal{K} = \bigcup_{i,j} D(b_j - a_i, \rho_i + r_j)$, and $\mathcal{F} = cl(\mathbb{R}^3 \setminus \mathcal{K})$. For a parameter $\varepsilon \geq 0$, let

$$\mathcal{K}(\varepsilon) = \bigcup_{i,j} D(b_j - a_i, (1-\varepsilon)(\rho_i + r_j)),$$

and $\mathcal{F}(\varepsilon) = cl(\mathbb{R}^3 \setminus \mathcal{K}(\varepsilon))$. We call a region $X \subseteq \mathbb{R}^3$ *ε-free* if $\mathcal{F} \subseteq X \subseteq \mathcal{F}(\varepsilon)$.

This notion of approximating $\mathcal{F}$ is motivated by some applications in which the data is noisy, and/or shallow penetration is allowed. For example, each atom in a protein is best modelled as a "fuzzy" ball rather than a hard ball [15]. We can model this fuzziness by allowing any atom $D(b,r)$ to be intersected by other atoms, but only within the shell

$D(b,r) \setminus D(b,(1-\varepsilon)r)$ for some $\varepsilon > 0$. In this way, the atoms of two docking molecules may penetrate a little at the desired placement. Although $\mathcal{F}$ can have large complexity, namely, up to $O(m^2n^2)$ in $\mathbb{R}^3$, we present a technique for constructing an ε-free region X of considerably smaller complexity. We thus compute X and a placement $t^* \in X$ such that $H_U(\mathcal{A}+t^*,\mathcal{B}) \leq (1+\varepsilon)\sigma_U(\mathcal{A},\mathcal{B};X)$. We refer to such an approximation $H_U(\mathcal{A}+t^*,\mathcal{B})$ as a *pseudo-ε-approximation* for $\sigma_U(\mathcal{A},\mathcal{B};\mathcal{F})$.

Lemma 3.6 *In three dimensions, an ε-free region X of size $O(mn/\varepsilon^3)$ can be computed in time*

$$O((mn/\varepsilon^3)\log(mn/\varepsilon)).$$

Proof: Let $\mathcal{D} = \{D_{ij} = D(b_j - a_i, \rho_i + r_j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$. We insert each ball $D_{ij} \in \mathcal{D}$ into an oct-tree T. Let C_v denote the cube associated with a node v of T. In order to insert D_{ij}, we visit T in a top-down manner. Suppose we are at a node v. If $C_v \subseteq D_{ij}$, we mark v as black and stop. If $C_v \cap D_{ij} \neq \emptyset$ and the size of C_v is at least $\varepsilon(\rho_i + r_j)/2$, then we recursively visit the children of v. Otherwise, we stop, leaving v unmarked. After we insert all balls from $\mathcal{D}$, if all eight children of a node v are marked black, we mark v as black too. Let $V = \{v_1, v_2, \ldots, v_k\}$ be the set of highest marked nodes, i.e., each v_i is marked black but none of its ancestors is black. It is easy to verify that each D_{ij} marks at most $O(1/\varepsilon^3)$ nodes as black, because the nodes a fixed D_{ij} marks are disjoint and of size at least $\varepsilon(\rho_i + r_j)/2$; thus $|V| = O(mn/\varepsilon^3)$. The whole construction takes $O((mn/\varepsilon^3)\log(mn/\varepsilon))$ time, and obviously $\mathcal{K}(\varepsilon) \subseteq \bigcup_{v\in V} C_v \subseteq \mathcal{K}$. Set $X = \mathrm{cl}(\mathbb{R}^3 \setminus \bigcup_{v\in V} C_v)$; it is an ε-free region, as claimed. ∎

Furthermore, let $r(B), r(A)$, and $\tau = r(\mathcal{B}) - r(\mathcal{A})$ be as defined earlier in this section. We prove the following result.

Lemma 3.7 *Let $t^* \in X$ be the closest point of τ in X. Then*

$$H_U(\mathcal{A}+t^*,\mathcal{B}) \leq (1+2\sqrt{3})\sigma_U(\mathcal{A},\mathcal{B};X).$$

Proof: Let $\hat{\delta} = \sigma_U(\mathcal{A},\mathcal{B};X)$ and $\hat{t} \in X$ the placement so that $H_U(\mathcal{A}+\hat{t},\mathcal{B}) = \hat{\delta}$. Then

$$\|\hat{t} - \tau\| = \|\hat{t} - r(\mathcal{B}) + r(\mathcal{A})\| = d(r(\mathcal{A}) + \hat{t}, r(\mathcal{B})).$$

A result in [3] implies that $d(r(\mathcal{A}) + \hat{t}, r(\mathcal{B})) \leq \sqrt{3}\hat{\delta}$. On the other hand,

$$\begin{aligned} H_u(\mathcal{A}+t^*,\mathcal{B}) &\leq \hat{\delta} + \|\hat{t} - t^*\| \leq \hat{\delta} + \|\hat{t} - \tau\| + \|\tau - t^*\| \\ &\leq \hat{\delta} + 2\|\tau - \hat{t}\| \leq \hat{\delta} + 2\sqrt{3}\hat{\delta} = (1+2\sqrt{3})\sigma_U(\mathcal{A},\mathcal{B};X). \end{aligned}$$

∎

The point $t^* \in X$ closest to τ can be computed as follows. Recall that in Lemma 3.6, $X = \mathrm{cl}(\mathbb{R}^3 \setminus \bigcup_{v\in V} C_v)$. Set $\bar{X} = \mathrm{cl}(\mathbb{R}^3 \setminus X) = \bigcup_{v\in V} C_v$; $\bar{X}$ consists of a set of openly disjoint cubes. We first check whether $\tau \in \bar{X}$ by a point-location operation. If the answer is no, then $\tau \in X$, and we return $t^* = \tau$. Otherwise, t^* is a point on $\partial X = \partial \bar{X}$ that is closest to τ. In that case, t^* is either a vertex of a cube in V, or lies in the interior of an edge or of

a face of a cube in V. For each node $v \in V$ and for each boundary feature $\xi \subset C_v$, that is, a face, an edge, or a vertex of C_v, we compute the point in ξ closest to τ. Let Q_v be the resulting set of closest points. We then check, for each $q \in Q_v$, whether $q \in \partial\bar{X}$, by testing whether at least one neighboring cube is unmarked. This can be achieved by performing point location operations in T. Finally, from among those points of Q_v that lie on $\partial\bar{X}$ (thus on ∂X), we return the one that is closest to τ. There are $O(mn/\varepsilon^3)$ cubes, and each has constant number of boundary features. Furthermore, at most a constant number of nodes in V contain a given point, and each point location operation takes $O(\log(mn/\varepsilon))$ time. Hence, t^* can be computed in $O((mn/\varepsilon^3)\log(mn/\varepsilon))$ time. MICHA SAYS: Is the thing inside the log ok? ⟵

We can compute $H_U(\mathcal{A}+t^*, \mathcal{B})$ in $O((n^2+m^2)\log^2 mn)$ time, as described in Section 3.2, so we can approximate $\sigma_U(\mathcal{A}, \mathcal{B}; X)$, up to a constant factor, in $O((n^2+m^2)\log^2 mn)$ time. We then draw an appropriate grid around t^* and use it to compute an ε-approximation of $\sigma_U(\mathcal{A}, \mathcal{B}; X)$, as in Section 3.2, with the difference that we only test those grid points that lie in X. We thus obtain the following result.

Theorem 3.8 *Given $\mathcal{A}$, $\mathcal{B}$ in $\mathbb{R}^3$ and $\varepsilon > 0$, we can compute in $O(((n^2+m^2)/\varepsilon^3)\log^2 mn)$ time, an ε-free region $X \subseteq \mathbb{R}^3$ and a placement $t \in X$ of $\mathcal{A}$, such that*

$$H_U(\mathcal{A}+t, \mathcal{B}) \leq (1+\varepsilon)\sigma_U(\mathcal{A}, \mathcal{B}; X).$$

4 RMS and Summed Hausdorff Distance between Points

We first establish a result on simultaneous approximation of the Voronoi diagrams of several point sets, which we believe to be of independent interest, and then we apply this result to approximate $\sigma_R(\mathcal{A}, \mathcal{B})$ and $\sigma_S(\mathcal{A}, \mathcal{B})$ for *point sets* $\mathcal{A} = \{a_1, \ldots, a_m\}$ and $\mathcal{B} = \{b_1, \ldots, b_n\}$ in any dimension.

4.1 Simultaneous approximation of Voronoi diagrams

Given a family $\{P_1, \ldots, P_l\}$ of point sets in $\mathbb{R}^d$, with a total of N points, and a parameter $\varepsilon > 0$, we wish to construct a subdivision of $\mathbb{R}^d$, so that, for any $x \in \mathbb{R}^d$, we can quickly compute points $p_i \in P_i$, for all $1 \leq i \leq l$, with the property that $d(x, p_i) \leq (1+\varepsilon)d(x, P_i)$, where $d(x, P_i) = \min_{q \in P_i} d(x, q)$. Our data structure is based on a recent result by Arya and Malamatos [9]: Given a set P of n points and a parameter $\varepsilon > 0$, they construct a partition Ξ of $\mathbb{R}^d$ into $O(n/\varepsilon^d)$ cells; each cell $\Delta \in \Xi$ is associated with a point $\phi(\Delta) \in P$, so that for any point $q \in \Delta$, $d(q, \phi(\Delta)) \leq (1+\varepsilon)d(q, P)$. Ξ is the partition induced by the leaves of a compressed quad tree T [22], built on an initial hypercube C that contains P. Ξ and T can be constructed in $O((n/\varepsilon^d)\log(n/\varepsilon))$ time, and the cell of Ξ containing a query point can be located in $O(\log(n/\varepsilon))$ time.

Let C be a hypercube containing $\bigcup_{i=1}^{l} P_i$. We construct the above compressed quad tree T_i for point set P_i, and let Ξ_i be the resulting subdivision. We then merge $T_1, \ldots, T_l$ into a single compressed quad tree T [22] and thus effectively overlay $\Xi_1, \ldots, \Xi_l$. In particular, we start with T_1 and insert cells of Ξ_i's one by one, for $2 \leq i \leq l$. We refine T after each insertion so that we still maintain a compressed quad tree structure [9]. Since all T_i's are built using the same initial hypercube C, any two hypercubes from T_i's are either disjoint or one containing another. Hence the insertion of each hypercube creates at most 2^d new leaves. Let Π be the resulting overlay of $\Xi_1, \ldots, \Xi_l$; Π is a refinement of each Ξ_i and $|\Pi| = O(N/\varepsilon^d)$. Since the merged tree T is also a compressed quad tree, the cell of Π containing any query point can be computed in $O(\log(N/\varepsilon))$ time. For any cell $\Delta \in \Pi$, let $\phi_i(\Delta) \in P_i$ denote the point associated with the cell $\Delta_i \in \Xi_i$ that contains Δ. Recall that, for any point $q \in \Delta$, $\phi_i(\Delta)$ is an ε-nearest neighbor w.r.t. P_i, i.e.,

$$d(q, P_i) \leq d(q, \phi_i(\Delta)) \leq (1 + \varepsilon) d(q, P_i).$$

If we store all the $\phi_i(\Delta)$'s for each cell $\Delta \in \Pi$ (i.e., in the leaf nodes of T), we need $O(l \cdot N/\varepsilon^d)$ space, which we cannot afford. So we instead store ϕ_i at appropriate internal nodes of T. More specifically, for a fixed $1 \leq i \leq n$, and for any cell $\Delta_i \in \Xi_i$, let $v \in T$ be the node in the merged tree T associated with the hypercube of Δ_i. We store $\phi_i(\Delta_i)$ at node v. **Sariel: Please verify and fix it if necessary.** Since $|\Xi_i| = O(|P_i|)/\varepsilon^d)$, the total storage needed to store $\phi_i(\cdot)$'s is $\sum_{i=1}^{l} O(|P_i|)/\varepsilon^d) = O(N/\varepsilon^d)$. To query with a point q lying in a cell $\Delta \in \Pi$, we collect $\phi_i(\Delta)$, $1 \leq i \leq l$, while traversing the path from the root to the leaf of T associated with Δ. As ϕ_i is stored at most twice along any path from the root to a leaf of T (once due to the outer hypercube, and possibly once due to the inner hypercube), we conclude the following.

Theorem 4.1 *Given a family $\{P_1, \ldots, P_l\}$ of point sets in $\mathbb{R}^d$, with a total of N points, and a parameter $\varepsilon > 0$, we can compute in $O((N/\varepsilon^d) \log(N/\varepsilon))$ time a subdivision of $\mathbb{R}^d$ of size $O(N/\varepsilon^d)$ so that, for any point $q \in \mathbb{R}^d$, one can ε-approximate $d(q, P_i)$, for all $1 \leq i \leq l$, in $O(\log(N/\varepsilon) + l)$ time.*

4.2 Approximating $\sigma_R(A, B)$

For $1 \leq i \leq m$, let $P_i = \mathcal{B} - a_i = \{b_j - a_i \mid 1 \leq j \leq n\}$. We construct the preceding decomposition, denoted as Π_A, and the associated compressed quad-tree T_A, for $P_1, \ldots, P_m$, with the given parameter ε; $|\Pi_A| = O(mn/\varepsilon^d)$. Define

$$f_i(t) = d^2(t, P_i) = \min_{1 \leq j \leq n} d^2(t, b_j - a_i),$$

and let

$$F_A(t) = m \cdot h_R^2(\mathcal{A} + t, \mathcal{B}) = \sum_{i=1}^{m} f_i(t).$$

For each cell $\Delta \in \Pi_A$, define

$$\widehat{F}_{A,\Delta}(t) = \sum_{i=1}^{m} d^2(t, \phi_i(\Delta)).$$

By construction, for any $t \in \Delta$,

$$\begin{aligned} F_A(t) &\leq \widehat{F}_{A,\Delta}(t) = \sum_{i=1}^{m} d^2(t, \phi_i(\Delta)) \\ &\leq \sum_{i=1}^{m} (1+\varepsilon)^2 \cdot d^2(t, P_i) = (1+\varepsilon)^2 F_A(t), \end{aligned}$$

implying that

$$\sqrt{\frac{1}{m}\widehat{F}_{A,\Delta}(t)} \leq (1+\varepsilon) h_R(\mathcal{A} + t, \mathcal{B}).$$

Hence, it suffices to store $\widehat{F}_{A,\Delta}(t)$ at each cell $\Delta \in \Pi_A$. Since $\widehat{F}_{A,\Delta}$ is a quadratic equation in $t \in \mathbb{R}^d$, it can be stored using $O(1)$ space (where the constant depends on d) and updated in $O(1)$ time for each change in $\phi_i(\Delta)$.

If we compute $\widehat{F}_{A,\Delta}$ for each cell $\Delta \in \Pi_A$ independently, the total time is $O(m^2 n/\varepsilon^d)$. We therefore proceed as follows. We perform an in-order traversal of the compressed quadtree T_A. For the cell Δ associated with the first leaf of T_A visited by the procedure, we compute $\widehat{F}_{A,\Delta}$ in $O(m)$ time. For the subsequent leaves we compute $\widehat{F}_{A,\Delta}$ from the value previously computed. Suppose we are currently visiting a cell Δ of Π_A, let Δ' be the previous cell visited by the procedure, let z (resp., z') be the leaf in T_A associated with Δ (resp., Δ'), and let

$$k_{\Delta,\Delta'} = \{i \mid \phi_i(\Delta') \neq \phi(\Delta)\}.$$

The values of $\phi_i(\Delta)$ and $\phi_i(\Delta')$, for all $i \in k_{\Delta,\Delta'}$, are stored along the two paths from z and z' to their nearest common ancestor, which is the portion of T traversed between z and z'. We can compute $k_{\Delta,\Delta'}$ while traversing T and charge the time spent in computing $k_{\Delta,\Delta'}$ to the traversal of T. Since

$$\widehat{F}_{A,\Delta}(t) = \widehat{F}_{A,\Delta'}(t) + \sum_{i \in k_{\Delta,\Delta'}} [d^2(t, \phi_i(\Delta)) - d^2(t, \phi_i(\Delta'))],$$

we can compute $\widehat{F}_{A,\Delta}$ from $\widehat{F}_{A,\Delta'}$ in $O(|k_{\Delta,\Delta'}|)$ time. As $\sum |k_{\Delta,\Delta'}| = O(mn/\varepsilon^d)$, the total time required to compute all $\widehat{F}_{A,\Delta}$'s is $O(mn/\varepsilon^d)$.

Next, we compute, in $O(mn/\varepsilon^d)$ time, a subdivision Π_B on the family $Q_j = \{b_j - a_i \mid 1 \leq i \leq m\}$, for $1 \leq j \leq n$, and a quadratic function $\widehat{F}_{B,\Delta}$ for each cell $\Delta \in \Pi_B$ so that $\widehat{F}_{B,\Delta}(t) \leq (1+\varepsilon)^2 F_B(t)$. We overlay Π_A and Π_B. The same argument as the one used to bound the complexity of Π_A shows that the resulting overlay Π has $O(mn/\varepsilon^d)$ cells and that

it can be computed in $O((mn/\varepsilon^d)\log(mn/\varepsilon))$ time. Finally, for each cell Δ in the overlay, we compute

$$\begin{aligned} t_\Delta &= \arg\min_{t\in\Delta} \max\left\{\sqrt{\widehat{F}_{A,\Delta}(t)/m}, \sqrt{\widehat{F}_{B,\Delta}(t)/n}\right\} \\ &\leq \arg\min_{t\in\Delta}(1+\varepsilon)H_R(\mathcal{A}+t,\mathcal{B}) \end{aligned}$$

and return

$$\min_{\Delta\in\Pi} H_R(\mathcal{A}+t_\Delta,\mathcal{B}) \quad\leq\quad (1+\varepsilon)\sigma_R(\mathcal{A},\mathcal{B}).$$

Hence, we obtain the following.

Theorem 4.2 *Given two sets $\mathcal{A}$ and $\mathcal{B}$ of m and n points in $\mathbb{R}^d$ and a parameter $\varepsilon > 0$, we can:*

i. compute a vector $t^* \in \mathbb{R}^d$ in $O((mn/\varepsilon^d)\log(mn/\varepsilon))$ time, so that

$$H_R(\mathcal{A}+t^*,\mathcal{B}) \quad\leq\quad (1+\varepsilon)\sigma_R(\mathcal{A},\mathcal{B});$$

ii. construct a data structure of size $O(mn/\varepsilon^d)$, in time $O((mn/\varepsilon^d)\log(mn/\varepsilon))$, so that for any query vector $t \in \mathbb{R}^d$, we can compute an ε-approximate value of $H_R(\mathcal{A}+t,\mathcal{B})$ in $O(\log(mn/\varepsilon))$ time.

4.3 Approximating $\sigma_S(A,B)$

Modifying the above scheme, we approximate $\sigma_S(\mathcal{A},\mathcal{B})$ as follows. Let P_i, Q_j, Π_A, and Π_B be as defined above. We define

$$\begin{aligned} G_A(t) &= \sum_{i=1}^{m} d(t,P_i) = m\cdot h_S(\mathcal{A}+t,\mathcal{B}), \\ G_B(t) &= \sum_{j=1}^{n} d(t,Q_j) = n\cdot h_S(\mathcal{B},\mathcal{A}+t). \end{aligned}$$

For each cell $\Delta \in \Pi_A$, let

$$\widehat{G}_{A,\Delta}(t) \quad=\quad \sum_{i=1}^{m} d(t,\phi_i(\Delta)) \leq m(1+\varepsilon)h_S(\mathcal{A}+t,\mathcal{B})$$

and for each cell $\Delta \in \Pi_B$, let

$$\widehat{G}_{B,\Delta}(t) \quad=\quad \sum_{j=1}^{n} d(t,\phi_j(\Delta)) \leq n(1+\varepsilon)h_S(\mathcal{B},\mathcal{A}+t).$$

As above, we overlay Π_A and Π_B. For each cell Δ in the overlay, we wish to compute

$$H_\Delta \;=\; \min_{t\in\Delta} \max\left\{\frac{1}{m}\widehat{G}_{A,\Delta}(t),\;\; \frac{1}{n}\widehat{G}_{B,\Delta}(t)\right\}.$$

Since $\widehat{G}_{A,\Delta}$ and $\widehat{G}_{B,\Delta}$ are not simple algebraic functions, we do not know how to compute, store, and update them efficiently. Nevertheless, we can compute an ε-approximation for $\widehat{G}_{A,\Delta}$ (resp., $\widehat{G}_{B,\Delta}$) that is easier to handle. More precisely, for a given set P of points in $\mathbb{R}^d$, define the 1-median function

$$\mathrm{med}_P(t) \;=\; \sum_{p\in P} d(p,t).$$

For any $\Delta \in \Pi_A$, $\widehat{G}_{A,\Delta}(t) = \mathrm{med}_{\Phi(\Delta)}(t)$, where $\Phi(\Delta) = \{\phi_i(\Delta) \mid 1 \le i \le m\}$. The same is true for $\widehat{G}_{B,\Delta}$, where $\Delta \in \Pi_B$. In Section 4.4, we describe a dynamic data structure that, given a point set P of size n, maintains an ε-approximation of the function $\mathrm{med}_P(\cdot)$ as a function consisting of $O((1/\varepsilon^d)\log(1/\varepsilon))$ pieces; the domain of each piece is a d-dimensional (or the complement of a d-dimensional) hypercube. A point can be inserted into or deleted from P in $O(\log^{d+1} n \log^2(n/\varepsilon)/\varepsilon^d)$ time. Furthermore, given two point sets P and Q in $\mathbb{R}^d$, this data structure can maintain an ε-approximation of $\max_t\{\mathrm{med}_P(t), \mathrm{med}_Q(t)\}$ within the same time bound.

Using this data structure, we can traverse all cells of the overlay of Π_A and Π_B, as in Section 4.2, and compute an ε-approximation of $\widehat{G}_{A,\Delta}$ and $\widehat{G}_{B,\Delta}$ (thus roughly a (2ε)-approximation of G_A and G_B) for each cell Δ of the overlay. However, given two adjacent leaves during the traversal, associated with cells Δ and Δ' respectively, we now spend

$$O(k_{\Delta,\Delta'} \cdot (1/\varepsilon^d)\,\mathrm{polylog}(mn, 1/\varepsilon))$$

time to compute an ε-approximation of $\widehat{G}_{A,\Delta'}$ from that of $\widehat{G}_{A,\Delta}$. Putting everything together, we conclude the following.

Theorem 4.3 *Given two sets $\mathcal{A}$ and $\mathcal{B}$ of m and n points in $\mathbb{R}^d$ and a parameter $0 < \varepsilon \le 1$, we can compute:*

i. a vector $t^* \in \mathbb{R}^d$ in $O(\frac{mn}{\varepsilon^{2d}}\,\mathrm{polylog}(mn, \frac{1}{\varepsilon}))$ time so that

$$H_S(\mathcal{A} + t^*, \mathcal{B}) \le (1+\varepsilon)\sigma_S(\mathcal{A}, \mathcal{B});$$

ii. a data structure of $O(\frac{mn}{\varepsilon^{2d}}\,\mathrm{polylog}(mn, \frac{1}{\varepsilon}))$ size in time $O(\frac{mn}{\varepsilon^{2d}}\,\mathrm{polylog}(mn, \frac{1}{\varepsilon}))$, so that for any query vector $t \in \mathbb{R}^d$, we can ε-approximate $H_S(\mathcal{A}+t, \mathcal{B})$ in time $O(\mathrm{polylog}(mn, \frac{1}{\varepsilon})$.

4.4 Maintaining the 1-median function

Let P be a set of n points in $\mathbb{R}^d$ and let $\varepsilon > 0$ be a parameter. For $x \in \mathbb{R}^d$, define the 1-median function $\mathrm{med}_P(x) = \sum_{p \in P} d(p, x)$, as above. We describe a dynamic data structure that maintains a function $f : \mathbb{R}^d \to \mathbb{R}$ as the points are inserted into or deleted from P so that

$$\mathrm{med}_P(x) \leq f(x) \leq (1 + \varepsilon)\mathrm{med}_P(x), \quad \forall x \in \mathbb{R}^d.$$

We maintain a height-balanced binary tree T with n leaves, each storing a point of P. For a node $v \in T$, let $P_v \subseteq P$ be the set of points stored at the leaves of the subtree rooted at v; set $n_v = |P_v|$. For each node v of height i (leaves have height 0), set $\lambda_i = i\varepsilon/2h$, where $h = O(\log n)$ is the height of the tree T. We associate with a node v, at height i, a function f_v that is a λ_i-approximation of med_{P_v}, i.e.,

$$\mathrm{med}_{P_v}(x) \leq f_v(x) \leq (1 + \lambda_i)\mathrm{med}_{P_v}(x), \quad \forall x \in \mathbb{R}^d.$$

The description complexity of f_v is $O((h/\varepsilon)^d \log(h/\varepsilon))$. Finally, we maintain a function f of description complexity $O((1/\varepsilon^d)\log(1/\varepsilon))$ that is an $(/\varepsilon/3)$-approximation of f_{root} and thus an ε-approximation of $\mathrm{med}_P(x)$.

More specifically, if a leaf v stores the point $p \in P$, then set $f_v(x) = d(x, p)$. For all internal nodes v, we compute f_v in a bottom-up manner as follows. Let w and z be the children of v. By induction, suppose we have already computed the functions f_w and f_v, each of descriptive complexity $O((h/\varepsilon)^d \log(h/\varepsilon))$. Set

$$g_v(x) = f_w(x) + f_z(x).$$

Since $\mathrm{med}_{P_v}(x) = \mathrm{med}_{P_w}(x) + \mathrm{med}_{P_z}(x)$, by induction hypothesis,

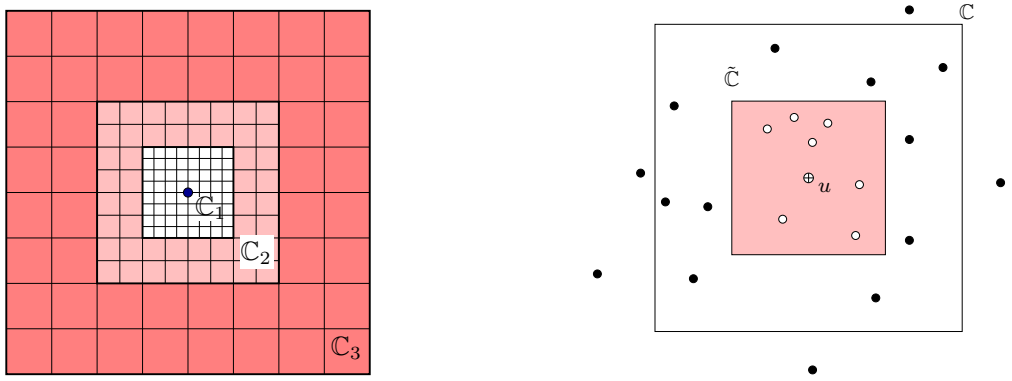
$$g_v \leq \left(1 + \frac{(i-1)\varepsilon}{2h}\right) \mathrm{med}_{P_v}(x) \qquad \forall x \in \mathbb{R}^d \tag{1}$$

However, the description complexity of g_v is more than what we desire. We therefore approximate g_v by a simpler function f_v as follows. For $x \in \mathbb{R}^d$ and $r \in \mathbb{R}$, let $\mathbb{C}(x, r)$ be the hypercube of side length $2r$ centered at x. For simplicity, let $\lambda = \varepsilon/2h$. We compute $u = \arg\min_x g_v(x)$ and set $\mu = g_v(u)$. Let $\mathbb{C}_j = \mathbb{C}(u, (8\mu/n_v)2^j)$ for $1 \leq j \leq \log(1/\lambda)$. Partition each cubic shell $\mathbb{C}_{j+1} \setminus \mathbb{C}_j$ into hypercubes by a d-dimensional grid G_j in which each cell has side length $2^j\lambda\mu/(10\sqrt{d}n_v)$; G_j has $O(1/\lambda^d)$ cells. The union of G_j's is an exponential grid with $O(1/\lambda^d \log(1/\lambda))$ cells that covers the hypercube $\mathbb{C} = \mathbb{C}(u, 8\mu/(\lambda n_v))$. See Figure 3 (a) for an illustration. In each cell $\Delta \in G_j$, pick an arbitrary point $y \in \Delta$ and set

$$f_v(x) \;=\; g_v(y) + 2^j\lambda\mu/4, \qquad \forall x \in \Delta. \tag{2}$$

For points outside $\mathbb{C}$, we set

$$f_v(x) \;=\; n_v d(u, x) + \mu, \quad \forall x \in \mathbb{R}^d \setminus \mathbb{C}. \tag{3}$$



(a) (b)

Figure 3: (a) An exponential grid with 3 layers. (b) The larger (resp., smaller) box is $\mathbb{C}$ (resp., $\tilde{\mathbb{C}}$), and the set of hollow circles is $\tilde{P}_v$.

Hence, the function f_v is piecewise-constant inside $\mathbb{C}$ and a quadratic function outside $\mathbb{C}$. The description complexity of f_v is $O((1/\lambda^d)\log(1/\lambda)) = O((h/\varepsilon)^d \log(h/\varepsilon))$. Since inductively f_w and f_z also have the same structure, the point u can be computed by evaluating the function g_v at the vertices of the exponential grids drawn for f_w and f_z. At each point x, we can evaluate $g_v(x)$ in time $O(\log(h/\varepsilon))$ time by simply locating x in the two exponential grids. Hence, we can compute the point u in time $O((h/\varepsilon)^d \log^2(h/\varepsilon))$. We spend another $O((h/\varepsilon)^d \log(h/\varepsilon))$ time to compute f_v. That $f_v(\cdot)$ is indeed a λ_i-approximation of $\mathrm{med}_{P_v}(\cdot)$ is proved in Lemmas 4.4 and 4.5. This finishes the induction step. Using the same procedure, we compute an $(\varepsilon/3)$-approximation, f, of f_{root}, of descriptive complexity $O((1/\varepsilon)^d \log(1/\varepsilon))$. By construction, for all $x \in \mathbb{R}^d$,

$$\begin{aligned} \mathrm{med}_P(x) \leq f(x) &\leq (1+\varepsilon/3) f_{\mathrm{root}}(x) \\ &\leq (1+\varepsilon/3)(1+\varepsilon/2)\mathrm{med}_P(x) \\ &\leq (1+\varepsilon)\mathrm{med}_P(x). \end{aligned}$$

Obviously, the size of the above data structure is $O((n/\varepsilon^d)\log^d(n)\log(n/\varepsilon))$. To insert or delete a point p, we follow a path from the leaf z storing p to the root of T and recompute f_v at all nodes along this path, and then compute f from f_{root}. Hence, the update time is $O(\log^{d+1} n \log^2(n/\varepsilon)/\varepsilon^d)$. The only missing component now is to show that f_v, as constructed above at each node $v \in T$, is indeed a λ_i-approximation of med_{P_v}.

Lemma 4.4 *Let v be a node of T at height i. For any $1 \leq j \leq \log(1/\lambda)$ and for any $\Delta \in G_j \subseteq \mathbb{C}$,*

$$\mathrm{med}_{P_v}(x) \leq f_v(x) \leq (1+\lambda_i)\mathrm{med}_{P_v}(x), \quad \forall x \in \Delta.$$

Proof: The triangle inequality implies that for any $x, y \in \mathbb{R}^d$,

$$| \operatorname{med}_{P_v}(x) - \operatorname{med}_{P_v}(y) | \quad \leq \quad n_v d(x,y) \quad \leq \quad \operatorname{med}_{P_v}(x) + \operatorname{med}_{P_v}(y). \tag{4}$$

Therefore, by construction of the exponential grid,

$$| \operatorname{med}_{P_v}(x) - \operatorname{med}_{P_v}(y) | \quad \leq \quad \frac{2^j \lambda \mu}{10}, \quad \forall x, y \in \Delta. \tag{5}$$

Equation (2) and (5) imply that

$$f_v(x) \quad = \quad g_v(y) + \frac{2^j \lambda \mu}{4} \quad \geq \quad \operatorname{med}_{P_v}(y) + 2^{j-2}\lambda\mu \quad \geq \quad \operatorname{med}_{P_v}(x).$$

Substituting u for y in (4), we obtain

$$\operatorname{med}_{P_v}(x) \quad \geq \quad n_v d(x,u) - \operatorname{med}_{P_v}(u) \geq 2^{j-1} \cdot 8\mu - \mu \geq 2^j \mu, \tag{6}$$

where the last inequality follows from the fact that for all $x \in \mathbb{C}_j \backslash \mathbb{C}_{j-1}$, $d(x,u) \geq 2^{j-1} 8\mu / n_v$. Hence, for any $x \in \Delta$,

$$\begin{aligned}
f_v(x) &= g_v(y) + 2^{j-2}\lambda\mu \\
&\leq \left(1 + (i-1)\frac{\varepsilon}{2h}\right) \operatorname{med}_{P_v}(y) + 2^{j-2}\lambda\mu \qquad \text{(using 1)} \\
&\leq \left(1 + (i-1)\frac{\varepsilon}{2h}\right) \left[\operatorname{med}_{P_v}(x) + \frac{2^j \lambda \mu}{10}\right] + 2^{j-2}\lambda\mu \qquad \text{(using (5))} \\
&\leq \left(1 + (i-1)\frac{\varepsilon}{2h}\right) \cdot \operatorname{med}_{P_v}(x) + 2^{j-1}\lambda\mu \\
&\leq \left(1 + (i-1)\frac{\varepsilon}{2h} + \frac{\lambda}{2}\right) \cdot \operatorname{med}_{P_v}(x) \qquad \text{(using (6))} \\
&\leq \left(1 + \frac{i\varepsilon}{2h}\right) \operatorname{med}_{P_v}(x) \quad = \quad (1 + \lambda_i) \operatorname{med}_{P_v}(x).
\end{aligned}$$

■

Lemma 4.5 *Let v be a node of T at height i. Then for any $x \in \mathbb{R}^d \setminus \mathbb{C}$,*

$$\operatorname{med}_{P_v}(x) \leq f_v(x) \leq (1 + \lambda_i) \operatorname{med}_{P_v}(x).$$

Proof: By (4),

$$| \operatorname{med}_{P_v}(x) - \operatorname{med}_{P_v}(u) | \quad \leq \quad n_v d(x,u) \quad \leq \quad \operatorname{med}_{P_v}(x) + \mu. \tag{7}$$

The first inequality of the lemma is now immediate because

$$\operatorname{med}_{P_v}(x) \leq n_v d(x,u) + \operatorname{med}_{P_v}(u) \leq n_v d(x,u) + \mu = f_v(x).$$

As for the second inequality, we first obtain an upper bound on μ in terms of $\mathrm{med}_{P_v}(x)$. Let

$$\tilde{\mathbb{C}} = \mathbb{C}(u, 4\mu/(\lambda n_v))$$

and $\tilde{P_v} = P_v \cap \tilde{\mathbb{C}}$ (see Figure 3 (b)). Then

$$\mu \;\geq\; \mathrm{med}_{P_v}(u) = \sum_{p \in P_v} d(p,u) \geq \sum_{p \notin \tilde{P_v}} d(p,u) \geq |P_v \setminus \tilde{P_v}| \cdot \frac{4\mu}{\lambda n_v}.$$

Therefore

$$|\tilde{P_v}| \;\geq\; n_v - \lambda n_v/4 \;=\; (1-\lambda/4)n_v.$$

On the other hand, for any $x \in \mathbb{R}^d \setminus \mathbb{C}$ and $y \in \tilde{\mathbb{C}}$, we have that

$$d(x,y) \geq 8\mu/\lambda n_v - 4\mu/\lambda n_v = 4\mu/\lambda n_v.$$

Hence, as long as $\lambda = \varepsilon/2h \leq 2$, we have

$$\mathrm{med}_{P_v}(x) \;\geq\; \sum_{p \in \tilde{P_v}} d(p,x) \geq \left(1 - \frac{\lambda}{4}\right) n_v \cdot \frac{4\mu}{\lambda n_v} \geq \frac{2\mu}{\lambda},$$

thereby implying that $2\mu \leq \lambda \mathrm{med}_{P_v}(x)$. Using (3) and (7),

$$\begin{aligned} f_v(x) &= n_v d(x,u) + \mu \\ &\leq \mathrm{med}_{P_v}(x) + 2\mu \\ &\leq \mathrm{med}_{P_v}(x) + \lambda \mathrm{med}_{P_v}(x) \leq (1+\lambda_i) \cdot \mathrm{med}_{P_v}(x). \end{aligned}$$

■

4.5 A randomized algorithm

We briefly describe below a simple randomized algorithm to approximate $\sigma_R(\mathcal{A}, \mathcal{B})$. The algorithm of approximating $\sigma_S(\mathcal{A}, \mathcal{B})$ is similar. Let t^* be the optimal translation, i.e., $H_R(\mathcal{A} + t^*, \mathcal{B}) = \sigma_R(\mathcal{A}, \mathcal{B})$.

Lemma 4.6 *For a random point a_k from $\mathcal{A}$, $d(a_k + t^*, \mathcal{B}) \leq 2\sigma_R(\mathcal{A}, \mathcal{B})$, with probability greater than $1/2$. The same claim holds for $\sigma_S(\mathcal{A}, \mathcal{B})$.*

Proof: Let a_k be a random point from $\mathcal{A}$, where each point of $\mathcal{A}$ is chosen with equal probability. Let Y be the random variable $Y = d(a_k + t^*, \mathcal{B})$. Then $E[Y] = \frac{1}{m}\sum_{i=1}^{m} d(a_i + t^*, \mathcal{B})$. Moreover,

$$\sigma_R(\mathcal{A}, \mathcal{B}) = H_R(\mathcal{A} + t^*, \mathcal{B}) = \frac{1}{m}\sum_{k=1}^{m} d(a_k + t^*, \mathcal{B}) = E[Y].$$

The lemma now follows immediately from Markov's inequality. ∎

Choose a random point $a_k \in \mathcal{A}$. Let $t_j = b_j - a_k$ and $\delta_j = H_R(\mathcal{A} + t_j, \mathcal{B})$, for $1 \leq j \leq n$. It then follows from Lemma 4.6 and the same argument as in Lemma 3.7, that $\min_j \delta_j$ is a constant-factor approximation of $\sigma_R(\mathcal{A}, \mathcal{B})$, with probability greater than $1/2$. Computing δ_j exactly is expensive in $\mathbb{R}^d$, therefore we compute an approximate value of δ_j, for $1 \leq j \leq n$, in time $O((m+n)\log mn)$, by performing approximate nearest-neighbor queries [9]. We can improve this constant-factor approximation algorithm to compute a $(1+\varepsilon)$-approximation of $\sigma_R(\mathcal{A}, \mathcal{B})$ using the same technique as in Section 3. We thus obtain the following result.

Theorem 4.7 *Given two sets $\mathcal{A}$ and $\mathcal{B}$ of m and n points, respectively, in $\mathbb{R}^d$, and a parameter $\varepsilon > 0$, we can compute, in* MICHA SAYS: randomized expected? *time $O((mn/\varepsilon^d)\log mn)$, two translation vectors t_1 and t_2, such that, with probability greater than $1/2$,* ⟵

$$H_R(\mathcal{A} + t_1, \mathcal{B}) \leq (1+\varepsilon)\sigma_R(\mathcal{A}, \mathcal{B}) \quad \textit{and} \quad H_S(\mathcal{A} + t_2, \mathcal{B}) \leq (1+\varepsilon)\sigma_S(\mathcal{A}, \mathcal{B}).$$

5 Conclusions

We provide in this paper some initial study of various problems related to minimizing Hausdorff distance between sets of points, disks, and balls. One natural question following our study is to compute exactly or approximately the smallest Hausdorff distance over all possible rigid motions in $\mathbb{R}^2$ and $\mathbb{R}^3$. Given two sets of points $\mathcal{A}$ and $\mathcal{B}$ of size n and m, respectively, let Δ be the maximum of the diameters of $\mathcal{A}$ and $\mathcal{B}$. We believe that there is a randomized algorithm with roughly $mn\sqrt{\Delta}$ expected time, that approximates the optimal summed-Hausdorff distance (or rms-Hausdorff distance) under rigid motions in the plane. The algorithm combines our randomized approach from Section 4.5, a framework to convert the original problem to a pattern matching problem [19], and a result by Amir *et al.* on string matching [8]. However, this approach does not extend to families of balls. We leave the problem of computing the smallest Hausdorff distance between sets of points or balls under rigid motions as an open question for further research. Another question is to approximate efficiently the best Hausdorff distance under certain transformations when partial matching is allowed. The traditional approaches using reference points break down with partial matching.

MICHA SAYS: Some comments about the bibliography too - not implemented yet. ⟵

References

[1] P. K. Agarwal and J. Matoušek, Ray shooting and parametric search, *SIAM J. Comput.*, 22 (1993), 540–570.

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

[3] O. Aichholzer, H. Alt, and G. Rote, Matching shapes with a reference point, *Intl. J. Comput. Geom. and Appl.*, 7 (1997), 349–363.

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

[5] H. Alt, P. Brass, M. Godau, C. Knauer, and C. Wenk, Computing the hausdorff distance of geometric patterns and shapes, in: *Discrete Computational Geometry — The Goodman-Pollack Festchrift* (B. Aronov, S. Baus, J. Pach, and M. Sharir, eds.), Springer-Verlag, Heidelberg, 2003, pp. 65–76.

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

[7] N. Amenta and R. Kolluri, The medial axis of a union of balls, *Comput. Geom: Theory Appl.*, 20 (2001), 25–37.

[8] A. Amir, E. Porat, and M. Lewenstein, Approximate subset matching with Don't Cares, *Proc. 12th ACM-SIAM Symp. Discrete Algorithms*, 2001, pp. 305–306.

[9] S. Arya and T. Malamatos, Linear-size approximate voronoi diagrams, *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, 2002, pp. 147–155.

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

[11] D. Cardoze and L. Schulman, Pattern matching for spatial point sets, *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.*, 1998, pp. 156–165.

[12] L. P. Chew, D. Dor, A. Efrat, and K. Kedem, Geometric pattern matching in d-dimensional space, *Discrete Comput. Geom.*, 21 (1999), 257–274.

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

[14] M. T. Goodrich, J. S. Mitchell, and M. W. Orletsky, Practical methods for approximate geometric pattern matching under rigid motion, *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 1994, pp. 103–112.

[15] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov, Principles of docking: An overview of search algorithms and a guide to scoring functions, *Proteins: Structure, Function, and Genetics*, 47 (2002), 409–443.

[16] S. Har-Peled, A replacement for Voronoi diagrams of near linear size, *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, 2001, pp. 94–103.

[17] D. P. Huttenlocher, K. Kedem, and J. M. Kleinberg, On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane, *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, 1992, pp. 110–120.

[18] D. P. Huttenlocher, K. Kedem, and M. Sharir, The upper envelope of Voronoi surfaces and its applications, *Discrete Comput. Geom.*, 9 (1993), 267–291.

[19] P. Indyk, R. Motwani, and S. Venkatasubramanian, Geometric matching under noise: Combinatorial bounds and algorithms, *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, 1999, pp. 457–465.

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

[21] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM*, 30 (1983), 852–865.

[22] H. Samet, *Spatial Data Structures: Quadtrees, Octrees, and Other Hierarchical Methods*, Addison-Wesley, Reading, MA, 1989.

[23] S. Seeger and X. Laboureux, Feature extraction and registration: An overview, *Principles of 3D Image Analysis and Synthesis*, (2002), 153–166.