Maximum Overlap Area of a Convex Polyhedron and a Convex Polygon Under Translation

Honglin Zhu
Massachusetts Institute of Technology, Cambridge, MA, USA

Hyuk Jun Kweon
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Let P be a convex polyhedron and Q be a convex polygon with n vertices in total in three-dimensional space. We present a deterministic algorithm that finds a translation vector $v \in \mathbb{R}^3$ maximizing the overlap area $|P \cap (Q+v)|$ in $O(n \log^2 n)$ time. We then apply our algorithm to solve two related problems. We give an $O(n \log^3 n)$ time algorithm that finds the maximum overlap area of three convex polygons with n vertices in total. We also give an $O(n \log^2 n)$ time algorithm that minimizes the symmetric difference of two convex polygons under scaling and translation.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases computational geometry, shape matching, arrangement

Digital Object Identifier 10.4230/LIPIcs.SoCG.2023.61

Related Version *Full Version*: `https://arxiv.org/abs/2301.02949`

Acknowledgements This paper is the result of the MIT SPUR 2022, a summer undergraduate research program organized by the MIT math department. The authors would like to thank the faculty advisors David Jerison and Ankur Moitra for their support and the math department for providing this research opportunity. We thank the anonymous referees for providing helpful comments that increased the quality of this paper.

1 Introduction

Shape matching is an important topic in computational geometry, with useful applications in areas such as computer graphics. In a typical problem of shape matching, we are supplied two or more shapes, and we want to determine how much the shapes resemble each other. More precisely, given a similarity measure and a set of allowed transformations, we want to transform the shapes to maximize their similarity measure.

There are many candidates for the similarity measure, such as the Hausdorff distance and the Fréchet distance between the boundaries of the shapes. We can also consider the area/volume of overlap or of symmetric difference. The advantage to these is that they are more robust against noise on the boundary of the shapes [6].

The maximum overlap problem of convex polytopes has been studied by many. In dimension 2, de Berg et al. [6] give an $O(n \log n)$ time algorithm for finding a translation maximizing the area of intersection of two convex polygons (where n denotes the total number of vertices of the polygons). In dimension 3, Ahn et al. [1] give an $O(n^3 \log^4 n)$ expected time algorithm finding the maximum overlap of two convex polyhedra under translation. For the same problem, Ahn et al. [3] present an algorithm that runs in $O(n \log^{3.5} n)$ time with probability $1 - n^{-O(1)}$ and an additive error. For $d > 3$, given two convex polytopes of dimension d with n facets in total, Ahn et al. [3] give an algorithm that finds the maximum overlap under translation in $O(n^{\lfloor d/2 \rfloor + 1} \log^d n)$ time with probability $1 - n^{O(1)}$ and an additive error.

In the plane, when all rigid motions are allowed, Ahn et al. [4] give an approximate algorithm that finds a rigid motion realizing at least $1-\epsilon$ times the maximal overlap in $O((1/\epsilon)\log n + (1/\epsilon^2)\log(1/\epsilon))$ time. In dimension 3, Ahn et al. [2] present an approximate algorithm that finds a rigid motion realizing at least $1-\epsilon$ times the maximal overlap in $O(\epsilon^{-3} n \log^{3.5} n)$ with probability $1-n^{-O(1)}$.

When considering the maximum overlap as a similarity measure, we obviously can only allow area/volume-preserving transformations. However, we may want to allow scaling as a transformation – two similar triangles are supposed to be very "similar," though they may have different sizes. In this case, the area of symmetric difference is a better measure of similarity. Yon et al. [14] give an algorithm minimizing the symmetric difference of two convex polygons under translation and scaling in $O(n\log^3 n)$ expected time.

Our results

While many have studied the matching problem for two convex polytopes of the same dimension, to our knowledge no one has examined the problem for polytopes of different dimensions or matching more than two polytopes.

The main result in this paper is a deterministic algorithm for the problem of matching a convex polyhedron and a convex polygon under translation in three-dimensional space.

▶ **Theorem 1.** *Let P be a convex polyhedron and Q a convex polygon with n vertices in total. We can find a vector $v \in \mathbb{R}^3$ that maximizes the overlap area $|P \cap (Q+v)|$ in $O(n\log^2 n)$ time.*

We also present two applications of our algorithm to other problems in computational geometry. First, we give a deterministic algorithm for maximizing the overlap of three convex polygons under translations.

▶ **Theorem 2.** *Let P, Q, R be three convex polygons with n vertices in total in the plane. We can find a pair of translations $(v_Q, v_R) \in \mathbb{R}^4$ that maximizes the overlap area $|P \cap (Q + v_Q) \cap (R + v_R)|$ in $O(n\log^3 n)$ time.*

We also give a deterministic $O(n\log^2 n)$ time algorithm for minimizing the symmetric difference of two convex polygons under a homothety (a translation and a scaling), which is an improvement to Yon et al.'s randomized algorithm [14].

▶ **Theorem 3.** *Let P and Q be convex polygons with n vertices in total. Then we can find a homothety φ that minimizes the area of symmetric difference $|P \setminus \varphi(Q)| + |\varphi(Q) \setminus P|$ in $O(n\log^2 n)$ time.*

The main ingredient in the proof of Theorem 1 is a new technique we introduce which generalizes Megiddo's prune-and-search [13]. This allows us to efficiently prune among n groups of m parallel lines.

Let $S = \bigcup_{i=1}^{n} S_i$ be a union of n sets of $O(m)$ parallel lines in the plane, none of which are parallel to the x-axis, and suppose the lines in each S_i are indexed from left to right.

▶ **Lemma 4.** *In $O(n)$ time, $\mathbb{R}^2$ can be partitioned into six regions $R_1, \ldots, R_6$ by three lines, and we can find six subsets $S^{R_1}, \ldots, S^{R_6} \subset S$ such that for each $i \le 6$, S^{R_i} contains all lines intersecting the interior of R_i and $|S^{R_i}| \le \frac{17}{18}|S|$.*

With this lemma, we can employ divide-and-conquer to obtain the following.

▶ **Theorem 5.** *Suppose there is an unknown point $p^* \in \mathbb{R}^2$ and we are given an oracle that decides in time T the relative position of p^* to any line in the plane. Then we can find the relative position of p^* to every line in S in $O((T+n)\log(mn))$ time.*

The omitted proofs can be found in the full version of this paper [12].

2 Preliminaries

Let $P \subset \mathbb{R}^3$ be a convex polyhedron and $Q \subset \mathbb{R}^2$ be a convex polygon with n vertices in total. Throughout the paper, we assume that Q is in the xy-plane, and that the point in P with minimal z coordinate is on the xy-plane. We want to find a translation vector $v = (x, y, z) \in \mathbb{R}^3$ that maximizes the overlap area $f(v) = |P \cap (Q + v)|$.

It is easy to observe that $f(v)$ is continuous and piecewise quadratic on the interior of its support. As noted in [6, 1, 3], f is smooth on a region R if $P \cap (Q + v)$ is combinatorially equivalent for all $v \in R$, that is, if we have the same set of face-edge incidences between P and Q. Following the convention of [1], we call the polygons that form the boundaries of these regions the *event polygons*, and as in [6], we call the space of translations of Q the *configuration space*. The arrangement of the event polygons partition the configuration space into cells with disjoint interiors. The overlap function $f(v)$ is quadratic on each cell. Thus, to locate a translation maximizing f, we need to characterize the event polygons.

For two sets $A, B \subset \mathbb{R}^d$, we write the *Minkowski sum* of A and B as

$$A + B := \{a + b | a \in A, b \in B\}.$$

We will make no distinction between the translation $A + v$ and the Minkowski sum $A + \{v\}$ for a vector v. We also write $A - B$ for the Minkowski sum of A with $-B = \{-b | b \in B\}$. We categorize the event polygons into three types and describe them in terms of Minkowski sums:

(I) When $Q + v$ contains a vertex of P. For each vertex u of P, we have an event polygon $u - Q$. There are $O(n)$ event polygons of this type.

(II) When a vertex of $Q + v$ is contained in a face of P. For each face F of P and each vertex v of Q, we have an event polygon $F - v$. There are $O(n^2)$ event polygons of this type.

(III) When an edge of $Q + v$ intersects an edge of P. For each edge e of P and each edge e' of Q, we have an event polygon $e - e'$. There are $O(n^2)$ event polygons of this type.

The reason that convexity is fundamental is due to the following standard fact, as noted and proved in [6, 14].

▶ **Proposition 6.** *Let P be a d'-dimensional convex polytope and let Q be a d-dimensional convex polytope. Suppose $d' \geq d$. Let $f(v) = \mathrm{Vol}(P \cap (Q + v))$ be the volume of the overlap function. Then, $f(v)^{1/d}$ is concave on its support* $\mathrm{supp}(f) = \{v | f(v) > 0\}$.

As in [5], we say a function $f : \mathbb{R} \to \mathbb{R}$ is *unimodal* if it increases to a maximum value, possibly stays there for some interval, and then decreases. It is *strictly unimodal* if it strictly increases to the maximum and then strictly decreases. Furthermore, we say a function $f : \mathbb{R}^d \to \mathbb{R}$ is (strictly) unimodal if its restriction to any line is (strictly) unimodal.

The following corollary of Proposition 6 allows us to employ a divide-and-conquer strategy in our algorithm.

▶ **Corollary 7** ([5]). *For any line l parameterized by $l = p + vt$ in $\mathbb{R}^{d'}$ for $v \neq 0$, the function $f_l(t) = f(p + vt)$ is strictly unimodal.*

We also use the following two techniques in our algorithm.

▶ **Lemma 8** ([11]). *Let M be an $m \times n$ matrix of real numbers, where $m \leq n$. If every row and every column of M is in increasing order, then we say M is a sorted matrix. For any positive integer k smaller or equal to mn, the k-th smallest entry of M can be found in $O(m \log(2n/m))$ time, assuming an entry of M can be accessed in $O(1)$ time.*

For our purposes, we will use this result in the weaker form of $O(m + n)$.

▶ **Lemma 9** ([8]). *Given n hyperplanes in $\mathbb{R}^d$ and a region $R \subset \mathbb{R}^d$, a $(1/r)$-cutting is a collection of simplices with disjoint interiors, which together cover R and such that the interior of each simplex intersects at most n/r hyperplanes. A $(1/r)$-cutting of size $O(r^d)$ can be computed deterministically in $O(nr^{d-1})$ time. In addition, the set of hyperplanes intersecting each simplex of the cutting is reported in the same time.*

3 Generalized two-dimensional prune-and-search

In this section, we prove Theorem 5, our generalization of Megiddo's prune-and-search technique [13]. This technique is of independent interest and can likely be applied to other problems.

In [13], Megiddo proves the following:

▶ **Theorem 10** ([13]). *Suppose there exists a point $p^* \in \mathbb{R}^2$ not known to us. Suppose further that we have an oracle that can tell us for any line $l \subset \mathbb{R}^2$ whether $p^* \in l$, and if $p^* \notin l$, the side of l that p^* belongs to. Let T be the running time of the oracle. Then given n lines in the plane, we can find the position of p^* relative to each of the n lines in $O(n + T \log n)$ time.*

We are interested in a generalized version of Megiddo's problem. Suppose, instead of n lines, we are given n sets of parallel lines $S_1, S_2, \dots, S_n$, each of size $O(m)$. In addition, suppose the lines in each S_i are indexed from left to right (assuming none of the lines are parallel to the x-axis). Again, we want to know the position of p^* relative to every line in $S = \bigcup_{i=1}^n S_i$. Megiddo's algorithm solves this problem in $O(mn + T \log(mn))$ time, but we want a faster algorithm for large m by exploiting the structure of S.

Without loss of generality, suppose that there are no lines parallel to the y-axis. For each i between 1 and n, let $S_i = \{l_i^1, l_i^2, \dots\}$ where l_i^j lies strictly to the left of l_i^{j+1} for all vaild j. Suppose that $p^* = (x^*, y^*) \in \mathbb{R}^2$. To report our final answer, we need to provide, for each S_i, the two consecutive indices a and $a+1$ such that p^* lies strictly between l_i^a and l_i^{a+1} or the single index a such that $p^* \in l_i^a$.

In our algorithm, we keep track of a feasible region R containing P^*, which is either the interior of a (possibly unbounded) triangle or an open line segment if we find a line l that p^* lies on. Together with R, we keep track of the $2n$ indices lower(i) and upper(i) such that $S^R = \bigcup_{i=1}^n S_i^R = \{l_i^j | j \in (\text{lower}(i), \text{upper}(i)]\}$ contains the set of lines intersecting R. In the beginning, $R = \mathbb{R}^2$. Each step, we find $O(1)$ lines to run the oracle on to find a new feasible region $R' \subset R$ such that $|S^{R'}| \leq \frac{17}{18}|S^R|$ and recurse on R'. An outline is given in Algorithm 3.1.

We will use the following well-known result:

▶ **Lemma 11** ([10]). *Suppose we are given n distinct real numbers with positive weights that sum to 1. Then we can find the weighted median of these numbers in $O(n)$ time.*

Given S^R and R, we want to find $R' \subset R$ to recurse on, as well as efficiently update $S^{R'}$. Note that $S^{R'}$ need not be exactly the set of lines intersecting the interior of R'; we only need it to contain those lines and have size a constant fraction smaller than S^R.

Algorithm 3.1 Pseudocode for Theorem 5.

```
input  : A set $S = \bigcup_{i=1}^{n} S_i = \{l_i^j\}$ of $O(mn)$ lines
output : A list of indices that indicate the position of $p^*$ to each $S_i$
1 $R \longleftarrow \mathbb{R}^2$
2 $S^R \longleftarrow S$
3 while $|S^R| \geq 18$ do
4     Find $O(1)$ lines to run the oracle on
5     Compute the piece $R' \subset R$ containing $p^*$
      /* We guarantee that $R'$ intersects at most 17/18 of the lines that
         intersect $R$                                                    */
6     Triangulate $R'$ with $O(1)$ lines to run the oracle on
7     Update $S^R \longleftarrow S^{R'}$
8 end
9 Compute relative position of $p^*$ to the remaining lines in $|S^R|$ by brute force
```

Proof of Lemma 4. We write $S^R = S = \bigcup_{i=1}^{n} S_i = \{l_i^j\}$. We first find the weighted median of the slopes of the lines in S, where the slope of the lines of S_i is weighted by $|S_i|/|S|$. This can be done in $O(n)$ time by Lemma 11.

If this slope is equal to the slope of some line in S_i and $|S_i| \geq \frac{1}{9}|S|$, then we can simply divide the plane using the median line of S_i and the x-axis and the interior of each quadrant will avoid at least 1/18 of the lines of S. The subsets S^{R_i} can be formed by removing either half of the lines of S_i.

Otherwise, at least 4/9 of the lines have slopes strictly greater than/less than the median slope. Without loss of generality, we assume at least 4/9 of the lines have positive slope and at least 4/9 of the lines have negative slope. Now let $S_+ = \bigcup_{i=1}^{k} S_i$ and $S_- = \bigcup_{i=k+1}^{n} S_i$ denote the set of lines with positive/negative slope, respectively. We remove lines from the larger of the two sets until they have the same size.

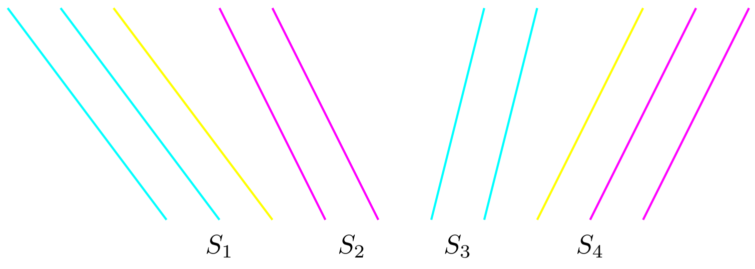


Figure 1 P_1, P_2 are P_3 are represented by colors.

We partition $S_+ \cup S_-$ into $O(n)$ subsets P_i each containing the same number of lines from S_+ and S_- in the following way: going in lexicographical order by the indices of the lines, we put a line from S_1 and a line from S_{k+1} into P_1 until we exhaust one of the sets (say it is S_{k+1}). Then, we move on to put a line from the remaining S_1 and a line from S_{k+2} into P_2 until we exhaust one of them, and so on. Each P_i is then of the form $\{l_{a(i)}^{b(i)}, \ldots, l_{a(i)}^{b(i)+|P_i|/2-1}, l_{c(i)}^{d(i)}, \ldots, l_{c(i)}^{d(i)+|P_i|/2-1}\}$, and can be represented by the indices $(a(i), b(i))$ and $(c(i), d(i))$ (see Figure 1). We can compute this partition in $O(n)$ time. For each P_i, we compute the intersection $p_i = (x_i, y_i)$ of the median line in P_i with positive slope and the median line with negative slope, and assign p_i a weight $w_i = |P_i|/(2|S_+|)$. Then, the weights of the p_i sum to 1. The significance of this is that the interior of each of the four quadrants of the plane defined by $x = x_i$ and $y = y_i$ is avoided by at least 1/4 of the lines in P_i, which is at least $\frac{2}{9}w_i$ of all the lines in $|S|$.

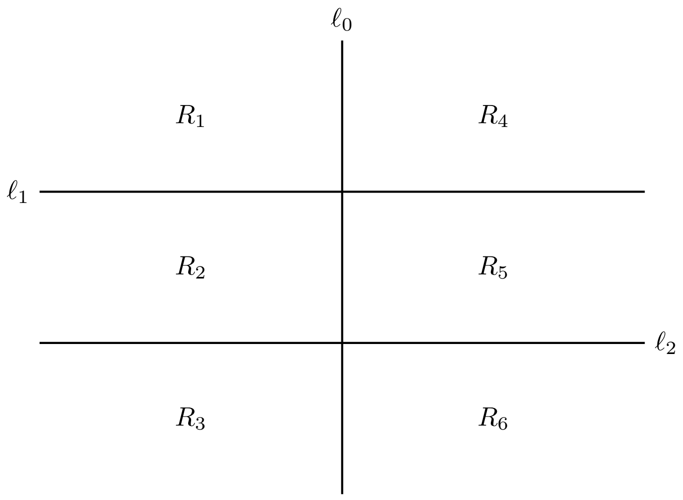


Figure 2 Dividing the plane into six regions.

We find the median point $q_0 = (x_q, y_q)$ of the p_i's by weight in x-coordinate in $O(n)$ time by Lemma 11. We have the line $\ell_0 : x = x_{q_0}$. We then find the median point of the p_i's to the left of ℓ_0 and the median point of those to the right of ℓ_0 by weight in y-coordinates, respectively. Suppose these are $q_1 = (x_{q_1}, y_{q_1})$ and $q_2 = (x_{q_2}, y_{q_2})$. Then let $\ell_1 : y = y_{q_1}$ and $\ell_2 : y = y_{q_2}$. The three lines ℓ_0, ℓ_1, and ℓ_2 partition the plane into six closed regions $R_1, \ldots, R_6$ as in Figure 2. By our construction, the weights of points in each of $R_1, R_2 \cup R_3, R_4 \cup R_5, R_6$ sum to at least 1/4. Thus, the interiors of $R_5 \cup R_6, R_4, R_3, R_1 \cup R_2$ each avoids at least $\frac{2}{9} \cdot \frac{1}{4} = \frac{1}{18}$ of all the lines in S. In particular, the interior of each R_i intersects no more than 17/18 of the lines in S.

We show how to compute $|S^{R_1}|$, and the others follow similarly. If $p_i \in R_6$, then the lines in P_i with positive slope and to the right of p_i avoid $R_1 \cup R_2$. We can remove these lines by updating the indices of the associated set S_j or parallel lines. This updating takes $O(1)$ time for each p_i and $O(n)$ time in total. ◀

Applying Lemma 4 on S^R, we obtain three lines on which we can run the oracle to get a new feasible region R_i and a subset S^{R_i}. We then triangulate it with $O(1)$ more oracle calls to get R' and set $S^{R'} = S^{R_i}$, in $O(T + n)$ time total.

Proof of Theorem 5. After $O(\log mn)$ recursive iterations of Lemma 4, we arrive at a feasible region whose interior intersects less than 18 lines in S, and we can finish by brute force. Therefore, our algorithm runs in $O((T + n)\log(mn))$ time. ◀

▶ Remark 12. A simpler and probably more practical algorithm for Lemma 4 is simply choosing a random line from S_+ and S_- to intersect and run the oracle on the horizontal and vertical line through the intersection. This method gives the same run time in expectation.

4 Maximum overlap of convex polyhedron and convex polygon

In this section, we give the algorithm that finds a translation $v \in \mathbb{R}^3$ maximizing the area of overlap function f. Following the convention in [6], we call such a translation a *goal placement*. In the algorithm, we keep track of a closed *target region* R which we know contains a goal placement and decrease its size until for each event polygon F, either $F \cap \text{interior}(R) = \varnothing$ or $F \supset R$. Then, f is quadratic on R and we can find the maximum of f on R using standard calculus. Thus, the goal of our algorithm is to efficiently trim R to eliminate event polygons that intersect it.

In the beginning of the algorithm, the target region is the interior of the Minkowski sum $P - Q$, where the overlap function is positive. By the unimodality of the overlap function, the set of goal placements is convex. Thus, for a plane in the configuration space, either it contains a goal placement, or all goal placements lie on one of the two open half spaces separated by the plane. If we have a way of knowing which case it is for any plane, we can decrease the size of our target region by cutting it with planes and finding the piece to recurse. More precisely, we need a subroutine **PlaneDecision** that decides the relative position of the set of goal placements to a plane S.

Whenever **PlaneDecision** reports that a goal placement is found on a plane, we can let the algorithm terminate. Thus, we can assume it always reports a half-space containing a goal placement.

As in Algorithm 4.1, we break down our algorithm into three stages.

Algorithm 4.1 Pseudocode for Theorem 1.

```
input : A convex polyhedron $P \in \mathbb{R}^3$ and a convex polygon $Q \in \mathbb{R}^3$ with $n$ vertices
        in total
output: A translation $v \in \mathbb{R}^3$ maximizing the area $|P \cap (Q + v)|$
1 Locate a horizontal slice containing a goal placement that does not contain any
    vertices of $P$ and replace $P$ by this slice of $P$
2 Find a "tube" $D + l_y$ whose interior contains a goal placement and intersects $O(n)$
    event polygons, where $D$ is a triangle in the $xz$-plane and $l_y$ is the $y$-axis
3 Recursively construct a $(1/2)$-cutting of the target region $D + l_y$ to find a simplex
    containing a goal placement that does not intersect any event polygon
```

4.1 Stage 1

In the first stage of our algorithm, we make use of [6] to simplify our problem so that P can be taken as a convex polyhedron with all of its vertices on two horizontal planes.

We sort the vertices of P by z-coordinate in increasing order and sort the vertices of Q in counterclockwise order. Next, we trim the target region with horizontal planes (planes parallel to the xy-plane) to get to a slice that does not contain any vertices of P.

▶ **Lemma 13.** *In $O(n \log^2 n)$ time, we can locate a strip $R = \{(x, y, z) | z \in [z_0, z_1]\}$ whose interior contains a goal placement and P has no vertices with $z \in [z_0, z_1]$.*

Proof. Starting with the median z-coordinate of the vertices of P, we perform a binary search on the levels containing a vertex of P. For a horizontal plane S, [6, Theorem 3.8] allows us to compute the maximum overlap of $P \cap S$ and Q under translation in $O(n \log n)$-time. The

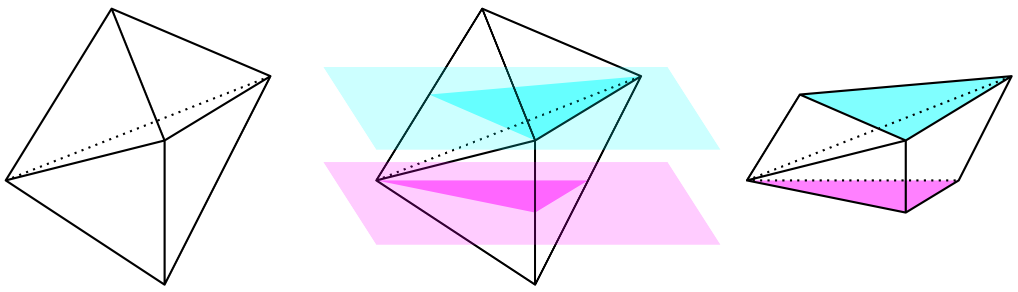


Figure 3 The slice of P with $z \in [z_0, z_1]$.

two planes S_1 and S_2 with the largest maximum values will be the bounding planes for the slice containing a goal placement by the unimodality of f. Thus, by a binary search, we can locate this slice in $O(n \log^2 n)$ time. ◀

By Chazelle's algorithm [7], the convex polyhedron $P' = \{(x, y, z) \in P | z \in [z_0, z_1]\}$ can be computed in $O(n)$ time. From now on, we replace P with P' (see Figure 3). Without loss of generality, assume $z_0 = 0$ and $z_1 = 1$.

The region in the configuration space where $|P \cap (Q + v)| > 0$ is the Minkowski sum $P - Q$. Since P only has two levels $P_0 = \{(x, y, z) \in P | z = 0\}$ and $P_1 = \{(x, y, z) \in P | z = 1\}$ that contain vertices, the Minkowski sum $P - Q$ is simply the convex hull of $(P_0 - Q) \cup (P_1 - Q)$, which has $O(n)$ vertices. We can compute $P_0 - Q$ and $P_1 - Q$ in $O(n)$ time and compute their convex hull in $O(n \log n)$ time by Chazelle's algorithm [9].

4.2 PlaneDecision

With the simplification of the problem in Stage 1, we now show that the subroutine **PlaneDecision** can be performed in $O(n \log n)$ time. Let S be a fixed plane in the configuration space. We call a translation v that achieves $\max_{v \in S} f(v)$ a *good placement*. First, we can compute the intersection of S with $P - Q$ in $O(n)$ time by Chazelle's algorithm [7]. If the intersection is empty, we just report the side of S containing $P - Q$. From now on assume this is not the case.

The following lemma shows that **PlaneDecision** runs in the same time bound as the algorithm that just finds the maximum of f on a plane.

▶ **Lemma 14.** *Suppose we can compute $\max_{v \in S} f(v)$ for any plane $S \subset \mathbb{R}^3$ in time T, then we can perform **PlaneDecision** for any plane in time $O(T)$.*

Proof. The idea is to compute $\max_{v \in S'} f(v)$ for certain S' that are perturbed slightly from S to see in which direction relative to S does f increase.

We compute over an extension of the reals $\mathbb{R}[\omega]/(\omega^3)$, where $\omega > 0$ is smaller than any real number. Let $A > 0$ be the maximum of f over a plane S. Let S_+ and S_- be the two planes parallel to S that have distance ω from S. We compute $A_+ = \max_{v \in S_+} f(v)$ and $A_- = \max_{v \in S_-} f(v)$ in $O(T)$ time. Since f is piecewise quadratic, A_+ and A_- as symbolic expression will only involve quadratic terms in ω. Since f is strictly unimodal on $P - Q$, there are three possibilities:

1. If $A_+ > A$, then halfspace on the side of S_+ contains the set of goal placements.
2. If $A_- > A$, then halfspace on the side of S_- contains the set of goal placements.
3. If $A \geq A_+$ and $A \geq A_-$, then A is the global maximum of f.

Thus, in $O(T)$ time, we can finish **PlaneDecision**. ◀

Finding a good placement on S is similar to finding a goal placement on the whole configuration space. S is partitioned into cells by the intersections of event polygons with S. We need to find a region on S containing a good placement that does not intersect any event polygons.

We present a subroutine **LineDecision** that finds, for a line $l \subset S$, the relative position of the set of good placements on S to l.

▶ **Proposition 15.** *For a line $l \subset S$, we can perform **LineDecision** in $O(n)$ time.*

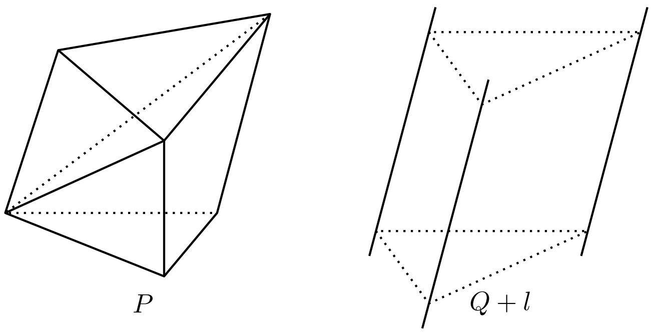


Figure 4 The convex polyhedron I is formed by interesecting P and $(Q + l)$.

Proof. First, we compute $\max_{v \in l} f(v)$ and a vector achieving the maximum. We parameterize the line l by $p + vt$ where t is the parameter and $p, v \in \mathbb{R}^3$. The horizontal cross-section of $I = P \cap (Q + l)$ at height t has area $f(p + vt)$. Since I is the intersection of two convex polytopes with $O(n)$ vertices (see Figure 4), Chazelle's algorithm [7] computes I in $O(n)$ time. Then, [5, Theorem 3.2] computes the maximum cross-section in $O(n)$ time.

Now, by the same argument and method as in the proof of Lemma 14, we can finish **LineDecision** in $O(n)$ time. In the case where $\max_{v \in l} f(v) = 0$, we report the side of l containing $S \cap (P - Q)$. ◀

Whenever our subroutine **LineDecision** reports a good placement is found on a line, we can let the algorithm terminate. Thus, we can assume it always reports a half-plane of S containing a good placement.

We now present **PlaneDecision**. If S is horizontal, then we only need to find the maximum overlap of the convex polygons $P \cap S$ and Q using De Berg et al.'s algorithm [6], which takes $O(n \log n)$ time. Thus, we assume S is non-horizontal.

Algorithm 4.2 Pseudocode for **PlaneDecision**.

```
input  : A plane $S \subset \mathbb{R}^3$
output : A translation $v \in S$ maximizing the area $|P \cap (Q + v)|$
1 Compute $S \cap (P - Q)$ and set it to be our initial target region
2 Locate a strip on $S$ containing a good placement whose interior intersects $O(n)$ event
    polygons
3 Recursively construct a (1/2)-cutting of the strip to find a triangle containing a good
    placement that does not intersect any event polygon
```

As in Algorithm 4.2, we break down **PlaneDecision** into three steps. We have already explained Step 1, where we compute $S \cap (P - Q)$, so we begin with Step 2.

4.2.1 PlaneDecision: Step 2

We want to find a strip on S strictly between $z = 0$ and $z = 1$ that intersects $O(n)$ event polygons. Since there are no vertices of P with z-coordinate in the interval $(0, 1)$, there are no event polygons of type I in this range, and we will only need to consider event polygons of type II and type III.

We look at the intersection points of S with the edges of the event polygons. These edges come from the set $\{e_i - v_j | e_i$ non-horizontal edge of P, v_j vertex of $Q\}$. Without loss of generality, assume that S is parallel to the y-axis. We are interested in the z-coordinates of the intersections, so we project everything into the xz-plane. Then, S becomes a line, which we denote by l_S, and each edge $e_i - v_j$ becomes a segment whose endpoints lie on $z = 0$ and $z = 1$. Suppose each edge e_i projects to a segment s_i, and each v_j projects to a point x_j on the x-axis. Then, we get $O(n^2)$ segments $s_i - x_j$ with endpoints on $z = 0$ and $z = 1$, and the line l_S that intersect them in some places.

▶ **Lemma 16.** *In $O(n \log n)$ time, we can locate a strip $R = \{(x, y, z) \in S | z \in [z_0, z_1]\}$ whose interior contains a good placement and intersects none of the edges of the event polygons.*

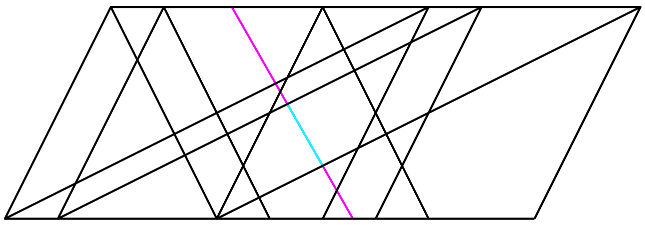


Figure 5 Projecting the configuration space onto the xz-plane. The projection of S is the magenta line segment, and the projection of the strip R obtained form Lemma 16 is the cyan line segment.

Our current target region, the strip R we obtained from Lemma 16 (see Figure 5), intersects few event polygons and we can compute them efficiently.

▶ **Lemma 17.** *The interior of the region R intersects $O(n)$ event polygons, and we can compute them in $O(n \log n)$ time.*

4.2.2 PlaneDecision: Step 3

Now we have a target region R as well as the $O(n)$ intersections it makes with the event polygons.

▶ **Lemma 18.** *In $O(n \log n)$ time, we can find a region $R' \subset R$ containing a good placement that does not intersect any of the $O(n)$ event polygons.*

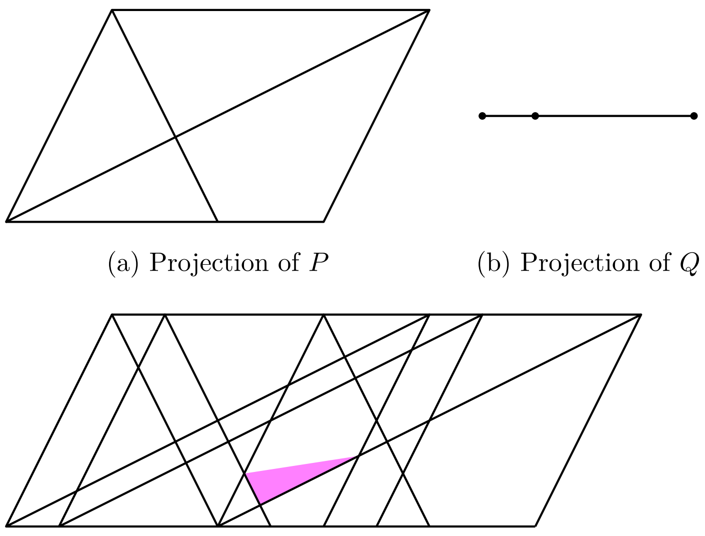
Proof. We recursively construct a (1/2)-cutting of the target region. By Lemma 9, a (1/2)-cutting of constant size can be computed in $O(n)$ time. We perform **LineDecision** on the lines of the cutting to decide on which triangle to recurse. After $O(\log n)$ iterations, we have a target region R' that intersects no event polygons. This procedure runs in $O(n \log n)$ time. ◀

Finally, since the overlap function is quadratic on our final region R', we can solve for the maximum using standard calculus. After finding $\max_{v \in S} f(v)$ and a vector achieving it $O(n \log n)$ time, by Lemma 14, we can perform **PlaneDecision** on S in the same time bound.

▶ **Proposition 19.** *For a plane S, we can perform **PlaneDecision** in $O(n \log n)$ time.*

4.3 Stage 2

With the general **PlaneDecision** at our disposal, we now move on to Stage 2, the main component of our algorithm. We project the entire configuration space and the event polygons onto the xz-plane in order to find a target region D whose preimage $D + l_y$ intersects few event polygons, where l_y is the y-axis (see Figure 6).


(a) Projection of P (b) Projection of Q

(c) Projection of the configuration space, and the target region D

Figure 6 Projecting onto the xz-plane.

The non-horizontal edges of the event polygons project to segments on the strip $0 < z < 1$ on the xz-plane. We characterize our desired region D in the following lemma.

▶ **Lemma 20.** *For a region D that does not intersect any of the segments that are the projections of the non-horizontal edges of the event polygons, the preimage $D + l_y$ intersects $O(n)$ event polygons.*

Now it remains to efficiently find such a region D with $D + l_y$ containing a goal placement and compute the $O(n)$ event polygons that intersect its interior.

▶ **Lemma 21.** *In $O(n \log^2 n)$ time, we can find a triangle D in the xz-plane such that the interior of $D + l_y$ contains a goal placement and intersects $O(n)$ event polygons. We can compute these $O(n)$ event polygons in the same time bound.*

Proof. The computation of D is a direct application of Theorem 5, where $m = O(n)$. Calling the oracle on a line l in the xz-plane is running the **PlaneDecision** algorithm on the plane parallel to the y-axis that projects to l. We compute a triangle for each of the four groups of segments, take their intersection, and triangulate the intersection using $O(1)$ calls to **PlaneDecision**. Thus, we can compute the desired triangle D in $O(n \log^2 n)$ time.

To compute the event polygons intersecting the interior of $D + l_y$ is simple, since we have shown in the proof of Lemma 20 that D intersects at most one projection of an event polygon of each type in each of the four groups for a fixed vertex x_j (for type II) or segment s_i (for type III). Once we have D, we can compute these polygons by binary search on each of the $O(n)$ groups of $O(n)$ non-intersecting segments to find the two between which R lies. Also, the event polygons all have constant complexity so computing all of them takes linear time. We can recover the event polygons from their projections and compute the planes that contain them in linear time. Thus, this entire process can be done in $O(n \log n)$ time. ◀

4.4 Stage 3

Now, we have a target region $R = D + l_y$ whose interior contains a goal placement, and we have the $O(n)$ event polygons that intersect it.

▶ **Lemma 22.** *In $O(n \log^2 n)$ time, we can find a region $R' \subset R$ containing a goal placement that does not intersect any of the $O(n)$ event polygons.*

Proof. We recursively construct a (1/2)-cutting of the target region. By Lemma 9, a (1/2)-cutting of constant size can be computed in $O(n)$ time. We perform **PlaneDecision** on the planes of the cutting to decide on which simplex to recurse. After $O(\log n)$ iterations, we have a target region R' that intersects no event polygons. This procedure runs in $O(n \log^2 n)$ time. ◀

Finally, since the overlap function is quadratic on our final region R', we can solve for the maximum using standard calculus. This concludes the proof of Theorem 1.

5 Maximum overlap of three convex polygons

Let P, Q, R be three convex polygons with n vertices in total in the plane. We want to find a pair of translations $(v_Q, v_R) \in \mathbb{R}^4$ that maximizes the overlap area $g(v_Q, v_R) = |P \cap (Q + v_Q) \cap (R + v_R)|$.

In this problem, the configuration space is four-dimensional. An easy extension of Proposition 6 and Corollary 7 shows that the function of overlap area is again unimodal. This time, we have four-dimensional *event polyhedra* instead of event polygons that divide the configuration space into four-dimensional cells on which $g(v_Q, v_R)$ is quadratic. We call a hyperplane containing an event polyhedron an *event hyperplane*, and they are defined by two types of events:

(I) When one vertex of P, $Q + v_Q$ or $R + v_R$ lies on an edge of another polygon. There are $O(n)$ groups of $O(n)$ parallel event hyperplanes of this type.

(II) When an edge from each of the three polygons intersect at one point. There are $O(n^3)$ event hyperplanes of this type.

To overcome the difficulty of dealing with the $O(n^3)$ event hyperplanes of type II, we first prune the configuration space to a region intersecting no event hyperplanes of type I. We then show that the resulting region only intersects $O(n)$ event hyperplanes of type II.

Similar to Theorem 1, we want an algorithm **HyperplaneDecision** that computes, for a hyperplane $H \subset \mathbb{R}^4$, the maximum $\max_{(v_Q, v_R) \in H} g(v_Q, v_R)$ and the relative location of the goal placement to H. In fact, we will only need to perform **HyperplaneDecision** on some hyperplanes.

▶ **Proposition 23.** *Suppose H is a hyperplane that satisfies one of the following three conditions:*

(1) *H is orthogonal to a vector $(x_1, y_1, 0, 0)$ for some $x_1, y_1 \in \mathbb{R}$.*

(2) *H is orthogonal to a vector $(0, 0, x_2, y_2)$ for some $x_2, y_2 \in \mathbb{R}$.*

(3) *H is orthogonal to a vector $(x_1, y_1, -x_1, -y_1)$ for some $x_1, y_1 \in \mathbb{R}$.*

*Then, we can perform **HyperplaneDecision** on H in $O(n \log^2 n)$ time.*

Using Proposition 23, we can prune the configuration space to a region that intersects no event hyperplanes of type I and $O(n)$ event hyperplanes of type II.

▶ **Proposition 24.** *We can compute a 4-polytope T_{PQR} of complexity $O(1)$ in $O(n \log^3 n)$ time such that*
(1) *the goal placement lies on T_{PQR},*
(2) *no hyperplane of type I intersects the interior of T_{PQR}, and*
(3) *only $O(n)$ event polyhedrons of type II passes through T_{PQR}.*
The hyperplanes of type II intersecting the interior of T_{PQR} are obtained in the same time bound. Furthermore, the 3-tuples of edges of P, Q and R defining the hyperplanes are also obtained in the same time bound.

In the rest of the section, we fix T_{PQR} as in Proposition 24. Moreover, let

$$f(v_P, v_Q) = \begin{cases} |P \cap (Q + v_Q) \cap (R + v_R)| & \text{if } (v_Q, v_R) \in T_{PQR} \\ 0 & \text{otherwise.} \end{cases}$$

▶ **Proposition 25.** *Let S be any m-flat in the configuration space. In $O(n)$ time, we can find a point in $S \cap \operatorname{supp} f$, or report that $S \cap \operatorname{supp} f$ is empty.*

Proof. Notice that $\operatorname{supp} f$ is a convex 4-polytope whose face are hyperplanes of type I or type II. Let H be a hyperplane of type II intersecting the interior of T_{PQR}. Then H contains a face of $\operatorname{supp} f$ if and only if a polygon $P \cap Q$ is tangent to R in $H \cap T_{PQR}$. This can be tested in constant time, so we can find all faces of $\operatorname{supp} f$ in $O(n)$ time. Our problem become a feasibility test of a linear programming of size $O(n)$, which can be solved in $O(n)$ time by Megiddo's algorithm [13]. ◀

Proof of Theorem 2. Take T_{PQR} as in Proposition 24. Let

$$f(v_P, v_Q) = \begin{cases} |P \cap (Q + v_Q) \cap (R + v_R)| & \text{if } (v_Q, v_R) \in T_{PQR} \\ 0 & \text{otherwise.} \end{cases}$$

Then f is unimodal and the maximum of f is the goal placement. Given an m-flat S, we want to compute the maximum of f on S in $O(n \log^{m-1})$ time by induction on $m \in \{1, 2, 3, 4\}$.

If $m = 1$, this can be done in $O(n)$ time by Proposition 15. Assume that $m > 1$. Then $S \cap T_{PQR}$ can be computed in $O(1)$ time. Given an $(m-1)$-flat $l \subset S$, we can use Proposition 25 and the perturbation method as in Lemma 14 to report the relative position of the maximum over S. There are $O(n)$ event hyperplane intersecting $S \cap T_{PQR}$. Thus, by Lemma 9, we can recursively construct (1/2)-cuttings to give an $O(n \log^{m-1})$ time algorithm to find the maximum of f on S. ◀

6 Minimum symmetric difference of two convex polygons under homothety

A homothety $\varphi\colon \mathbb{R}^2 \to \mathbb{R}^2$ is a composition of a scaling and a translation. Let $\lambda > 0$ be the scaling factor and v be the translation vector of φ. Then

$$\varphi(A) = \lambda A + v = \{\lambda p + v \mid p \in A\}.$$

Define the *symmetric difference* of sets $A, B \subset \mathbb{R}^2$ to be

$$\begin{aligned} A \triangle B :=& (A \cup B) \setminus (A \cap B) \\ =& (A \setminus B) \cup (B \setminus A). \end{aligned}$$

Let P and Q be convex polygons with n vertices in total. We want to find a homothety φ of Q that minimizes the area of symmetric difference

$$h(\varphi) = h(x, y, \lambda) = |P \triangle \varphi(Q)|,$$

where $\varphi(Q) = \lambda Q + (x, y)$.

Yon et al. [14] consider a slightly more general problem, where they minimize the function

$$h(\varphi) = (2 - 2\kappa)|P \setminus \varphi(Q)| + 2\kappa|\varphi(Q) \setminus P|,$$

where $\kappa \in (0, 1)$ is some constant. When $\kappa = 1/2$, this is the area of symmetric difference function. They give a randomized algorithm that solves this problem in $O(n \log^3 n)$ expected time. We present a faster determinisitc algorithm by relating this problem to the polyhedron-polygon matching problem and then applying a modified version of Theorem 1.

As in [14], we rewrite the objective function $h(\varphi)$:

$$\begin{aligned} h(\varphi) &= 2(1 - \kappa)|P| + 2\kappa|\varphi(Q)| - 2|P \cap \varphi(Q)| \\ &= 2(1 - \kappa)|P| + 2\kappa|Q|\lambda^2 - 2|P \cap \varphi(Q)|. \end{aligned}$$

Thus, minimizing $h(\varphi)$ is the same as maximizing $f(\varphi) = |P \cap \varphi(Q)| - c\lambda^2$, where $c = \kappa|Q|$.

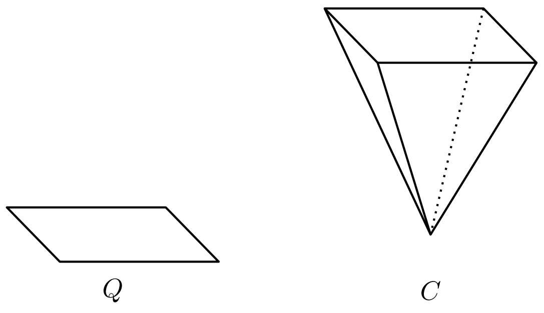


Figure 7 Formation of the cone C.

Consider the cone $C = \{(x, y, \lambda) | \lambda \in [0, M], (x, y) \in \lambda Q\}$, where $M = \sqrt{|P|/c}$ (see Figure 7). Then f is negative for $\lambda > M$ so it is never maximized. We also put P into $\mathbb{R}^3$ by $P = \{(x, y, 0) | (x, y) \in P\}$. Since $f(x, y, \lambda) = |C \cap (P + (-x, -y, \lambda))| - c\lambda^2$, the problem reduces to maximizing the overlap area of the cone C and P under translation subtracted by a quadratic function. To show that we can still use a divide-and-conquer strategy, we identify a region where f is strictly unimodal.

▶ **Lemma 26** ([14]). *The closure $\mathcal{D}$ of the set $\{\varphi \in \mathbb{R}^3 | f(\varphi) > 0\}$ is convex. Furthermore, $f(x, y, \lambda)$ is strictly unimodal on $\mathcal{D}$.*

Proof. This follows from [14, Lemma 2.2] and [14, Lemma 2.7]. ◀

Although it is difficult to directly compute $\mathcal{D}$, note that $-P \subset \mathcal{D}$. With this observation, we show that we can still find the relative position of the set of goal placements to certain planes S in $O(n \log n)$ time with some modifications to **LineDecision** and **PlaneDecision**.

▶ **Lemma 27.** *For any $l \subset \mathbb{R}^3$, we can compute $\max_{\varphi \in l} f(\varphi)$ or report it is a negative number in $O(n)$ time.*

▶ **Proposition 28.** *Let $S \subset \mathbb{R}^3$ be a plane. If S is horizontal or if S intersects the polygon $-P \subset \mathcal{D}$, then we can perform **PlaneDecision** on S in $O(n \log n)$ time.*

▶ **Theorem 29.** *Let P and Q be convex polygons with n vertices in total. Suppose $\kappa \in (0,1)$ is a constant. We can find a homothety φ that minimizes*

$$h(\varphi) = 2(1-\kappa)|P \setminus \varphi(Q)| + 2\kappa|\varphi(Q) \setminus P|$$

in $O(n \log^2 n)$ time.

Proof. We want to maximize $f(x, y, \lambda) = |C \cap (P + (-x, -y, \lambda))| - c\lambda^2$ over $\mathbb{R}^3$, where $c = \kappa|Q|$. In order to apply our algorithm for Theorem 1, we need to show that we only run **PlaneDecision** on horizontal planes and planes that intersect $-P$.

In the first stage (as outlined in Algorithm 4.1), we only run **PlaneDecision** on horizontal planes.

In the second stage, we apply Theorem 5 to the $O(n)$ groups of $O(n)$ lines that are the projections of the lines containing edges of event polygons on the xz-plane. Observe that these lines all intersect the projection of $-P$ on the xz-plane. In each recursive step of our algorithm, we query a horizontal (parallel to the x-axis) line and a line that goes "between" two lines in the $O(n^2)$ lines. The planes they represent both satisfy the condition for Proposition 28. Then we run **PlaneDecision** $O(1)$ more times to triangulate our feasible region. Here, we make a small modification: instead of maintaining a triangular feasible region, we maintain a trapezoidal one by making $O(1)$ horizontal cuts to make the region a trapezoid.

In the third stage, we have a "tube" and $O(n)$ event polygons that intersect it. As usual, we recursively construct a $(1/2)$-cutting by Lemma 9. Chazelle's algorithm [8] picks $O(1)$ planes intersecting the target region as the cutting, along with $O(1)$ extra planes to triangulate each piece. All the planes containing the event polygons intersect $-P$, so we can run **PlaneDecision** on them. Instead of triangulating our target region, it suffices to reduce it to constant complexity. We do this by cutting it with $O(1)$ horizontal planes such that the remaining region only has vertices on two levels. Then, let e be any non-horizontal edge. With $O(1)$ planes through e, we can cut the target region into prisms and pyramids with triangular bases. These planes all intersect $-P$ since they are between the two faces of the target region containing e, and the planes containing them intersect $-P$.

Therefore, with slight modifications to Theorem 1, we obtain a deterministic $O(n \log^2 n)$ algorithm for minimizing $h(\varphi)$. ◀

Theorem 3 follows as a direct corollary of Theorem 29.

References

1 Hee-Kap Ahn, Peter Brass, and Chan-Su Shin. Maximum overlap and minimum convex hull of two convex polyhedra under translations. *Comput. Geom.*, 40(2):171–177, 2008. `doi:10.1016/j.comgeo.2007.08.001`.

2 Hee-Kap Ahn, Siu-Wing Cheng, Hyuk Jun Kweon, and Juyoung Yon. Overlap of convex polytopes under rigid motion. *Comput. Geom.*, 47(1):15–24, 2014. `doi:10.1016/j.comgeo.2013.08.001`.

3 Hee-Kap Ahn, Siu-Wing Cheng, and Iris Reinbacher. Maximum overlap of convex polytopes under translation. *Comput. Geom.*, 46(5):552–565, 2013. `doi:10.1016/j.comgeo.2011.11.003`.

4 Hee-Kap Ahn, Otfried Cheong, Chong-Dae Park, Chan-Su Shin, and Antoine Vigneron. Maximizing the overlap of two planar convex sets under rigid motions. *Comput. Geom.*, 37(1):3–15, 2007. `doi:10.1016/j.comgeo.2006.01.005`.

5 David Avis, Prosenjit Bose, Thomas C. Shermer, Jack Snoeyink, Godfried Toussaint, and Binhai Zhu. On the sectional area of convex polytopes. In *Communication at the 12th Annu. ACM Sympos. Comput. Geom.*, page C. Association for Computing Machinery, New York, NY, 1996.

6 Mark de Berg, Olivier Devillers, Marc van Kreveld, Otfried Schwarzkopf, and Monique Teillaud. Computing the maximum overlap of two convex polygons under translations. In *International Symposium on Algorithms and Computation*, pages 126–135. Springer, 1996.

7 Bernard Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput.*, 21(4):671–696, 1992. `doi:10.1137/0221041`.

8 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993. `doi:10.1007/BF02189314`.

9 Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10(4):377–409, 1993. `doi:10.1007/BF02573985`.

10 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

11 Greg N. Frederickson and Donald B. Johnson. Generalized selection and ranking: sorted matrices. *SIAM J. Comput.*, 13(1):14–30, 1984. `doi:10.1137/0213002`.

12 Hyuk Jun Kweon and Honglin Zhu. Maximum overlap area of a convex polyhedron and a convex polygon under translation, 2023. `doi:10.48550/ARXIV.2301.02949`.

13 Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31(1):114–127, 1984. `doi:10.1145/2422.322418`.

14 Juyoung Yon, Sang Won Bae, Siu-Wing Cheng, Otfried Cheong, and Bryan T. Wilkinson. Approximating convex shapes with respect to symmetric difference under homotheties. In *32nd International Symposium on Computational Geometry*, volume 51 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 63, 15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.