On Computing All Immobilizing Grasps of a Simple Polygon with Few Contacts

Jae-Sook Cheong Herman J. Haverkort
A. Frank van der Stappen
Institute of Information and Computing Sciences,
Utrecht University, P.O.Box 80089, 3508 TB Utrecht, the Netherlands
{jaesook,herman,frankst}@cs.uu.nl

Abstract: We study the output-sensitive computations of all the combinations of the edges and vertices of a simple polygon P that allow a form closure grasp with less than four point contacts. More specifically, we present an $O(m^{\frac{4}{3}} \log^{\frac{1}{3}} m + K)$-time algorithm to compute all K pairs of concave vertices, an $O(n^2 \log^2 n + K)$-time and $O(m^2 \log^2 m + nm^{\frac{2}{3}} \log^{\frac{4}{3}} m + K)$-time algorithm to compute all K triples of one concave vertex and two edges, and two concave vertices and an edge respectively, where n is the number of edges, and m is the number of concave vertices of P. We also present an $O(n^2 \log^4 n + K)$-time algorithm that enumerates all the edge triples with a second-order immobility grasp using Czyzowicz's conditions in [4].

1 Introduction

Many applications such as robot hand grasping and manufacturing operations require an object to be *immobilized*, such that any motion of the object violates the rigidity of the object or the contacts.

An attractive theoretical model for immobility was formulated by Reuleaux [11] in 1876. He defines a rigid body to be in *form closure* if a set of contacts along its boundary constrains all finite and *infinitesimal* motions of the body. This notion is stronger than immobility, as for instance an equilateral triangle with a point contact in the middle of each edge is immobilized, but is not in form closure (it permits an infinitesimal rotation around its center). Form closure depends only on the position of the contacts and their normals, and is invariant with respect to the curvature of body and contacts. This is not true for immobility in general (if we replace the equilateral triangle in the example by its inscribed circle, contacts and normals remain identical, but the body is no longer immobilized). Markenscoff et al. [8] and Mishra et al. [9] independently showed that, with the exception of a circle, any two-dimensional body can be put in form closure with four frictionless point contacts, and that almost any three-dimensional body can be put in form closure with seven such contacts. We will call a configuration of frictionless point contacts that put an object in form closure a *form-closure grasp*.

We consider the problem of computing all form-closure grasps of a polygonal part with at most four frictionless point fingers. The availability of *all* grasps of a certain part allows a user–usually a machinist–to select the grasps that best meet specific additional requirements, for instance with respect to accessibility, which may vary from one operation to another. As the computation of all grasps along a given combination of edges and concave vertices can be accomplished in constant time [10, 16], the algorithmic challenge is to efficiently report all combinations of edges and vertices that yield at least one grasp.

An algorithm to compute, for a simple n-vertex polygon, all the edge combinations that have a form-closure grasp with four fingers was presented by van der Stappen et al. [16]. The algorithm runs in $O(n^{2+\epsilon} + K)$-time, where K is the number of edge quadruples reported. Brost and Goldberg [1] studied the same problem in modular settings, where the contact positions are restricted to a grid.

In general, four point contacts are required for planar objects, but sometimes form-closure grasps with fewer contacts are possible by using contacts in concave vertices of the object. Form-closure grasps with less than four fingers were first studied by Gopalakrishnan and Goldberg [7], who gave an $O(n^2)$-time algorithm to find all the concave vertex pairs that allow a two-finger form-closure grasp. In Section 2.1, we improve this to $O(m^{\frac{4}{3}} \log^{\frac{1}{3}} m + K)$, where m is the number of concave vertices of the polygon. Combinations of one concave vertex and two edges can be reported using the general algorithm by van der Stappen et al. We improve on that by presenting an $O(n^2 \log^2 n + K)$-time algorithm. Finally, we show how to report combinations of two concave vertices and one edge in time $O(m^2 \log^2 m + nm^{\frac{2}{3}} \log^{\frac{4}{3}} m + K)$-time.

In Section 3, we turn our attention away from form-closure to general immobility. Analogous to the above, we call a configuration of frictionless point contacts that immobilizes a rigid body an *immobility grasp*. Czyzowicz et al. [4] provided a necessary and sufficient geometric condition for a simple polygon without parallel edges to be immobilized by three point contacts. A more general analysis applicable to arbitrary objects was given by Rimon and Burdick [12, 13, 14], who define the term *second-order immobility*, as it not only takes position and normal, but also curvature of object, contacts, and possible motions into account.

An algorithm that reports, for a simple n-vertex polygon without parallel edges, all the edge triples that yield at least one immobility grasp was given by van der Stappen [16]. Its running time is $O(n^2 \log^2 n + K')$, where K' is the number of triples considered according to some criterion. This criterion is a necessary, but not sufficient condition, and so the algorithm may consider triples that do not yield immobility grasps. We resolve this shortcoming by giving a truly output-sensitive algorithm with a running time of $O(n^2 \log^4 n + K)$-time, where K is the number of edge triples yielding grasps satisfying Czyzowicz et al.'s necessary and sufficient conditions [4] for immobilization.

2 Projection of the wrenches of a polygon on a plane

A wrench in two-dimensional space is a vector (F_x, F_y, τ), where F_x, F_y is the force along the normal direction of the edge, and τ is the moment/torque. A set of wrenches that achieves form-closure satisfies a condition, stated in the following theorems: one in an algebraic form and another in a geometric form [6, 15, 9, 10].

Theorem 2.1 *For $k \geqslant 4$ for 2D and $k \geqslant 7$ for 3D, k wrenches $w_1, w_2, \cdots w_k$ achieve form closure if and only if there exists a set of k non-negative and non-trivial constants $\lambda_1, \lambda_2, \cdots \lambda_k$, such that $\lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_k w_k = w$, for any wrench w.*

Theorem 2.2 *For $k \geqslant 4$ for 2D and $k \geqslant 7$ for 3D, k wrenches $w_1, w_2, \cdots, w_k$ achieve form closure if and only if the origin lies strictly inside the convex hull of $w_1, w_2, \cdots, w_k$.*

Placing a point contact at a concave vertex gives us two wrenches, (w_1, w_2), because the contact touches two edges at the same time. Therefore, we must take advantage of concave vertices to achieve form closure with less than four point contacts. Computing all form closure grasps using a concave vertex is equivalent to the problem of finding all pairs of wrenches that satisfy the condition in Theorem 2.2 with a given wrench pair.

First we introduce some notation. If a ray from the origin O in the direction of a wrench w hits a plane Π, the intersection of Π and the ray is called the projection of w. The reflection of w is the intersection point of Π and a ray from O in the opposite direction of w. (See Figure 1 (a).) We call the projection of a wrench induced by a point contact on a polygon edge *edge wrench*. The edge wrenches of a simple polygon P are a collection of vertical line segments in the wrench space, so the projection of them will be vertical line segments as in Figure 1 (b). Sometimes, we will refer this projection of the set of edge wrenches for an edge *edge wrench line segment*.

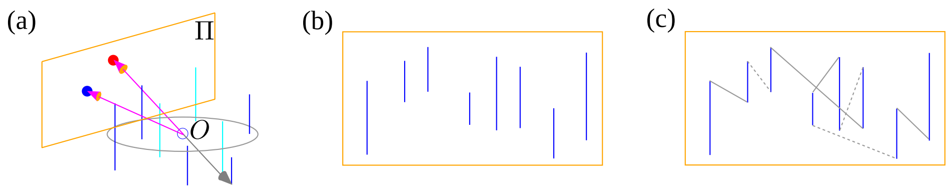
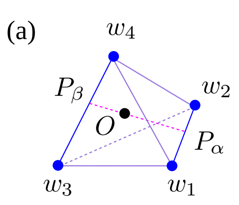


Figure 1: (a) Edge wrenches in the wrench space and a projection and a reflection of two wrenches on a plane. (b) A projection of the wrenches of polygon edges. (c) A projection of the concave vertex line segments in solid line, and convex vertex line segments in dotted line.

When we move from one edge to the next, the wrench will suddenly jumps from one edge wrench to the next one as well, because the normal direction suddenly jump. Naturally this jumping occurs only at vertices, and it can be represented as a line segment connecting the previous edge wrench and the next one. The connecting line for concave and convex vertices are called concave and convex line segments respectively. Figure 1 (c) shows the edge wrenches with concave and convex line segments in solid and in dotted lines respectively. Note that a vertex line segment does not represent the range of the wrench from one edge wrench to the next. (In the wrench space, the range of the wrenches from one edge to the next is a sine curve.) Also note that the order of the edge wrenches in Figure 1 (c) is not monotone with respect to a horizontal line.

Lemma 2.1 *Let w_1, w_2, w_3, w_4 be four edge wrenches for a simple polygon P. Two wrenches w_3 and w_4 with a given edge wrenches w_1 and w_2 form a tetrahedron that contains the origin strictly inside, if and only if the projection of $\overline{w_1 w_2}$ and the reflection of $\overline{w_3 w_4}$ onto a plane intersect at an interior point.*

Proof: $(\Rightarrow)$ Suppose that the origin O lies strictly inside the tetrahedron formed by w_1, w_2, w_3 and w_4. By Theorem 2.1, O can be expressed as follows: $\lambda_1 w_1 + \lambda_2 w_2 + \lambda_3 w_3 + \lambda_4 w_4 = 0$, where $\lambda_i > 0, (1 \leqslant i \leqslant 4)$. This can be rewritten as $\lambda_1 w_1 + \lambda_2 w_2 = -\lambda_3 w_3 - \lambda_4 w_4$, which implies that there exist two points P_α and P_β in the middle of $\overline{w_1 w_2}$ and $\overline{w_3 w_4}$ respectively, such that P_α, P_β and O are collinear as in Figure 2 (a). Hence the projection of P_α and the reflection of P_β on a plane Π are one point on Π, which is the intersection of the projection of $\overline{w_1 w_2}$ and reflection of $\overline{w_3 w_4}$. (See Figure 2 (b).) Note that the intersection is in the interior of $\overline{w_1 w_2}$ and $\overline{w_3 w_4}$, because O is strictly inside the tetrahedron.



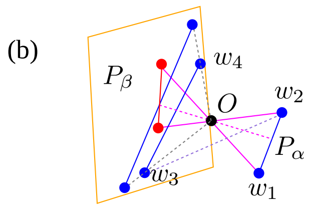


Figure 2: (a) A description of Lemma 2.1. (b) The origin O lies on a line $\overline{P_\alpha P_\beta}$, where P_α is on $\overline{w_1 w_2}$ and P_β on $\overline{w_3 w_4}$.

$(\Leftarrow)$ Suppose that the projected and reflected line segments intersect at an interior point. If we walk from the intersection point p along the line $\rightarrow pO$, we hit $\overline{w_3 w_4}$, then O, and then $\overline{w_1 w_2}$. The origin O is not on $\overline{w_3 w_4}$, if it does, it implies that the edges corresponding to w_3 and w_4 touch each other, which contradicts that P is a simple polygon. Thus O is strictly inside the tetrahedron of w_1, w_2, w_3 and w_4. ◻

We color the projections of the wrenches blue, and the reflections red. Because one plane cannot have all the projections of the wrenches, we use three pairwise non-parallel planes perpendicular to the $F_x F_y$-plane. Now we have a red and blue line arrangement on each of the three planes. According to Lemma 2.1, the intersections between one red and one blue line segments should be reported. To report these intersections efficiently, we do not want to report the intersections between blue segments or between red segments. Theorem 2.3 will be used to solve Red-Blue intersection problem. Throughout this section, n denotes the number of the edges of a polygon P, m denote the number of concave vertices, and K denotes the output size.

Theorem 2.3 *[2] There is an algorithm to report all the intersecting line segments for a given query line segment. The query time is $O(n^{\frac{1}{3}} \log^{\frac{1}{3}} n + k)$, and preprocessing time is $O(n \log n)$, where n is the number of the line segments, and k is the output size.*

2.1 Form closure with pairs of concave vertices

We study the problem of reporting all pairs of concave vertices which allow a form closure grasp by placing two frictionless point contacts at the concave vertices. The algorithm works as follows. Identify the concave vertex line segments, and place the red projections and the blue reflections on the planes.

There are m blue and m red line segments. Using Theorem 2.3, we query the intersecting blue line segments for each of the red line segments. The preprocessing time is $O(m \log m)$, and it runs in $O(m^{\frac{4}{3}} \log^{\frac{1}{3}} m + K)$-time to report K Red-Blue intersections. The output size K can be $\Omega(n^2)$, but it can also be zero.

Theorem 2.4 *All form closure grasps on a polygon with two concave vertices can be computed in $O(m^{\frac{4}{3}} \log^{\frac{1}{3}} m + K)$ time.*

When the polygon is rectilinear, all the pairs of concave vertices with form closure configurations can be enumerated in $O(m \log^2 m + K)$ time, using orthogonal range searching trees.

2.2 Form closure with triples of one concave vertex and two edges

Placing three frictionless point contacts at a concave vertex v_c and the interior of two edges e, e' can achieve form closure. In this section, we given an algorithm to report all such triples of (v_c, e, e'). One difference from Lemma 2.1 is that two wrenches w_3 and w_4 can be chosen from each of the edge wrenches for e and e'.

Lemma 2.2 *One concave vertex and two edges have a form closure grasp if and only if the projections of the quadrilateral formed by two edge wrenches for e and e', and the reflection of the concave vertex line segment intersect each other in the interior.*

The quadrilateral defined by two edge wrenches for e, e' is the collection of the line segments $\overline{w_1 w_2}$, where w_1 is in the interior of the edge wrenches for e, and w_2 in the interior of those for e'. To report all the triples of (v_c, e, e'), we first make a blue quadrilateral with the edge wrenches for each pair of edges. Then we reflect the concave vertex line segments, and color them red. Figure 3 (c) shows an arrangement of the red line segments and the blue quadrilaterals. A blue quadrilateral intersecting a red line segment in the interior make a triple (v_c, e, e').

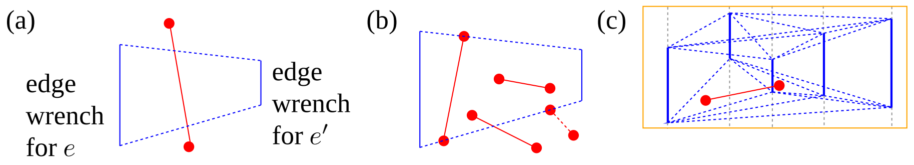


Figure 3: (a) Both the endpoints lie strictly outside of the quadrilateral. (b) At least one endpoint lies in the quadrilateral including its boundaries. (c) An arrangement of blue quadrilaterals and red line segments.

There are two cases of a red line segment intersecting a blue quadrilateral in the interior: (i) when both the endpoints lie strictly outside of the quadrilateral, and (ii) when at least one endpoint of the line segment is in the quadrilateral including its boundaries. (See Figure 1 (a) and (b).) A red line segment of case (i) intersects at least one side of the blue quadrilateral in the interior. Such pairs can be reported by the Red-Blue line segment intersection algorithm. A red line segment of case (ii) has at least one endpoint or its midpoint inside the quadrilateral. Each quadrilateral can be divided into two triangles, and we use a multi-level data structure with segment trees and line segment intersection data structures in [2] to report all the triangles that contain a given point inside. The endpoints and the midpoints of the red line segments are the query points. Remember that the query point should be strictly inside of the triangles, but it can lie on the diagonal of the quadrilateral. This way, the algorithm reports all the quadrilateral intersecting a line segment two endpoints of which are on its boundaries, but nothing for the line segment that intersects a quadrilateral only on the side. (See Figure 3 (b).)

There are $O(n^2)$ quadrilaterals, but there are $O(n)$ vertical lines from the endpoints of the triangles, thus $O(n)$ vertical slabs. These slabs can be stored in a segment tree with line segment intersection data structures at the second level that store the duals of the triangle boundary lines. The construction time for all cases is $O(n^2 \log^2 n)$, and the query time for each of the $O(m)$ queries is $O(n^{\frac{2}{3}} \log^{\frac{1}{3}} n + \log n \times n^{\frac{2}{3}} \log^{\frac{1}{3}} n + k) = O(n^{\frac{2}{3}} \log^{\frac{4}{3}} n + k)$. The total time to report all triples of (v_c, e, e') is $O(n^2 \log^2 n + K)$. There exist polygons for which K is $O(n^3)$, and for which K is $O(n^2)$.

Theorem 2.5 *All form closure grasps on a polygon with one concave vertex and two edges can be computed in $O(n^2 \log^2 n + K)$-time.*

When the polygon is rectilinear, all the triples of (v_c, e, e') that can achieve form closure can be enumerated in $O(m \log n + K)$ time, using orthogonal range searching trees.

2.3 Form closure with triples of two concave vertices and one edge

Placing two frictionless point contacts at a pair of concave vertices v_c, v'_c may not achieve form closure. Placing one more point contact in the interior of an appropriate edge e, however, can achieve form closure with v_c and v'_c. In this section, we study the problem of reporting all triples of (v_c, v'_c, e) with a form closure grasp.

Lemma 2.3 *Two concave vertices and one edge have a form closure grasp if and only if the reflection of the edge wrenches for e and the convex hull of the two projected concave vertex line segments intersect each other in the interior.*

Proof: $(\Rightarrow)$ Suppose that a grasp at v_c, v'_c and at edge e achieves form closure. Let (w_1, w_2) and (w_3, w_4) be the wrenches induced by v_c and v'_c, and w_5 be the wrench induced by a contact at e. By Theorem 2.1, there exists a wrench w_5 at the edge wrench for e such that $w_5 = \frac{-1}{\lambda_5}(\lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_4 w_4)$ for some non-negative and non-trivial constants $\lambda_1, \lambda_2, \cdots \lambda_5$. A similar argument as in Lemma 2.1 says that the reflection of the edge wrench intersect the quadrilateral of the projections of $w_1, \cdots, w_4$ in the interior. (See Figure 4 (a).)

$(\Leftarrow)$ Let p be an intersection point in the interior of the blue quadrilateral and the red lien segment. The line from p to O will hit the interior of one of the triangular regions formed by $w_1, \cdots, w_4$, then O, and then the interior of the edge wrench for e. This implies that O is strictly inside the convex hull of $w_1, \cdots, w_5$. $\square$

With Lemma 2.3, we have the following algorithm. Project the concave vertex line segments on the planes. Compute a convex hull for each pair of the line segments, and color them blue. Reflect the edge wrenches on the planes, and color them red. Report all the red-blue pairs that have intersections in the interior, as in Section 2.2.

We have $O(m^2)$ triangles, $O(m)$ slabs, and $O(n)$ query line segments, so the construction time for both data structures is bounded by $O(m^2 \log^2 m)$, and the query time in both data structure for each of the $O(n)$ red linesegments is bounded by

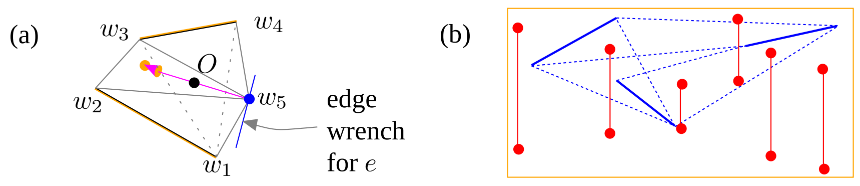


Figure 4: (a) The convex hull of $w_1, \cdots, w_5$ contains O strictly inside. (b) An arrangement of the blue convex hulls and the red line segments.

$O(\log m \times m^{\frac{2}{3}} \log^{\frac{1}{3}} m + k)$. Thus the total time to report all the form closure triples is $O(m^2 \log^2 m + nm^{\frac{2}{3}} \log^{\frac{4}{3}} m + K)$. The output size K can be $\Omega(n^3)$, but it can also be $O(m^2)$.

Theorem 2.6 *All triples of one concave vertex and two edges of a simple polygon with a form closure grasp can be reported in $O(m^2 \log^2 m + nm^{\frac{2}{3}} \log^{\frac{4}{3}} m + K)$-time.*

When the polygon is rectilinear, all the triples of (v_c, v_c', e) with a form closure grasp can be enumerated in $O(n \log^2 m + K)$ time, using orthogonal range searching trees.

2.4 Extension

Wentink [17] studied the problems of immobilizing a polygon not only with point contacts but also with edge contacts. She provided a $O(n^2 \log^2 n + K)$-time algorithms to enumerate all form closure configurations of one edge contact and two point contacts and of a fixed or adjustable angle contacts and one point contact. Using Lemma 2.1, this algorithm can also be improved to $O(n \log n + hn^{\frac{1}{3}} \log^{\frac{1}{3}} n + K)$-time, where h is the number of edges of the convex hull, and K is the number of pairs consisting of one convex hull edge E and one edge e of the polygon, such that a form closure configuration exists with an fixed or adjustable angle contact along E and a point contact at e.

3 All edge triples that immobilize a simple polygon

We first introduce some notations and definitions used in this section. Facing the interior of a simple polygon P from an edge e, the normals at the left and the right endpoints of e are denoted as $s_l(e)$ and $s_r(e)$ respectively. The set of normal lines of e between $s_l(e)$ and $s_r(e)$ is denoted as $\hat{s}(e)$. Since placing a contact at vertices may damage the part, vertices are not generally accepted as contact positions. If contacts are allowed to be at concave vertices, the corresponding slab boundaries should be included in the normal slabs. In this section, however, we do not include slab boundaries in normal slabs.

Let $l(e)$ be the supporting line of an edge e, and let $H(e)$ be an open half-plane bounded by $l(e)$, which locally contains the interior of P. (See Figure 5.) When the intersection of $H(e), H(e')$ and $H(e'')$ forms a triangle, then e, e', e'' are said to be a *triangular triple*. (Compare Figure 5 (a) with (b).) A necessary and sufficient condition for three edges to have a configuration of three point contacts to immobilize a simple polygon is provided by Czyzowicz et al. [4].

Lemma 3.1 *Three point contacts along three edges e, e', e'' can immobilize a two-dimensional polygon if and only if:*
1. $\hat{s}(e) \cap \hat{s}(e') \cap \hat{s}(e'') \neq \emptyset$ (common intersection condition) and
2. $H(e) \cap H(e') \cap H(e'')$ is a triangle (triangular triple condition).

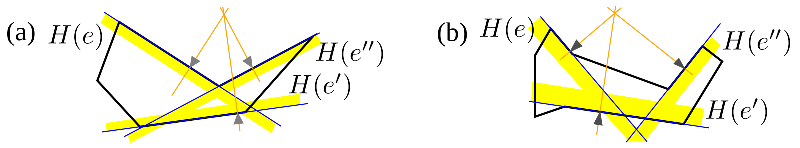


Figure 5: The edges e, e', e'' in (a) are a triangle triple, while those in (b) are not.

We take a similar approach as in [16]. Interval trees and orthogonal range search trees (see [5]) will be used to find all the edge triples that have a common normal intersection, and among these, triangular triples will be filtered out using orthogonal range search trees and convex layer structures (see [3]). These data structures will be combined as a multi-level data structure.

The sketch of the global approach is as follows. A data structure will be built to store the information of all the edges assuming each directed edge e as the positive x-axis, so $O(n)$ data structures will be built. In each data structure, we query with each edge e' of $n-1$ edges to find all the edges that satisfy the conditions in Lemma 3.1 with e and e'.

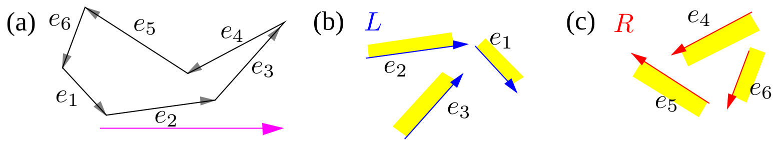


Figure 6: (a) A polygon with directed edges. (b) The edges in L, and (c) in R.

We first give each edge a direction so that the interior of the edge lies on the left. (See Figure 6 (a).) Further, we divide the edges into two groups L and R; when an edge forms an angle between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ with the positive x-axis, it will be in L; otherwise, in R. (See Figure 6 (b) and (c).) *Top boundary* is the slab boundary lying above the other among $s_l(e')$ and $s_r(e')$ of $\hat{s}(e')$. (See Figure 7.)

From now on, we focus in the case when the geometric information of all the edges are described with a directed edge e as the positive x-axis, unless it is stated otherwise. Now we present a necessary and sufficient condition for three edges to have a non-empty common normal intersection region. Figure 7 shows a visual description of the following condition.

Condition 3.1 *[16] Two normal slabs $\hat{s}(e')$ and $\hat{s}(e'')$ have a common normal intersection with $\hat{s}(e)$ if and only if:*
(i) when the open intervals of $s_l(e) \cap \hat{s}(e')$ and $s_l(e) \cap \hat{s}(e'')$ overlap, or
(ii) when the open intervals of $s_r(e) \cap \hat{s}(e')$ and $s_r(e) \cap \hat{s}(e'')$ overlap, or
(iii) the top boundaries of $\hat{s}(e')$ and $\hat{s}(e'')$ intersect in $\hat{s}(e)$.

In cases (i) and (ii), an interval tree is used to identify all the intervals that overlap a given interval $s_l(e) \cap \hat{s}(e')$ or $s_r(e) \cap \hat{s}(e')$. In case (iii), observe that the order of

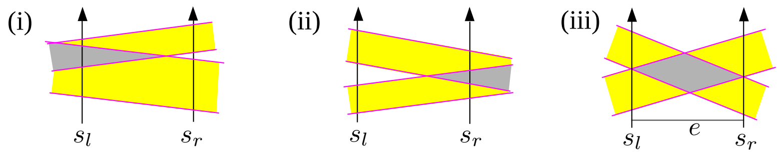


Figure 7: Three cases of three normal slabs having common normal intersection.

the top boundary positions at $s_l(e)$ and $s_r(e)$ are swapped. Let $x(e')$ and $y(e')$ be the y-coordinate of the intersection point at $s_l(e)$ and $s_r(e)$ respectively, $\hat{s}(e')$, and $x(e''), y(e'')$ are defined likewise. The top boundaries intersect each other in $\hat{s}(e)$, if $((x' \leqslant x'') \wedge (y' \geqslant y''))$, or if $((x' \geqslant x'') \wedge (y' \leqslant y''))$. All the edges satisfying the condition (iii) with e' can be reported using a two-dimensional orthogonal range search tree. Thus we have the edges that have a common normal intersection with e and e'.

Among these edges, the triangular triples will be filtered out in the dual space. Let a' be the slope of $l(e')$, and q be $l(e) \cap l(e')$. Remember that the directed line $l(e)$ is the positive x-axis now. The slope a' can be either positive or negative. In each case, there are only two different directions of $l(e')$, i.e., $l(e')$ is from L or R. Thus $l(e)$ and $l(e')$ make four cases as in Figure 8.

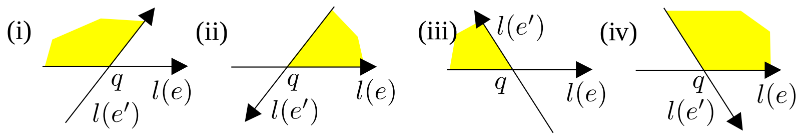


Figure 8: (i) $(a' > 0) \wedge (l(e') \in L)$, (ii) $(a' > 0) \wedge (l(e') \in R)$, (iii) $(a' < 0) \wedge (l(e') \in L)$, and (iv) $(a' < 0) \wedge (l(e') \in R)$. The intersection of $H(e)$ and $H(e')$ is shaded.

In each case, the queries to find the triangular triples are as follows. In cases (i) and (iv), we find all the lines in R, which are above q, and the slopes of which are between 0 and a', and in case (ii), we find all the lines in R, which are above q, and the slopes of which are smaller than 0, and find all the lines in L, which are below q, and the slopes of which are greater than a'. Finally in case (iii), we find all the lines in R, which are above q and the slopes of which are greater than 0, and find all the lines in L, which are below q and the slopes of which are smaller than a'. Figure 9 shows the query regions in the dual space of $l(e'')$'s, where the dual of $q(a_q, b_q)$ is a line $q^* : a_q x + b_q$. The dual points of L and R are first stored in one-dimensional orthogonal range search trees according to the slopes, and the points at each node are stored in a convex layer structure for half plane queries. This concludes the algorithm.

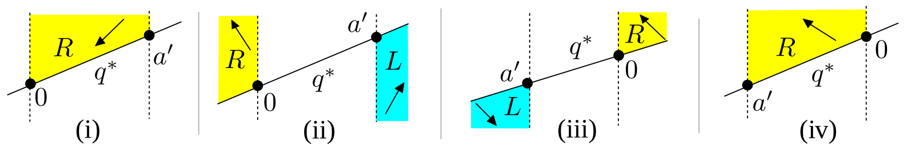


Figure 9: The query regions in the dual space.

Now we analyze the time complexity of this algorithm. Case (i) and (ii) use one-dimensional interval trees with one-dimensional range search trees (for L and R) at the second level, and a convex layer structure at the third level. Case (iii) uses a three-dimensional range search trees with a convex layer structure at the fourth level.

One-dimensional interval and range search trees, and convex layer structures use $O(n)$ storage, have $O(n \log n)$ construction time, and $O(\log n)$ query time. Three-dimensional range search trees use $O(n \log^2 n)$ storage, have $O(n \log^2 n)$ construction time, and $O(\log n^3)$ query time. In total, case (i) and (ii) use $O(n \log n)$ storage, and has $O(n \log n + n \log n \log(n \log n)) = O(n \log^2 n)$ construction time, and $O(\log n^3 + K)$ query time. On the other hand, case (iii) use $O(n \log^2 n)$ storage, has $O(n \log^2 n + n \log^2 n \log(n \log^2 n)) = O(n \log^3 n)$ construction time, and $O(\log n^4 + K)$ query time in total. These multi-level data structures are built for each edge, so the whole algorithm takes $O(n \log^2 n)$ storage, and runs in $O(n^2 \log^4 n + K)$ time.

Theorem 3.1 *All K edge triples (e, e', e'') of a simple polygon such that $\hat{s}(e) \cap \hat{s}(e') \cap \hat{s}(e'') \neq \emptyset$ and $H(e) \cap H(e') \cap H(e'')$ is a triangle can be reported in $O(n^2 \log^4 n + K)$.*

4 Conclusion

We have studied the problem of computing all the combinations of edges and vertices of a simple polygon that allow a form closure grasp with less than four point contacts. Using the red-blue intersection algorithm and the point intersection searching algorithm, we have proposed an $O(m^{\frac{4}{3}} \log^{\frac{1}{3}} m + K)$-time algorithm for all K pairs of concave vertices, $O(n^2 \log^2 n + K)$-time algorithm for all K triples of one concave vertex and two edges, and an $O(m^2 \log^2 m + nm^{\frac{2}{3}} \log^{\frac{4}{3}} m + K)$-time algorithm for all K triples of two concave vertices and an edge, where n is the number of edges, and m is the number of concave vertices of a simple polygon. An $O(n^2 \log^4 + K)$-time algorithm is proposed to compute all K edge triples with a second-order immobility grasp for a simple polygon, using the condition in [4].

References

[1] R. Brost and K. Goldberg. A complete algorithm for designing planar fixtures using modular components. In *IEEE Transactions on Robotics and Automation*, volume 12, pages 31–46, 1996.

[2] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9(1):145–158, 1993.

[3] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. In *IEEE Symposium on Foundations of Computer Science*, pages 217–225, 1983.

[4] J. Czyzowicz, I. Stojmenovic, and J. Urrutia. Immobilizing a shape. *International Journal of Computational Geometry and Applications*, 9(2):181–206, 1999.

[5] M. de Berg, M. van Kreveld, M. Overmars, , and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997. Heidelberg.

[6] A.J. Goldman and A.W. Tucker. Polyhedral convex cones. *Linear Inequalities and Related Systems*, pages 19–40, 1956.

[7] Gopal Gopalakrishnan and Ken Goldberg. Gripping parts at concave vertices. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1590–1596, May 2002.

[8] X. Markenscoff, L. Ni, and C. H. Papadimitriou. The geometry of grasping. *International Journal of Robotics Research*, 9(1):61–74, 1990.

[9] B. Mishra, J. T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2:541–558, 1987.

[10] V-D. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3):3–16, 1988.

[11] F. Reuleaux. *The Kinematics of Machinery*. Macmilly and Company, 1876. Republished by Dover in 1963.

[12] E. Rimon and J. W. Burdick. New bounds on the number of frictionless fingers required to immobilize planar objects. *J. of Robotic Systems*, 12(6):433–451, 1995.

[13] E. Rimon and J. W. Burdick. Mobility of bodies in contact—part I: A second-order mobility index for multiple-finger graps. *IEEE Transactions on Robotics and Automation*, 14:696–708, 1998.

[14] E. Rimon and J. W. Burdick. Mobility of bodies in contact—part II: How forces are generated by curvature effects. *IEEE Transactions on Robotics and Automation*, 14:709–717, 1998.

[15] K. Salisbury. *Kinematic and force analysis of articulated hands*. PhD thesis, Standford University, 1982.

[16] A. F. van der Stappen, C. Wentink, and M. H. Overmars. Computing immobilizing grasps of polygonal parts. *International Journal of Robotics Research*, 19(5):467–479, 2000.

[17] C. Wentink. *Fixture Planning—Geometry and Algorithms*. PhD thesis, Department of Computer Science, Utrecht University, 1998.