Computing immobilizing grasps of polygonal parts

A. F. van der Stappen,

C. Wentink,

M. H. Overmars

UU-CS-1999-14

Computing Immobilizing Grasps of Polygonal Parts[*]

A. Frank van der Stappen Chantal Wentink Mark H. Overmars

Department of Computer Science, Utrecht University,
P.O.Box 80089, 3508 TB Utrecht, the Netherlands.

Abstract

We present the first algorithms for computing *all* placements of (frictionless) point fingers that put a polygonal part in form closure and *all* placements of point fingers that achieve 2nd-order immobility of a polygonal part. Our algorithms run in $O(n^{2+\epsilon}+K)$ and $O(n^2 \log^2 n+K)$ time in the case of form closure and 2nd-order immobility respectively, where n is the number of vertices of the polygon, K is the description size of the resulting set of finger placements, and ϵ is an arbitrarily small constant. The basis of our algorithm is a translation of the problem into geometric searching problems, which are solved using efficient data structures. Our results can be extended to the problem of computing all placements of a line and two points that put a polygonal part in form closure. The resulting algorithm runs in $O(n^2 \log^2 n + K)$ time, where K is again the description size of the output.

1 Introduction

Many manufacturing operations, such as machining, assembly, and inspection, require objects to be fixtured, that is, to be held in such a way that they can resist all external wrenches. The concept of *form closure*, formulated by Reuleaux [13] in 1876, provides a sufficient condition for constraining, despite the application of an external wrench, all finite and infinitesimal motions of a rigid object by a set of contacts along its boundary. Any motion of an object in form closure has to violate the rigidity of the contacts. Markenscoff et al. [4] (and Mishra et al. [7]) showed that—in the absence of friction—four points are necessary and sufficient to put almost any planar object in form closure. We refer to a specific placement of the contacts as a *grasp*. A grasp that puts a part in form closure is referred to as a *form-closure grasp*.

Results by Rimon and Burdick [14, 15] show that form closure is not a necessary condition for constraining all motions of an object. Three points turn out to be sufficient to achieve what they call *2nd-order immobility* of an object; the corresponding analysis of a placement of three points takes the curvature of the object boundary into account. We refer to a grasp that achieves 2nd-order immoblity as a *2nd-order-immobility grasp*. Unlike with form closure (which is a 1st-order notion of immobility in the terminology of [14]), a small arbitrary shift along the object boundary of one or more of the three points is unlikely to keep an object 2nd-order immobilized. As a result, the three points have to be placed with infinite precision.

We consider the computation of all form-closure grasps of a polygonal part P with (at most) four frictionless point fingers and of all 2nd-order-immobility grasps of P with (at most) three frictionless point fingers. The availability of all form-closure or 2nd-order-immobility grasps allows a user—usually a machinist—to select the grasps that best meet specific additional requirements (e.g. with respect to accessability), which may vary from one operation to another. We shall use the common assumption that the placement of the frictionless point fingers is restricted to interiors of edges or endpoints of edges that are concave vertices of P. The continuous nature of the sets of form-closure or 2nd-order-immobility grasps makes their computation difficult. Our algorithms

[*]A preliminary extended abstract of this paper appeared as [16].

are the first to solve the problem of computing these sets efficiently. The running times of our algorithms depend to a large extent on the size of their outputs; they run in $O(n^{2+\epsilon} + K)$ time in the case of form closure and in $O(n^2 \log^2 n + K)$ time in the case of 2nd-order immoblity, where ϵ is an arbitrarily small positive constant and K is the description size—or complexity for short—of the set all grasps. The value of K is trivially bounded by $O(n^4)$ and $O(n^3)$ respectively, but is smaller in most practical cases.

Ponce et al. [10, 11, 12] studied the computation of form-closure grasps of two-dimensional and three-dimensional polyhedral objects in the context of point fingers with friction. The essential difference with our work lies in the number of fingers involved in the grasps. Whereas they computed three-finger grasps of two-dimensional objects and four-finger grasps of polyhedra we must compute all form-closure and 2nd-order-immobility grasps of polygons with at most four and three fingers respectively, simply because these numbers of fingers may be necessary to obtain form closure or 2nd-order immobility in the absence of friction. Moreover, the running times of our algorithms are largely determined by the size of their outputs. Other papers study the problem of computing the single optimal grasp according to some quality criterion [3, 5, 6].

A related piece of work is a paper by Nguyen [8]. The algorithm outlined in that paper computes a subset of the set of all form-closure grasps. This subset consists of all quadruples of parts of edges—to which we will refer as *Nguyen regions*—such that any placement of a point on each of these four parts will put P in form closure. Nguyen's algorithm runs in $O(n^4)$. We point out later how a minor modification of our algorithm makes it suitable for the computation of Nguyen regions. The resulting computation also takes $O(n^{2+\epsilon} + K)$, where K is now the number of Nguyen regions. The difference in running times between Nguyen's algorithm and our output-sensitive algorithm becomes evident when we realize that in most cases the complexity of the set of all form-closure grasps and the set of Nguyen regions remains well below the obvious upper bound of $O(n^4)$. When the set of solutions is small, our algorithm runs in almost-quadratic time while Nguyen's algorithm still takes $O(n^4)$ time.

A related topic that received considerable attention is the computation of all *modular* form-closure and 2nd-order-immobility grasps. In the modular setting, the placement of the fingers is restricted to a regular orthogonal grid of holes. Brost and Goldberg [2] gave an algorithm for computing all form-closure grasps with three point fingers that are placed at grid holes and a fourth finger that is placed on a horizontal or vertical line through these holes. Their algorithm for this modular setting runs in $O(n^4 d^5)$ time, where d is the diameter of the part in grid units. Other results on the computation of modular grasps are reported in [9, 17, 18, 19, 20, 21, 22].

The key feature of our approach to solving the problem of computing all 2nd-order-immobility and form-closure grasps is that we first identify all triples or quadruples of edges that induce at least one 2nd-order-immobility or form-closure grasp. (Note that trying all triples and quadruples would inevitably lead to $\Omega(n^3)$ and $\Omega(n^4)$ algorithms.) We transform the problem of identifying all such triples and quadruples into a number of geometric searching problems. These problems are then solved efficiently by applying data structures [1] from the field of computational geometry. For each of the reported triples or quadruples we can compute the (non-empty) set of induced 2nd-order-immobility grasps and form-closure grasps in constant time.

Overmars et al. [9] (see also [21]) proposed to extend the set of contacts for holding parts with line contacts (or walls), and studied the computation of form-closure grasps involving one line and two points in a modular setting. We extend our results for four point contacts to the case of one line and two points, obtaining an $O(n^2 \log^2 n + K)$ algorithm, where K is again the output size. The value of K is trivially bounded by $O(n^3)$, but is smaller in most practical cases.

This paper is organized as follows. Section 2 discusses the notions of form closure and 2nd-order immobility. In Section 3, we use the properties of form-closure grasps to compute all such grasps with four, three, and two point fingers. In Section 4, we use the properties of 2nd-order-immobility grasps and the results from Section 3 to tackle the problem of computing all 2nd-order-immobility grasps with three and two point fingers. Section 5 extends the results from Section 3 to form-closure grasps with a line and to point fingers. Section 6 concludes the paper.

2 Analysis of grasps

We are given a polygonal part P and a set of points (and lines), and we want to determine all placements of these points (and lines) in contact with the boundary of P such that these frictionless contacts achieve form closure or 2nd-order immobility. Analysis of these placements—or grasps—can be performed in several ways. An intuitive method for form-closure analysis [13] considers grasps in the two-dimensional space of the part itself. Its graphical nature will allow us to translate the problem of finding all form-closure grasps into a geometric searching problem. A similar translation can also be made for 2nd-order-immobility grasps. Before we outline the graphical method for point contacts and its extension to line contacts, we first introduce the notation used throughout the paper.

2.1 Definitions

We define n to be the number of edges (or vertices) of the polygon P. Recall that the point fingers must be placed at edge interiors or endpoints that are concave vertices of P. For the sake of simplicity, we treat concave vertices as part of their incident edges. Convex vertices do not belong to their incident edges. As a result, edges are closed at endpoints that are concave vertices of P and open at endpoints that are convex vertices of P. Let a be a point on an edge e of P. We denote by $l(a)$ be the directed line through a, perpendicular to e, and directed towards the interior of P. The half-line $l^+(a)$ is the part of $l(a)$ extending towards the interior of P from (but not including) a; $l^-(a)$ is the other (closed) part of $l(a)$. For an edge e, we define $slab(e)$ to be the slab directed towards the interior of P defined by the union of all lines $l(a)$ with a on e. Note that the boundaries of $slab(e)$ may or may not belong to $slab(e)$, depending on the types of the endpoints of e. The half-slab $slab^+(e)$ is the part of $slab(e)$ extending towards the interior of P from (but not including) e, or in other words the union of all $l^+(a)$, with a on e.

A set of (direction) vectors $v_1, \ldots, v_k$ *positively spans* $\mathbb{R}^2$ if any vector $v \in \mathbb{R}^2$ can be written as $v = \Sigma_{i=1}^k \alpha_i v_i$, where all α_i are non-negative scalars. (Note that k should be at least 3.)

We define $\mathcal{T}(P)$ to be the set of all triples (e_1, e_2, e_3) of edges of P with the property that the directions of $slab(e_1)$, $slab(e_2)$, and $slab(e_3)$ positively span $\mathbb{R}^2$. The set $\mathcal{T}(P)$ consists of two disjoint subsets $\mathcal{T}_0(P)$ and $\mathcal{T}_1(P)$: $\mathcal{T}_0(P) = \{(e_1, e_2, e_3) \in \mathcal{T}(P) | slab(e_1) \cap slab(e_2) \cap slab(e_3) = \emptyset\}$ and $\mathcal{T}_1(P) = \{(e_1, e_2, e_3) \in \mathcal{T}(P) | slab(e_1) \cap slab(e_2) \cap slab(e_3) \neq \emptyset\}$. Finally, we let $\mathcal{T}_1^+(P) = \{(e_1, e_2, e_3) \in \mathcal{T}(P) | slab^+(e_1) \cap slab(e_2) \cap slab(e_3) \neq \emptyset\}$.

2.2 Form closure and 2nd-order immobility

Our objective is to constrain all infinitesimal (and thus also finite) motions of the polygonal part P by a collection of fingers. Form closure and 2nd-order immobility are different conditions that both guarantee that no motion of the part is possible.

We examine what infinitesimal motions are ruled out by a point finger—or point contact—at a point a on an edge e of P. Forces and torques applied to P will make it translate or rotate. We confine ourselves to rotations, with the understanding that a translation can be regarded as a rotation about a point at infinity. In this manner, an infinitesimal motion of P can be represented by a point in the plane and a sign, being the center-of-rotation and the direction of the rotation respectively. A negative (or clockwise) rotation about a point in the half-plane left of the directed line $l(a)$ causes the point finger at a to enter P. Hence, such a motion is impossible. A positive (or counterclockwise) rotation about the same point is still possible as it will cause the part to move away from the finger. Similarly, we can show that the finger at a excludes positive rotations about points in the right half-plane, but still allows for negative rotations about these points. The signs in Figure 1a indicate the types of rotations that are possible about points in both half-planes despite the presence of a point contact at a. The constraint imposed in the particular case in which a is a concave vertex is easily understood by realizing that a lies on both incident edges e and e'. The resulting constraint combines the constraints imposed by a being on e and by a being on e', leading to half-wedges of allowed rotation centers (see Figure 1b). In our case of a polygonal

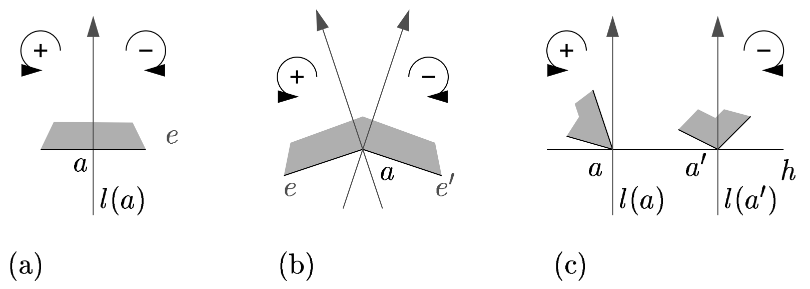

Figure 1: Motions allowed by (a) a point finger at an interior point a of an edge e, (b) a point finger at a concave vertex a with incident edges e and e', and (c) a line contact along a convex-hull edge aa'.

part P and point contacts, the difference between form closure and 2nd-order immobility lies in their treatment of the line $l(a)$ itself.

Form-closure analysis conservatively assumes that negative as well as positive rotations are possible about all points on the directed line $l(a)$. In other words, the line $l(a)$ belongs to both half-planes of allowed rotation centers. Each contact induces a closed half-plane of assumed possible centers of negative rotations and a closed half-plane of assumed possible centers of positive rotations. A part is in form closure if all motions are excluded by the constraints imposed by the contacts placed along the boundary of P. As a result, we must intersect the closed half-planes of possible centers of positive rotations and the closed half-planes of possible centers of negative rotations. If both intersections are empty then the part is in form closure: no motion is possible. Four point contacts are necessary and in almost all cases—including polygons—also sufficient [4] to put a part in form closure. We consider in Section 3 the computation of all form closure grasps with four point fingers, with three point fingers, in which case one finger must be at a concave vertex, and with two fingers, both at concave vertices.

The more careful 2nd-order-immobility analysis takes into account the curvature of the part boundary at the point contact. It shows that a negative or positive rotation about a point on the upper part $l^+(a)$ of $l(a)$ causes the point contact to penetrate P, whereas a similar rotation about a point on the lower part $l^-(a)$ causes the point to move away from or slide along P. In conclusion, only the lower part $l^-(a)$ belongs to both half-planes of allowed rotations. The subtle difference with form-closure analysis has important implications. Consider three point contacts a, a', and a'' that are placed such that the lines $l(a)$, $l(a')$, and $l(a'')$ intersect in a single point p and their directions positively span $\mathbb{R}^2$. The point fingers do not put P in form closure since the corresponding analysis says that positive and negative rotations remain possible about p. The 2nd-order-immobility analysis learns that this motion becomes impossible if it is excluded by at least one of the contacts a, a', and a'', or, in other words, if p lies on at least one of $l^+(a)$, $l^+(a')$, and $l^+(a'')$. Hence, the three points achieve 2nd-order immobility. When the lines $l(a)$, $l(a')$, and $l(a'')$ do not intersect in a single point, the part P can never be 2nd-order immobilized. Lemma 1 summarizes the conclusion of this paragraph.

Lemma 1 *Three points a, a', a'' achieve 2nd-order immoblity of a part if (i) the directions of $l(a)$, $l(a')$, and $l(a'')$ positively span $\mathbb{R}^2$, and (ii) $l^+(a)$, $l(a')$, and $l(a'')$, or $l(a)$, $l^+(a')$, and $l(a'')$, or $l(a)$, $l(a')$, and $l^+(a'')$ intersect in a single point.*

Observe that the set $\mathcal{T}_1^+(P)$ defined above consists of all triples of edges such that there exists at least one placement of point fingers on these edges that achieves 2nd-order immobility. We consider in Section 4 the computation of all 2nd-order-immobility grasps with three point fingers, and with two points fingers of which one must be at a concave vertex.

Let us finally consider grasps with lines. Although a line h tangent to P can touch either a vertex or an edge of the convex hull of P, we shall concentrate on the latter case only. (See [22] for results on grasps involving a line touching a convex-hull vertex.) Let h be in contact with a convex-hull edge e with endpoints a and a', and $l(a)$ and $l(a')$ be the directed lines perpendicular to e and through a and a' respectively. The resulting constraint only allows for positive rotations about points in the half-plane left of $l(a)$ and for negative rotations about points in the half-plane right of $l(a')$ (see Figure 1c).

Wentink [22] (see also [21]) showed that any polygonal part can be put in form closure with one line and two point fingers. We will extend the results from Section 3 to the computation of all form-closure grasps with one line and two points. The fact that the line is placed along an edge of the convex hull of P allows us to simplify the form-closure condition for a grasp with one line contact and two point contacts as given by Lemma 2 [22]. Note that since e is an edge of the convex hull, both e and $slab(e)$ are open.

Lemma 2 *A part P is in form closure with point contacts at a and a' and a line contact along a convex-hull edge e of P if and only if (i) the directions of $l(a)$, $l(a')$, and $slab(e)$ positively span $\mathbb{R}^2$, and (ii) $l(a) \cap l(a') \subseteq slab(e)$.*

3 Form closure with point fingers

In this section we focus on the computation of all form-closure grasps with point fingers. Let e_1, e_2, e_3, and e_4 be edges of P. We define $\mathcal{F}(e_1, e_2, e_3, e_4)$ to be the set of all placements of point contacts $a_1 \in e_1$, $a_2 \in e_2$, $a_3 \in e_3$, and $a_4 \in e_4$ that put P in form closure. We denote by $\mathcal{Q}(P)$ the set of all quadruples (e_1, e_2, e_3, e_4) for which $\mathcal{F}(e_1, e_2, e_3, e_4) \neq \emptyset$. Our solution to the problem of computing all form-closure grasps proceeds in two steps: we first compute the set $\mathcal{Q}(P)$, and then we compute for each $(e_1, e_2, e_3, e_4) \in \mathcal{Q}(P)$ the set $\mathcal{F}(e_1, e_2, e_3, e_4)$. Section 3.1 shows that each set $\mathcal{F}(e_1, e_2, e_3, e_4)$ has constant complexity and can be computed in constant time. We will also see that the sets $\mathcal{F}(e_1, e_2, e_3, e_4)$ implicitly give the form-closure grasps involving fewer than four fingers. Section 3.2 deals with the output-sensitive computation of $\mathcal{Q}(P)$ in $O(n^{2+\epsilon} + K)$ time, where K is the size of the solution and ϵ is an arbitrarily small constant. Using the results from Section 3.1, the bounds for the computation and size of $\mathcal{Q}(P)$ immediately carry over to the set of all form-closure grasps with at most four fingers.

3.1 Grasps induced by one edge quadruple

Let us assume that we are given a quadruple of edges (e_1, e_2, e_3, e_4) such that $\mathcal{F}(e_1, e_2, e_3, e_4) \neq \emptyset$. We can characterize the placement of one point contact along an edge e_i by a value α_i in the unit interval, saying that the contact is at a distance $\alpha_i \cdot |e_i|$ from a designated endpoint of e_i. (Note that the endpoints may or may not belong to the interval.) A placement of four point contacts along $e_1, \ldots, e_4$ can thus be described by quadruple $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ in the four-dimensional unit cube. We want to subdivide this cube of grasp representations into cells of form-closure grasps and cells of non-form-closure grasps. The form-closure grasps are separated from non-form-closure grasps by grasps that correspond to structural changes in the arrangement of the contact normals. We refer to these as *critical grasps*. A grasp is critical if (i) the normals at three of the point contacts intersect in a point or (ii) two normals at two of the points contacts have opposite directions and share a single supporting line. See Figures 2 and 3 for examples of both critical grasps.

Consider three edges e_1, e_2, and e_3 and the representation by the unit cube of placements of point contacts on these edges. For every placement of contacts on e_1 and e_2 there is at most one placement on e_3 such that the normals at these contacts intersect in a single point. The representations $(\alpha_1, \alpha_2, \alpha_3)$ of all such placements of contacts on e_1, e_2, and e_3 turn out to satisfy a single linear equation. Hence, these representations define a two-dimensional planar face in $(0,1)^3$. Moreover, the representations of all placements of contacts on e_1, e_2, e_3, and e_4 with the property that the normals at the contacts on e_1, e_2, and e_3 intersect in a point define a three-dimensional planar face in the four-dimensional unit cube of all placements on e_1, e_2, e_3, and e_4. Note that there are four triples of edges and thus four faces.

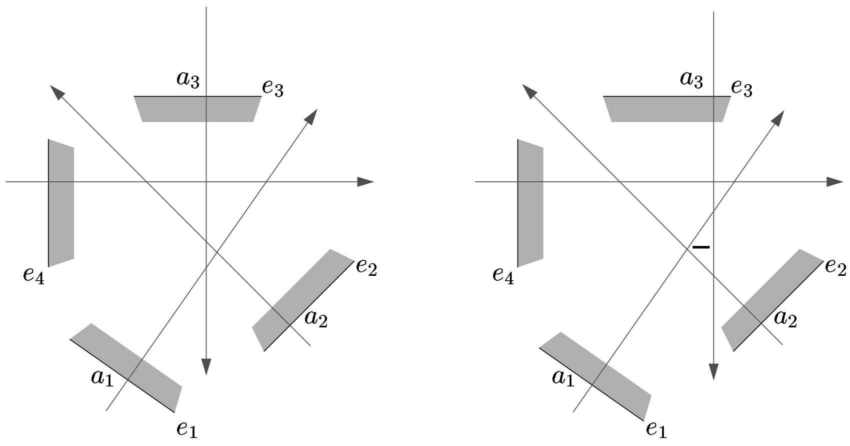

Figure 2: A small shift of the point contact a_3 on e_3 transforms a form-closure grasp into a non-form-closure grasp.

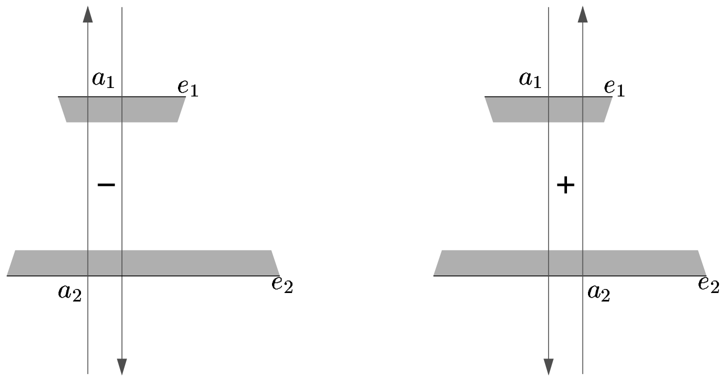

Figure 3: A small shift of the point contact a_2 on e_2 reverses the sign of the composite constraint imposed by a_1 and a_2, and may transform a form-closure grasp into a non-form-closure grasp.

Consider two parallel edges e_1 and e_2 with opposite normal directions and the representation by the unit square of placements of point contacts on these edges. For every placement of a contact on e_1 there is at most one placement on e_2 such that the normals at these contacts share the same supporting line. The representations (α_1, α_2) of all such placements of contacts on e_1 and e_2 define a line segment in the unit square. Again, we obtain a three-dimensional planar face in the four-dimensional unit cube when we incorporate the contacts on e_3, and e_4. The number of faces of this type is at most two, since three (of four) contacts on parallel edges will never yield a form closure grasp.

The supporting planes of the (at most six) planar faces subdivide the four-dimensional unit cube into a constant number of convex cells that entirely consist of either form-closure grasps or non-form-closure grasps; each cell has constant complexity. As a result, the union of the form-closure cells—and thus $\mathcal{F}(e_1, e_2, e_3, e_4)$—has constant complexity. To report the form-closure cells, we simply enumerate all cells and then select those that consist of form-closure grasps by checking an arbitrary point in the cell. The entire computation clearly takes constant time and outputs a representation of $\mathcal{F}(e_1, e_2, e_3, e_4)$.

In certain cases, two of the edges in a quadruple (e_1, e_2, e_3, e_4) meet at a single concave vertex a. A two-dimensional facet of the four-dimensional unit cube representing $\mathcal{F}(e_1, e_2, e_3, e_4)$ then corresponds to the placement of two different fingers at the same vertex a. Since it is useless to

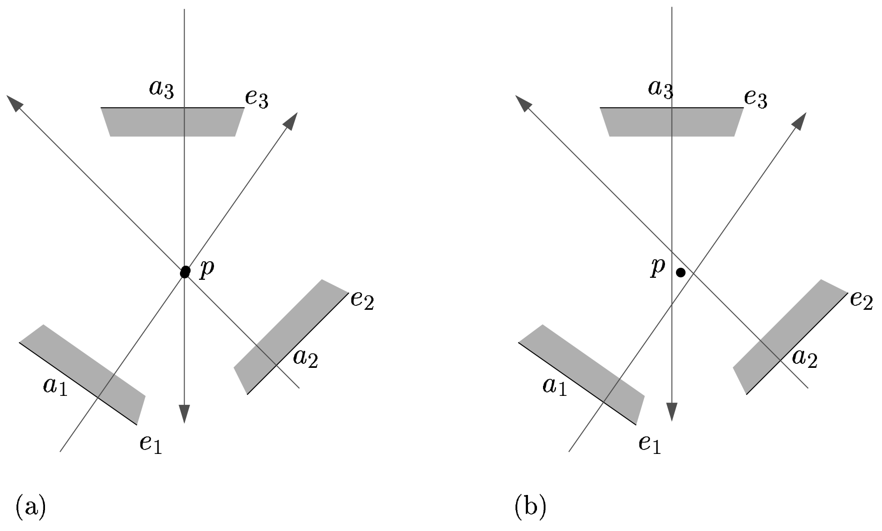

Figure 4: Turning a grasp allowing for negative and positive rotations about a single point p only into a grasp allowing for positive rotations about points in an arbitrarily small triangular region and excluding all negative rotations.

place two fingers at exactly the same location, such a placement can be viewed as a grasp with three fingers. Likewise, we can find the grasps involving two fingers at concave vertices at the corners of specific four-dimensional unit cubes. It is easy to see that every grasp with two or three fingers can be regarded as a particular grasp with four fingers. We conclude that the union of all sets $\mathcal{F}(e_1, e_2, e_3, e_4)$ does not only contain all four-finger grasps of P but also grasps with two and three fingers.

3.2 Computing all form-closure grasps

We use the results from the previous section to obtain a $O(n^{2+\epsilon} + K)$ bound for computing all form-closure grasps for P, where K is the complexity of the solution. Besides all form closure grasps we can also compute all Nguyen regions in time $O(n^{2+\epsilon} + K)$, where K is the number of such regions. The key observation is that an edge quadruple (e_1, e_2, e_3, e_4) defines Nguyen regions if and only if $\mathcal{F}(e_1, e_2, e_3, e_4) \neq \emptyset$. As a consequence, we can first compute $\mathcal{Q}(P)$ and then apply the ideas of Nguyen [8] (for finding the Nguyen regions defined by a single edge quadruple) to all quadruples in $\mathcal{Q}(P)$. The latter computation takes constant time per quadruple, leading to the quoted overall bound. This bound is a considerable improvement over the (n^4) bound by Nguyen.

We turn our attention to the computation of $\mathcal{Q}(P)$. A necessary condition for a quadruple (e_1, e_2, e_3, e_4) to satisfy $\mathcal{F}(e_1, e_2, e_3, e_4) \neq \emptyset$—and thus be in $\mathcal{Q}(P)$—is that the contact normals (and slabs) to three of the edges must positively span $\mathbb{R}^2$. Without loss of generality we assume that e_1, e_2, and e_3 are these edges, so $(e_1, e_2, e_3) \in \mathcal{T}(P)$. We know that either $(e_1, e_2, e_3) \in \mathcal{T}_0(P)$ or $(e_1, e_2, e_3) \in \mathcal{T}_1(P)$; we treat both cases separately, in reverse order.

Consider the case where $(e_1, e_2, e_3) \in \mathcal{T}_1(P)$ and let R be the non-empty intersection of *slab*(e_1),*slab*(e_2), and *slab*(e_3). Every point $p \in R$ corresponds to a placement of point contacts at $a_1 \in e_1$, $a_2 \in e_2$, and $a_3 \in e_3$ with $l(a_1) \cap l(a_2) \cap l(a_3) = p$, excluding all motions except positive and negative rotations about p. Now choose p such that it is in the *interior* of R (see Figure 4a). We can now slightly shift each of the points a_1, a_2, and a_3 in the appropriate direction along the respective edges to create an arbitrarily small triangular region around p of centers of positive rotations and exclude all negative rotations (see Figure 4b). Likewise, we can create an arbitrarily small triangle of centers of negative rotations and exclude all positive rotations.

Now let e_4 be an edge of P. Choose a point a_4 on e_4 such that $l(a_4)$ does not contain p. If p

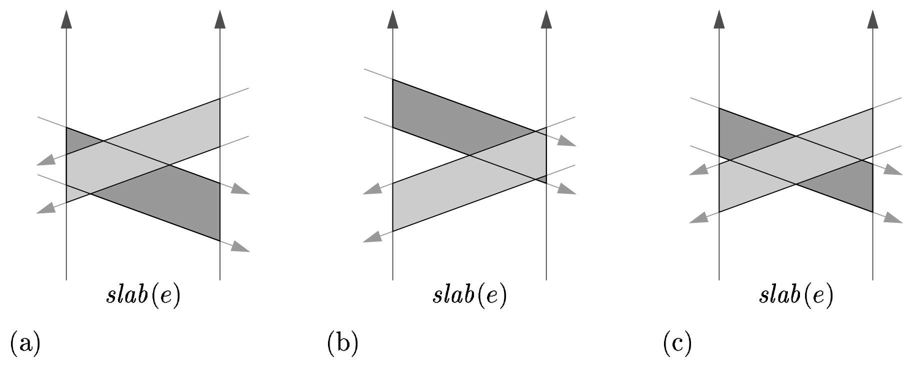

Figure 5: Two parallelograms intersect inside a slab if and only if (a) their left edges overlap, or (b) their right edges overlap, or (c) their top edges intersect.

lies in the interior of the half-plane left of $l(a_4)$, we move a_1, a_2, and a_3 in the manner described above to establish that the contacts at these points only allow for negative rotations about points in a small triangle that lies entirely inside the interior of the half-plane left of $l(a_4)$. Incorporation of the constraint imposed by a_4, however, will rule out all these rotations: P is in form closure. We can act similarly when p lies to the right of $l(a_4)$. As a result, the quadruple (e_1, e_2, e_3, e_4) will be in $\mathcal{Q}(P)$ regardless of the choice of e_4. Hence, our aim is simply to report all triples $(e_1, e_2, e_3) \in \mathcal{T}_1(P)$, as each such triple contributes n quadruples to $\mathcal{Q}(P)$.

We fix one edge e and solve the problem of finding all combinations of two edges e_1 and e_2 such that $(e, e_1, e_2) \in \mathcal{T}_1(P)$. To simplify the discussion, we assume without loss of generality that $slab(e)$ is vertical and directed upward. We call a slab red if it crosses $slab(e)$ from left to right and blue if it crosses $slab(e)$ from right to left. Notice that $slab(e_1)$ and $slab(e_2)$ must have different colors. We query with each red slab to find the appropriately-oriented blue slabs intersecting it inside $slab(e)$.

Query A *Given a slab v with direction α, a set S of n slabs, and a query slab q with direction β, report all $s \in S$ such that α, β, and the direction of s positively span $\mathbb{R}^2$ and $v \cap s \cap q \neq \emptyset$.*

The intersection of a slab with $slab(e)$ is a parallelogram with two vertical edges. A parallelogram inherits the color and direction of its defining slab. Certain parts of the boundary of the parallelogram may not belong to the parallelogram. Regardless of this fact, we have that a red parallelogram intersects a blue parallelogram if and only if (i) its left edge overlaps the left edge of the blue parallelogram, (ii) its right edge overlaps the right edge of the blue parallelogram, or (iii) the *interior* of its top edge intersects the *interior* of the top edge of the blue parallelogram. (Observe that a pair of vertical edges must overlap if the top edges intersect at their endpoints.) We first compute all pairs satisfying condition (i). Notice that all left edges are intervals on a single line. Each interval inherits the associated direction from its defining parallelogram. We store the blue intervals such that we can report for each query red interval all appropriately-oriented overlapping blue intervals. This can be accomplished by a two-level tree, of which the first-level range tree selects the intervals with the appropriate associated orientations and the second-level interval tree selects the intervals that overlap the query interval. (See [1] for details on all tree structures used in this paper.) The tree can be constructed in $O(n \log^2 n)$ time and offers a query time of $O(\log^2 n + k)$, where k is the number of answers. Since we query with $O(n)$ red intervals, we can enumerate all pairs satisfying condition (i) (and the equivalent condition (ii)) in $O(n \log^2 n + k')$ time, where k' is the number of reported pairs.

To find all pairs of parallelograms satisfying condition (iii), we must report all pairs of intersecting differently-colored top edges. All (blue and red) left endpoints of the top edges lie on a single line. We represent each top edge by a point in the plane: the x-coordinate and y-coordinate being the height of its left and right endpoint respectively. A query red edge represented by (x_r, y_r)

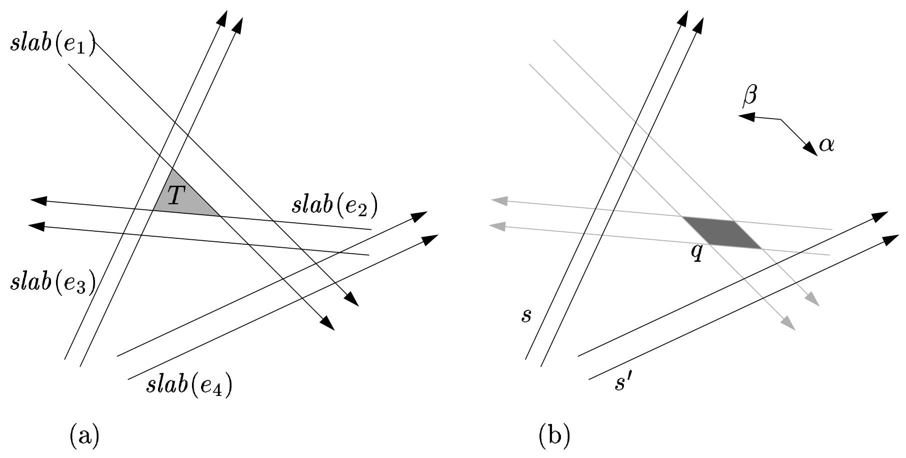

Figure 6: (a) Three slabs $\mathit{slab}(e_1)$, $\mathit{slab}(e_2)$, and $\mathit{slab}(e_3)$ bounding a triangular region T and a fourth slab $\mathit{slab}(e_4)$ leaving $\mathit{slab}(e_1) \cap \mathit{slab}(e_2)$ to its left; (b) two slabs with appropriate directions leaving the query parallelogram defined by two slabs on opposite sides.

intersects a blue edge (x_b, y_b) if $x_r > x_b \wedge y_r < y_b$ or $x_r < x_b \wedge y_r > y_b$. Thus our aim is to solve—for a query point (x_q, y_q)—the orthogonal range queries for points (x, y) with $x > x_q \wedge y < y_q$ or $x < x_q \wedge y > y_q$.

We store the points representing the blue edges in a two-level tree, in which the first-level range tree serves to select the edges originating from parallelograms with appropriate orientations and the second-level two-dimensional range tree is used to perform the above orthogonal range queries (and thus select the intersecting edges). This tree is constructed in time $O(n \log^2 n)$ and the $O(n)$ queries take $O(n \log^2 n + k)$ time, where k is the number of reported pairs.

To find all triples of $\mathcal{T}_1(P)$ we must solve Query A for all edges e of P. This amounts to a total of $O(n^2 \log^2 n + K)$ time for finding all K triples.

We must still consider the case where $(e_1, e_2, e_3) \in \mathcal{T}_0(P)$. Now the slabs $\mathit{slab}(e_1)$, $\mathit{slab}(e_2)$, and $\mathit{slab}(e_3)$ bound a triangle T (which may degenerate into a point) that is either to the left or to the right of these slabs. Let us assume T lies to the right of the slabs so that T is defined by the right slab boundaries. The only motions that remain possible when we place contacts at some $a_1 \in e_1$, $a_2 \in e_2$, and $a_3 \in e_3$ are negative rotations about points in the triangle bounded by $l(a_1)$, $l(a_2)$, and $l(a_3)$. The size of this triangle, which is congruent with T, is minimized when a_1, a_2, and a_3 are placed as close as possible to the (right) endpoints of e_1, e_2, and e_3. In that case the region of centers of negative rotations becomes T. Notice that an edge e_4 of P satisfies $(e_1, e_2, e_3, e_4) \in \mathcal{Q}(P)$ if and only if there exists a point $a_4 \in e_4$ such that the contact at a_4 excludes the negative rotations about points in T.

Let e_4 be an edge of P such that $(e_1, e_2, e_3, e_4) \in \mathcal{Q}(P)$. Since there is a point $a_4 \in e_4$ with the properties mentioned in the previous paragraph, no part of T lies to the right of $\mathit{slab}(e_4)$. The slab $\mathit{slab}(e_4)$ then either intersects one of the parallelograms $\mathit{slab}(e_1) \cap \mathit{slab}(e_2)$, $\mathit{slab}(e_1) \cap \mathit{slab}(e_3)$, and $\mathit{slab}(e_2) \cap \mathit{slab}(e_3)$, or has all these parallelograms to its left. It turns out that in the first case $\mathit{slab}(e_4)$ positively spans $\mathbb{R}^2$ together with the slabs defining the intersected parallelogram. As a result, this intersecting triple—along with all the quadruples it contributes to $\mathcal{Q}(P)$—was already reported in the case handled earlier. Hence, we may assume that $\mathit{slab}(e_4)$ leaves all three parallelograms to its left. This implies that $\mathit{slab}(e_4)$ either leaves $\mathit{slab}(e_1) \cap \mathit{slab}(e_2)$ to its left and positively spans $\mathbb{R}^2$ with $\mathit{slab}(e_1)$ and $\mathit{slab}(e_2)$, or leaves $\mathit{slab}(e_1) \cap \mathit{slab}(e_3)$ to its left and positively spans $\mathbb{R}^2$ with $\mathit{slab}(e_1)$ and $\mathit{slab}(e_3)$, or leaves $\mathit{slab}(e_2) \cap \mathit{slab}(e_3)$ to its left and positively spans $\mathbb{R}^2$ with $\mathit{slab}(e_2)$ and $\mathit{slab}(e_3)$. Each of these cases can be handled in a similar way. Let us assume that $\mathit{slab}(e_4)$ leaves the parallelogram $\mathit{slab}(e_1) \cap \mathit{slab}(e_2)$ to its left and positively spans $\mathbb{R}^2$ with $\mathit{slab}(e_1)$ and $\mathit{slab}(e_2)$ (see Figure 6a). From the assumption that the right boundaries of $slab(e_1)$, $slab(e_2)$, and $slab(e_3)$ bound a triangle T it follows that $slab(e_3)$ leaves the parallelogram $slab(e_1) \cap slab(e_2)$ to its right and positively spans $\mathbb{R}^2$ with $slab(e_1)$ and $slab(e_2)$. Hence, the slabs $slab(e_3)$ and $slab(e_4)$ leave the parallelogram $slab(e_1) \cap slab(e_2)$ on opposite sides and both positively span $\mathbb{R}^2$ with $slab(e_1)$ and $slab(e_2)$. Our approach is to query with all $O(n^2)$ parallelograms—defined by the intersection of two slabs—for all pairs of appropriately-oriented pairs of slabs that leave the parallelogram on opposite sides (see Figure 6b). If we associate with each parallelogram the directions of both defining slabs we obtain the following query.

Query B *Given a set S of n slabs and a query parallelogram q with associated directions α and β, report all pairs $(s, s') \in S^2$ such that α, β, and the direction of s as well as α, β, and the direction of s' positively span $\mathbb{R}^2$ and q lies to the left of s and to the right of s'.*

We can use a four-level cutting tree to store the slabs for efficient reporting of all slabs that leave a query parallelogram q to a given side. Each level selects the slabs that leave one specific vertex of q to the given side. This data structure offers $O(\text{polylog}\, n + k)$ query time and can be computed in $O(n^{2+\epsilon})$ time, for an arbitrarily small positive constant ϵ. We add a range-tree level to restrict the search to slabs within the appropriate range of orientations. The resulting five-level tree has the same asymptotic query and preprocessing time as the original tree.

It remains to show how we can use the five-level tree to report pairs of appropriately-oriented slabs that leave q on opposite sides. We first check for emptiness of the set of slabs that leave q to the left and for emptiness of the set of slabs that leave q to the right. We can use the five-level tree to perform this check (by equipping the nodes with cardinalities of the stored sets) in $O(\text{polylog}\, n)$ time. If one of the sets turns out to be empty then the number of pairs is zero and we have detected so in $O(\text{polylog}\, n)$ time. If both sets turn out to be non-empty, we enumerate the sets in $O(\text{polylog}\, n+k)$ and $O(\text{polylog}\, n+k')$ time, where k and k' are the numbers of slabs leaving q to the left and right respectively. The kk' solution pairs are now easily reported in additional $O(kk')$ time. Because $k, k' \neq 0$, the time for reporting the pairs dominates the time for the two separate sets of slabs. As a result, we conclude that the pairs are reported in $O(\text{polylog}\, n + k'')$ time, where k'' is the number of pairs. Querying with all $O(n^2)$ parallelograms amounts to an $O(n^{2+\epsilon} + K)$ time bound for solving Query B, where K is the number of answers.

Combining the $O(n^{2+\epsilon} + K)$ and $O(n^2 \log^2 n + K)$ bounds for computing the subsets of $\mathcal{Q}(P)$ with the results from Section 3.1, we obtain Theorem 3.

Theorem 3 *The set of all form-closure grasps with at most four point contacts for a polygon can be computed in time $O(n^{2+\epsilon} + K)$, where K is the complexity of the solution.*

We recall that the bound of Theorem 3 also applies to the computation of Nguyen regions. The straightforward bounds on K are $\Omega(1)$ and $O(n^4)$. Figure 7 shows that the $O(n^4)$ upper bound is tight. The polygonal part consists of four convex chains of approximately $(n/4) - 1$ edges each, connected by two long horizontal edges and two vertical edges. Consider the line l bisecting the two long edges. The chains are constructed such the slabs of any two edges left/right of l intersect strictly left/right of l. As a result, any placement of four contacts on different chains will provide form closure. Since the number of quadruples consisting of one edge from each of the chains is $\Omega(n^4)$, we find that the set of form-closure grasps for this polygon has complexity $\Omega(n^4)$, showing that the upper bound of $O(n^4)$ on K is tight.

Theorem 4 reports two non-trivial bounds on the asymptotic complexity of the set of all form closure grasps. The $\Omega(n^2)$ lower bound for rectilinear polygons shows that the algorithm outlined in this section is almost optimal for rectilinear polygons.

Theorem 4 *The complexity K of the set of all form-closure grasps with four point contacts for a polygon P is*

(i) $\Omega(n)$ if P is convex and has no parallel edges,

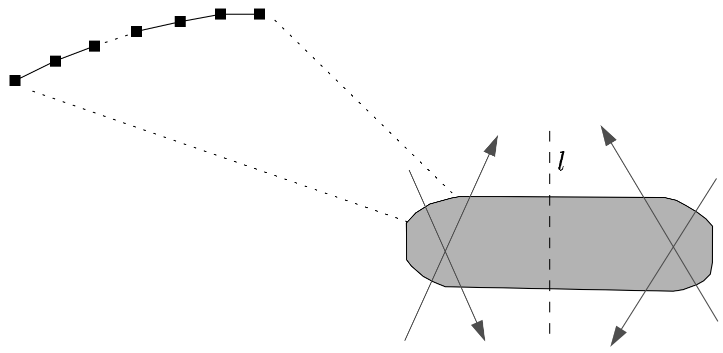

Figure 7: An n-gon with a set of form-closure grasps of complexity $\Omega(n^4)$.

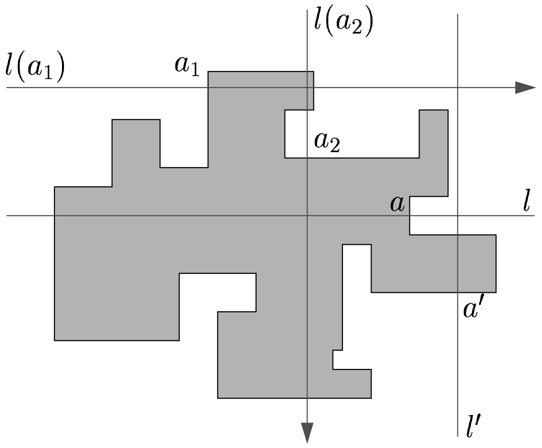

Figure 8: Point contacts at a and a' together with the point contacts at a_1 and a_2 put P in form closure.

(ii) $\Omega(n^2)$ if P is rectilinear.

Proof. (i) The maximal inscribed circle of P touches ∂P in three points, lying on three edges e_1, e_2, and e_3; the slabs *slab*(e_1), *slab*(e_2), and *slab*(e_3) have a common intersection and their directions positively span $\mathbb{R}^2$. By the considerations above, the edges e_1, e_2, and e_3 contribute $\Theta(n)$ quadruples to $\mathcal{Q}(P)$.

(ii) Assume that the $n/2$ horizontal and the $n/2$ vertical edges of P are parallel to the respective coordinate axes. Moreover, we assume without loss of generality that the normals to at least $n/4$ of the edges are directed to the right and that the normals to at least $n/4$ of the edges are directed downward. We show that for any of the $\Omega(n^2)$ pairs consisting of an edge e_1 with a normal directed to the right and an edge e_2 with a normal directed downward, there exist edges e and e' such that $\mathcal{F}(e_1, e_2, e, e') \neq \emptyset$. Let a_1 and a_2 be arbitrary points on e_1 and e_2. A horizontal line l below $l(a_1)$ intersecting P must always intersect a vertical edge e with a normal directed to the left. If this intersection occurs at an endpoint of e, we slightly move l such that it intersects the interior of e. Let a be the intersection of l and e (see Figure 8). Similarly, we let e' be the edge with a normal directed upward intersected by a vertical line l' right of $l(a_2)$. Let a' be the intersection of l' and e'. It is easily verified that point contacts at a_1, a_2, a, and a' put P in form closure, thus $\mathcal{F}(e_1, e_2, e, e') \neq \emptyset$. □

4 2nd-order immobility with point fingers

In the spirit of the previous section, we define $\mathcal{I}(e_1, e_2, e_3)$ to be the set of all placements of point contacts a_1, a_2, a_3 on the edges e_1, e_2, e_3 respectively that achieve 2nd-order immobility of the part. We recall from Section 2 that $\mathcal{T}_1^+(P)$ is exactly the set of edge triples (e_1, e_2, e_3) for which $\mathcal{I}(e_1, e_2, e_3) \neq \emptyset$. Section 4.1 deals with the computation of the constant-complexity sets $\mathcal{I}(e_1, e_2, e_3)$, which implicitly also give all 2nd-order-immobility grasps involving only two fingers. Section 4.2 shows that $\mathcal{T}_1^+(P)$ can be found by (repeatedly) solving a slightly modified version of Query A from Subsection 3.2. This approach leads to an $O(n^2 \log^2 n + K)$ algorithm for finding all 2nd-order-immobility grasps with at most three point fingers.

4.1 Grasps induced by one edge triple

At least one permutation of a triple (e_1, e_2, e_3) with $\mathcal{I}(e_1, e_2, e_3) \neq \emptyset$ occurs in the set $\mathcal{T}_1^+(P)$. Assume without loss of generality that $(e_1, e_2, e_3) \in \mathcal{T}_1^+(P)$. The definition of $\mathcal{T}_1^+(P)$ says that $\mathit{slab}^+(e_1) \cap \mathit{slab}(e_2) \cap \mathit{slab}(e_3) \neq \emptyset$. This non-empty intersection, which is bounded by at most seven line segments, consists of all intersections p of lines $l^+(a_1)$, $l(a_2)$, and $l(a_3)$ with $a_1 \in e_1$, $a_2 \in e_2$, and $a_3 \in e_3$ and represents as such the set $\mathcal{I}(e_1, e_2, e_3)$. (The point p represents the placement of point fingers obtained by projecting p perpendicularly onto e_1, e_2, and e_3.) This representation is clearly computable in constant time.

In the case that two of the three edges of a triple (e_1, e_2, e_3) meet at a concave vertex, one vertex of the representation of $\mathcal{I}(e_1, e_2, e_3)$ may correspond to a placement in which two of the fingers coincide at that concave part vertex. Since it is again useless to place two fingers at the same location, we can simply omit one finger, obtaining a two-finger 2nd-order-immobility grasp.

4.2 Computing all 2nd-order-immobility grasps

The preceding subsection shows that the time required to compute all 2nd-order-immobility grasps with at most three fingers equals (asymptotically) the time to compute the set $\mathcal{T}_1^+(P)$. We fix an edge e and solve the problem of finding all combinations of edges e_1 and e_2 such that $(e, e_1, e_2) \in \mathcal{T}_1^+(P)$, or, in other words, such that $\mathit{slab}^+(e) \cap \mathit{slab}(e_1) \cap \mathit{slab}(e_2) \neq \emptyset$. We simplify the discussion by assuming that the half-slab $\mathit{slab}^+(e)$ is vertical and directed upward. A slab is red if it crosses $\mathit{slab}^+(e)$ from left to right and blue if it crosses $\mathit{slab}^+(e)$ in the opposite direction. The slabs e_1 and e_2 must have different colors. We obtain the following modified version of Query A.

Query C *Given a half-slab v^+ with direction α, a set S of n slabs, and a query slab q with direction β, report all $s \in S$ such that α, β, and the direction of s positively span $\mathbb{R}^2$ and $v^+ \cap s \cap q \neq \emptyset$.*

The (red or blue) intersection of a (red or blue) slab with $\mathit{slab}^+(e)$ is now (the upper part of) a parallelogram with two vertical edges clipped by a horizontal line. It has one or two vertical edges, at most one horizontal edge, and one or two non-vertical edges. A red shape intersects a blue shape if (i) their left vertical edges overlap, (ii) their right vertical edges overlap, (iii) their horizontal edges overlap, or (iv) the interiors of their top edges intersect. Cases (i)-(iii) are similar to cases (i) and (ii) in Section 3.2, and the only difference between the present case (iv) and case (iii) in Section 3.2 is that a top edge can now have one of its endpoints on the horizontal boundary of the half-slab $\mathit{slab}^+(e)$. We distinguish (a) edges that connect the left and right vertical boundary, (b) edges that connect the left vertical and the horizontal boundary, and (c) edges that connect the horizontal and right vertical boundary of $\mathit{slab}^+(e)$. We store the blue edges in three separate data structures according to their type, and query with the red edges. As an example, assume we are given a red edge with its left endpoint at height x_r on the left boundary and its right endpoint at a horizontal coordinate z_r on the horizontal boundary of $\mathit{slab}^+(e)$. We query for blue edges of type (a) with left endpoints at heights x satisfying $x < x_r$, blue edges of type (b) with left endpoints

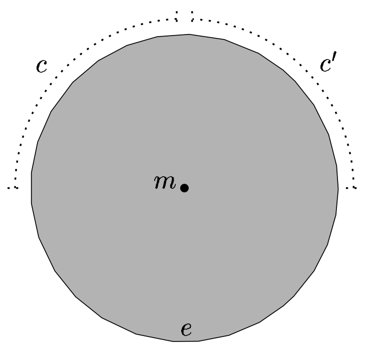

Figure 9: An n-gon for which the set of 2nd-order-immobility grasps with three fingers and the set of form-closure grasps with one line and two points has complexity $\Omega(n^3)$.

at heights x and right endpoints with coodinates z satisfying $x > x_r \wedge z < z_r$ or $x < x_r \wedge z > z_r$, and blue edges of type (c) with left endpoints with coordinates z satisfying $z < z_r$. All subqueries are orthogonal range queries and can thus be solved efficiently by using the range-tree structure from Section 3.2. Similar arguments apply if the red edge is of one of the other two types. This leads to the following result.

Theorem 5 *The set of all 2nd-order-immobility grasps with at most three point contacts for a polygon can be computed in time $O(n^2 \log^2 n + K)$, where K is the complexity of the solution.*

The almost regular n-gon P in Figure 9 shows that the upper bound of $O(n^3)$ on K is tight. The half-slabs $\mathit{slab}^+(e)$ of all edges e contain the center-of-mass m. Assume that e is horizontal, as in Figure 9. Now, each combination consisting of edges e_1 and e_2 from the chains c and c' (both consisting of approximately $n/4$ edges) satisfies $(e, e_1, e_2) \in \mathcal{T}_1^+(P)$. Repeating the argument for all choices of e yields that the size of $\mathcal{T}_1^+(P)$ can be $\Omega(n^3)$.

Figure 10 shows a polygon P for which the number of triples in $\mathcal{T}_1^+(P)$ is constant. It consists of two long edges e_1 and e_2 and a concave chain, which in turn consists of two subchains of short edges connected by a long edge e_c. Every triple in $\mathcal{T}_1^+(P)$ must include both e_1 and e_2 to meet the requirement that the directions of the slabs positively span $\mathbb{R}^2$. The concave chain is constructed in such a way that the slab $\mathit{slab}(e)$ induced by any of its short edges e does not intersect $\mathit{slab}(e_1) \cap \mathit{slab}(e_2)$. The only remaining triple that may belong to $\mathcal{T}_1^+(P)$ is therefore (e_1, e_2, e_c).

5 Form closure with a line and two point fingers

We now return to form closure and consider the problem of computing all grasps with one line and two point fingers. We shall obtain $O(n^2 \log^2 n + K)$ and $O((n + K) \log n)$ time algorithms for arbitrary and rectilinear polygons, where K is the complexity of the output. Let e be an edge of the convex hull of the part P. We define $\mathcal{F}_e(e_1, e_2)$ to be the set of all placements of point contacts $a_1 \in e_1$ and $a_2 \in e_2$ that—together with a line along e—put P in form closure. We denote by $\mathcal{T}'(P)$ the set of all triples (e, e_1, e_2) such that $\mathcal{F}_e(e_1, e_2) \neq \emptyset$. We will again follow the two-step approach of first computing $\mathcal{T}'(P)$ and then the sets $\mathcal{F}_e(e_1, e_2)$ for all $(e, e_1, e_2) \in \mathcal{T}'(P)$.

5.1 Grasps of arbitrary polygons

From Lemma 2 we deduce that the problem of computing all pairs of edges (e_1, e_2) such that $(e, e_1, e_2) \in \mathcal{T}'(P)$ is exactly the problem formulated in Query A. We can therefore obtain $\mathcal{T}'(P)$ in $O(n^2 \log^2 n + K)$ time, where $K = |\mathcal{T}'(P)|$.

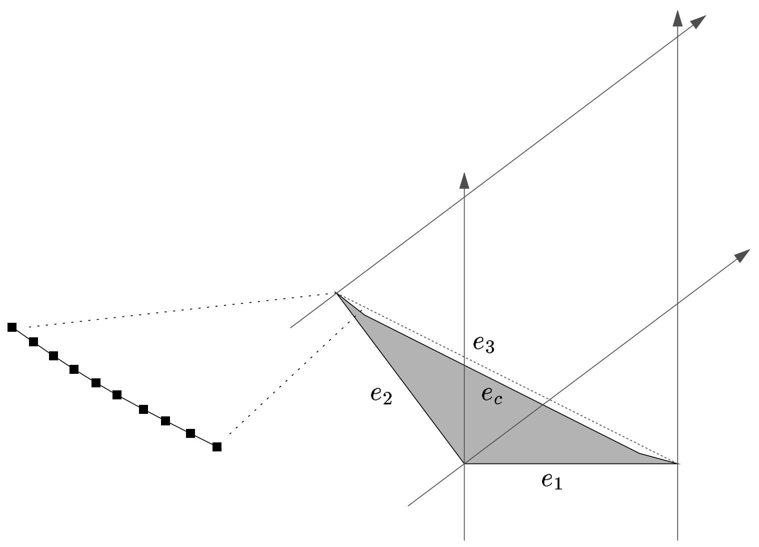

Figure 10: An n-gon for which the set of 2nd-order-immobility grasps with three fingers and the set of form-closure grasps with one line and two points has complexity $O(1)$.

Let us now assume that we are given a triple of edges $(e, e_1, e_2) \in \mathcal{T}'(P)$. The non-empty region $slab(e) \cap slab(e_1) \cap slab(e_2)$ consists of all intersections p of lines $l(a_1)$ and $l(a_2)$ with $a_1 \in e_1$ and $a_2 \in e_2$ inside $slab(e)$ and represents as such the set $\mathcal{F}_e(e_1, e_2)$. (The point p represents the pair of points (a_1, a_2) obtained by projecting p perpendicularly onto e_1 and e_2.) This representation of $\mathcal{F}_e(e_1, e_2)$ has constant complexity as it is bounded by at most six lines and is clearly computable in constant time.

If we apply the ideas of the preceding paragraphs to all edges of the convex hull of P we obtain Theorem 6.

Theorem 6 *The set of all form-closure grasps with one line and at most two point contacts for a polygon can be computed in time $O(n^2 \log^2 n + K)$, where K is the complexity of the solution.*

Figures 9 and 10 show that the trivial upper and lower bounds of $O(n^3)$ and $\Omega(1)$ on K are tight. In the first case, we now use the edge e for the line contact. In the second case, the edge contact can only be placed along the three convex hull edges e_1, e_2, and e_3. The placement of the line contact along e_1 (e_2) dictates that form closure may only be obtained by placing the two points along e_c and e_2 (e_1). Form closure with the line contact along e_3 can only be achieved when the point contacts are placed along e_1 and e_2. In conclusion, we find that $|\mathcal{T}'(P)| = O(1)$ for the polygon in Figure 10.

5.2 Grasps of rectilinear polygons

The computation of $\mathcal{T}'(P)$ can be accomplished in $O((n + K) \log n)$ time if the polygon P is rectilinear. We consider placements of the line contact along convex-hull edges that are not parallel to any of the polygon edges, with the understanding that the placements along the remaining edges will never yield form closure with two additional points.

Let e be a convex-hull edge with an appropriate orientation and assume that the slab $slab(e)$ is vertical and directed upward. Our aim is again to report all pairs of edges e_1 and e_2 such that $slab(e)$, $slab(e_1)$, and $slab(e_2)$ positively span $\mathbb{R}^2$ and such that $slab(e) \cap slab(e_1) \cap slab(e_2) \neq \emptyset$. As our polygon P is rectilinear, we now face four sets of equally-directed slabs. It turns out that the union of all slabs in each set is again a single slab. A slab is red if it is directed downward and crossing $slab(e)$ from left to right and blue if it is directed downward and crossing $slab(e)$ from

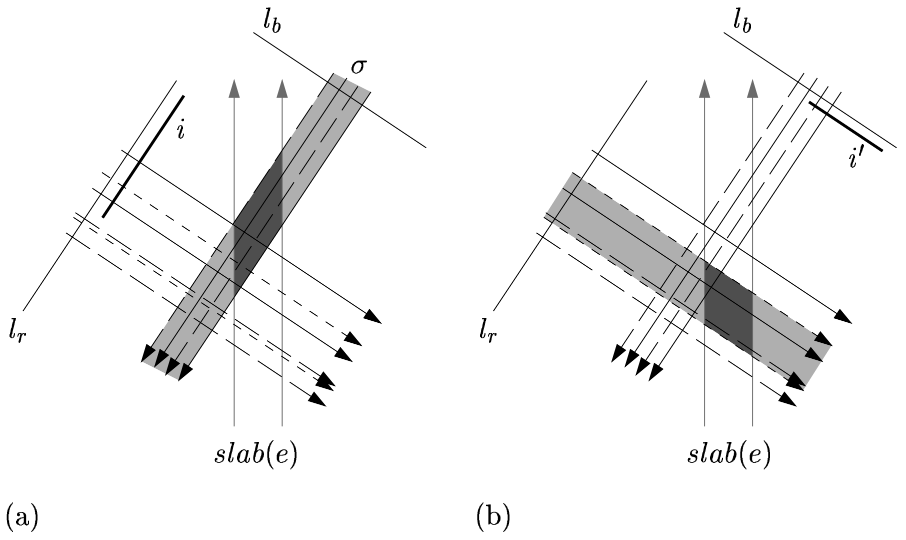

Figure 11: (a) Selection of the red slabs that intersect at least one blue slab, and (b) a query with one of those red slabs to determine all blue slabs that intersect it.

right to left. Notice that all red slabs and all blue slabs are equally-directed. We observe that one of $slab(e_1)$ and $slab(e_2)$ must be red and the other one blue in order for $slab(e)$, $slab(e_1)$, and $slab(e_2)$ to positively span $\mathbb{R}^2$. We solve the following query.

Query D *Given a slab v and two sets S_0 and S_1 of $O(n)$ parallel slabs whose unions are again single slabs, report all pairs $(s_0, s_1) \in S_0 \times S_1$ such that $v \cap s_0 \cap s_1 \neq \emptyset$.*

Querying with all red slabs in a data structure storing the blue slabs would cost at least linear time in the number of red slabs—and thus result in an algorithm with at least quadratic overall running time. Instead we choose to query only with the red slabs that intersect at least one blue slab inside $slab(e)$. Clearly, this approach only pays off if this subset of slabs can be determined efficiently.

Let l_r and l_b be lines perpendicular to the red and blue slabs respectively. We store the intervals defined by the intersections of the red slabs and l_r in an interval tree. Note that the union of these red intervals is again a single interval. The interval tree allows us to report all red intervals that intersect a query interval in $O(\log n + k)$ time, where k is the number of reported answers. The intersections of the blue slabs and l_b are stored in a similar structure. The two data structures (as well as the similar structures for the remaining two sets of parallel slabs) can be constructed in $O(n \log n)$ time.

Let σ be the union of all blue slabs. Furthermore, denote by i the projection onto l_r of $\sigma \cap slab(e)$. Querying with i in the tree storing the red intervals yields exactly those intervals that are defined by red slabs intersecting $\sigma \cap slab(e)$. Because σ contains no holes, each reported red slab must intersect at least one of the blue slabs (see Figure 11a). We find this subset of the red slabs in $O(\log n + k)$ time, where k is the size of the subset. Next, we take each of the selected red slabs s and query the blue structure with the projection i' into l_b of $s \cap slab(e)$ (see Figure 11b). The reported intervals are exactly those that are defined by blue slabs s' intersecting $s \cap slab(e)$. The queries with all red slabs take $O(k \log n + k')$ time, where k' is now the number of pairs of slabs intersecting inside $slab(e)$. Using the fact that k' can in the worst case be as small as k, we obtain that the running time is $O(k' \log n)$. Applying the same ideas to all $O(n)$ convex-hull edges leads to an overall query time of $O(n \log n + K \log n)$, where K is the total number of reported answers. The four necessary data structures can be built in $O(n \log n)$ time. Theorem 7 summarizes the result.

Theorem 7 *The set of all form-closure grasps with one line and at most two point contacts for a rectilinear polygon can be computed in time $O((n+K)\log n)$, where K is the complexity of the solution.*

6 Conclusion

We have shown how the problems of computing *all* form-closure and 2nd-order-immobility grasps with at most four and three points respectively and the problem of computing *all* form-closure grasps with one line and two points can be transformed into efficiently solvable geometric searching problems. The resulting algorithms are the first output-sensitive algorithms to solve these problems. The running time of the algorithm for form-closure grasps with at most four points is $O(n^{2+\epsilon}+K)$, where K is the description size of the solution and ϵ is an arbitrarily small constant. This algorithm is also capable of solving—within the same time bound—Nguyen's problem of finding quadruples of parts of edges that allow for independent placement of point contacts [8]. Our result marks a significant improvement of Nguyen's $O(n^4)$ algorithm. In addition, we have reported an $O(n^2\log^2 n+K)$ algorithm for computing the 2nd-order-immobility grasps involving three or two fingers. Finally, we have extended the form-closure result to grasps with one line and two point contacts, resulting in $O(n^2\log^2 n+K)$ and $O((n+K)\log n)$ algorithms for arbitrary and rectilinear polygons respectively, where K is the size of the output.

In the process of reformulating the problems of computing all grasps into efficiently-solvable geometric searching problems we seem to rely quite heavily on the fact that the part under consideration is polygonal. An obvious question that arises is whether it is possible to obtain output-sensitive algorithms for non-polygonal parts as well.

Another open question concerns the extension of our results to three-dimensional (polyhedral) parts. Intuitively, it seems much harder to obtain output-sensitive algorithms for computing form-closure grasps, which require up to seven fingers [4], than for computing 2nd-order-immobility grasps, which may take up to four fingers [14, 15]. In order to translate the problem of computing three-dimensional form-closure grasps into a geometric searching problem, it would be extremely useful to have a three-dimensional version of Reuleaux' graphical method.

Acknowledgement

The authors thank Elon Rimon for useful discussions.

References

[1] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry–Algorithms and Applications*, Springer-Verlag Berlin (1997).

[2] R.C. Brost and K.Y. Goldberg, A complete algorithm for synthesizing modular fixtures for polygonal parts, *IEEE Tr. on Robotics and Automation* **12** (1996), pp. 31-46.

[3] C. Ferrari and J.F. Canny, Planning optimal grasps, *Proc. IEEE Int. Conf. on Robotics and Automation* (1992), pp. 2290-2295.

[4] X. Markenscoff, L. Ni, and C.H. Papadimitriou, The geometry of grasping, *Int. J. of Robotics Research* **9** (1990), pp. 61-74.

[5] X. Markenscoff and C.H. Papadimitriou, Optimum grip of a polygon, *Int. J. of Robotics Research* **8** (1989), pp. 17-29.

[6] B. Mirtich and J.F. Canny, Easily computable optimum grasps, *Proc. IEEE Int. Conf. on Robotics and Automation* (1994), pp. 739-747.

[7] B. Mishra, J.T. Schwarz, and M. Sharir, On the existence and synthesis of multifinger positive grips, *Algorithmica* **2** (1987), pp. 541-558.

[8] V.-D. Nguyen, Constructing force-closure grasps, *Int. J. of Robotics Research* **7** (1988), pp. 3-16.

[9] M. Overmars, A. Rao, O. Schwarzkopf, and C. Wentink, Immobilizing polygons against a wall, *Proc. 11th Ann. ACM Symp. on Computational Geometry* (1995), pp. 29-38.

[10] J. Ponce and B. Faverjon, On computing three-finger force-closure grasps of polygonal objects, *IEEE Tr. on Robotics and Automation* **11** (1995), pp. 868-881.

[11] J. Ponce, D. Stam, and B. Faverjon, On computing force-closure grasps of curved two-dimensional objects, *Int. J. of Robotics Research* **12** (1993), pp. 263-273.

[12] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet, On computing four-finger equilibrium and force-closure grasps of polyhedral objects, *Int. J. of Robotics Research* **16** (1997), pp. 11-35.

[13] F. Reuleaux, *The Kinematics of Machinery*, Macmilly and Company (1876); republished by Dover (1963).

[14] E. Rimon and J.W. Burdick, Mobility of bodies in contact–Part I: A 2nd-order mobility index for multiple-finger grasps, *IEEE Tr. on Robotics and Automation* **14** (1998), pp. 696-708.

[15] E. Rimon and J.W. Burdick, Mobility of bodies in contact–Part II: How forces are generated by curvature effects, *IEEE Tr. on Robotics and Automation* **14** (1998), pp. 709-717.

[16] A.F. van der Stappen, C. Wentink, and M.H. Overmars, Computing form-closure configurations, *Proc. IEEE Int. Conf. on Robotics and Automation* (1999).

[17] A. Sudsang, J. Ponce, and N. Srinivasa, Algorithms for constructing immobilizing fixtures and grasps of three-dimensional objects, in *Algorithms for Robotic Motion and Manipulation* (J.-P. Laumond and M. Overmars (Eds.)), A.K. Peters, Wellesley MA (1997), pp. 363-380.

[18] R. Wagner, Y. Zhuang, and K.Y. Goldberg, Fixturing parts with seven modular struts, *Proc. IEEE Int. Symp. on Assembly and Task Planning* (1995), pp. 133-139.

[19] A. Wallack and J.F. Canny, Planning for modular and hybrid fixtures, *Proc. IEEE Int. Conf. on Robotics and Automation* (1994), pp. 520-527.

[20] A. Wallack and J.F. Canny, Modular fixture design for generalized polyhedra, *Proc. IEEE Int. Conf. on Robotics and Automation* (1996), pp. 830-837.

[21] C. Wentink, A.F. van der Stappen, and M.H. Overmars, Algorithms for fixture design, in *Algorithms for Robotic Motion and Manipulation* (J.-P. Laumond and M. Overmars (Eds.)), A.K. Peters, Wellesley MA (1997), pp. 321-346.

[22] C. Wentink, Fixture planning–Geometry and algorithms, PhD thesis, Dept. of Computer Science, Utrecht University (1998).