Geometric Pattern Matching in d-Dimensional Space*

L. Paul Chew[1] Dorit Dor[2] Alon Efrat[2] Klara Kedem[3]

[1]Department of Computer Science, Cornell University, Ithaca, NY 14853, USA
[2]School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel
[3]Department of Math and CS, Ben Gurion University, Beer-Sheva 84105, Israel

Abstract

We show that, using the L_∞ metric, the minimum Hausdorff distance under translation between two point sets of cardinality n in d-dimensional space can be computed in time $O(n^{(4d-2)/3}\log^2 n)$ for $3 < d \leq 8$, and in time $O(n^{5d/4}\log^2 n)$ for any $d > 8$. Thus we improve the previous time bound of $O(n^{2d-2}\log^2 n)$ due to Chew and Kedem. For $d = 3$ we obtain a better result of $O(n^3 \log^2 n)$ time by exploiting the fact that the union of n axis-parallel unit cubes can be decomposed into $O(n)$ disjoint axis-parallel boxes. We prove that the number of different translations that achieve the minimum Hausdorff distance in d-space is $\Theta(n^{\lfloor 3d/2 \rfloor})$. Furthermore, we present an algorithm which computes the minimum Hausdorff distance under the L_2 metric in d-space in time $O(n^{\lceil 3d/2 \rceil + 1 + \delta})$, for any $\delta > 0$.

1 Introduction

We consider the problem of finding the resemblance, under translation, of two point sets in d-dimensional space for $d \geq 3$. In many matching applications, objects are described by d parameters; thus a single object corresponds to a point in d-dimensional space. One would like the ability to determine whether two sets of such objects resemble each other. A 3D example comes from molecular matching, where a molecule can be described by its atoms, represented as points in 3-space.

The tool that we suggest here for measuring resemblance is the well-researched minimum Hausdorff distance under translation. The distance function we use (except in Section 8) is

*Work by Chew and Kedem was partly supported by US-Israel Binational Science Foundation Grant 94-00279. Chew was also supported by ONR Grant N00014-89-J-1946, by ARPA under ONR contract N00014-88-K-0591, by the US Army Research Office through the Mathematical Sciences Institute of Cornell University under contract DAAL03-91-C-0027, and by the Cornell Theory Center which receives funding from its Corporate Research Institute, NSF, New York State, ARPA, NIH, and IBM Corporation. Kedem was also supported by a grant from the Israel Science Foundation founded by The Israel Academy of Sciences and Humanities.

the L_∞ metric. One advantage of using the Hausdorff distance is that it does not assume equal cardinality of the point sets. It measures the maximal *mismatch* between the sets when one point set is allowed to translate in order to minimize this mismatch. Two point sets are considered to be similar if this mismatch is small. To simplify our presentation, we assume that the cardinalities of the sets are n and $m = O(n)$ and we express our results in terms of n.

There have been several papers on the subject of point set resemblance using the minimum Hausdorff distance under translation. Huttenlocher et al. [9, 10] find the minimum Hausdorff distance for point sets in the plane in time $O(n^3 \log n)$ under the L_1, L_2, or L_∞ metrics. For point sets in 3-dimensional space their algorithm, using the L_2 metric, runs in time $O(n^{5+\epsilon})$. The method used in [10] cannot be extended to work under L_∞.

Chew and Kedem [6] show that, when using the L_∞ metric in the plane, the minimum Hausdorff distance can be computed in time $O(n^2 \log^2 n)$. This is a somewhat surprising result, since there can be $\Omega(n^3)$ different translations that achieve the (same) minimum [6, 14]. They [6] further extend their technique to compute the minimum Hausdorff distance between two point sets in d-dimensional space using the L_∞ metric, achieving a time bound of $O(n^{2d-2} \log^2 n)$ for a fixed dimension d.

We show in this paper that, using the L_∞ metric, the minimum Hausdorff distance between two point sets can be found in time $O(n^3 \log^2 n)$ for $d = 3$, and in time $O(n^{5d/4} \log^2 n)$ for $d > 3$. In an earlier version of this paper [5], we have shown how a time bound of $O(n^{(4d-2)/3} \log^2 n)$ can be achieved for $d > 3$. For $3 < d < 8$, the time bound of [5] is slightly better than the one we present here.

To estimate the quality of the time complexity of our algorithms, it is natural to seek the number of different translations that achieve the minimum Hausdorff distance. More precisely, the number of connected components in the set of feasible translations in the d-dimensional translation space. We show that this number is $\Theta(n^{\lfloor 3d/2 \rfloor})$ in the worst case. Note that, as for the planar case solved in [6], the runtime of the algorithms which we present for a fixed $d \geq 3$ is significantly lower than the number of the connected components in the d-dimensional translation space.

Many optimization problems are solved parametrically by finding an oracle for a *decision problem* and then using this oracle in some parametric optimization scheme. In this paper we follow this line by developing an algorithm for the *Hausdorff distance decision problem* (see definition in the next section) and then using it as an oracle in the Frederickson and Johnson [8] optimization scheme. For the oracle in 3-space we prove that a set of n unit cubes can be decomposed into $O(n)$ disjoint axis-parallel boxes. We then apply the orthogonal partition trees (OPTs) described by Overmars and Yap [13] to find the maximal depth of disjoint axis-parallel boxes. We show that this suffices to answer the Hausdorff distance decision problem in 3-space. For $d > 3$ there is a super-linear lower bound on the number of boxes obtained by disjoint decomposition of a union of boxes (see [3]); thus we cannot use a disjoint decomposition of unit hypercubes. Instead, we build a decision-problem oracle by developing and using a modified, nested version of the OPT.

When using L_2 as the underlying metric we show that there can be $\Omega(n^{\lfloor 3d/2 \rfloor})$ connected components in the translation space, and that the complexity of the space of feasible translations is $O(n^{\lceil 3d/2 \rceil})$. We present an algorithm which computes the minimum Hausdorff

distance under the L_2 metric in d-space in time $O(n^{\lceil 3d/2 \rceil + 1 + \delta})$ for any $\delta > 0$.

The paper is organized as follows: In Section 2 we define the *minimum Hausdorff distance problem*, and describe the *Hausdorff distance decision problem*. In Section 3 we show that the union of n axis-parallel unit cubes in 3-space can be decomposed into $O(n)$ disjoint axis-parallel boxes, and use the orthogonal partition trees of [13] to solve the Hausdorff distance decision problem in 3-space. For $d > 3$, our algorithm is more involved and hence its description is separated into two sections: Section 4 contains a relaxed version of our data structures and an oracle which runs in time $O(n^{3d/2-1} \log n)$; in Section 5 we modify the data structures of the relaxed version and obtain an $O(n^{5d/4} \log n)$ runtime oracle. In Section 6 we show briefly how we plug the decision algorithm into the Frederickson and Johnson optimization scheme. Bounds on the number of translations that minimize the Hausdorff distance are presented in Section 7. The algorithm for the minimum Hausdorff distance under the L_2 metric is discussed in Section 8. Conclusions and open questions appear in Section 9.

Since all the spatial objects we deal with in this paper are axis-parallel cubes, axis-parallel boxes and axis-parallel cells, we omit from now on the words 'axis-parallel' and talk about cubes, boxes and cells. We call a box in d-space a d-box.

2 The Hausdorff Distance Decision Problem

The well-known *Hausdorff distance* between point sets A and B is defined as

$$H(A, B) = \max(h(A, B), h(B, A))$$

where the *one-way* Hausdorff distance from A to B is

$$h(A, B) = \max_{a \in A} \min_{b \in B} \rho(a, b).$$

Here, $\rho(\cdot, \cdot)$ represents a familiar metric on points: for instance, the standard Euclidean metric (the L_2 metric) or the L_1 or L_∞ metrics. In this paper, *unless otherwise noted*, we use the L_∞ metric. In dimension d, an L_∞ "sphere" (i.e., a set of points equidistant from a given center point) is a d-cube.

The *minimum Hausdorff distance* between two point sets is the Hausdorff distance minimized with respect to all possible translations of the point sets. Huttenlocher and Kedem [9] observe that the minimum Hausdorff distance is a metric on point sets (and more general shapes) that is independent of translation. Intuitively, it measures the maximum mismatch between two point sets after the sets have been translated to minimize this mismatch. For the minimum Hausdorff distance the sets do not have to have the same cardinality, although to simplify our presentation, we assume that both point sets are of size $\Theta(n)$.

As in [6] we approach this optimization problem by using the *Hausdorff distance decision problem* with parameter $\varepsilon > 0$ as a subroutine for a search in a sorted matrix of ε values. We define the *Hausdorff distance decision problem* for a given ε to be the question of whether the minimum Hausdorff distance under translation is bounded by ε. We say that the Hausdorff distance decision problem for sets A and B and for ε is *true* if there exists a translation t such that the Hausdorff distance between A and B shifted by t is less than or equal to ε.

We follow the approach taken in [6], solving the Hausdorff distance decision problem by solving a problem of the intersection of unions of cubes in the (d-dimensional) *translation space*. Let A and B be two sets of points as above, let ε be a positive real value, and let C_ε be a d-dimensional cube, with side size 2ε and with the origin at its center (the L_∞ "sphere" of radius ε). We define the set A_ε to be $A \oplus C_\varepsilon$, where $\oplus$ represents the Minkowski sum. Consider the set $A_\varepsilon \oplus (-b)$ where $-b$ represents the reflection of the point b through the origin. This set is the set of translations that map b into A_ε. The set of translations that map all points $b \in B$ into A_ε is then $\cap_{b \in B}(A_\varepsilon \oplus (-b))$;

we denote this set by $S(A, \varepsilon, B)$. It can be shown [6] that the Hausdorff distance decision problem for point sets A and B and for ε is true iff $S(A, \varepsilon, B) \cap -S(B, \varepsilon, A) \neq \emptyset$. We restrict our attention to the problem of determining whether $S(A, \varepsilon, B)$ is empty; extending our method to determining whether the intersection of this set with $-S(B, \varepsilon, A)$ is empty is reasonably straightforward.

Another way to look at the Hausdorff distance decision problem is to assign a different color, call it i, to each $b_i \in B$, $i = 1, \ldots, n$. Now we can look at $A_\varepsilon \oplus -b_i$ as a union of cubes of one color which we call a *layer*. We thus have n layers in n different colors, one layer for each point $b_i \in B$. A point $p \in \mathbb{R}^d$ is *covered* by a color i if $p \in A_\varepsilon \oplus -b_i$. The *color-depth* of p is the number of layers that cover p. Our aim is thus to determine if there is a point p of color-depth n.

3 The Decision Problem in 3 Dimensions

Overmars and Yap [13] address the question of determining the volume of a union of N d-boxes (all boxes are axis-parallel). Using a data structure they call an *orthogonal partition tree*, which we abbreviate as OPT, they achieve a runtime of $O(N^{d/2} \log N)$. They also observe that their data structure can be used to report other measures within the same time bound. One problem that can easily be solved using their data structure is the *maximum coverage* for a set S of N d-boxes. Defining the *coverage* of a point $p \in \mathbb{R}^d$ to be the number of (closed) d-boxes that contain p, the *maximum coverage* is $\max\{coverage(p) \| p \in \mathbb{R}^d\}$.

Maximum coverage is almost what we need for the Hausdorff distance decision problem, but instead we need the maximum color-depth. The difference is that maximum coverage counts the number of different boxes while we need to count the number of different colors (layers). These two concepts are the same if, for each color, all the boxes of that color are disjoint. Actually it is enough to require that all the boxes of the same color are disjoint in their *interiors*. To achieve this, we first decompose each layer into $O(n)$ boxes disjoint in their interiors; then we apply the OPT method to compute the maximum coverage (which will now equal the maximum color-depth).

Theorem 1 *The union of n unit cubes in $\mathbb{R}^3$ can be decomposed, in time $O(n \log n)$, into $O(n)$ boxes whose interiors are disjoint.*

Proof: We slice the 3-dimensional space by planes parallel to the z axis at $z = 0, 1, 2, \ldots$ (without loss of generality, all cubes have nonnegative coordinates). For each integer i, let n_i denote the number of cubes intersected by the plane $z = i$. Surely $\sum n_i \leq n \leq 2 \sum n_i$. Let

E be the portion of the union of cubes that lies within the (closed) slab bounded by $z = i$ and $z = i + 1$. It is known (*e.g.*, [3]) that the complexity of the boundary of the union of n unit 3-cubes is linear in the number of cubes. Therefore, the complexity of the boundary of E is $O(n_i + n_{i+1})$.

To end the proof, we show how to decompose E into $O(n_i + n_{i+1})$ boxes with disjoint interiors. As all cubes are unit cubes, the intersection of any *vertical* line (parallel to the z-axis) with E is either empty, a unit segment, or up to two "short" segments emanating either from the plane $z = i$ or from the plane $z = i + 1$. Let the *silhouette* of E be the projection on both $z = i$ and $z = i + 1$ of all such vertical lines whose intersection with E is one unit long. Clearly the complexity of the silhouette of E is $O(n_i + n_{i+1})$.

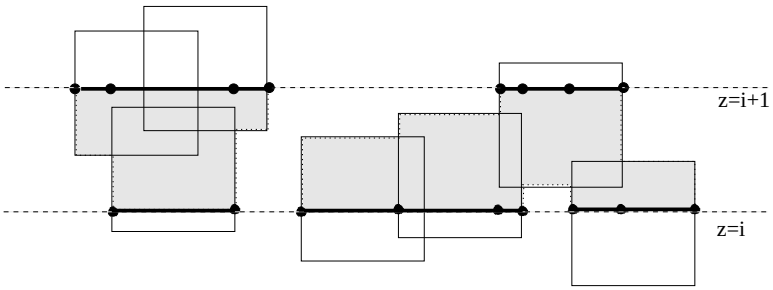

Figure 1: E is shaded, E' and E" are the thick lines on $z = i$ and $z = i + 1$ resp., F' and F" consist of the black points on $z = i$ and $z = i + 1$ resp.

Consider E', the intersection of E with the plane $z = i$ (see Figure 1 for a 2-dimensional illustration). For every point p of E', observe the vertical segment emanating from p towards the opposite boundary of E. Let F' be all points p of E' at which the length of the vertical segment changes. Clearly F' forms a rectilinear shape of up to $O(n_i + n_{i+1})$ vertices and edges (which are not self intersecting) on the plane $z = i$. We perform in time $O((n_i + n_{i+1}) \log(n_i + n_{i+1}))$ a vertical decomposition of F' and extend this decomposition in the z direction until we either hit an end of a short segment or we hit the other plane $z = i + 1$.

Similarly, we can form E'' (the intersections of E with the plane $z = i + 1$) and F''; F'' forms a rectilinear shape of up to $O(n_i + n_{i+1})$ vertices and edges (which are not self intersecting) on the plane $z = i + 1$. We can form a vertical decomposition of F'' and extend this in the z direction (toward the plane $z = i$). Note that the parts of F' and F'' due to the silhouette of E are identical. Since the silhouette of E appears in both the planar arrangements it is clear that the unit long-segments' vertical decompositions coincide while the short-segments' decompositions are disjoint.

This produces a decomposition of the slab E into $O(n_i + n_{i+1})$ disjoint boxes. Summing over all the slabs produces the final decomposition of $O(n)$ disjoint boxes. □

Applying this theorem to each of the n colors, we decompose each layer (recall that a layer is the union of all cubes of a single color) into $O(n)$ disjoint boxes, getting a total

of $N = O(n^2)$ boxes, where boxes of the same color do not overlap in their interiors. We can now apply the Overmars and Yap algorithm on these boxes, getting an answer to the Hausdorff distance decision problem in time $O(n^3 \log n)$. This gives us the following theorem.

Theorem 2 *For point sets in 3-space, the Hausdorff distance decision problem can be answered in time* $O(n^3 \log n)$.

4 Higher Dimensions: the Relaxed Version

The decomposition method used for 3-space cannot be extended efficiently to work for $d > 3$, since as Boissonnat et al. [3] have recently shown, the complexity of the union of n d-dimensional unit cubes is $\Theta(n^{\lfloor d/2 \rfloor})$; thus a single layer (the union of cubes for a single color) cannot be decomposed into $O(n)$ disjoint boxes. Note that we cannot use the Overmars and Yap data structure (OPT) and algorithm directly for the set of n^2 colored cubes because of the possible overlapping of cubes of the same color. Our method is therefore to augment the OPT adding capabilities that efficiently handle the overlapping of same-color cubes.

We describe very briefly the OPT of Overmars and Yap [13]. Let $Q = \{q_1, \ldots, q_N\}$ be a set of N boxes contained in d-space. An OPT $\mathcal{T}$ defined for Q is a binary tree such that each node δ is associated with a box-like d-cell C_δ that contains some part of the d-space. For each node δ, the cell C_δ is the disjoint union of the cells associated with its children. Note that the ancestor/descendent relation in the tree $\mathcal{T}$ corresponds to the containment relation between cells.

Consider a cell C of the OPT and a box $q \in Q$. If $C \subseteq q$ we say that q *covers* C. We say that box q is a *pile* with respect to a cell C if (1) q does not cover C and (2) for at least $d-1$ of the axes, the projection of q on these axes contains the projection of C (see Figure 2). Intuitively, q is a pile with respect to C if q "looks like" a simple planar slab (a thickened plane or hyperplane) from within C. A pile q divides a cell C into at most three parts: (1) $q \cap C$, (2) the portion of C "above" q, and (3) the portion of C "below" q. Some attributes of the OPT are ([13]):

- **(A1)** Each cell C stores those boxes of Q that cover C, but don't cover the parent of C. (In this way, the OPT is an extension of the well known segment tree.)
- **(A2)** Every leaf-cell also stores the boxes of Q that partially cover it (as piles) and for each leaf cell there are $O(\sqrt{N})$ such boxes.
- **(A3)** Each box q partially covers $O(N^{(d-1)/2})$ leaf-cells. Each q is a pile with respect to those leaf cells that it partially covers.
- **(A4)** The height of the OPT tree is $O(\log N)$. The number of nodes in the tree is $O(N^{d/2})$.
- **(A5)** A box q of Q can be inserted into or deleted from the OPT in time $O(N^{(d-1)/2} \log N)$.

Overmars and Yap use the OPT to compute the measure of the union of N d-boxes in time $O(N^{d/2} \log N)$ by treating this *static* d-dimensional problem as a *dynamic* problem in dimension $d-1$. They build an OPT of dimension $d-1$, then they use it to sweep d-space

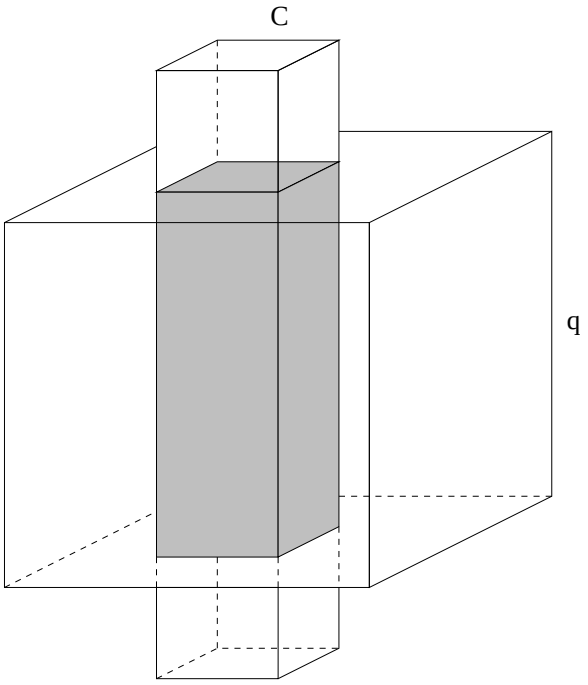

Figure 2: The box q is a pile with respect to the cell C in 3-space. The grey part represents the intersection between q and C.

using a hyperplane h of dimension $d-1$. The set Q of boxes that they use to build the OPT is the set of projections of their original d-boxes onto the $(d-1)$-hyperplane h. During the sweep, each box q is inserted into the OPT when it starts intersecting h and is deleted as h sweeps past it. Note that, because the OPT is of dimensions $d-1$ (and not d), the time to insert/delete a single box q is $O(N^{(d-2)/2}\log N)$. Both insertions and deletions involve some computation concerning the required measure.

We would like to implement a type of OPT for $N = O(n^2)$ *colored* cubes (n cubes in each of the n colors) to find whether there is a point covered by n colors. It is easy to use the OPT to count straightforward coverage, but we need to know the color-depth. The fact that cubes of the same color can overlap makes this difficult. To determine color-depth, we use secondary OPTs, one for each leaf of a primary OPT. Our primary tree will be an OPT of dimension d (instead of $d-1$ as used by Overmars and Yap).

First, note that our Hausdorff problem is more restricted than the measure problem solved by Overmars and Yap in the sense that we don't have boxes of arbitrary size. Instead we have only cubes. In addition they are all unit cubes. (The "unit" here is 2ε where ε is the size parameter of our Hausdorff distance decision problem.)

Second, we don't have to look at all of d-space. As a matter of fact we can restrict our attention to a single unit d-cube. This is because, given point sets A and B and given parameter ε, any translation t that makes the minimum Hausdorff distance between A and $B+t$ less than ε must also bring the minimum x-value of A within ε of the minimum x-value of $B+t$. The same holds true for each of the d axes. Thus, in translation space, the set of translations that could potentially bring A and B within Hausdorff distance ε of each other is restricted to a single unit d-cube, a cube of size 2ε.

Third, since leaf cells are small (smaller than a unit d-cube as shown above), the cubes that partially cover a leaf cell C (the piles of C) intersect C in a more restricted way. Here, a pile q will divide a cell C into at most two parts since if any part of C is "above" q then, since q is large with respect to C, there cannot be any part of C "below" q. Intuitively, a pile q with respect to C "looks like" a half-space from within C.

This last observation implies that, if we restrict our attention to one color, say green, then the part of leaf-cell C that is not-green (the portion left after all intersections of C with green cubes have been removed) is a single d-box within C. Similarly, for each color i, the not-i portion of leaf-cell C is a single d-box within C. We refer to the d-box within C that is not-i as G_i. Note that it is possible for a particular G_i to be empty or to equal the entire cell C.

Now observe that there exists a point p within C that is covered by all colors if and only if there is a point in C that is outside all boxes G_i. This question, in turn, is equivalent to determining whether the measure of the union $\cup_i G_i$ is equal to the measure of C. By posing the problem in this way we have converted our color-depth problem into a set of measure problems in d space which can be answered by applying the algorithm of Overmars and Yap on each leaf cell separately. This gives us the following straightforward algorithm for the Hausdorff distance decision problem.

1. Using the set Q of $N = O(n^2)$ colored d-cubes, determine the $O(N^{d/2})$ leaf-cells of the OPT of Q (restricted to a unit d-cube as explained above).

2. For each cell C:

   (a) For each color i, determine the d-box $G_i \subseteq C$ that is not-i.

   (b) Determine (using a secondary OPT) whether the measure of $\cup_i G_i$ is equal to the measure of C. If not then report **True** and halt.

3. Report **False** and halt.

Time and Space Analysis

Overmars and Yap have shown that their algorithm can work using only $O(N)$ space. This is done by creating one leaf cell at a time and then performing the measure computation for this leaf cell. In our case the space requirement can be improved to just $O(n)$ space, even though there are $N = \Theta(n^2)$ d-cubes. This is because, for the Hausdorff distance decision problem, the N cubes are generated from just $O(n)$ points; thus, the N cubes can be stored implicitly in $O(n)$ space.

We can afford to be a bit sloppy in the time needed to build the primary OPT since this portion of the algorithm is far from the most time-consuming part. We attempt to keep space costs low. Instead of building all leaf-cells at once, we build each one as we need it, taking time $O(N)$ for each leaf-cell. By property (A2), the space needed to store a cell C along with the list of those d-cubes that partially cover C is $O(\sqrt{N}) = O(n)$. In addition, we need to retain the list of colors that completely cover C, requiring $O(n)$ additional space. The not-i d-boxes G_i can be built in time $O(n)$ and there are of course n of them, one for each color.

Once we have all the boxes G_i, we build secondary OPTs to compute the measure of their union. For each leaf-cell C of the primary OPT, this takes time $O(n^{d/2} \log n)$ and space $O(n)$ [13]. Multiplying this time by the number of primary-OPT leaf-cells $O(N^{d/2}) = O(n^d)$, we get the following intermediate result (which we improve in the next Section).

Lemma 3 *For $d > 3$, the Hausdorff distance decision problem can be answered in time $O(n^{3d/2} \log n)$ using $O(n)$ space.*

5 Higher Dimensions: the Improved Version

In this section we improve the relaxed algorithm described in Section 4. In the relaxed algorithm we used two classes of OPT: a primary OPT and, for each leaf cell, a secondary OPT. Our final time bound was due to multiplying $O(n^d)$, the number of leaf cells in the primary OPT, by $O(n^{d/2})$, the time needed to compute the measure of a union of n d-boxes using a secondary OPT. We develop an improved algorithm by finding a better balance between these two quantities. The idea is to decrease the number of cells in the primary OPT, thus making more work for the secondary OPTs.

Decreasing the number of leaf cells in the primary OPT has two effects: (1) there are more boxes per leaf cell and (2) some of the boxes that partially cover a leaf-cell are nonpiles. (Recall that for a standard OPT, each box that intersects a leaf cell either completely covers

the cell or intersects it as a *pile*; see Property (A3) in Section 4.) We show that, as long as the number of nonpile boxes is relatively small, secondary OPTs can be built without severe performance penalty.

To explain our technique, we first discuss the way in which a standard OPT is built [13]. To make the $O(N^{d/2})$ leaf-cells, we first divide d-space into $2\sqrt{N}$ slabs by cutting with $(d-1)$-flats perpendicular to axis x_1. This is done in such a way that there are $\sqrt{N}$ 1-*boundaries* in each slab (an i-boundary is a cube boundary — a $(d-1)$-flat — that is perpendicular to axis x_i). Now we split each of these slabs with respect to x_2. We first split at every $\sqrt{N}$th 2-boundary of those d-boxes that intersect the slab. In addition, for each slab, we split at all those 2-boundaries that are boundaries of d-cubes that have a 1-boundary in the slab; there are $O(\sqrt{N})$ of these. Intuitively, this ensures that the subslabs contain no "corners." After both these kinds of splits, each slab has been divided into $O(\sqrt{N})$ subslabs, each of which contains $O(\sqrt{N})$ 2-boundaries and $O(\sqrt{N})$ 1-boundaries. This process continues. At dimension i, we first split at every $\sqrt{N}$th i-boundary. In addition, we split at all those i-boundaries that are boundaries of d-cubes that have a j-boundary (for $j < i$) in the current slab; there are $O(\sqrt{N})$ of these. The end result is a structure that satisfies properties (A1) through (A5) in Section 4. See [13] for additional details.

We modify this construction. We first divide d-space, with $(d-1)$-flats perpendicular to axis x_1, into N^α slabs where each slab contains $O(N^{1-\alpha})$ 1-boundaries; α is a parameter representing a fixed constant whose value will be determined later in the proof. Now we split each of these slabs with $(d-1)$-flats perpendicular to axis x_2. We first split at every $N^{1-\alpha}$th 2-boundary, creating $O(N^\alpha)$ subslabs. In addition, for each slab we split at *some* of those 2-boundaries of d-cubes with 1-boundaries in the slab. We cannot afford to split at *all* such boundaries as we did in the construction of the standard OPT, since we want to have the same number of splits ($O(N^\alpha)$) for each type of split. So we have to use $O(N^\alpha)$ splits, leaving $O(N^{1-2\alpha})$ of these cube corners (the 2-boundaries of cubes that also have 1-boundaries within the slab) within each subslab. In a sense, these cube corners disrupt the OPT structure, so our solution is to set these aside. In other words, as we continue subdividing the current slab, we will ignore these $O(N^{1-2\alpha})$ disrupting cubes. That's not to say that these cubes are gone forever: they are only ignored for this current slab and, even though ignored for the rest of the construction, they remain associated with the current slab (so we can find them later).

A similar construction method is used for the other dimensions. At dimension i, working on a single slab, we divide the slab with $(d-1)$-flats perpendicular to axis x_i. We create $O(N^\alpha)$ subslabs each containing $N^{1-\alpha}$ i-boundaries and $O(N^{1-2\alpha})$ cube corners. The cube corners disrupt the OPT structure so they are set aside and the construction continues.

When this process ends, we have $O(N^{d\alpha})$ leaf-cells, each having (well-behaved) partial intersections with $O(N^{1-\alpha})$ cubes; these are the cubes that intersect a leaf-cell to form piles. In addition, we have all the disrupting cubes that were set aside during the construction process. Each cell inherits $O(N^{1-2\alpha})$ of these from each dimension i, giving a total of $O(dN^{1-2\alpha})$. We can ignore the constant factor of d, absorbing it into the big-O notation. So each leaf cell has $O(N^{1-\alpha})$ "good" cubes (piles) that partially intersect it and $O(N^{1-2\alpha})$ "bad" cubes that partially intersect it.

Now, as we did before, we want to build a secondary OPT for each leaf-cell. There are

$N^{1/2}$ colors, so if we didn't have to worry about the bad cubes, we could build a standard OPT structure and check the leaf-cell for color depth in time $O(N^{d/4}\log N)$ (there is one box G_i for each of the colors i, $i = 1, \ldots, N^{1/2}$). The key observation is that the bad cubes can be handled in a naive way without significantly messing up the structure of our secondary OPT. During the building of the secondary OPT we do some extra splitting: basically, we split slabs at every bad cube boundary. Thus, each slab is split $O(N^{\max(\frac{1}{4},1-2\alpha)})$ times: $N^{\frac{1}{4}}$ due to the standard construction for $N^{\frac{1}{2}}$ boxes, $N^{1-2\alpha}$ due to splitting at the bad-cube boundaries. In the end we have $O(N^{d*\max(\frac{1}{4},1-2\alpha)})$ secondary leaf-cells. Note that by splitting the slabs in this way, there are no partial intersections of secondary leaf-cells with bad cubes: for each secondary leaf-cell and for each bad cube, either the cube completely covers the leaf-cell or they don't intersect at all. Thus color-depth for a primary leaf-cell can be determined in time $O(N^{d*\max(\frac{1}{4},1-2\alpha)}\log N)$.

The total time for the Hausdorff distance decision problem comes from the number of leaf-cells in the primary OPT structure ($O(N^{d\alpha})$) multiplied by the time to determine color-depth in the secondary OPT structure ($O(N^{d*\max(\frac{1}{4},1-2\alpha)}\log N)$). Total time is thus $O(N^{d*\max(\alpha+\frac{1}{4},1-\alpha)}\log N)$. The optimal time is where $\alpha+\frac{1}{4}$ and $1-\alpha$ are equal. This occurs at $\alpha = \frac{3}{8}$. Total time is then $O(N^{5d/8}\log N)$ or, in terms of n, $O(n^{5d/4}\log n)$.

This gives us the following theorem. As in Lemma 3, we keep our space requirement low by never building all the primary OPT leaf-cells at once. Instead, we build each leaf-cell only as it is needed.

Theorem 4 *For $d > 3$, the Hausdorff distance decision problem can be answered in time $O(n^{5d/4}\log n)$ using $O(n)$ space.*

Combining this result with the 2-dimensional result [6], with our 3-dimensional result from Section 3 and with results descibed in an earlier version of this paper [5], we summarize the best time bounds known.

Theorem 5 *The Hausdorff distance decision problem can be answered in time $O(n^d\log n)$ for dimension $d = 2$ and $d = 3$, in time $O(n^{(4d-2)/3}\log n)$ for dimensions $4 \le d \le 8$ and in time $O(n^{5d/4}\log n)$ for dimensions $d \ge 8$.*

6 Finding the Minimum Hausdorff Distance

Now we want to determine the minimum ε for which the intersection is still non-empty. It is easy to see that the desired minimum value is achieved at some ε_0 for which two cubes just touch each other.

We need to search among all possible values of ε where two cubes touch at their boundaries. We compute the pairwise distances for the cubes for each axis separately. We thus perform d searches, each over a set of $O(n^4)$ ε values, and find for each axis the smallest ε for which the intersection is not empty. The largest ε of these d minima is the required minimum Hausdorff distance.

We can afford to use a simple binary search on these values for dimension $d \ge 4$ (note though that this raises the space requirement to $\Theta(n^4)$) because the algorithm for finding

the maximal color-depth is of the same complexity as for sorting all the pairwise distances. For $d = 3$, however, this is too costly, since the Hausdorff distance decision problem is solved in time $O(n^3 \log n)$. Hence we apply here, as in [6], the method of Frederickson and Johnson [8] for solving an optimization problem by using a sorted matrix (stored implicitly): Given a sorted matrix of size N by N it takes time $O(N + D \log N)$ to solve the optimization problem where D is the runtime of the decision problem. For us, $N = n^2$ and $D = O(n^3 \log n)$.

Theorem 6 *Using the L_∞ metric, the minimum Hausdorff distance under translation between point sets A and B in d-space can be determined in time $O(n^3 \log^2 n)$ if $d = 3$, in time $O(n^{(4d-2)/3} \log^2 n)$ if $3 < d \leq 8$, and in time $O(n^{5d/4} \log^2 n)$ if $d \geq 8$, where $n = \max\{|A|, |B|\}$.*

Proof: The proof follows directly from the discussion above and Theorem 5. □

7 Combinatorial Bounds

In this section we show combinatorial results on the region (in translation space) of all those translations that minimize the Hausdorff distance between two point sets.

Theorem 7 *Let $\mathcal{L} = \{L_1, \ldots, L_m\}$ be a collection of layers in d-space where each layer L_i is the union of n unit cubes, and let T denote the intersection of the layers. Then the number of vertices of T is $O(m^d n^{\lfloor d/2 \rfloor})$.*

Proof: Assume for simplicity that the cubes are in general position. Let v be a vertex of T, and let $\mathcal{L}' \subseteq \mathcal{L}$ denote the d or fewer layers which contain v on their boundary. Surely, v is a vertex of the intersection of the layers of $\mathcal{L}'$. However, v is also a vertex of the union of the layers of $\mathcal{L}'$, since v cannot lie in the interior of any of the layers of $\mathcal{L}'$. Thus, each vertex of T is also a vertex of a union of at most d layers of $\mathcal{L}$.

The proof now follows from a simple counting argument. There are $O(m^d)$ subsets of $\mathcal{L}$ containing d or fewer of the layers. Each subset contains at most dn cubes, and by the result of [3] cited above, the union of dn unit cubes in d-space has complexity $O((nd)^{\lfloor d/2 \rfloor})$. Hence the total number of vertices generated in the union of d or fewer layers of $\mathcal{L}$ is $O(m^d) \times O((dn)^{\lfloor d/2 \rfloor})$ which is the bound we are after. Note that factors involving only d can be treated as constant and absorbed into the big-O notation. □

Observe that since the regions of T are all axis-parallel, the bound on the number of vertices of T also bounds the complexity T, and of course the number of connected regions of T. We now show that this bound is tight in the worst case.

Theorem 8 *Let $\mathcal{L}$ and T be defined as in Theorem 7. Then the number of connected regions in T is $\Omega(m^d n^{\lfloor d/2 \rfloor})$.*

Proof: Our construction extends an earlier construction due to Rucklidge [14] that he developed for the two-dimensional case. We first show how to construct $\Omega(m^2 n)$ connected regions in T for $d = 2$. We define a *corridor of color* i as the region enclosed between two stairs of squares of color i, as shown in Figure 3(a). We are assuming that layer L_i corresponds to the union of all squares of color i. Each step of the "stairs" is of size δ, where $\delta > 0$ is a small fixed parameter (depending on n) choosen to ensure that the entire corridor fits within a single one of our unit squares. The corridor is generated by n cubes and has $n/2$ stairs. We refer to this first corridor as the *base corridor*.

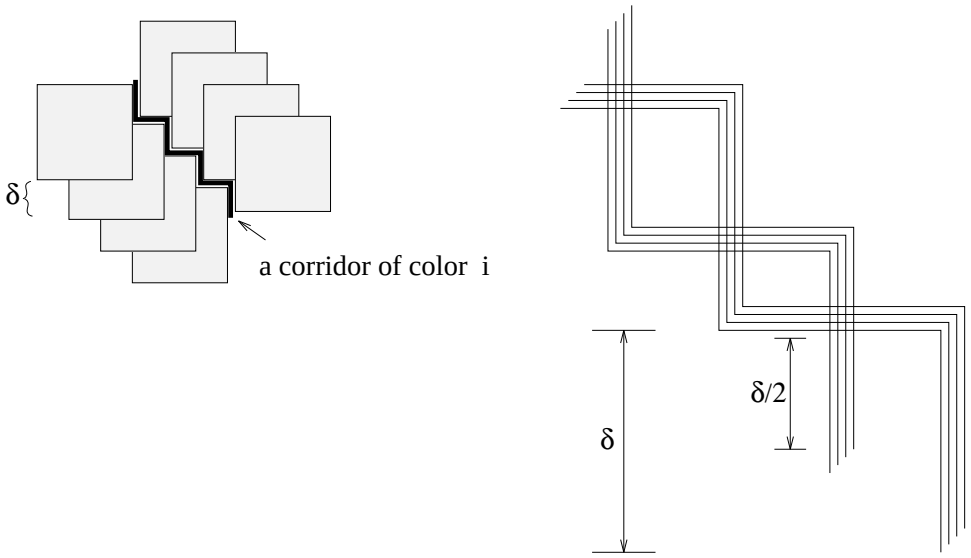

Figure 3: Lower bound construction

We generate two batches of $m/2$ layers, where each layer has its own corridor: a translated copy of the base corridor. Note that for layer L_i, L_i's corridor is a narrow region that is *not* covered by color i. The ith corridor in the first batch (for $i = 1, \ldots, m/2$) is generated by shifting the base corridor by $i\delta'$ (in both the x and y directions) where $\delta' = \delta/2m$. The ith corridor of the second batch (for $i = m/2 + 1, \ldots, m$) is generated by translating the first batch by $\delta/2$ in both the x and y directions. See Figure 3(b).

Note that each corridor in batch one intersects each corridor in batch two $\Omega(n)$ times, yielding $\Omega(m^2 n)$ distinct maximally-colored regions. Most of these regions (all but $O(mn)$ of them) are convex, rectangle-shaped, and adjacent along all four sides to corridors. These rectangular regions are the ones we use in the d-dimensional construction.

We turn to the proof in $\mathbb{R}^d$. Let us first assume that d is even, and set $k = d/2$. As before, we associate layer L_i with color i. We divide our m colors into k groups, each of size $m' = m/k$. Note that we can choose m so that it is divisibe by k.

For each group of m' colors, we build a set of $m'n$ d-cubes in $\mathbb{R}^d$ so that, for group j ($j = 1, \ldots, k$), the projection of the cubes on the $2j$ and $2j + 1$ axes produces a set of squares and corresonding corridors identical to the two-dimensional Rucklidge construction

given above. The other $d-2$ coordinates of the cube centers are identically zero. Note that for each group j, there is a set of maximally colored (colored with all m' color of group j) d-bricks that are rectangles when projected on the $2j$ and $2j+1$ axes. These d-bricks are of unit size along all axes except for axes $2j$ and $2j+1$. By the earlier construction, there are $\Omega((m')^2 n) = \Omega(m^2 n)$ such d bricks for each group j.

Our claim is that there are $\Omega(m^2 k n^k)$ connected components of T where T is the intersection of the layers and each layer L_i corresponds to the union of all the cubes of color i. To verify this claim, consider the number of d-bricks generated as the cross-product of our 2-dimensional rectangles. It's easy to see that there are $\Omega(m^{2k} n^k) = \Omega(m^d n^{d/2})$ of these.

If d is odd, we embed the even-dimensional construction in the first $d-1$ dimensions of $\mathbb{R}^d$, but we use just $m/2$ of the colors (this affects only the constant in the big-omega notation) — all cubes have center-coordinate zero for the dth axis. The remaining $m/2$ colors are used to create a "striped" pattern on the dth axis. For each of these colors, we create two cubes, centered along all axes except the dth. Along the dth axis the two cubes have a very small gap between them. The gap for each color is in a slightly different place leading to the result that, for these $m/2$ colors, there are $1 + m/2$ intervals along the dth axis that are covered by all the $m/2$ colors, separated by $m/2$ gaps where each gap is missing a different color. It's easy to see that each of the bricks created in the even-dimensional construction is cut into $\Omega(m)$ pieces by these gaps. This is enough to give us the desired bound for the theorem. □

The bounds of Theorems 7 and 8 actually show that there can be $\Theta(n^{\lfloor 3d/2 \rfloor})$ translations t that minimize the *one-way* Hausdorff distance between A and $B+t$, where A and B contain n points each. One can show, by slightly modifying the proofs, that the same bounds hold for the number of translations that minimize the two-way Hausdorff distance.

If the cubes defining $\mathcal{L}$ are axis-parallel but not necessarily of the same size, then the proofs of Theorems 7 and 8 can be modified to obtain the larger (upper and lower) bound of $\Theta(m^d n^{\lceil d/2 \rceil})$.

Theorem 9 *Let $\mathcal{L} = \{L_1, \ldots, L_m\}$ be a collection of layers, where each layer L_i is the union of n axis-parallel cubes of arbitrary size in $\mathbb{R}^d$. Then the complexity of $\cap_i L_i$ is $\Theta(m^d n^{\lceil d/2 \rceil})$.*

Proof: The upper bound follows in a straightforward way from [3], cited above, where it is shown that the complexity of the union of dn such cubes in d-space is $O(\,(dn)^{\lceil d/2 \rceil})$. To show the matching lower bound, we modify the proof of Theorem 8. First, note that no change is necessary for d even. For d odd, we modify the construction used to create gaps along the dth axis. The idea is to use small cubes to create more gaps along the dth axis. For a particular color, we can use n small cubes of that color, arranged one after the other along the dth axis, with small gaps between them. Similar versions, shifted slightly along the dth axis can be arranged for each of the other colors. Combining these we create $\Omega(mn)$ all-color intervals separated by gaps. By choosing δ, the stair-step size, sufficiently small, we can ensure that our d bricks are cut by these gaps, completing the proof of the theorem. □

8 Hausdorff Distance under the L_2 Norm

Our upper bound technique described above is useful for showing similar bounds for balls instead of cubes. The following theorem turns out to have important algorithmic implications.

Theorem 10 *Let $\mathcal{L} = \{L_1, \ldots, L_m\}$ be a collection of m layers, where each layer L_i is the union of n balls of arbitrary size in $\mathbb{R}^d$. Then the complexity of $\cap_i L_i$ is $O(m^d n^{\lceil d/2 \rceil})$.*

Proof: The standard lifting transformation $\lambda : \mathbb{R}^d \to \mathbb{R}^{d+1}$ is defined by $\lambda((x_1, \ldots, x_d)) \equiv (x_1, \ldots, x_d, x_1^2 + \ldots + x_d^2)$ (see, e.g., [7]). The vertices of the union of balls in each layer can be expressed as (a subset of) the vertices of the upper envelope of n hyperplanes in $\mathbb{R}^{(d+1)}$. The number of such vertices is $O(n^{\lfloor (d+1)/2 \rfloor}) = O(n^{\lceil d/2 \rceil})$ [7]. The remainder of the proof is similar to that of Theorem 7. □

Let $A = \{a_1, \ldots a_n\}$ and $B = \{b_1, \ldots b_n\}$ be two sets of points in d-space, for $d \geq 4$, and let $H_2(A, B)$ denote the Hausdorff distance between them, when L_2 is the underlying metric. In [9, 10] an $O(n^{5+\delta})$-time algorithm[1] for finding a translation t that minimizes $H_2(A, B + t)$ when $A, B \subseteq \mathbb{R}^3$ was presented. In this section, we extend this result for higher dimensions, by obtaining an $O(n^{\lceil 3d/2 \rceil + 1 + \delta})$-time algorithm for the case that A and B are in $\mathbb{R}^d$, for $d \geq 4$.

Our algorithm is analogous to the one in Section 2. Let ε^* be $\min_t H_2(A, t + B)$, for $t \in \mathbb{R}^d$, and let ε be a given fixed number. Our goal is to determine whether ε is equal to, smaller or larger than ε^*. Let D_ε denote a ball (under the L_2 norm) of radius ε centered at the origin. Let

$$\bar{T}(\varepsilon) \equiv \cap_{b \in B}(b - (A \oplus D_\varepsilon)) \bigcap \cap_{a \in A}((B \oplus D_\varepsilon) - a)$$

As discussed in Section 2, $\bar{T}(\varepsilon)$ is empty, if and only if $\varepsilon < \varepsilon^*$. For every $a_i \in A$ let us define the layer $L_i(\varepsilon) \equiv (B \oplus D_\varepsilon) - a_i$ to be the union of the ε-balls about $b_j - a_i$ (for all $b_j \in B$). We denote these balls by $\mathcal{B}(L_i)$. We obtain an oracle for determining whether $T(\varepsilon) = \bigcap_i L_i(\varepsilon)$ is empty. Determining whether $\bar{T}(\varepsilon)$ is empty is obtained in a straightforward way. The oracle consists of two phases; In the *Generation Phase*, we generate a set $\mathcal{S}$ of points which is a superset of the vertices of $T(\varepsilon)$. In the second phases, the *Decision Phase*, we check if any element of $\mathcal{S}$ lies in (the closure of) $T(\varepsilon)$, which implies that $T(\varepsilon) \neq \emptyset$.

The Generation Phase: Following the proof of Theorem 7 and Theorem 10, for each subset $\mathcal{L}' \subseteq \mathcal{L}$ of d or less layers, we compute $\mathcal{S}'$, the set of all vertices of the union of the balls of $\mathcal{B}(\mathcal{L}')$. Under the lifting transform $\lambda(\cdot)$ described in the proof of Theorem 10, the boundary of each ball of $\mathcal{B}(\mathcal{L}')$ is transformed into a region in a hyperplane in $\mathbb{R}^{d+1}$. A vertex of $\mathcal{S}'$ corresponds to a vertex of the upper envelope of these hyperplanes. We use the algorithm of Seidel [16], to generate this upper envelope in time $O(n^{\lceil d/2 \rceil} \log n)$ for each such $\mathcal{L}'$. Let $\mathcal{S}$ denote the union of these vertices generated for each such $\mathcal{L}' \subseteq \mathcal{L}$. The upper envelope, however, might contain edges not containing a vertex. Such an edge could be created, for example, in the case where $\bigcap \mathcal{L}$ consist only of the intersection of less than

[1] Throughout the section, δ stands for a positive constant which can be chosen arbitrarily small with an appropriate choice of other constants of the algorithms.

d balls. For each such edge e appearing on the upper envelope of hyperplanes, we pick an arbitary point on e, and add this point to $\mathcal{S}$. The time required for generating $\mathcal{S}$ is therefore

$$d\binom{n}{d} \times O(n^{\lceil d/2 \rceil} \log n) \;=\; O(n^{\lceil 3d/2 \rceil} \log n) \;.$$

As shown in the proof of Theorem 10, the vertices of $T(\varepsilon)$ (if it is not empty) are contained in $\mathcal{S}$.

The Decision Phase: Here we check each point $q \in S$ to find if it lies in $T(\varepsilon)$. The transformation $\lambda(\cdot)$ transforms each ball b of $\mathcal{B}(\mathcal{L}_i)$ into a halfspace $\lambda(b) \subseteq \mathbb{R}^{d+1}$. Let H_i denote the intersection of the complements of these halfspaces (for $i = 1 \ldots n$). Surely, $q \in L_i$ if and only if $\lambda(q) \notin H_i$. To check this condition, we preprocess H_i in time $O(n^{\lceil d/2 \rceil + \delta})$ into a point-location data structure $\mathcal{D}_i$, of Matoušek [11], so that determining if $q \in H_i$ is obtained in time $O(\log n)$. Hence determining if $q \in T(\varepsilon)$ is obtained in time $O(n \log n)$. We perform this query for each point of $\mathcal{S}$ (in time $O(n^{\lceil 3d/2 \rceil + 1} \log n)$. Note that $\mathcal{D}_i$ needs to be constructed only for a single L_i, since the layers are just translation copies of each other. This completes the description of the oracle.

Optimization:

We turn now to the problem of finding ε^*. Note that a ball in d-space is determined by $d+1$ points, and hence there are $\Omega(n^{2(d+1)})$ critical values ε at which the layer structures combinatorially changes. So in contrast to the L_∞-case, we cannot generate all the critical values. To overcome this difficulty, we use the parametric searching paradigm of Megiddo [12]. We assume familiarity of the reader with the technique, and refer to [1] for a description of the technique and some of its geometric applications. The technique requires the construction of a parallel version of the oracle. Consider the Generation Phase. Constructing all subsets $\mathcal{L}' \subseteq \mathcal{L}$ can be performed sequentially, since this process is not effected by ε. For a subset $\mathcal{L}'$, we need to construct the vertices of the intersection of the halfspaces $\lambda(b)$ (in $\mathbb{R}^{d+1}$) for each ball $b \in \mathcal{B}(L')$. Under the "standard" primal-dual transformation, this intersection is transformed into the convex hull (in $\mathbb{R}^{d+1}$) of the points dual to hyperplanes bounding these halfspaces. It is computed using the algorithm of Amato et al. [2] in $O(\log n)$ parallel steps, using $O(n^{\lceil d/2 \rceil} \log^c n)$ processors, for some constant $c > 0$ (these bounds refer to d-space for $d \geq 4$). Applying this procedure in parallel to each $\mathcal{L}' \subseteq \mathcal{L}$, we can generate $\mathcal{S}$ in $O(\log n)$ parallel steps using $O(n^{\lceil 3d/2 \rceil} \log^c n)$ processors.

Turning to parallel implementation of the Decision Phase. The data structure $\mathcal{D}_i$ can be constructed in $O(n^\delta)$ parallel steps, using similar construction as in [4]. Performing the $O(n^{\lceil 3d/2+1 \rceil})$ queries in $\mathcal{D}_i$ is trivially done in $O(\log n)$ parallel steps. Plugging the parallel algorithm into the parametric search paradigm, we observe that the algorithm performs $O(n^\delta)$ parallel steps, consulting the oracle $O(\log n)$ times at each step. Hence the total time spent for consulting the oracle is $O(n^{d+\lceil d/2 \rceil + 1 + \delta})$, which bounds the total time of the algorithm. Thus we have

Theorem 11 *Let A and B be two point-sets of n points each in d-space, ($d \geq 4$) Then a translation t minimizing $H_2(A,\, B + t)$ can be found in time $O(n^{\lceil 3d/2 \rceil + 1 + \delta})$, for every $\delta > 0$.*

9 Conclusions and Open Problems

For point sets in dimension d the minimum Hausdorff distance under translation can be found in time $O(n^{5d/4}\log^2 n)$ when L_∞ is the underlying metric and in time $O(n^{\lceil 3d/2\rceil+1+\delta})$ for any $\delta > 0$ when the underlying metric is L_2. For $d = 3$ under the L_∞ metric, we obtain the better result of $O(n^3 \log^2 n)$. For $3 < d < 8$ under the L_∞ metric, we can obtain a slightly better result of $O(n^{(4d-2)/3}\log^2 n)$ as shown in our earlier paper [5].

There are some interesting questions that remain open.

- Are L_∞ Hausdorff distance decision problems inherently easier than the ones for L_2? Is there a single technique that solves both types of problems efficiently, perhaps with better time bounds than those achieved here?

- Given a d-dimensional box C and n boxes $G_1, \ldots, G_n$ lying inside C, is it possible to determine if $\cup_1^n G_i = C$ in time $o(n^{d/2})$? Finding such a technique would immediately improve the running time of the relaxed version of our L_∞ algorithm and would be likely to speed up the improved version as well.

- For large d, our time bounds—and our constant factors, hidden by the big-O—are such that our exact algorithms are likely to be impractical. What kinds of nontrivial approximations are useful?

- The set of cubes for one color is really just a translation of the set of cubes for each other color. Our algorithm uses this fact to reduce the storage capacity, but not to improve the running time. Is there some way to use this information to design a faster algorithm?

Our algorithms can be modified within the same asymptotoic time bounds to tackle the problem of *pattern matching in the presence of spurious points*, sometimes called *outliers*. In this problem we seek a small subset $X \subseteq A \cup B$ of points (whose existence is perhaps a result of noise) containing at most a pre-determined number k of points, such that $A \setminus X$ can be optimally matched, under translation, to $B \setminus X$. The modification needed is the following: In the algorithm we used the assumption that our attention could be restricted to a single unit cube in the translation space. This assumption is not valid if we assume the existence of spurious points. To overcome this difficulty, we divide the translation d-space into unit cubes, and solve our problem independently in each such non-empty cube. This does not increase the overall running time, since each original cube hits just $O(2^d)$ of these new unit cubes.

Acknowledgments.

We would like to thank Micha Sharir for useful discussions and help in the proof of Theorem 7. Thanks also to Reuven Bar-Yehuda for contributing to the algorithm of Section 4.

References

[1] P. K. Agarwal, M. Sharir and S. Toledo, Applications of Parametric Searching in Geometric Optimization, *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, 1992, 72–82.

[2] N. M. Amato, M. T. Goodrich and E. R. Ramos, Parallel Algorithms for Higher-Dimensional Convex Hulls, *Proceedings 6th Annual ACM Symposium on Theory of Computing*, 1994, 683–694.

[3] J. Boissonnat, M. Sharir, B. Tagansky and M. Yvinec, Voronoi Diagrams in Higher Dimensions under Certain Polyhedral Distance Functions, *Proc. 11th Annual ACM Symposium on Computational Geometry*, 1995, 79–88.

[4] B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Diameter, width, closest line pair and parametric searching, *Discrete Comput. Geom.* 10 (1993), 183–196.

[5] L. P. Chew, D. Dor, A. Efrat, and K. Kedem, Geometric Pattern Matching in d-Dimensional Space, *Proceedings of the Third Annual European Symposium on Algorithms* 1995, editted by Paul Spirakis, 264–279.

[6] L.P. Chew and K. Kedem, Improvements on Geometric Pattern Matching Problems, *Algorithm Theory – SWAT '92*, edited by O. Nurmi and E. Ukkonen, Lecture Notes in Computer Science #621, Springer-Verlag, July 1992, 318–325.

[7] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, Germany, 1987.

[8] G. Frederickson and D. Johnson, Generalized selection and ranking: sorted matrices, *SIAM J. Comput. 13* (1984), 14–30.

[9] D.P. Huttenlocher and K. Kedem, Efficiently Computing the Hausdorff Distance for Point Sets under Translation, *Proceedings of the Sixth ACM Symposium on Computational Geometry*, 1990, 340–349.

[10] D.P. Huttenlocher, K. Kedem and M. Sharir, The Upper Envelope of Voronoi Surfaces and its Applications, *Discrete and Computational Geometry*, 9 (1993), 267–291.

[11] J. Matoušek, Reporting Points in Halfspaces, *Comput. Geom. Theory Appl.*, 2 (1992), 169–186.

[12] N. Megiddo, Applying Parallel Computation Algorithms in the Design of Serial Algorithms, *J. ACM* 30 (1983), 852–865.

[13] M. Overmars and C.K. Yap, New Upper Bounds in Klee's Measure Problem, *SIAM Journal of Computing*, 20 (1991), 1034–1045.

[14] W.J. Rucklidge, Lower Bounds for the Complexity of the Hausdorff Distance, *Discrete and Computational Geometry*, 16(2) 1996, pp. 135–153.

[15] W.J. Rucklidge, Private Communication.

[16] R. Seidel, Constructing Higher-Dimensional Convex Hulls at Logarithmic Cost Per Face, *Proc. 18th Annu. ACM Sympos. Theory Comput.* 1986, 404–413.