Hausdorff Distance under Translation for Points and Balls

Pankaj K. Agarwal
Department of Computer Science, Duke University, Durham, NC 27708-0129, U.S.A.
and
Sariel Har-Peled
Department of Computer Science, University of Illinois, Urbana, IL 61801, U.S.A.
and
Micha Sharir
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.
and
Yusu Wang
Department of Computer Science and Engineering, Ohio State University.

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 variants that are more robust — 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.

Categories and Subject Descriptors: F.2.2 [**Nonnumerical Algorithms and Problems**]: Pattern Matching

General Terms: Algorithms, Theory

P.A. is supported 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 grant CCR-01-32901. M.S. is 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, CCR-02-04118 and DOE grant DE-FG02-06ER25735.Part of this work was done while Y.W. was at Duke University.

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 [Alt and Guibas 2000; Halperin et al. 2002; Seeger and Laboureux 2002]. 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 [Halperin et al. 2002].

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 [Halperin et al. 2002]. Moreover, because of efficiency considerations, one sometimes restricts the set of allowed transformations, such as to translations only. For example, to reduce the time complexity, one approach to approximate the optimal Hausdorff distance under the rigid motion is by first discretizing the rotational space. Next, for each rotation, we compute the optimal Hausdorff distance under 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* [Alt and Guibas 2000], 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 [Amenta and Kolluri 2001], which is therefore the main type of input assumed in the first part of this paper.

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 $\mathcal{A} + t = \left\{ a + t \;\middle|\; a \in \mathcal{A} \right\}$. 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 [Alt and Guibas 2000], 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

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

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" [Chew et al. 1999], 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

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

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 [Indyk et al. 1999], 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.

Previous results. It is beyond the scope of this paper to discuss all the results on shape matching. We refer the reader to [Alt and Guibas 2000; Halperin et al. 2002; Seeger and Laboureux 2002] 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 [Alt et al. 1995]. 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 [Agarwal and Matoušek 1993]. Huttenlocher *et al.* [Huttenlocher et al. 1993] 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.* [Chew et al. 1999] 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 $O(m^2n^2(m+n)\log^2 mn)$ time [Chew et al. 1997].

Faster approximation algorithms to compute $\sigma(\mathcal{A},\mathcal{B})$ were first proposed by Goodrich *et al.* [Goodrich et al. 1994]. Aichholzer *et al.* proposed a framework of approximation algorithms using *reference points* [Aichholzer et al. 1997]. 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.* [Indyk et al. 1999] study the partial matching problem defined as follows: given a query $r>0$, compute a rigid transformation 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 [Cardoze and Schulman 1998] 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 [Agarwal et al. 1994; Alt et al. 1995; Alt et al. 2003]. Atallah [Atallah 1983] presents an algorithm for computing $H_U(\mathcal{A},\mathcal{B})$ for two convex polygons in $\mathbb{R}^2$. Agarwal *et al.* [Agarwal et al. 1994] 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 segments in the plane can be computed in time $O((mn)^3\log^2(mn))$ (Chew *et al.* [Chew et al. 1997]). Aichholzer *et al.* [Aichholzer et al. 1997] 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 refer-

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

ence 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.

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 $D(c, r)$ be 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 [Halperin et al. 2002]. 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 union of disks. 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 Voronoi diagram of the boundary of the union of balls in $\mathbb{R}^3$ [Amenta and Kolluri 2001]. 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 $H_U(\mathcal{A}+t, \mathcal{B}) \le (1+\varepsilon)\sigma_U(\mathcal{A}, \mathcal{B})$. 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

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

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 [Halperin et al. 2002].

Finally, consider 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 algorithm that computes, in $O\big((mn/\varepsilon^d)\log(1/\varepsilon)\log(mn/\varepsilon)\big)$ time, an ε-approximation of $\sigma_R(\mathcal{A}, \mathcal{B})$.

It also provides a data structure of size $O((mn/\varepsilon^d)\log(1/\varepsilon))$ 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 $P_1, \ldots, P_\ell$ of point sets in $\mathbb{R}^d$, with a total of N points, we construct a decomposition of $\mathbb{R}^d$ into $O((N/\varepsilon^d)\log(1/\varepsilon))$ cells, which is an ε-approximation of each of the Voronoi diagrams of $P_1, \ldots, P_\ell$, in the sense defined in [Arya and Malamatos 2002; Har-Peled 2001]. We can preprocess this decomposition in $O((N/\varepsilon^d)\log(1/\varepsilon)\log(N/\varepsilon))$ time into a data structure of size $O((N/\varepsilon^d)\log(1/\varepsilon))$, so that for a query point q, an ε-approximate nearest neighbor of q in every P_i can be computed in a total time of $O(\log(N/\varepsilon)+\ell)$. Moreover, given a semigroup operation $+$, 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 for maintaining an ε-coreset of P for 1-median. That is, we show that we can maintain a weighted subset $Q \subseteq P$ of size $O((1/\varepsilon^d)\log(1/\varepsilon))$ so that the weighted sum of distances from any point $x \in \mathbb{R}^d$ to the points of Q is an ε-approximation of the sum of distances from x to the points of P. Q can be updated efficiently as the points in P are inserted or deleted. Using similar ideas, an algorithm for computing an ε-coreset of P for k-medians has been proposed in [Har-Peled and Mazumdar 2004]. Finally, we also present a simple randomized algorithm to approximate $\sigma_R(\mathcal{A}, \mathcal{B})$ and $\sigma_S(\mathcal{A}, \mathcal{B})$ efficiently. We remark that our computation model assumes that we can compute the floor function in constant time.

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 balls $A_i = D(a_i, \rho_i) \in \mathcal{A}$ and $B_j = D(b_j, r_j) \in \mathcal{B}$, we define

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

Note that if A_i and B_j intersect, then $d(A_i, B_j) \leq 0$. Otherwise, it is 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.

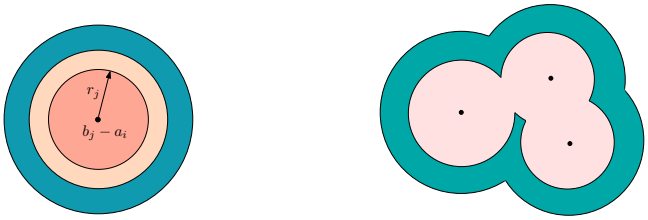
2.1 Computing $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$ in $\mathbb{R}^2$ and $\mathbb{R}^3$

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 $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F}) \leq \delta$. We then use the parametric-searching technique [Agarwal et al. 1994; Megiddo 1983] to compute $\sigma(\mathcal{A}, \mathcal{B}; \mathcal{F})$. Given $\delta > 0$, for $1 \leq i \leq m$, let $V_i \subseteq \mathbb{R}^d$ be the set of vectors $t \in \mathbb{R}^d$ such that

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

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 $U_i^+ = \bigcup_{j \leq n} D_{ij}^+$ is the set of vectors that satisfy $\min_{1 \leq j \leq n} d(A_i + t, B_j) \leq \delta$, and the interior of



(a) (b)

Fig. 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).

$U_i^- = \bigcup_{j\le n} D_{ij}^-$ violates $0 \le \min_{1\le j\le n} d(A_i + t, B_j)$. Hence, $V_i = \mathrm{cl}(U_i^+ \setminus U_i^-)$ is the set of valid translations for the ball A_i. Let

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

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 $h(\mathcal{A}+t,\mathcal{B}) \le \delta$. Similarly, we define

$$V(\mathcal{B},\mathcal{A}) \subseteq \mathcal{F} = \Big\{ t \in \mathcal{F} \;\Big|\; h(\mathcal{B},\mathcal{A}+t) \le \delta \Big\}.$$

Thus $\sigma(\mathcal{A},\mathcal{B};\mathcal{F}) \le \delta$ 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 $1 \le i \le m$, or an intersection point of an edge in V_i and an edge in V_k, for some $1 \le i \ne k \le m$. 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 $1 \le i,k \le m$. 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 $1 \le i,k \le m$. 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 [Kedem et al. 1986]. 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 $O(m^3n^2)$ vertices. □

Now, using the same argument as above, but switching the role of $\mathcal{A}$ and $\mathcal{B}$, it follows that the complexity of $V(\mathcal{B},\mathcal{A})$ is $O(n^2m)$ in $\mathbb{R}^2$ and $O(n^3m^2)$ 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 $\mathbb{R}^2$. In three dimensions, the bound is $O(m^2n^2(m+n))$.*

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 $O(m^2n)$ vertices in $\mathbb{R}^2$.

In $\mathbb{R}^2$, we use divide-and-conquer together with a plane-sweep to compute $V(\mathcal{A},\mathcal{B})$, $V(\mathcal{B},\mathcal{A})$, and their intersections. 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, in $O((m+n)mn \log mn)$ total time. $V(\mathcal{B},\mathcal{A})$ and $V(\mathcal{A},\mathcal{B}) \cap V(\mathcal{B},\mathcal{A})$ can be computed in a similar manner with the same time complexity.

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 $D \in \left\{ D_{ij}^-, D_{ij}^+ \;\middle|\; 1 \le i \le m, 1 \le j \le n \right\}$. 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 $O(m^2n^2(m+n)\log mn)$ 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 [Agarwal et al. 1994]. 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 conquer stage, which is not easy to parallelize. Instead, we use the algorithm described in Section 5 of [Agarwal et al. 1994] 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 or not in $O(\log^2 mn)$ time using $O(mn(m+n)\log mn)$ processors in $\mathbb{R}^2$, and $O(m^2n^2(m+n)\log mn)$ 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 $O(m^2n^2(m+n)\log^4 mn)$ in $\mathbb{R}^3$.*

2.2 Partial matching

Given an integer k, let $h_k(\mathcal{A}, \mathcal{B})$ be the k^{th} largest value in the set $\left\{ d(a, \mathcal{B}) \;\middle|\; a \in \mathcal{A} \right\}$; 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. This definition of the partial Hausdorff distance $H_k(\mathcal{A}, \mathcal{B})$ is the same as the one introduced in [Indyk et al. 1999], which allows $k-1$ outliers.

We briefly illustrate the two-dimensional case of computing the partial Hausdorff similarity $\sigma_k(\mathcal{A}, \mathcal{B}; \mathcal{F})$. 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 $T_1 = \left\{ t \;\middle|\; h_k(\mathcal{A}+t, \mathcal{B}) \leq \delta \right\}$. 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 $T_2 = \left\{ t \;\middle|\; h_k(\mathcal{B}, \mathcal{A}+t) \leq \delta \right\}$ 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}^-. 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 $O(m^2n^2(m+n)\log^4 mn)$ 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. As above, let $A_i = D(a_i, \rho_i)$, 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 $\delta > 0$; i.e, to decide whether there exists some translation t such that $H_U(\mathcal{A}+t, \mathcal{B}) \leq \delta$.

For any point p, we have

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

This value is greater than δ if and only if

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

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

$$T_1 = \left\{ t \;\middle|\; h_U(\mathcal{A} + t, \mathcal{B}) \leq \delta \right\};$$

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

$$T_2 = \left\{ t \;\middle|\; h_U(\mathcal{B}, \mathcal{A} + t) \leq \delta \right\},$$

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 $|Q| \leq |\Gamma| \leq 6n - 12$ [Kedem et al. 1986].

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 $d(x, \xi) \leq d(x, \xi')$, for all $\xi' \in Q \cup \Gamma$. For each $\gamma \in \Gamma$, let $W(\gamma)$ denote the circular sector spanned by γ within the disk $B_j(\delta)$ whose boundary contains γ, and let $U' = U \setminus \bigcup_{\gamma \in \Gamma} W(\gamma)$. The diagram has the following structure (see Figure 2 (a)).

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.
(c) The complexity of $\mathrm{Vor}(Q \cup \Gamma)$ is $O(n)$.

Proof. We prove claim (a), which will then imply claim (b). First, observe that for any $\gamma \in \Gamma$, $W(\gamma) \subseteq V(\gamma)$. Indeed, let $p \in W(\gamma)$ be a point whose nearest neighbor on ∂U is not a point of γ. Let $q \notin \gamma$ be a nearest neighbor of p, and let D be the disk containing γ on its boundary. Let q' be the point on ∂D closest to p, which lies on γ. Since $|qp| < |q'p|$, q lies in the interior of D, thereby contradicting the assumption that $q \in \partial U$. Hence $W(\gamma) \subseteq V(\gamma)$.

We now show that for any point $p \notin W(\gamma)$ and $p \in U$, none of its nearest neighbors in ∂U lies on γ. This will imply claim (a) above. In particular, let p_1 and p_2 denote the two endpoints of γ, o the origin of the disk D containing γ on its boundary, and $F(\gamma)$ the wedge bounded by the rays $\overrightarrow{op_1}$ and $\overrightarrow{op_2}$ and containing $W(\gamma)$; see Figure 2 (b) for an illustration. For any point q outside $F(\gamma)$, the nearest neighbor of q in $\gamma \cup \{p_1, p_2\}$ must be either p_1 or p_2. Hence $V(\gamma) \subseteq F(\gamma)$. Furthermore, for any point $q \in (F(\gamma) \setminus W(\gamma)) \cap U$, the segment connecting q to any point $p \in \gamma$ intersects ∂U in its interior. Hence $q \notin V(\gamma)$, implying that $W(\gamma) = V(\gamma)$. This proves both claims (a) and (b) above.

We now prove claim (c). First, observe that any edge in $\partial U'$ is of the form $e = bq$ where b is the center of some disk $D = B_j(\delta)$, $q \in \partial D$, and $q \in Q$. Furthermore, any point p in the interior of e has q as its *unique* nearest neighbor among all points from Q. Hence the interior of e does not intersect any Voronoi edge from $\mathrm{Vor}(Q)$. Since each vertex in $U' \cap \mathrm{Vor}(Q)$ is a vertex of $\mathrm{Vor}(Q)$, a vertex of $\partial U'$, or

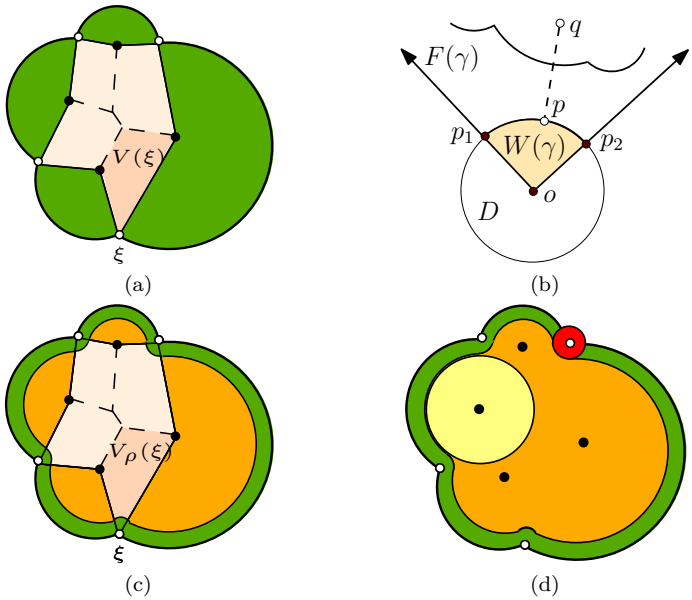


Fig. 2. (a) The Voronoi diagram of the boundary decomposes the union into 8 cells: 4 dark circular sectors and 4 light-colored polygonal regions. (b) The region $F(\gamma)$ is bounded by the rays $\overrightarrow{op_1}$ and $\overrightarrow{op_2}$. For any point $q \in F(\gamma) \setminus W(\gamma)$ and $p \in \gamma$, the interior of the segment pq must intersect ∂U (the solid curve). (c) Shrinking by ρ the Voronoi cell $V(\xi)$ of each boundary element ξ of the union. (d) The boundary of the lighter-colored disk contains a convex arc, and the boundary of the darker-colored disk contains a concave arc.

an intersection point between an edge of $\mathrm{Vor}(Q)$ and an edge from $\partial U'$, it follows that the complexity of $U' \cap \mathrm{Vor}(Q)$ is proportional to the complexity of $\partial U'$ plus that of $\mathrm{Vor}(Q)$, which is $O(n)$. Hence the complexity of $\mathrm{Vor}(Q \cup \Gamma)$ is $O(n)$.

□

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

$$V_\rho(\xi) = \left\{ x \in V(\xi) \;\middle|\; d(x, \xi) \geq \rho \right\}.$$

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(c) for an illustration. The combinatorial complexity of $T_1(A)$ is still $O(n)$.

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 $1 \le k \le n$, 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) $D(\xi - a_i, \rho_i) \cap (T_1(A_i) \setminus \partial(T_1(A_i)) = \emptyset$ for any $\xi \in Q$. See Figure 2 (d) 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. $T_1(A_i) \cap T_1(A_j)$ 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

$$\begin{aligned} &\{D(b_k - a_i, r_k + \delta - \rho_i), D(b_k - a_j, r_k + \delta - \rho_j)\}_{1 \le k \le n} \\ \cup\ &\{D(\xi - a_i, \rho_i), D(\xi - a_j, \rho_j)\}_{\xi \in Q}\,. \end{aligned}$$

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 $1 \le k \le n$. 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 [Kedem et al. 1986], 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. The computation of T_1 is by the same divide-and-conquer procedure combined with a plane sweep, as the computation of $V(\mathcal{A}, \mathcal{B})$ at the end of Section 2.1.

LEMMA 3.3. *T_1 has complexity $O(m^2 n)$, and 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 $O(mn^2 \log mn)$. We can then determine whether $T_1 \cap T_2 \neq \emptyset$, by a plane sweep, in time $O(mn(m+n)\log mn)$. Finally, the optimization problem, that is, to compute $\sigma_U(\mathcal{A}, \mathcal{B})$, can be solved by the parametric search technique [Agarwal et al. 1994], similar to the one used in Section 2.1. In order to apply the parametric search technique, we again need a parallel version of the above procedure. We use the algorithm of [Agarwal et al. 1994] (specifically, see Section 5 in [Agarwal et al. 1994]) to compute the union/intersection of two planar regions. It yields a parallel algorithm to determine whether $T_1 \cap T_2 \neq \emptyset$ in $O(\log^2 mn)$ time using $O(mn(m+n)\log mn)$ processors in $\mathbb{R}^2$. The standard technique of parametric searching then implies that $\sigma_U(\mathcal{A}, \mathcal{B})$ can be computed in $O(mn(m+n)\log^4 mn)$ time.

We compute $\sigma_U(\mathcal{A}, \mathcal{B}; \mathcal{F})$ by following the same approach as computing $\sigma_U(\mathcal{A}, \mathcal{B})$. Specifically, we need to modify the definitions of T_1 and 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

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

Symmetrically, we can redefine T_2. These new definitions of T_1 and T_2 do not change their asymptotic size complexity. In particular, by modifying the set $\mathcal{B}'_{ij}$, as introduced in the proof of Lemma 3.2, to also include the set of disks

$$\{D(b_k - a_i, r_k + \rho_i), D(b_k - a_j, r_k + \rho_j)\}_{1 \leq k \leq n},$$

the same argument holds. We thus conclude with 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 [Amenta and Kolluri 2001]. 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.

3.2.1 *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 [Aichholzer et al. 1997; Alt et al. 1995]. 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 [Alt et al. 1995] that in $\mathbb{R}^d$,

$$\sigma_U(\mathcal{A}, \mathcal{B}) \leq H_U(\mathcal{A} + \tau, \mathcal{B}) \leq (1 + \sqrt{d})\sigma_U(\mathcal{A}, \mathcal{B}),$$

and that the optimal translation lies in a disk of radius $H_U(\mathcal{A}+\tau,\mathcal{B})$ centered at τ. Computing τ takes $O(m+n)$ time. We compute $H_U(\mathcal{A}+\tau,\mathcal{B})$ using the parametric search technique [Agarwal et al. 1994], 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 $H_U(\mathcal{A}+t,\mathcal{B})\le\delta$ 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 $\mathbb{R}^3$, as it can be reduced to computing a convex hull of $m+n$ points in $\mathbb{R}^3$ and $\mathbb{R}^4$, respectively [Sharir and Agarwal 1995]. Plugging this into the parametric searching technique, the time spent in computing $H_U(\mathcal{A}+t,\mathcal{B})$ in $\mathbb{R}^2$ and $\mathbb{R}^3$ is $O((m+n)\log^3(mn))$ and $O((m^2+n^2)\log^2(mn))$, respectively.

In order to compute an ε-approximation of $\sigma_U(\mathcal{A},\mathcal{B})$ from this constant-factor approximation, we use the standard trick [Aichholzer et al. 1997] of placing a grid of cell size $\frac{\varepsilon}{1+\sqrt{d}}\cdot H_U(\mathcal{A}+\tau,\mathcal{B})$ in the disk of radius $H_U(\mathcal{A}+t,\mathcal{B})$ centered at τ, and returning the smallest $H_U(\mathcal{A}+t,\mathcal{B})$, 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 $\mathbb{R}^3$.*

3.2.2 *Pseudo-approximation for $\sigma_U(\mathcal{A},\mathcal{B};\mathcal{F})$.* Currently, we do not have an efficient ε-approximation algorithm for $\sigma_U(\mathcal{A},\mathcal{B};\mathcal{F})$ 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 $\varepsilon\ge 0$, 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 applications where data is noisy, and/or shallow penetration is allowed. For example, an atom in a protein is best modeled as a "fuzzy" ball rather than a hard ball [Halperin et al. 2002]. We can model this fuzziness by allowing an 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 $\varepsilon>0$. Hence 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 an algorithm to construct an ε-free region X of considerably smaller complexity. We also compute a placement $t^*\in X$ such that $H_U(\mathcal{A}+t^*,\mathcal{B})\le(1+\varepsilon)\sigma_U(\mathcal{A},\mathcal{B};X)$. 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. *Let $\mathcal{A}$ and $\mathcal{B}$ be two sets of balls of size m and n, respectively, in $\mathbb{R}^3$. 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 $\mathcal{D} = \left\{ D_{ij} = D(b_j - a_i, \rho_i + r_j) \;\middle|\; 1 \le i \le m, 1 \le j \le n \right\}$. We insert each ball $D_{ij} \in \mathcal{D}$ into an octree 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. 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 $\varepsilon(\rho_i + r_j)/2$. This implies that $|V| = O(mn/\varepsilon^3)$. 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 $r(\mathcal{B}), r(\mathcal{A})$, 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*

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

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

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

It follows from a result in [Alt et al. 1995] (Page 5, Section 5) that $d(r(\mathcal{A}) + \hat{t}, r(\mathcal{B})) \le \sqrt{3}\hat{\delta}$. On the other hand,

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

□

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 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; there are $O(1)$ neighboring cubes of q. 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.

We can compute $H_U(\mathcal{A}+t^*,\mathcal{B})$ in $O((n^2+m^2)\log^2 mn)$ time, as described in Section 3.2.1 to approximate $\sigma_U(\mathcal{A},\mathcal{B})$. Hence 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 3.2.1, 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*

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

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 $\{P_1,\ldots,P_\ell\}$ 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 $1\le i\le\ell$, such that $d(x,p_i)\le(1+\varepsilon)d(x,P_i)$, where $d(x,P_i)=\min_{q\in P_i}d(x,q)$. Arya and Malamatos [Arya and Malamatos 2002] proposed a data structure that can answer an ε-approximate nearest-neighbor query among a set of n points, in time $O(\log(n/\varepsilon))$ using $O((n/\varepsilon^d)\log(1/\varepsilon))$ space and preprocessing time $O((n/\varepsilon^d)\log(n/\varepsilon)\log(1/\varepsilon))$. Constructing this data structure for each P_i separately, one can answer the above query in time $O(\ell\log(N/\varepsilon))$, using $O((N/\varepsilon^d)\log(1/\varepsilon))$ space, and after a preprocessing in $O((N/\varepsilon^d)\log(N/\varepsilon)\log(1/\varepsilon))$ time. We adapt this data structure so that the query time can be improved to $O(\log(N/\varepsilon)+\ell)$. This data structure also constructs a subdivision of $\mathbb{R}^d$ of size $O((N/\varepsilon^d)\log(1/\varepsilon))$, which is an ε-approximate Voronoi diagram of each P_i. Besides being interesting in its own right, the modified data structure will be used in the subsequent subsections.

We begin by describing how we adapt the data structure by Arya and Malamatos [Arya and Malamatos 2002]. Let S be a set of n points in $\mathbb{R}^d$, let $\varepsilon>0$ be a parameter, and let $\mathcal{H}\supset S$ be a hypercube so that any point of S is at least $\operatorname{diam}(S)/\varepsilon$ away from $\partial\mathcal{H}$. Note that any point of S is an ε-approximate nearest neighbor for a point outside $\mathcal{H}$. A *quad-tree box* of $\mathcal{H}$ is a hypercube that can be obtained by recursively dividing each side of $\mathcal{H}$ into two equal parts. A useful property of quad-tree boxes (of $\mathcal{H}$) is that any two of them are either disjoint or one of them is contained in the other. Arya and Malamatos choose a set $\mathcal{B}(S)$ of $O((n/\varepsilon^d)\log(1/\varepsilon))$ quad-tree boxes of $\mathcal{H}$ that cover $\mathcal{H}$; each box $\Delta\in\mathcal{B}(S)$ is associated with a point $p_\Delta\in S$. $\mathcal{B}(S)$ has the following crucial property: For a point $x\in\mathcal{H}$, let $\Delta\in\mathcal{B}(S)$ be the smallest box containing x; then $d(x,p_\Delta)\le(1+\varepsilon)d(x,S)$. In order to find the smallest box containing x, they store $\mathcal{B}(S)$ in a BBD-tree, proposed by Arya *et al.* [Arya et al. 1998]. Instead, we store them in a *compressed quad tree* (see e.g. [Har-Peled 2005]), as follows.

We first construct a quad tree $\mathcal{Q}$ on $\mathcal{B}(S)$. Let H_v be the hypercube associated with the node v of $\mathcal{Q}$; the root is associated with $\mathcal{H}$. A box $\Delta \in \mathcal{B}(S)$ is stored at a node $v \in \mathcal{Q}$ if $H_v = \Delta$. If an interior node $v \in \mathcal{Q}$ does not store a box of $\mathcal{B}(S)$ and if the degree of both v and $p(v)$, the parent of v, is one, we compress v, i.e., we delete v and the child of v becomes the child of $p(v)$. We repeat this step until there is no such node in $\mathcal{Q}$. The size of $\mathcal{Q}$ is $O(|\mathcal{B}(S)|)$. For a node $v \in \mathcal{Q}$, if a box in $\mathcal{B}(S)$ contains H_v, then we set $\varphi(v) := p_\Delta$ where Δ is the smallest such box; if there is no such box, then $\varphi(v)$ is undefined. Note that if $\varphi(v)$ is defined, then Δ is associated with the lowest ancestor of v that stores some box from $\mathcal{B}(S)$. Hence we can set up $\varphi(v)$ for every node v in $\mathcal{Q}$ by traversing $\mathcal{Q}$ once in a top-down manner. The compressed quadtree $\mathcal{Q}$ can be constructed in time $O(|\mathcal{B}(S)| \log |\mathcal{B}(S)|) = O((n/\varepsilon^d) \log(1/\varepsilon) \log(n/\varepsilon))$ [Har-Peled 2005].

We call a node $v \in \mathcal{Q}$ *exposed* if its degree is at most one. We associate a region R_v with each exposed node v. If v is a leaf, then $R_v = H_v$. Otherwise, v has one child w, and $R_v = H_v \setminus H_w$. For a point $x \in R_v$, v is the lowest node in $\mathcal{Q}$ such that $x \in H_v$. The regions R_v form a partition of $\mathcal{H}$, which is also an ε-approximate Voronoi diagram of S. Indeed, for every exposed node v, $\varphi(v)$ is defined because R_v is not covered by the boxes of $\mathcal{B}(S)$ stored at the children of v and thus it is covered by a box stored at v or one of its ancestors. Now by construction,

$$d(x, S) \le d(x, \varphi(v)) \le (1 + \varepsilon) d(x, S) \qquad \forall x \in R_v.$$

The depth of $\mathcal{Q}$ is linear in the worst case. In order to quickly find the node v for which R_v contains a query point x, we construct another tree $\mathcal{T}$, of depth $O(\log(n/\varepsilon))$, on the nodes of $\mathcal{Q}$, as follows. We identify in $O(|\mathcal{Q}|)$ time a node ξ in $\mathcal{Q}$ so that the removal of ξ decomposes $\mathcal{Q}$ into at most $u \le 2^d + 1$ connected components, each of size at most $|\mathcal{Q}|/2$. Let $\mathcal{Q}_0(\xi)$ be the subtree that contains the ancestors of ξ, and let $\mathcal{Q}_1(\xi), \cdots, \mathcal{Q}_{u-1}(\xi)$ be the subtrees of $\mathcal{Q}$ rooted at the children of ξ. We recursively construct a tree $\mathcal{T}_i(\xi)$ on each $\mathcal{Q}_i(\xi)$, for $0 \le i < u$, and attach it as a subtree of ξ. $\mathcal{T}$ can be constructed in time $O(|\mathcal{Q}| \log |\mathcal{Q}|) = O((n/\varepsilon^d) \log(1/\varepsilon) \log(n/\varepsilon))$. Given a point $x \in \mathcal{H}$, we visit a path in $\mathcal{T}$, starting from its root. Suppose, we are at a node ξ. If ξ is exposed and $x \in R_\xi$, we return $\varphi(\xi)$ and stop. If $x \notin H_\xi$, we recursively visit the subtree $\mathcal{T}_0(\xi)$; otherwise, we visit the child η so that $x \in H_\eta$. The query time is proportional to the depth of $\mathcal{T}$, which is $O(\log(n/\varepsilon))$.

We now return to the problem of computing ε-approximate nearest neighbors of $P_1, \ldots, P_\ell$. Let C be a hypercube containing $P = \bigcup_{i=1}^{\ell} P_i$ so that any point of P is at least $\operatorname{diam}(P)/\varepsilon$ away from ∂C. For each $1 \le i \le \ell$, we first construct the family $\mathcal{B}(P_i)$ of quad-tree boxes of C, using the algorithm by Arya and Malamatos [Arya and Malamatos 2002], and then construct the trees $\mathcal{Q}$ and $\mathcal{T}$ on $\bigcup_i \mathcal{B}(P_i)$, as described above. For a node $v \in \mathcal{Q}$, let $\varphi_i(v) = p_{\Delta_i}$, where Δ_i is the smallest box of $\mathcal{B}(P_i)$ that contains H_v; if there is no such box, then $\varphi_i(v)$ is undefined. We define exposed nodes and the regions R_v as above. By definition, for any point $x \in R_v$ and for every $1 \le i \le \ell$,

$$d(x, P_i) \le d(x, \varphi_i(v)) \le (1 + \varepsilon) d(x, P_i).$$

The regions R_v form a subdivision of C, which is an ε-approximate Voronoi diagram of each P_i.

For a node v, let $\Phi(v)$ be a list of ℓ items whose ith item is $\varphi_i(v)$ if it is defined and NULL otherwise. If we store $\Phi(v)$ explicitly at each node v of $\mathcal{Q}$, then the size of the data structure will be $O((\ell N/\varepsilon^d)\log(1/\varepsilon))$. We therefore store $\Phi(v)$ in an implicit manner, using persistence [Sarnak and Tarjan 1986]. The total space used is $O((N/\varepsilon^d)\log(1/\varepsilon))$, and $\Phi(v)$, for a node v, can be reported in $O(\ell)$ time.

Specifically, first note that $\varphi_i(v) \neq \varphi_i(p(v))$ if a box $\Delta \in \mathcal{B}(P_i)$ is stored at v, i.e., $H_v = \Delta \in \mathcal{B}(P_i)$, and $\varphi_i(v) = p_\Delta$ in this case. Let $J(v) = \left\{ i \;\middle|\; H_v \in \mathcal{B}(P_i) \right\}$. Then $\Phi(v)$ can be constructed from $\Phi(p(v))$ by updating $\varphi_i(v)$ for $i \in J(v)$. We therefore perform an in-order traversal of $\mathcal{Q}$ and maintain the lists $\Phi(v)$ in a linked list Ψ, using persistence, so that $\Phi(u)$ for the nodes u that have been visited so far can be retrieved in $O(\ell)$ time. If we are currently visiting a node w, then the "current version" of Ψ contains $\Phi(w)$. Moreover, we maintain an array that stores pointers to each item in the current version of Ψ so that the ith item, for any $i \leq \ell$, can be accessed in $O(1)$ time. When we arrive at a node $v \in \mathcal{Q}$ for the first time, for each $i \in J(v)$, we do the following. Let $\Delta \in \mathcal{B}(P_i)$ be a box stored at v. Then we set $\varphi_i(v) = p_\Delta$ and update the ith item of Ψ. We store at v a pointer $\pi(v)$ to the head of the current version of the list Ψ. We also store the values of $\varphi_i(p(v))$, for $i \in J(v)$, in a stack at v. After we have processed the subtree rooted at v, the current version of Ψ contains $\Phi(v)$. We then delete the values stored in the stack at v and restore them in Ψ so that its current version contains $\Phi(p(v))$. Hence, we perform $2|J(v)|$ updates on Ψ while processing v. Sarnak and Tarjan [Sarnak and Tarjan 1986] showed that the amortized time and space of performing an update in a persistent linked list is $O(1)$, therefore the total time spent to build such a persistent data structure is $\sum_{v\in\mathcal{Q}} O(|J(v)|) = O((N/\varepsilon^d)\log(1/\varepsilon))$.

Let q be a query point. Following the procedure described above, we first find in $O(\log(N/\varepsilon))$ time the node $\xi \in \mathcal{T}$ such that $q \in R_\xi$. Next, following the pointer $\pi(\xi)$, we access the version of Ψ that stores $\Phi(\xi)$ and report $\varphi_i(\xi)$ for all $1 \leq i \leq \ell$ in $O(\ell)$ time. Putting everything together, we conclude the following.

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

4.2 Approximating $\sigma_R(\mathcal{A}, \mathcal{B})$

For $1 \leq i \leq m$, let $P_i = \mathcal{B} - a_i = \left\{ b_j - a_i \;\middle|\; 1 \leq j \leq n \right\}$, and for $m < i \leq m+n$, let $P_i = b_{i-m} - \mathcal{A} = \left\{ b_{i-m} - a_j \;\middle|\; 1 \leq j \leq m \right\}$. We construct the compressed quadtree $\mathcal{Q}$ for $P_1, \ldots, P_{m+n}$, with the given parameter ε; $|\mathcal{Q}| = O((mn/\varepsilon^d)\log(1/\varepsilon))$. Define

$$f_i(t) = d^2(t, P_i) = \begin{cases} \min\limits_{1\leq j\leq n} d^2(t, b_j - a_i), & 1 \leq i \leq m, \\ \min\limits_{1\leq j\leq m} d^2(t, b_{i-m} - a_j), & m < i \leq m+n. \end{cases}$$

Let

$$F_{\mathcal{A}}(t) \;=\; h_R^2(\mathcal{A}+t,\mathcal{B}) = \frac{1}{m}\sum_{i=1}^{m} d^2(a_i+t,\mathcal{B}) = \frac{1}{m}\sum_{i=1}^{m} f_i(t)$$

$$F_{\mathcal{B}}(t) \;=\; h_R^2(\mathcal{B},\mathcal{A}+t) = \frac{1}{n}\sum_{i=1}^{n} d^2(\mathcal{A}+t,b_i) = \frac{1}{n}\sum_{i=m+1}^{m+n} f_i(t).$$

For each exposed node $v \in \mathcal{Q}$, $\varphi_i(v)$ is defined for all $1 \le i \le m+n$. Let

$$\widehat{F}_{\mathcal{A},v}(t) = \frac{1}{m}\sum_{i=1}^{m} d^2(t,\varphi_i(v)) \quad \text{and} \quad \widehat{F}_{\mathcal{B},v}(t) = \frac{1}{n}\sum_{i=1}^{n} d^2(t,\varphi_{m+i}(v)).$$

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

$$F_{\mathcal{A}}(t) \;\le\; \widehat{F}_{\mathcal{A},v}(t) = \frac{1}{m}\sum_{i=1}^{m} d^2(t,\varphi_i(v)) \le \frac{1}{m}\sum_{i=1}^{m}(1+\varepsilon)^2\cdot d^2(t,P_i) \le (1+\varepsilon)^2 F_{\mathcal{A}}(t),$$

implying that

$$\sqrt{\widehat{F}_{\mathcal{A},v}(t)} \le (1+\varepsilon)h_R(\mathcal{A}+t,\mathcal{B}).$$

Similarly, $\sqrt{\widehat{F}_{\mathcal{B},v}(t)} \le (1+\varepsilon)h_R(\mathcal{B},\mathcal{A}+t)$. Hence, it suffices to store $\widehat{F}_{\mathcal{A},v}(t), \widehat{F}_{\mathcal{B},v}(t)$ at each exposed node $v \in \mathcal{Q}$. Since they are quadratic functions in $t \in \mathbb{R}^d$, they can be stored using $O(1)$ space (where the constant depends on d) and updated in $O(1)$ time for each change in $\varphi_i(v)$.

If we compute $\widehat{F}_{\mathcal{A},v}$ for each exposed node $v \in \mathcal{Q}$ independently, then the total time spent is $O\big((m^2n/\varepsilon^d)\log(1/\varepsilon)\big)$. We therefore proceed as in the previous section. We perform an in-order traversal of $\mathcal{Q}$. For each node $v \in \mathcal{Q}$, let $\Phi^*(v) = \Big\{ i \;\Big|\; \varphi_i(v) \text{ is defined} \Big\}$, and let

$$\widehat{F}_{\mathcal{A},v}(t) = \frac{1}{m}\sum_{i\in\Phi^*(v), i\le m} d^2(t,\varphi_i(v)) \quad \text{and} \quad \widehat{F}_{\mathcal{B},v}(t) = \frac{1}{n}\sum_{i\in\Phi^*(v), i> m} d^2(t,\varphi_i(v)).$$

If v is an exposed node then $\Phi^*(v) = \{1,2,\ldots,..m+n\}$, therefore the above definition of $\widehat{F}_{\mathcal{A},v}, \widehat{F}_{\mathcal{B},v}$ is consistent with that for exposed nodes defined earlier. Let $J^*(v) = \Big\{ i \;\Big|\; H_v \in \mathcal{B}(P_i) \Big\}$. Recall that $\varphi_i(v) \ne \varphi_i(p(v))$ for $i \in J^*(v)$. Then, following the same idea as in the previous subsection, $\widehat{F}_{\mathcal{A},v}$ can be reconstructed from $\widehat{F}_{\mathcal{A},p(v)}$ by updating based on $J^*(v)$. Specifically,

$$\widehat{F}_{\mathcal{A},v}(t) \;=\; \widehat{F}_{A,p(v)}(t) + \frac{1}{m}\sum_{i\in J^*(v), i\le m} [d^2(t,\varphi_i(v)) - d^2(t,\varphi_i(p(v)))],$$

$$\widehat{F}_{\mathcal{B},v}(t) \;=\; \widehat{F}_{B,p(v)}(t) + \frac{1}{n}\sum_{i\in J^*(v), i> m} [d^2(t,\varphi_i(v)) - d^2(t,\varphi_i(p(v)))].$$

Hence, $\widehat{F}_{\mathcal{A},v}, \widehat{F}_{\mathcal{B},v}$ can be computed in $O(|J^*(v)|)$ time. As such, the total time spent in the traversal and in computing $\widehat{F}_{\mathcal{A},v}, \widehat{F}_{\mathcal{B},v}$ for all nodes in $\mathcal{Q}$ is $O((n/\varepsilon^d)\log(1/\varepsilon))$.

Finally, for each exposed node $v \in \mathcal{Q}$, we compute

$$t_v = \arg\min_{t \in R_v} \max\left\{\sqrt{\widehat{F}_{A,v}(t)}, \sqrt{\widehat{F}_{B,v}(t)}\right\}$$

in constant time (as $\widehat{F}_{\mathcal{A},v}$, and $\widehat{F}_{\mathcal{B},v}$ are quadratic functions), and return

$$\min_v H_R(\mathcal{A} + t_v, \mathcal{B}) \leq (1+\varepsilon)\sigma_R(\mathcal{A}, \mathcal{B})$$

where the minimum is taken over all exposed nodes v of $\mathcal{Q}$.

If we wish to compute an ε-approximate value of $H_R(\mathcal{A} + t, \mathcal{B})$ for any query $t \in \mathbb{R}^d$, we construct the tree $\mathcal{T}$ as described in the previous section. Instead of storing the lists $\Phi(\cdot)$ at each node, we now store $\widehat{F}_{\mathcal{A},v}, \widehat{F}_{\mathcal{B},v}$ at each node $v \in \mathcal{T}$. The total storage needed is $O((mn/\varepsilon^d)\log(1/\varepsilon))$. 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 $O((mn/\varepsilon^d)\log(mn/\varepsilon)\log(1/\varepsilon))$ time, so that

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

ii. construct, in time $O((mn/\varepsilon^d)\log(mn/\varepsilon)\log(1/\varepsilon))$, a data structure of size $O((mn/\varepsilon^d)\log(1/\varepsilon))$, 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 $\sigma_S(\mathcal{A}, \mathcal{B})$

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

$$G_{\mathcal{A}}(t) = \frac{1}{m}\sum_{i=1}^{m} d(t, P_i) = h_S(\mathcal{A} + t, \mathcal{B}),$$

$$G_{\mathcal{B}}(t) = \frac{1}{n}\sum_{j=1}^{n} d(t, P_{m+j}) = h_S(\mathcal{B}, \mathcal{A} + t).$$

For each exposed node $v \in \mathcal{Q}$, let

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

$$\widehat{G}_{\mathcal{B},v}(t) = \frac{1}{n}\sum_{j=1}^{n} d(t, \varphi_{m+j}(v)) \leq (1+\varepsilon)h_S(\mathcal{B}, \mathcal{A} + t)$$

$$t_v = \arg\min_{t \in R_v} \max\left\{\widehat{G}_{\mathcal{A},v}(t), \ \ \widehat{G}_{\mathcal{B},v}(t)\right\}.$$

Since $\widehat{G}_{\mathcal{A},v}$ and $\widehat{G}_{\mathcal{B},v}$ 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 $\widehat{G}_{\mathcal{A},v}$ (resp., $\widehat{G}_{\mathcal{B},v}$) that is easier to handle. More precisely, for a given set P of points in $\mathbb{R}^d$, define the 1-median function

$$\mathrm{med}_P(t) = \frac{1}{|P|}\sum_{p \in P} d(p, t).$$

For any $v \in \mathcal{Q}$, let $\Phi^*(v) = \left\{ i \;\middle|\; \varphi_i(v) \text{ is defined} \right\}$ as before. Let $\Phi_{\mathcal{A}}(v) = \left\{ \varphi_i(v) \;\middle|\; i \le m, i \in \Phi^*(v) \right\}$ and $\Phi_{\mathcal{B}}(v) = \left\{ \varphi_i(v) \;\middle|\; i > m, i \in \Phi^*(v) \right\}$. Then we have that $\widehat{G}_{\mathcal{A},v}(t) = \mathrm{med}_{\Phi_{\mathcal{A}}(v)}(t)$, and $\widehat{G}_{\mathcal{B},v}(t) = \mathrm{med}_{\Phi_{\mathcal{B}}(v)}(t)$. 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 defined by a set of $O((1/\varepsilon^d)\log(1/\varepsilon))$ weighted points. This approximation can be maintained in $O((1/\varepsilon^d)\log((\log n)/\varepsilon)\log^{d+1} n)$ time as a point is inserted or deleted from P. Furthermore, given two point sets P and Q in $\mathbb{R}^d$, this can be used to maintain, within the same time bound, an ε-approximation of $\arg\min_t \max\{\mathrm{med}_P(t), \mathrm{med}_Q(t)\}$.

Using this dynamic data structure that will be introduced shortly in Section 4.4, we can traverse all cells of $\mathcal{Q}$, as in Section 4.2, and compute an $(\varepsilon/3)$-approximation of $\widehat{G}_{\mathcal{A},v}$ and $\widehat{G}_{\mathcal{B},v}$ (thus an ε-approximation of $G_{\mathcal{A}}$ and $G_{\mathcal{B}}$) for each node $v \in \mathcal{Q}$. However, we now spend

$$O(|J^*(v)| \cdot (1/\varepsilon^d)\,\mathrm{polylog}(mn, 1/\varepsilon))$$

time, instead of spending $O(|J^*(v)|)$ time as in Section 4.2, to compute an $(\varepsilon/3)$-approximation of $\widehat{G}_{\mathcal{A},v}$ from that of $\widehat{G}_{\mathcal{A},p(v)}$. 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 $O((mn/\varepsilon^{2d})\,\mathrm{polylog}(mn, 1/\varepsilon))$ time, so that

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

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

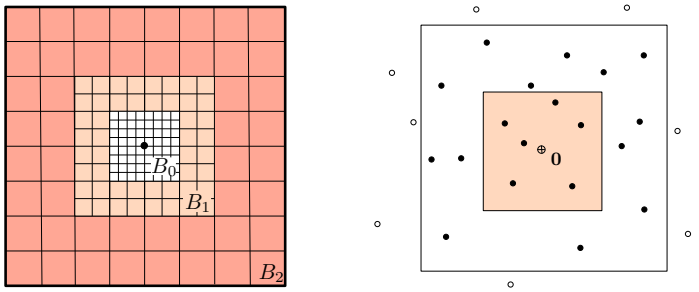
4.4 Maintaining the 1-median function

Let P be a set of n points in $\mathbb{R}^d$. In this subsection, we describe an algorithm for maintaining an ε-approximation of the 1-median of a point set P as points are inserted into or deleted from P. Using the ideas in [Agarwal et al. 2004; Har-Peled and Mazumdar 2004], we construct a *coreset*, a weighted subset of P, whose 1-median approximates that of P, and argue that it can be updated efficiently as the set P changes.

Let (P, ω) be a *weighted* point set in $\mathbb{R}^d$ with the weight function $\omega : P \to \mathbb{R}^+$. For a subset $A \subseteq P$, let $\omega(A) = \sum_{p \in A} \omega(p)$. If the weight function is not important or obvious from the context, we will use P to denote (P, ω). For a point $x \in \mathbb{R}^d$, we define $\nu((P, \omega), x) = \sum_{p \in P} \omega(p)\,\|px\|$ as the *price* of the 1-median placed at x. Furthermore, let $\nu_{\mathrm{opt}}((P, \omega)) = \min_{x \in \mathbb{R}^d} \nu((P, \omega), x)$ denote the price of the *optimal* 1-*median* for (P, ω).

Definition 4.4 *Coreset.* Let (P, ω) be a weighted point set in $\mathbb{R}^d$. A weighted set $(\mathcal{S}, \chi)$ with $\mathcal{S} \subseteq P$ is an *ε-coreset* of (P, ω) for 1-median if

$$(1-\varepsilon)\nu((P, \omega), x) \le \nu((\mathcal{S}, \chi), x) \le (1+\varepsilon)\nu((P, \omega), x) \qquad \forall x \in \mathbb{R}^d. \tag{1}$$



(a) (b)

Fig. 3. (a) An exponential grid with 3 layers. (b) The larger (resp., smaller) box is $\mathcal{H}$ (resp., $\mathcal{G}$), and the set of hollow circles is P_O.

The proof of the following lemma describes an algorithm for computing a small ε-coreset for 1-median.

LEMMA 4.5. *Let (P,ω) be a weighted set of n points in $\mathbb{R}^d$, and let $0<\varepsilon\le 1/2$ be a parameter. An ε-coreset $(\mathcal{S},\chi)$ for (P,ω) of size $O((1/\varepsilon^d)\log(1/\varepsilon))$ for 1-median can be computed in time $O(n\log(1/\varepsilon)+(1/\varepsilon^d)\log(1/\varepsilon))$.*

PROOF. Let $\mathbf{0}\in\mathbb{R}^d$ be the point realizing $\nu_{\text{opt}}(P)$. Set $r=\nu_{\text{opt}}(P)/\omega(P)$, $M=\lceil\log_2(c_1 d/\varepsilon^2)\rceil$, where $c_1\ge 40$ is a constant, and $\ell=2^M r=c_1 dr/\varepsilon^2$. Let $\mathcal{C}(x,\rho)$ be the axis-parallel hypercube of side length ρ centered at x. Let $B_0=\mathcal{C}(\mathbf{0},r)$, and let $B_i=\mathcal{C}\big(\mathbf{0},2^i r\big)\setminus\mathcal{C}\big(\mathbf{0},2^{i-1}r\big)$ for $1\le i\le M$. Next, we partition each B_i into $O(1/\varepsilon^d)$ hypercubes of side length $\varepsilon 2^i r/(c_2 d)$ by drawing a uniform grid, where $c_2>10$ is another constant. This forms an exponential grid G that covers the hypercube $\mathcal{H}=\mathcal{C}(\mathbf{0},\ell)$; see Figure 3. Set $P_I=P\cap\mathcal{H}$ and $P_O=P\setminus P_I$.

For every grid cell $\square\in G$, we pick a "representative" point $p\in P\cap\square$, add it to $\mathcal{S}$ and set its weight $\chi(p)=\omega(P\cap\square)$. Finally, let $u\in P$ be the point farthest away from $\mathbf{0}$. We add u to $\mathcal{S}$ and set its weight $\chi(u)=\nu(P_O,\mathbf{0})\,/\,\|u\mathbf{0}\|$. By construction, $|\mathcal{S}|=O((1/\varepsilon^d)\log(1/\varepsilon))$. We claim that $(\mathcal{S},\chi)$ is a ε-coreset of (P,ω) for 1-median, i.e., it satisfies Eq. (1).

Let $\mathcal{S}_I=\mathcal{S}\cap\mathcal{H}$ and $\mathcal{S}_O=\mathcal{S}\setminus\mathcal{S}_I=\{u\}$. First, for any $x\in\mathbb{R}^d$, we claim that

$$|\nu(P_I,x)-\nu(\mathcal{S}_I,x)|\le(\varepsilon/4)\nu(P,\mathbf{0})=(\varepsilon/4)\nu_{\text{opt}}(P). \tag{2}$$

Indeed, every point in a cell of $P\cap\mathcal{H}$ can be interpreted as "traveling" to its representative point in that cell. The total (weighted) distance traveled by all the points inside $\mathcal{H}$ to the coreset is smaller than $(\varepsilon/4)\nu_{\text{opt}}(P)$. As such, the error contributed by the points of P_I is smaller than $(\varepsilon/4)\nu_{\text{opt}}(P)$. This implies that we only need to consider the error contributed by points in P_O.

In particular, we need to bound the difference in the price of $\nu(P_O,x)$ and $\nu(\mathcal{S}_O,x)$. (Of course, if $P_O=\emptyset$, the lemma trivially holds and the following is unnecessary.) Let $\mathcal{G}=\mathcal{C}(\mathbf{0},c_1 r/4\varepsilon)\subseteq\mathcal{H}$; see Figure 3. We consider two cases.

(i) $x\in\mathcal{G}$: By the triangle inequality, for any $x,y\in\mathbb{R}^d$,

$$\|y\mathbf{0}\|-\|x\mathbf{0}\|\le\|yx\|\le\|y\mathbf{0}\|+\|x\mathbf{0}\|\,.$$

Therefore,

$$\nu(P_O, \mathbf{0}) - \omega(P_O) \|x\mathbf{0}\| \leq \nu(P_O, x) \leq \nu(P_O, \mathbf{0}) + \omega(P_O) \|x\mathbf{0}\| . \tag{3}$$

Since $\ell \geq c_1 dr/\varepsilon^2$ and the side length of $\mathcal{G}$ is at most $c_1 r/4\varepsilon$, $\|p\mathbf{0}\| \geq 4 \|x\mathbf{0}\| /\varepsilon$, for all $p \in P_O$, and thus Eq. (3) becomes

$$(1 - \varepsilon/4)\nu(P_O, \mathbf{0}) \leq \nu(P_O, x) \leq (1 + \varepsilon/4)\nu(P_O, \mathbf{0}) . \tag{4}$$

Similarly, we have

$$(1 - \varepsilon/4)\nu(P_O, \mathbf{0}) \leq \nu(\mathcal{S}_O, x) \leq (1 + \varepsilon/4)\nu(P_O, \mathbf{0}) . \tag{5}$$

Putting Eq. (4) and Eq. (5) together, we have

$$|\nu(P_O, x) - \nu(\mathcal{S}_O, x)| \leq (\varepsilon/2)\nu(P_O, \mathbf{0}) . \tag{6}$$

Using Eq. (2) and Eq. (6), we conclude

$$\begin{aligned} |\nu(P, x) - \nu(\mathcal{S}, x)| &\leq |\nu(P_I, x) - \nu(\mathcal{S}_I, x)| + |\nu(P_O, x) - \nu(\mathcal{S}_O, x)| \\ &\leq (\varepsilon/4)\nu(P, \mathbf{0}) + (\varepsilon/2)\nu(P_O, \mathbf{0}) \\ &\leq \varepsilon\nu(P, \mathbf{0}) \leq \varepsilon\nu(P, x) . \end{aligned}$$

(ii) $x \in \mathbb{R}^d \setminus \mathcal{G}$: We first claim that

$$\omega(P_O) \leq \varepsilon^2 \omega(P)/c_1. \tag{7}$$

Indeed,

$$\nu_{\text{opt}}(P) = \sum_{p \in P} \omega(p) \|p\mathbf{0}\| \geq \sum_{p \in P_O} \omega(p)\ell \geq \omega(P_O) \cdot \frac{c_1 d \nu_{\text{opt}}(P)}{\varepsilon^2 \omega(P)},$$

which implies the claim. Next,

$$\omega(P) \|x\mathbf{0}\| \geq \omega(P)\frac{c_1 r}{4\varepsilon} \geq c_1 \nu_{\text{opt}}(P)/4\varepsilon \tag{8}$$

Using Eq. (7) and Eq. (8), we obtain

$$\begin{aligned} \nu(P_I, x) &\geq \omega(P_I) \|x\mathbf{0}\| - \nu(P_I, \mathbf{0}) \\ &\geq \left(1 - \varepsilon^2/c_1\right) \omega(P) \|x\mathbf{0}\| - \nu_{\text{opt}}(P) \\ &\geq \left(1 - \varepsilon^2/c_1\right) \omega(P) \|x\mathbf{0}\| - (4\varepsilon/c_1)\omega(P) \|x\mathbf{0}\| \\ &\geq (1 - \varepsilon/8)\omega(P) \|x\mathbf{0}\| . \end{aligned} \tag{9}$$

The last inequality follows since $c_1 \geq 40$ and $\varepsilon \leq 1/2$. A similar argument shows that

$$\begin{aligned} \nu(P_O, x) &\leq \omega(P_O) \|x\mathbf{0}\| + \nu(P_O, \mathbf{0}) \\ &\leq \frac{\varepsilon^2}{c_1}\omega(P) \|x\mathbf{0}\| + \nu_{\text{opt}}(P) \\ &\leq \frac{\varepsilon}{8}\omega(P) \|x\mathbf{0}\| \qquad \text{(using Eq. (8))}. \end{aligned} \tag{10}$$

Plugging Eq. (9) into Eq. (10), we obtain

$$\nu(P_O, x) \leq \frac{\varepsilon}{8}\frac{\nu(P_I, x)}{1 - \varepsilon/8} \leq \frac{\varepsilon}{4}\nu(P, x) .$$

Similarly, we can argue that

$$\nu(\mathcal{S}_O, x) \leq (\varepsilon/4)\nu(P, x)\,. \tag{11}$$

Thus, using Eq. (2), Eq. (10), Eq. (11), we obtain

$$\begin{aligned} |\nu(P,x) - \nu(\mathcal{S},x)| &\leq |\nu(P_I,x) - \nu(\mathcal{S}_I,x)| + |\nu(P_O,x) - \nu(\mathcal{S}_O,x)| \\ &\leq (\varepsilon/4)\nu(P,x) + (\varepsilon/2)\nu(P,x) \\ &\leq \varepsilon\nu(P,x)\,. \end{aligned}$$

This completes the proof that $(\mathcal{S}, \chi)$ is an ε-coreset of (P, ω). The above argument works even if $\nu(P, \mathbf{0}) \leq c\nu_{\text{opt}}(P)$ for some constant $c \geq 1$. Specifically, in the beginning of the above proof, we need to set $r = \nu(P_O, \mathbf{0})/\omega(P)$, $M = \lceil \log_2(cc_1 d/\varepsilon^2) \rceil$, and the grid size in each B_i to be $\varepsilon 2^i r/(cc_2 d)$. The remaining argument is the same.

In order to compute $\mathcal{S}$, we first compute in $O(n)$ time the centroid $\bar{\mathbf{c}} \in \mathbb{R}^d$. It is well known that $\nu(P, \bar{\mathbf{c}}) \leq 2\nu_{\text{opt}}(P)$. We then compute the B_i's and the exponential grid in time $O((1/\varepsilon^d)\log(1/\varepsilon))$ and find the grid cell that contains each point of P in a total time of $O(n \log(1/\varepsilon))$. Hence, the total time spent in computing $\mathcal{S}$ is $O(n\log(1/\varepsilon) + (1/\varepsilon^d)\log(1/\varepsilon))$. □

Let (P_1, ω) and (P_2, ω) be two weighted point sets so that $P_1 \cap P_2 = \emptyset$, and let $(\mathcal{S}_i, \chi)$ be an ε-coreset of (P_i, ω) for 1-median. Then $(\mathcal{S}_1 \cup \mathcal{S}_2, \chi)$ is a ε-coreset of $(P_1 \cup P_2, \omega)$. Moreover, if $(\mathcal{S}_2, \chi_2)$ is an ε_2-coreset of $(\mathcal{S}_1, \chi_1)$, and $(\mathcal{S}_1, \chi_1)$ is an ε_1-coreset of (P, ω), then $(\mathcal{S}_2, \chi_2)$ is a $2(\varepsilon_1 + \varepsilon_2)$-coreset of (P, ω). Using these observations and plugging Lemma 4.5 into the dynamic data structure by Agarwal et al. [Agarwal et al. 2004], one can maintain an ε-coreset of (P, ω) of size $O((1/\varepsilon^d)\log(1/\varepsilon))$ for 1-median of (P, ω) efficiently. Omitting all the details, we conclude the following.

THEOREM 4.6. *Let P be a set of n points in $\mathbb{R}^d$, and let $\varepsilon > 0$ be a parameter. One can maintain an ε-coreset of P of size $O((1/\varepsilon^d)\log(1/\varepsilon))$ under insertions and deletions of points in P so that each update takes $O((1/\varepsilon^d)\log(\log(n)/\varepsilon)\log^{d+1} n)$ time.*

4.5 A randomized algorithm

We briefly describe below a simple randomized algorithm to approximate $\sigma_R(\mathcal{A}, \mathcal{B})$. The algorithm for 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 4.7. *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 $1/2$. Similarly, $d^2(a_k + t^*, \mathcal{B}) \leq 2\sigma_S^2(\mathcal{A}, \mathcal{B})$ with probability greater than $1/2$.*

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

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

The first half of the claim now follows immediately from Markov's inequality. The same argument also holds for the second half of the claim. □

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 4.7 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 [Arya and Malamatos 2002]. 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. Finally, by using the same algorithm, but defining $\delta_j = H_S(\mathcal{A} + t_j, \mathcal{B})$, we can obtain the same result for $\sigma_S(\mathcal{A}, \mathcal{B})$. We thus conclude with the following result.

THEOREM 4.8. *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 $O((mn/\varepsilon^d)\log mn)$ randomized expected time, two translation vectors t_1 and t_2, such that, with probability greater than $1/2$,*

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

5. CONCLUSIONS

In this paper we studied 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 that we envisage combines our randomized approach from Section 4.5, a framework to convert the original problem to a pattern matching problem [Indyk et al. 1999], and a result by Amir *et al.* on string matching [Amir et al. 2001]. 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.

REFERENCES

AGARWAL, P. K., HAR-PELED, S., AND VARADARAJAN, K. R. 2004. Approximating extent measures of points. *J. Assoc. Comput. Mach. 51,* 4, 606–635.

AGARWAL, P. K. AND MATOUŠEK, J. 1993. Ray shooting and parametric search. *SIAM J. Comput. 22*, 540–570.

AGARWAL, P. K., SHARIR, M., AND TOLEDO, S. 1994. Applications of parametric searching in geometric optimization. *J. Algorithms 17*, 292–318.

AICHHOLZER, O., ALT, H., AND ROTE, G. 1997. Matching shapes with a reference point. *Internat. J. Comput. Geom. Appl. 7*, 349–363.

ALT, H., BEHRENDS, B., AND BLÖMER, J. 1995. Approximate matching of polygonal shapes. *Ann. Math. Artif. Intell. 13*, 251–266.

ALT, H., BRASS, P., GODAU, M., KNAUER, C., AND WENK, C. 2003. Computing the Hausdorff distance of geometric patterns and shapes. In *Discrete and Computational Geometry — The Goodman-Pollack Festchrift*, B. Aronov, S. Basu, J. Pach, and M. Sharir, Eds. Springer-Verlag, Heidelberg, 65–76.

Alt, H. and Guibas, L. J. 2000. 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, 121–153.

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

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

Arya, S. and Malamatos, T. 2002. Linear-size approximate Voronoi diagrams. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*. 147–155.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. Assoc. Comput. Mach. 45,* 6, 891–923.

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

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

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

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

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

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

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

Har-Peled, S. 2005. Class notes on compressed quadtrees. Comp. Sci. Dept, Univ. of Illinois, http://valis.cs.uiuc.edu/~sariel/teach/2004/a_aprx/lec/.

Har-Peled, S. and Mazumdar, S. 2004. Coresets for k-means and k-median clustering and their applications. In *Proc. 36th Annu. ACM Sympos. Theory Comput.* 291–300.

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

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

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

Megiddo, N. 1983. Applying parallel computation algorithms in the design of serial algorithms. *J. Assoc. Comput. Mach. 30,* 4, 852–865.

Sarnak, N. and Tarjan, R. E. 1986. Planar point location using persistent search trees. *Commun. ACM 29,* 7 (July), 669–679.

Seeger, S. and Laboureux, X. 2002. Feature extraction and registration: An overview. In *Principles of 3D Image Analysis and Synthesis*. Kluwer Academic Publishers, 153–166.

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

...