Caging Polygons with Two and Three Fingers

Mostafa Vahedi and A. Frank van der Stappen

Department of Information and Computing Sciences, Utrecht University, {vahedi, frankst}@cs.uu.nl

Abstract: We present algorithms for computing *all* placements of two and three fingers that cage a given polygonal object with n edges in the plane. A polygon is caged when it is impossible to take the polygon to infinity without penetrating one of the fingers. Using a classification into squeezing and stretching cagings, we provide an algorithm that reports all caging placements of two disc fingers in $O(n^2 \log n)$ time. Our result extends and improves a recent solution for point fingers. In addition, we construct a data structure requiring $O(n^2)$ storage that can answer in $O(\log n)$ whether two fingers in a query placement cage the polygon. We also study caging with three point fingers. Given the placements of two so-called base fingers, we report all placements of the third finger so that the three fingers jointly cage the polygon. Using the fact that the boundary of the set of placements for the third finger consists of equilibrium grasps, we present an algorithm that reports all placements of the third finger in $O(n^6 \log^2 n)$ deterministic time and $O(n^6 \log n (\log \log n)^3)$ expected time. Our results extend previous solutions that only apply to convex polygons.

1 Introduction

The caging problem was posed by Kuperberg in [7] as a problem of designing an algorithm for finding a set of points that prevents a polygon from moving arbitrarily far from a position. In other words, a polygon is caged when it is impossible to take it to infinity without penetrating a finger. However, to solve the problem it is easier to keep the polygon fixed and move the fingers instead, keeping their mutual distances fixed.

Caging is related to the notions of form (and force) closure grasps (see e.g. Mason's text book [9]), and immobilizing and equilibrium grasps [12]. Rimon et al. [11] introduced the notion of a caging set (or capture region) as the set of placements of fingers that may not immobilize the object but may prevent it from escaping to infinity. A comprehensive review on caging and related problems can be found in [1]. Caging sets have been applied to several problems in manipulation, such as grasping and in-hand manipulation, mobile

robot motion planning, parts feeding, and stable pose computation (see [4] for the references).

Using stratified Morse theory, Rimon and Blake [11] showed that in a two-fingered one-parameter gripping system, the hand's configuration at which the cage is broken corresponds to a frictionless equilibrium grasp. These results are extended by Davidson and Blake [2] to a three-fingered one-parameter gripper.

In the problem of caging a polygon with two disc fingers, it is required to compute all placements of two disc fingers that cage a polygonal object. This problem was first tackled by Sudsang and Luewirawong [14] by computing an acceptable distance for every pair of immobilizing vertices independent of the entire body of the polygon. As a result, the algorithm is not complete as it reports only a subset of all caging placements of two disc fingers. Independently Pipattanasomporn and Sudsang [10] have recently also solved the problem for point fingers in $O(n^2 \log n)$, and also provided a data structure capable of answering queries in $O(\log n)$. However the disc-finger problem has not been analyzed in their paper.

In the problem of caging a polygon with three fingers, the placements of two fingers, called the base fingers, are given. It is required to find all placements of the third finger, such that the resulting fingers cage the polygon. Sudsang [13] stated a sufficient (but not necessary) condition for caging a convex object in the plane with more than two fingers using the width of the object. For a non-convex object it was proposed to divide the object into convex parts and to consider the maximum width sub-part. The concept of the caging set was used by Sudsang and Ponce [15] as a basis for computing a plan for manipulating polygonal objects using three discs. The resulting caging set was very small and hence not complete, as the computation only takes three edges into account. Erickson et al. [4] provided the first complete algorithm for three-finger cagings of convex polygons. Two fingers are placed along the boundary of the polygon and then a region —the caging set— is computed for the third finger. An exact algorithm is provided that determines this region in $O(n^6)$ time, and also an approximation algorithm is provided with pre-specified accuracy. However, the problem of computing the set of all caging placements for non-convex polygons remained open, and is tackled in this paper.

In the first part of this paper (in Section 3) a solution for computing all caging placements of two disc fingers is presented. In addition a data structure is presented that requires $O(n^2)$ storage and is capable of answering in $O(\log n)$ whether a given placement of two fingers is caging. A given placement of two fingers is *squeezing* (*stretching*) caging, if it is caging and anyhow closing (opening) the fingers without penetrating the polygon the fingers remain caging until they reach a minimum (maximum) in which both fingers are on the boundary. It can be proven that any caging is *squeezing* or *stretching*. Using this fact, our work for two fingers extends and improves the result in [10] by providing an algorithm for computing all caging placements of two

disc fingers that runs in $O(n^2 \log n)$. To solve the problem we have employed pseudo triangulation, cell decomposition, connectivity graphs, and an event processing technique.

In the second part of this paper (in Section 4), the solution for computing all caging placements of three fingers with a fixed pair of base fingers is presented. It is shown that the boundary of the caging regions made by the third finger corresponds to equilibrium grasps. Our work on three-finger caging uses this fact to extend the result in [4] by providing the first complete algorithm for computing the set of all caging placements for non-convex polygons. To solve this problem we have used similar techniques as in our two fingers solution. The running time of the proposed three point-finger caging algorithm is $O(n^6 \log^2 n)$ deterministic time, and $O(n^6 \log n (\log \log n)^3)$ expected time.

2 Definitions and Assumptions

The given simple closed polygon P has no holes and is bounded by n edges. Let P_d be the Minkowski sum outer-face of P and a closed disc of radius d. More formally $P_d = P \oplus \Delta_d$[1], where Δ_d is the disc of radius d centered at the origin. Placing disc fingers of radius d around P is equivalent to placing point fingers around the generalized polygon P_d. A generalized polygon is a shape bounded by straight segments and circular arcs. As the holes of P_d correspond to placements of a single finger caging P, we discard these holes in our computations of two and three finger cagings.

Without loss of generality we can assume that P_d is enclosed in a sufficiently large rectangle B. Let $F \subset \mathbb{R}^2$ be $F = B \setminus \text{int}(P_d)$. Clearly F is the set of all possible placements for a finger. Throughout the paper we assume that the fingers are points, P is P_d, and we refer to F^2 $(= F \times F)$ and F^3 $(= F \times F \times F)$ as the admissible space for two and three fingers.

To solve the caging problem with two disc fingers, we decompose F into pseudo triangles. A pseudo triangle here is defined as a triangle that has at most one concave circular arc of radius d, and arc angle less than π. To obtain a pseudo-triangulation, new vertices may be added, but no vertex of a pseudo triangle should lie inside the edge or arc of one of its neighbor triangles.

A force vector with its application point fixed uniquely determines a wrench. Our model for wrenches induced by a given polygon differs from that of Markenscoff et al. [8] in the case of a point contact and a convex vertex. Consider a point on a convex vertex of a polygon. In our model, the possible wrenches for this point, similar to the concave case, contains the set of all convex combinations of the two unit normal wrenches to both incident edges which makes a cone. The intuition behind this model is that an ϵ-radius disc-finger on a convex vertex can apply any wrench being a combination of the two normal wrenches with ϵ adjustment.

[1] $A \oplus B = \{a + b | a \in A, b \in B\}$

Using Corollary 4.1 from [12], the grasp made by fingers on edges is an equilibrium grasp if and only if the wrench vectors meet in a common point and the angle between two consecutive wrenches is not more than π. When some of the fingers are at vertices, the grasp is an equilibrium grasp provided that there is a point that lies on the intersection of cones and wrenches and satisfies the angle condition. The number of fingers that make an equilibrium grasp is the number of fingers that exert a non-zero wrench on the object.

3 Two Fingers Caging

In this section the solution for the caging problem of a polygon P with two disc fingers with the same fixed radius is presented. In this problem all placements of two fingers that cage the polygon is computed. It is also required to answer quickly whether a given placement of two fingers is caging.

Let C_P be the set of all two-finger caging placements of a polygon P and let $C_{\delta,P}$ be the set of all caging placements for which the fingers are δ apart. Clearly $C_P = \bigcup_{0<\delta\in R} C_{\delta,P}$. Increasing or decreasing δ, the topology of $C_{\delta,P}$ changes at certain critical δ's. We use this fact to construct the set of all cagings.

Let (p_1, q_1) and $(p_2, q_2) \in F^2$ be two placements of a two-finger hand. These placements are δ-*reachable* if $|p_1q_1| = |p_2q_2| = \delta$ and both of them lie in the same connected component of the admissible space of finger placements that are δ apart. They are δ-*max-reachable* (δ-*min-reachable*) if $|p_1q_1|, |p_2q_2| \leq \delta$ ($|p_1q_1|, |p_2q_2| \geq \delta$) and both of them lie in the same connected component of the admissible space of finger placements that are at most (at least) δ apart. When two placements are δ-*reachable*, δ-*max-reachable*, or δ-*min-reachable*, it is possible to move the two-finger hand between the placements keeping the distance of the fingers fixed, at most δ, or at least δ respectively. Note that when two placements are δ-*max-reachable* (δ-*min-reachable*) they are δ'-*max-reachable* (δ'-*min-reachable*) for any $\delta' \geq \delta$ ($\delta' \leq \delta$). Hence, if a δ-apart placement is not δ-*max-reachable* (δ-*min-reachable*) to a δ-apart placement being remote from the polygon, it is squeezing (stretching) caging. If the reachability type is clear from the context or if we want to define something for all types of reachability we will use just the word *reachable*. Because of the lack of space we just mention the following important fact that provides the basis for our approach outlined in Subsection 3.2.

Lemma 3.1 *Given one obstacle in the plane, if two placements* (p_1, q_1) *and* (p_2, q_2) *of a two-finger hand satisfying* $|p_1q_1| = |p_2q_2| = \delta$ *are both* δ-max-reachable *and* δ-min-reachable*, then they are* δ-reachable*.*

The direct result of Lemma 3.1 is that, if a placement is caging, then it is squeezing caging, stretching caging, or both. In Figure 1(a) a shaded polygon and four δ apart placements (p_1, q_1), (p_2, q_2), (p_3, q_3) and (p_4, q_4) are

shown. No two placements are δ-reachable, but (p_1, q_1) and (p_2, q_2) are δ-max-reachable while (p_1, q_1) and (p_3, q_3) are δ-min-reachable. Moreover (p_1, q_1) is not caging, (p_2, q_2) is stretching caging, (p_3, q_3) is squeezing caging, and (p_4, q_4) is both stretching and squeezing caging.

Based on above reachability notions and a pseudo triangulation of the set F, the space F^2 is decomposed into constant-complexity 4D cells for every δ, such that all placements inside each cell are *reachable* from each other. The required property of the pseudo triangulation is stated in the following lemma. The construction is relatively easy and therefore we confine ourselves to mentioning the result.

Lemma 3.2 *It is possible to decompose F in $O(n \log n)$ time in $O(n)$ pseudo triangles such that every pseudo triangle has a constant number of neighbors.*

Based on δ and the cell decomposition of F^2, a *connectivity graph* is defined in Subsection 3.1, with cells as the nodes, and two neighbor nodes are connected by an edge if there are *reachable* placements inside the corresponding cells. Note that all the corresponding placements of one node are either all caging or all noncaging. Hence we associate with a node the caging status (caging or noncaging) of all its placements. Since all noncaging placements are *reachable* from each other, the noncaging nodes form a connected component in the connectivity graph. Therefore, all components in the graph except the one containing noncaging placements represent a set of caging placements.

To compute all caging placements of two fingers, it is possible to start from zero and increase (or equivalently start from a largely enough distance and decrease) the distance of the fingers. Meanwhile, there are *critical distances* at which the connectivity graph changes. The idea is to compute all critical distances and sort them increasingly. Clearly between two consecutive critical distances, the connectivity graph does not change. Therefore it is possible to compute all possible connectivity graphs for all distances by considering the critical distances one by one, and updating the connectivity graph accordingly in a reasonable time (instead of computing the whole connectivity graph from scratch every time). When a *caging* cell merges to or becomes disconnected from the *noncaging* cell, equivalently a connected component of the graph respectively merges to or becomes disconnected from the component of the graph that represents the noncaging placements. To update the graph for every merging or splitting of cells some edges respectively should be added to or deleted from the graph as the update operation. If the update operation includes deletion of edges the components of the graphs should be maintained during the process and it is not possible to do this operation in constant time (at least easily). But if it consists of just addition of edges the components will just merge or emerge and therefore it is possible to maintain the two types of components in constant amortized time. By using the squeezing/stretching fact, defined formally in Subsection 3.1, and increasing/decreasing the distance the cells just merge or emerge and therefore the update operation just include addition of nodes and edges in constant time. Whenever a caging node

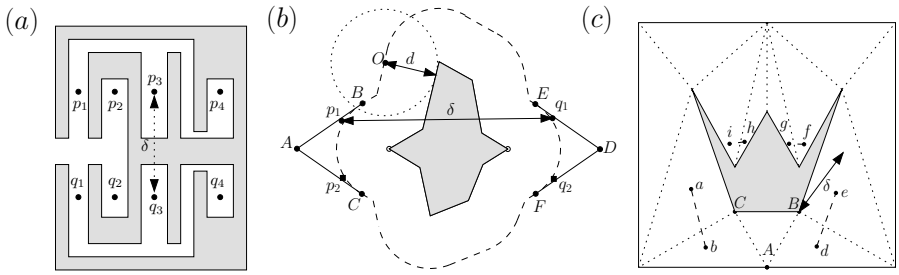

Fig. 1. (a) Reachability notions and caging types, (b) $R^M_\delta(ABC, DEF)$ has one cell, (c) connectivity subgraph of distance δ when fingers are points and the first finger is inside ABC

is going to join a noncaging node by a path, the corresponding 4D cell of the caging node is reported as a set of caging placements; any placement inside this cell is caging. The complete algorithm and the running time analysis is explained in Subsection 3.2.

3.1 Two Fingers Squeezing and Stretching Caging

Let s and s' be two closed subsets of F. The set of admissible placements induced by a pair of subsets of F for two fingers with distance δ is the set

$$R_\delta(s, s') = \{(p, q) \in s \times s' \mid |pq| = \delta\}.$$

The set $R_\delta(s, s')$ consists of a number of 4D connected components. Every connected component corresponds to a set of δ-*reachable* placements. Let $R^M_\delta(s, s')$ be the set of connected components of $R_\delta(s, s')$. Therefore every member of $R^M_\delta(s, s')$ is a subset of F^2 and is called a *cell*. If s and s' have constant complexity, the number of cells in $R^M_\delta(s, s')$ and their complexity will be constant too. Figure 1(b) shows a shaded polygon, its Minkowski-sum outer-face with a disc of radius d (displayed dashed), and two pseudo-triangles ABC and DEF. Here $R^M_\delta(ABC, DEF)$ has one cell. It seems that (p_2, q_2) is not δ-*reachable* from (p_1, q_1) using the placements inside the two pseudo-triangles; but the placement is reachable from (p_1, q_1) by moving p_1 toward B and moving q_1 toward F and then moving p_1 toward C.

Let T be a *suitable* pseudo triangulation of F (i.e. satisfies Lemma 3.2). The connectivity graph $CG_{\delta,T}(V, E)$ for T and distance δ is defined by

$$\begin{cases} V = \{r \subset F^2 \mid \exists\ t, t' \in T : r \in R^M_\delta(t, t')\}, \\ \\ E = \{\ (r_1, r_2) \in V^2 \mid \exists\ t_1, t'_1, t_2, t'_2 \in T : r_1 \in R^M_\delta(t_1, t'_1), \\ r_2 \in R^M_\delta(t_2, t'_2) \wedge \exists r \in R^M_\delta(t_1 \cup t_2, t'_1 \cup t'_2) : r = r_1 \cup r_2\ \}. \end{cases}$$

By definition every cell is assigned a unique node and therefore every admissible placement of fingers is assigned a node in the graph; there is no edge

between the cells of a set $R^M_\delta(t, t')$. There is an edge between two cells in $R^M_\delta(t_1, t'_1)$ and $R^M_\delta(t_2, t'_2)$ when there is a cell in $R^M_\delta(t_1 \cup t_2, t'_1 \cup t'_2)$ that contains the two cells. In other words there is an edge between two nodes when their corresponding pairs of pseudo triangles are neighbor and their corresponding placements are *reachable*. As every pseudo triangle has a constant number of neighbors in T, the total number of edges is linear in the total number of nodes. Therefore if there are $O(n)$ pseudo triangles in T, there will be $O(n^2)$ nodes and edges in $CG_{\delta,T}(V, E)$.

In Figure 1(c) a shaded polygon bounded in a rectangle, the polygon exterior triangulated (displayed dotted), and one of the triangles ABC are shown. Since it is not easy to draw the whole connectivity graph for distance δ, we show a subset of the graph for point fingers while the first finger is inside ABC. For any triangle if there are δ apart points inside that triangle and ABC, then consider a node in the graph (the nodes are displayed with small letters). Since there are no two points of distance δ inside ABC no node is considered for it. Note that here since no $R^M_\delta(ABC, t)$ has more than one cell, every pair of (ABC, t) has at most one node in the graph. When it is possible to move the second finger from one triangle to its neighbor while keeping the first finger inside ABC and the distance δ, then connect the two corresponding nodes by an edge (displayed with dashed segments).

Lemma 3.3 *Let* $(p_1, q_1) \in r_1 \in R^M_\delta(t_1, t'_1)$ and $(p_2, q_2) \in r_2 \in R^M_\delta(t_2, t'_2)$ *be two placements.* (p_1, q_1) *and* (p_2, q_2) *are* δ-reachable*, if and only if there is a path between* r_1 *and* r_2 *in* $CG_{\delta,T}(V, E)$.

Let v_δ be a noncaging node for which $v_\delta \in R^M_\delta(t_\delta, t'_\delta)$, and $t_\delta, t'_\delta \in T$. Without loss of generality, we can assume that it is possible to compute v_δ based on B, T, and δ such that is not caging (clearly v_δ may change when δ changes). The points on the boundary of B at distance δ e.g. do not cage the polygon being remote from the polygon. Using the fact that all noncaging nodes form a connected component in the graph, a given placement of fingers is caging if and only if there is no path between the corresponding node and v_δ in the graph. Let the set of caging nodes for the polygon P and distance δ be the set

$$CN_{\delta,P} = \{v \in V(CG_{\delta,T}) \mid \text{There is no path in } CG_{\delta,T} \text{ between } v \text{ and } v_\delta\}.$$

and let the set of caging placements obtained by CG graphs be the set

$$CN_P = \{(p, q) \in F^2 \mid \exists v \in CN_{|pq|,P} : (p, q) \in v\}.$$

Consider the following definitions of $R'_\delta(s, s')$ and $R''_\delta(s, s')$ that correspond to the δ-*max-reachable* and δ-*min-reachable* set of placements induced by s and s':

$$R'_\delta(s, s') = \{(p, q) \in s \times s' \mid |pq| \leq \delta\},$$

$$R''_\delta(s, s') = \{(p, q) \in s \times s' \mid |pq| \geq \delta\}.$$

Replacing $R_\delta(s, s')$ with $R'_\delta(s, s')$ and $R''_\delta(s, s')$, and *δ-reachable* with *δ-max-reachable* and *δ-min-reachable* in above definitions results in new definitions of ${R'}_\delta^M(s, s')$ and ${R''}_\delta^M(s, s')$, $CG'_{\delta,T}$ and $CG''_{\delta,T}$, $CN'_{\delta,P}$ and $CN''_{\delta,P}$, and CN'_P and CN''_P respectively in order. The adjusted Lemma 3.3 still holds for $CG'_{\delta,T}$ and $CG''_{\delta,T}$. We refer to $CG'_{\delta,T}$ and $CG''_{\delta,T}$ as the max and min connectivity graph and to CN'_P and CN''_P as the set of all squeezing and stretching caging placements respectively. Because of the lack of space we just mention the following important facts. Lemma 3.5 is the direct result of Lemma 3.1.

Lemma 3.4 *Given a polygon P and a distance δ, it is possible to compute $CG_{\delta,T}$, $CG'_{\delta,T}$, $CG''_{\delta,T}$, $CN_{\delta,P}$, $CN'_{\delta,P}$, and $CN''_{\delta,P}$ in $O(n^2)$. Then using T it is possible to determine in $O(\log n)$ time whether a given placement of two disc fingers that are δ apart cages P.*

Lemma 3.5 $C_P = CN_P = CN'_P \cup CN''_P$.

As it was mentioned in Section 3, it is possible to maintain the graph components in computing the squeezing and stretching caging placements in constant amortized time. Therefore, based on Lemma 3.5 that states the relation between the common notion of caging on one hand and squeezing and stretching caging at the other hand, we compute all caging placements by computing CN'_P and CN''_P separately in Subsection 3.2.

3.2 Two Disc Fingers Caging Algorithm

In this section we present our approach to solving the problem of finding all caging placements of two disc fingers of equal radius. To report all cagings the algorithm uses Lemma 3.5 and reports the two sets CN'_P and CN''_P instead, which both consist of 4D cells corresponding to squeezing and stretching cagings respectively. Each point inside every cell corresponds to a placement of two disc fingers on the plane that cages P. The algorithm consists of three steps for both types of cagings. Since these steps are similar for both types, we focus on the computation of the set CN'_P of squeezing cagings:

1. find and sort the critical distances (see below) induced by all pairs of pseudo-triangles in T (a *suitable* pseudo-triangulation of F),
2. compute $CG'_{\delta,T}$ for all δ by processing the critical distances and updating $CG'_{\delta,T}$ accordingly, meanwhile reporting the possible squeezing caging-placements,
3. report the remaining squeezing caging-placements.

Hereafter we use pseudo triangle and triangle interchangeably. The first step is based on the fact that the structure of $CG'_{\delta,T}$ only changes at particular values of δ, to which we shall refer as *critical distances*. Increasing δ from zero, we distinguish three types of critical distances induced by a single pair (t, t'):

1. $|R'^{M}_{\delta}(t,t')|$ increases,
2. $|R'^{M}_{\delta}(t,t')|$ decreases,
3. for neighbor pairs of (t_1,t'_1) and (t_2,t'_2) in $R'^{M}_{\delta}(t_1 \cup t_2, t'_1 \cup t'_2)$, a member of $R'^{M}_{\delta}(t_1,t'_1)$ merges with a member of $R'^{M}_{\delta}(t_2,t'_2)$.

The cells of a pair of triangles can only merge and not split, because when two placements become δ-*max-reachable* they remain so for any bigger δ. Therefore, the first type only occurs when a cell emerges, and the second only occurs when two cells merge together. Since the first two types of critical distances depend on t and t' only, the number of such distances is constant for a given t and t'. From the fact that all pseudo-triangles in T have a constant number of neighbors and also $R'^{M}_{\delta}(t,t')$ has constant cardinality, it follows that the number of critical distances of the latter type is constant as well. As a result, we can accomplish the computation and sorting of all $O(n^2)$ critical distances in $O(n^2 \log n)$ time.

In the last two steps we use a graph-based data structure to keep track of the changes in $CG'_{\delta,T}$ while increasing δ. When $R'^{M}_{\delta}(t,t')$ changes topologically for a critical distance δ, a new cell emerges or two cells merge into a single cell containing the original ones. For each newly emerging or merging cell there is a corresponding node in the graph and all nodes are included in the graph from the start. With every node in the graph we associate a caging status, *caging* or *noncaging*. Initially all nodes are *caging*. Every node also has a *critical distance* that will be determined later; initially it is set to zero. Every pair of triangles has a corresponding set of nodes in the graph and a set that specifies the current nodes in the graph for the current value of δ.

Starting from zero with $CG'_{0,T}$, which is built from scratch, the second step processes the critical distances in order to update the connectivity graph. Since P does not contain holes, there is no caging node in $CG'_{0,T}$. At each critical distance some actions are taken to determine $CG'_{\delta,T}$ from $CG'_{\delta',T}$, in which δ and δ' are the current and previous critical distances respectively. We recall that between two consecutive critical distances the graph does not change. The actions taken to update the graph depend on the type of critical distance. The first two sets of actions are the same and they are not repeated. They follow in order:

1,2. The current set of nodes for the corresponding pair of triangles is updated. The edges for the new node are computed. If there is no edge or the new edges only connect to caging nodes, the caging status of the node is 'caging'. Otherwise the caging status is 'noncaging'. If the corresponding node is a bridge between a caging node and a noncaging node, a maximal report (see below) is performed; otherwise the *critical distances* of the old nodes (if existing) are set to the current value of δ.

3. An edge is added between the corresponding nodes. If the nodes had a different caging status, a maximal report is performed.

A *maximal report* is done, when the caging status of a node changes from caging to noncaging. This happens for squeezing caging-placements for which

δ reaches the critical maximum distance, at which any distance larger than that distance allows the fingers to escape. Look at Figure 2(a) for some of the critical maximum distances that lead to *maximal report*. In this operation, the corresponding 4D cells of all the nodes in the graph that are in the same connected component of the changing node are reported, and their associated *critical distances* are set to the current value of δ and their associated caging statuses are set to 'noncaging'. If a node becomes 'noncaging', it can never become 'caging' again. Hence every node is reported at most once, and the time devoted to reporting the caging cells is linear in the number of nodes and therefore is $O(n^2)$.

Every update operation takes constant time. Clearly every change is local to a node and its neighbors. Since no node is added to the graph, the addition of edges is the only performed operation; the number of neighbor nodes and the number of edges for each node is constant. Therefore, the updates induced by a single critical distance take constant time in total.

Since some of the squeezing cagings may have no critical maximum distance (e.g. (p_4, q_4) in Figure 1(a)), all the remaining squeezing cagings are reported in a separate step at the end. In the third step the final connectivity graph is traversed for nodes that are 'caging' but their *critical distance* fields are still zero. For every such node, the field is set to infinity and its 4D cells is reported. The following theorem follows from the preceding discussion.

Theorem 3.6 *Given a polygon with n edges and two disc fingers of equal radius, it is possible to report all the caging placements in $O(n^2 \log n)$ time.*

Theorem 3.7 *It is possible to compute a data structure with $O(n^2)$ space and time complexity, with which it is possible to answer in $O(\log n)$ whether a given placement of fingers is caging.*

Proof. To answer the caging query using the two final (squeezing and stretching) computed data structures, the corresponding pseudo triangles are located in $O(\log n)$ [3]. Then for the given distance of the fingers, the corresponding two nodes in the graphs can be determined in constant time. There are two cases; If the caging status of any node is caging, the answer is caging. Otherwise, the critical distance field of every node is compared with the query distance. For squeezing/stretching caging, if the query is smaller/larger the answer is caging; otherwise the answer is noncaging. Therefore the total time to answer a query is $O(\log n)$. Clearly the space needed to store the data structure is $O(n^2)$.

4 Three Fingers Caging

In this section, the caging problem for three point fingers is presented. In this problem the placement of two fingers, called the base fingers, is given. It is required to find all placements of the third finger, such that the fingers cage

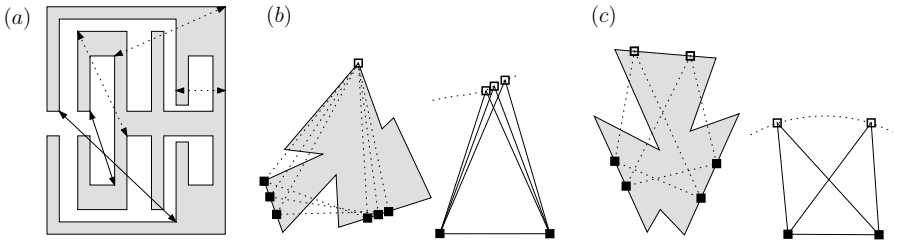

Fig. 2. (a) Three critical maximum distances displayed with dotted arrows and two critical minimum distances displayed with solid arrows. (*b* and *c*) Two loci are displayed for two polygons at the right side in which the filled boxes represent the base fingers and dotted triangles represent equilibrium grasps

the polygon. The caging placements form some regions on the plane of which the boundaries should be reported. We assume that the base fingers do not cage the polygon without the third finger.

Similar to the solution of the two fingers case, F is triangulated and the *connectivity graph* for F^3 is defined for a given triangulation T and a given vector of distances of three fingers δ. Therefore $R_\delta(s, s', s'')$, $R^M_\delta(s, s', s'')$, $CG_{\delta,T}$, and $CN_{\delta,P}$ are defined similarly. Because of the similarity we have not repeated the definitions and the lemmas, except for the following important lemma.

Lemma 4.1 *Given T and δ, it is possible to compute $CG_{\delta,T}$ and $CN_{\delta,P}$ in $O(n^3)$ and to answer queries about caging status of given fingers placements with δ distances in $O(\log n)$.*

In Subsection 4.2 it is shown that the third finger placed on a point on the caging boundary jointly with the given placements of the base fingers correspond to some equilibrium grasps. It does not mean that the fingers necessarily make an equilibrium grasp at that placement, but there is a corresponding placement, reachable from that placement, at which the fingers make an equilibrium grasp. Every curve on the caging boundaries corresponds to a set of equilibrium grasps that induced by the same pair or triple of features of the polygon. Therefore, it is possible to compute all curves on the caging boundaries by considering every pair or triple of features and computing the possible equilibrium grasps. The resulting grasps when moved to the fixed placement of the base fingers define some 2D curves which we call *curves of equilibrium grasps*; only some parts of these curves constitute the caging boundaries. Based on these facts, the idea is to compute all possible caging intervals on every curve, and then to compute the caging boundaries using the caging intervals. In Subsection 4.1 all equilibrium grasps involving the two base fingers are computed. Note that the base fingers have a fixed distance.

To compute the caging intervals on each *curve of equilibrium grasps* the vector of distances, δ, is altered by changing the position of the third finger

along that curve. Therefore for every point on the curve, $CG_{\delta,T}$ is computed accordingly. Similar to the two fingers case, there are critical points on the curve at which $CG_{\delta,T}$ changes and it does not change between two consecutive critical points. The same event processing approach is employed here to compute all possible $CG_{\delta,T}$ when the third finger moves along the curve. It is shown that the total number of possible nodes is $O(n^3)$ and all of them are included in the graph from the start; so there is no need to add or remove nodes from the graph. Here in contrast to the two fingers case, the update operation may require deletion of edges beside addition because an existing cell may split. Therefore, we have to use a special data structure called fully dynamic graph to efficiently query the caging property each time. The complete algorithm and the running time analysis is explained in Subsection 4.2.

4.1 Locus of Three Fingers Equilibrium Grasps

Consider all possible equilibrium grasps involving three fingers, two of which —referred to as base fingers— have a fixed distance, and the triangles defined by the fingers for every such grasp. Now consider two fixed points in the plane with distance equal to that between the base fingers, and draw the triangles such that the fixed points are on the base fingers. It is required to find the locus of the third finger in the plane. There are two general cases depending on the number of fingers that make the equilibrium grasp:

1. two fingers on
   a) two edges: the two edges should be parallel, hence it is possible to move the fingers together in one direction along the edges: the locus of all placements of the third finger describes a circular arc;
   b) an edge and a vertex, or two vertices: the locus of all placements of the third finger describes a circular arc;
2. three fingers with
   a) a base finger at a vertex: the locus of all placements of the third finger describes a line segment;
   b) the third finger at a vertex: the locus of all placements of the third finger describes a limaçon of Pascal or a line segment;
   c) all fingers on edges: the locus of all placements of the third finger describes a circular arc or a line segment;
   d) a base finger and the third finger at vertices: the locus of all placements of the third finger describes a finite number of points.

In Figure 2(b and c) two loci are displayed for two polygons. The filled boxes represent the base fingers and the empty boxes represent the third finger. Each dotted triangle represents an equilibrium grasp and is redisplayed at the right side with solid lines. For the polygon (b), all three fingers are on edges for which the locus is a circular arc. For the polygon (c), the third finger is at a vertex and the base fingers are on two edges for which the locus is a limaçon

of Pascal arc. The loci are displayed with dotted curves at right side above the triangles for each polygon.

Since the boundary of the caging regions consists of continues curves, the points of case 2.d are not relevant and can be discarded. Since the number of features is at most three, there are $O(n^3)$ curves.

Theorem 4.2 *The locus of all equilibrium grasps made by three fingers of which the distance of base fingers is fixed, when moved to a fixed placement of the base fingers, defines $O(n^3)$ constant complexity 2D curves.*

4.2 Three Fingers Caging Algorithm

We report all placements of a point finger such that it cages P together with the two given base fingers. The output of the algorithm is a set of regions. Each point inside every region corresponds to a placement of the third finger that cages the polygon jointly with the base fingers.

Before explaining the algorithm, it should be shown that the boundary of the caging regions correspond to the boundary of the polygon or to sets of equilibrium grasps. Consider an intersection point of the caging boundary (not on the polygon boundary) and an arbitrary line. The intersection point is a puncture point (see [11] for the definition), because moving the third finger along the line, the caging status changes at that point. Considering the set of fingers consisting of the base fingers and the third finger moving on the line, Proposition 3.3 of [11] states that the corresponding placement corresponds to an equilibrium grasp. Therefore the caging boundaries correspond to the boundary of the polygon or to sets of equilibrium grasps.

Lemma 4.3 *The caging regions of a polygon are bounded by curves of equilibrium grasps or the polygon boundary.*

The algorithm consists of four steps:

1. Compute the locus of the third finger in all possible equilibrium grasps made with three fingers,
2. determine critical points related to a triple of triangles and a curve, and sort all of the critical points for every curve on that curve,
3. determine the caging intervals by computing all possible $CG_{\delta,T}$ for every curve by processing the sorted critical points,
4. report the caging boundaries using the computed caging intervals.

In the first step, all the equilibrium grasps induced by three fingers are computed as a set of curves, *curves of equilibrium grasps*. To ease the computation of caging intervals on the caging boundaries (in the last step), the polygon edges are added to the set of curves.

To explain the second step, the notion of a critical point should be defined first. Consider a curve E of equilibrium grasps and a point p on E at which the third finger is placed. It is possible to build a connectivity graph for p

and the base fingers. Moving the third finger along E, there are two groups of critical points on E related to a triple of triangles (t, t', t''):

1. $|R^M_\delta(t, t', t'')|$ changes,
2. (t_2, t'_2, t''_2) and (t_1, t'_1, t''_1) are neighbors and a member of $R^M_\delta(t_1, t'_1, t''_1)$ merges with or becomes disconnected from a member of $R^M_\delta(t_2, t'_2, t''_2)$ inside $R^M_\delta(t_1 \cup t_2, t'_1 \cup t'_2, t''_1 \cup t''_2)$.

In the second step, all critical points related to a triple of triangles and a curve are calculated, and then all of the critical points for every curve are sorted along that curve. Since both the complexity of the triangles and their neighbors and the complexity of each curve are constant, there are constant number of critical points for every triple of triangles. Considering all possible ordered triples of triangles, there are $O(n^3)$ critical points for each curve, including the intersection points of that curve with the polygon and other curves of equilibrium grasps.

In the third step, all possible $CG_{\delta,T}$ are computed for every curve by taking the critical points in order and updating $CG_{\delta,T}$ for that curve; meanwhile the caging intervals are calculated. The approach for every curve E is as follows. One of the critical points p on E is taken as the starting point. The same data structure that was used in the two fingers case is used, without critical distance field. Initially $CG_{\delta,T}$ is built from scratch for p and the base fingers. Changing the position of p on E according to sorted critical points, one of the following actions is taken to update the current $CG_{\delta,T}$, depending on the type of the critical point:

1. the current set of nodes for the corresponding ordered triple of triangles is updated and the edges for the new set of nodes are computed, or
2. an edge is added between or removed from the two corresponding nodes.

Similar to the two fingers case, the update operation on the graph can be done in constant time for every critical point. In addition, however, we need to know the caging status of the current placement. Therefore, it is required to maintain a special data structure to quickly answer whether the current placement is caging.

A placement is caging, if there is no path in the current $CG_{\delta,T}$ between the corresponding node and v_δ, a *noncaging* node (here too, the v_δ may change when δ changes). Using the fully dynamic graph data structure [6, 5], it is possible to query for the connectivity of two nodes in the graph in $O(\log n / \log\log n)$ time and to update the mentioned data structure in $O(\log^2 n)$ deterministic amortized time and in $O(\log n(\log\log n)^3)$ expected amortized time. To find v_δ, choose a placement remote from the polygon in F, and find the corresponding node in the graph by locating the containing ordered triple of triangles in $O(\log n)$ [3].

To properly compute the caging status on the boundary of caging regions we use the trapezoidal map of the arrangement of the curves of equilibrium

grasps. Since all the points inside a trapezoid have similar caging status, instead of choosing points on curves, we choose points exactly inside the trapezoids.

In the fourth step, the caging boundaries are reported from the previously computed caging intervals. To report the caging boundaries, first a curve is found on the boundary of each caging region. To do this, every caging interval of every curve is taken and one of its starting points is considered. Clearly, the starting point is on the caging boundary. Considering the caging intervals that include this point, there is a caging interval such that all the other ones lie on one side of it. This interval is on the boundary of a caging region, and the other intervals are inside this caging region. Walking along the corresponding curve such that the interior of the caging region is on the left, the next intersection point on the curve is considered. On every intersection point, the caging intervals are considered that include the intersection point and the rightmost curve is selected. The next intersection point along the selected curve is considered and the same steps are repeated till the same starting point is reached. The same is done for every unvisited caging interval. Recalling that our algorithm computes all three-finger caging regions, we get the following final result.

Theorem 4.4 *Given a polygon with n edges and given placements of base fingers, it is possible to report all placements of the third finger such that the three fingers jointly cage the polygon in $O(n^6 \log^2 n)$ deterministic time, and in $O(n^6 \log n(\log\log n)^3)$ expected time.*

Proof. F can be triangulated in $O(n \log n)$ time [3], and the number of triangles is linear. Since the total number of locus curves is $O(n^3)$ and computing every one takes a constant time, the first step can be done in $O(n^3)$ time (Theorem 4.2). Since every ordered triple of triangles has constant number of critical points, the total number of critical points on each curve is $O(n^3)$. Therefore the second step takes $O(n^6 \log n)$. In the fourth step, for each critical point, adding and removing edges takes constant time, but testing the connectivity takes $O(\log^2 n)$ deterministic and $O(\log n(\log\log n)^3)$ expected amortized time. Since there are $O(n^6)$ critical points, the third step takes $O(n^6 \log^2 n)$ deterministic time and $O(n^6 \log n(\log\log n)^3)$ expected time. Since every intersection point is visited at most once the fourth step takes $O(n^6)$ time. Hence, the algorithm takes totally $O(n^6 \log^2 n)$ deterministic time and $O(n^6 \log n(\log\log n)^3)$ expected time.

5 Conclusion

In this paper we have presented algorithms for computing all possible caging placements of two disc fingers of equal radius, and three point fingers of which the placements of two base fingers are given. In the case of three fingers, extending the results to disc-shaped fingers is straightforward. Although the

curves of equilibrium grasps become more complicated, their degrees remain constant. We intend to implement the algorithms to gain more insight into the shapes of caging regions and their combinatorial complexities with the purpose of improving the worst-case running time of our algorithm. In addition we would like to consider the three-finger caging query as well. Finally, extending the results to 3D seems challenging, because of the problem of decomposing a polyhedron into few simple cells. Hence we will look for alternative ways to tackle the caging problems.

References

1. A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *ICRA*, pages 348–353. IEEE, 2000.
2. C. Davidson and A. Blake. Caging planar objects with a three-finger one-parameter gripper. In *ICRA*, pages 2722–2727. IEEE, 1998.
3. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwartzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 1997.
4. J. Erickson, S. Thite, F. Rothganger, and J. Ponce. Capturing a convex object with three discs. In *ICRA*, pages 2242–2247. IEEE, 2003.
5. M. R. Henzinger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, 1999.
6. J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. In *STOC '98: Proceedings of the 30th annual ACM symposium on Theory of computing*, pages 79–89, New York, NY, USA, 1998. ACM Press.
7. W. Kuperberg. Problems on polytopes and convex sets. *DIMACS Workshop on Polytopes*, pages 584–589, 1990.
8. X. Markenscoff, L. Ni, and C. H. Papadimitriou. The geometry of grasping. *Int. J. Robotics Res.*, 9(1):61–74, 1990.
9. M. Mason. *Mechanics of Robotic Manipulation*. MIT Press, August 2001. Intelligent Robotics and Autonomous Agents Series, ISBN 0-262-13396-2.
10. P. Pipattanasomporn and A. Sudsang. Two-finger caging of concave polygon. In *ICRA*, pages 2137–2142. IEEE, 2006.
11. E. Rimon and A. Blake. Caging 2d bodies by 1-parameter two-fingered gripping systems. In *ICRA*, volume 2, pages 75–91. IEEE, 1995.
12. 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(5):696–717, 1998.
13. A. Sudsang. A sufficient condition for capturing an object in the plane with disc-shaped robots. In *ICRA*, pages 682–687. IEEE, 2002.
14. A. Sudsang and T. Luewirawong. Capturing a concave polygon with two disc-shaped fingers. In *ICRA*, pages 1121–1126. IEEE, 2003.
15. A. Sudsang and J. Ponce. A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane. In *ICRA*, pages 1068–1075. IEEE, 2000.