Largest Placements and Motion Planning of a Convex Polygon*

Pankaj K. Agarwal† Nina Amenta‡ Boris Aronov§ Micha Sharir¶

Abstract

We study two problems involving collision-free placements of a convex m-gon P in a planar polygonal environment: (i) We first show that the largest similar copy of P inside another convex polygon Q with n edges can be computed in $O(mn^2 \log n)$ time. We also show that the combinatorial complexity of the space of all similar copies of P inside Q is $O(mn^2)$, and that it can also be computed in $O(mn^2 \log n)$ time. (ii) We then consider the case where Q is an arbitrary polygonal environment with n edges. We give the first (and relatively simple) algorithm that constructs the *entire* free configuration space (the 3-dimensional space of all free placements of P in Q) in time that is near-quadratic in mn, which is nearly optimal in the worst case. Previous solutions of the second problem were either incomplete, more expensive, or produced only part of the free configuration space. Combining our solution with parametric searching, we obtain an algorithm that finds the largest placement of P in Q in time that is also near-quadratic in mn. In addition, we describe an algorithm that preprocesses the computed free configuration space so that 'reachability' queries can be answered in polylogarithmic time.

*Pankaj Agarwal has been supported by NSF Grant CCR-93–01259, an NYI award, and by matching funds from Xerox Corp. Nina Amenta has been supported by the Geometry Center, which is officially the Center for Computation and Visualization of Geometric Structures, supported by NSF/DMS-8920161. Boris Aronov has been supported by NSF Grant CCR-92-11541 and a Sloan Research Fellowship. Micha Sharir has been supported by NSF Grants CCR-94-24398 and CCR-93-11127, by a Max-Planck Research Award, and by grants from the U.S.-Israeli Binational Science Foundation, and the G.I.F., the German-Israeli Foundation for Scientific Research and Development.

†Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA

‡Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

§Department of Computer and Information Science, Polytechnic University, Brooklyn, NY 11201 USA

¶School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, ISRAEL and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA

1 Introduction

Problem statement. Let P be a (closed) convex m-gon. We consider two problems involving placements of P inside a (closed) planar polygonal *environment* Q bounded by a total of n edges. We allow P to translate, rotate, and sometimes also to scale. A *placement* of P is thus any congruent or similar copy of P (without reflections). A placement of P is *free* if it is fully contained in Q. A placement of a similar (resp. congruent) copy of P can be represented by four (resp. three) real parameters. The *free configuration space* $\mathcal{C}$ of P in Q is the space of all free placements of P in Q. In general, $\mathcal{C}$ is a four-dimensional space; if scaling is not allowed, $\mathcal{C}$ is three-dimensional. There are two types of problems that we wish to consider in this setup:

Motion Planning: Assuming no scaling is allowed, construct the full configuration space $\mathcal{C}$. Also, preprocess $\mathcal{C}$ so that, given any two (congruent) placements of P, one can determine whether they lie in the same connected component of $\mathcal{C}$ (in which case there exists a collision-free motion of P inside Q from one of these placements to the other).

Largest Placement: Allowing scaling, find the largest similar copy of P inside Q.

Previous results: General environment. Both problems are important basic problems in robotics

and manufacturing, and have been studied intensively in computational geometry, starting about 15 years ago. Some of the initial results on this problem can be found in [6, 13, 20, 23]; these algorithms are either inefficient or consider only special cases (e.g., where P is assumed to be a line segment). The first significant development was made by Leven and Sharir [14], who showed that the combinatorial complexity of $\mathcal{C}$, when no scaling is allowed, is $O(mn\lambda_6(mn))$. Here $\lambda_s(q)$ is the maximum length of (q,s)-Davenport-Schinzel sequences [21], which is nearly linear in q for any fixed s. Thus the complexity of $\mathcal{C}$ is near-quadratic in mn. The goal then was to compute $\mathcal{C}$ in time that is also near-quadratic in mn. The first result in this direction is in [11], where an $O(mn\lambda_6(mn)\log mn)$-time algorithm was proposed. However, this algorithm turned out to have a technical difficulty: it computes a *superset* of all the vertices of $\mathcal{C}$ (where each such vertex is a free 'critical' placement of P in Q at which P makes three distinct contacts with the boundary of Q), and then aims to filter out the spurious (non-free) placements. However, this filtering was not handled correctly in some cases.

Two subsequent papers aimed to fix this problem. The first solution is given by Sharir and Toledo [22]. It processes Q into several range-searching data structures, and then it queries these structures with each placement of P produced by the algorithm. This allows us to detect and discard non-free placements, but the cost of each query is $O(m\log n)$, resulting in an overall complexity close to $O(m^3n^2)$, which is significantly more expensive (when m is large, which is what we assume here). A second solution is given by Kedem et al. [12]. It remains within the time complexity $O(mn\lambda_6(mn)\log mn)$ of the algorithm of [11], but it may fail to produce the entire space $\mathcal{C}$. More precisely, given an initial free placement Z of P, the algorithm is guaranteed to construct the connected component of $\mathcal{C}$ that contains Z (which suffices for most motion planning applications), and may compute some other components, but it is not guaranteed to produce all components of $\mathcal{C}$. Consequently, their solution can determine in $O(mn\lambda_6(mn)\log mn)$ time whether there exists a collision-free path between two placements of P, but it cannot determine in time that is near-quadratic in mn whether a congruent copy of P can be placed inside Q.

The case in which scaling is allowed, and we seek the largest placement of P inside Q, has been studied in [8, 22]. Using generalized Delaunay triangulations induced by P in Q, Chew and Kedem [8] gave an $O(m^4n^2\alpha(n)\log n)$-time algorithm for computing a largest free similar placement of P in Q; here $\alpha(n)$ is the inverse Ackermann's function. Sharir and Toledo [22] proposed another algorithm that combines parametric searching [16] with a construction of the entire configuration space for the fixed-size case, as in the preceding paragraph; the running time of their algorithm is $O(m^2n\lambda_4(mn)\log^3 mn\log\log mn)$, which is close to $O(m^3n^2)$. If only translations and scalings are allowed, the largest homothetic placement of P inside Q can be computed in time $O(mn\log n)$, using the generalized Voronoi diagram of ∂Q induced by P [15].

Previous results: Convex environment. One faces a much simpler situation when Q is also a convex polygon. The resulting problems are still challenging and have an interesting geometric structure. They also have several applications (see, e.g, [5] for a computer vision application). The only previous attack on the problem, in which P is assumed to have a fixed size, is in [6]. It is shown there that one can determine in $O(mn^2)$ time whether P can be placed inside Q. Computing a largest homothetic copy of P inside Q (i.e., allowing only translations and scalings) can be done in $O(m+n)$ time, using a linear-programming approach [22]. We are not aware of any previous work where P is also allowed to translate, rotate, and scale (except, of course, specializations of the algorithms mentioned above to the case where Q is convex; however, no improvements in the time bounds in such a specialization have been studied in these works).

New results and methods. We first study the problem where Q is convex, and we seek a largest placement of P inside Q. We show that such a placement can be computed in $O(mn^2\log n)$ time. We also show that the combinatorial complexity of the space $\mathcal{C}$ of all similar placements of P inside Q is $O(mn^2)$, that this bound is tight in the worst case, and that $\mathcal{C}$ can also be computed in $O(mn^2\log n)$ time. These results are obtained by a careful analysis of the structure of $\mathcal{C}$: we show that $\mathcal{C}$ can be represented as a convex polytope in 4-space with mn facets (this already implies that its complexity is $O(m^2n^2)$). A more refined analysis yields the improved bounds noted above. To find a largest placement of P in Q, it suffices to consider a certain 2-dimensional projection of $\mathcal{C}$. We also analyze the structure of this projection, which has some additional properties, and give a simpler, $O(mn^2\log n)$-time algorithm to compute this projection directly. Both algorithms are very simple to implement; they use some straightforward processing, followed by constructions of 3-dimensional convex hulls (for which optimal 'off-the-shelf' code is available).

We next study the case where Q is a general polygonal environment. We adapt a recent randomized algorithmic technique from [1, 3], to obtain a ran-

domized algorithm that constructs (the boundary of) $\mathcal{C}$ in expected time $O(mn\lambda_6(mn)\log^2 mn)$. A somewhat more complex variant of the algorithm runs in expected $O(mn\lambda_6(mn)\log mn)$ time. This is the first correct solution whose running time is near quadratic in mn and which produces the entire configuration space $\mathcal{C}$. These algorithms (in particular the first one) are also relatively simple to implement (though not as simple as in the case of a convex Q, because here we need to process curved algebraic surfaces and arcs). Even for the task of computing only a portion of $\mathcal{C}$, our algorithms are simpler than the ones in [12, 22]. We can preprocess $\mathcal{C}$ in additional $O(mn\lambda_6(mn)\log^2 mn)$ time so that we can answer efficiently *reachability quesries*: for any two placements of P, we can determine in $O(\log^2 mn)$ time whether there is a collision-free motion from one to the other (i.e., whether they lie in the same connected component of $\mathcal{C}$).

Using an approach based on parametric searching, similar to that of [22], we can find the largest similar placement of P in Q, in randomized expected time $O(mn\lambda_6(mn)\log^5 mn)$, thus improving significantly over the previous bounds in [8, 22]. We note that the parametric searching requires an 'oracle' procedure that has to determine, for a given size of P, whether the corresponding $\mathcal{C}$ is nonempty, which we can do using our algorithm for computing the entire $\mathcal{C}$. Notice that we cannot use the algorithm by Kedem et al. [12] here, because it may miss some of the components of $\mathcal{C}$.

2 Largest Placement of One Convex Polygon Inside Another

Let P be a convex polygon with m edges and Q a convex polygon with n edges. Our goal is to find a largest similar copy of P inside Q (allowing translation, rotation, and scaling of P); see Figure 1.

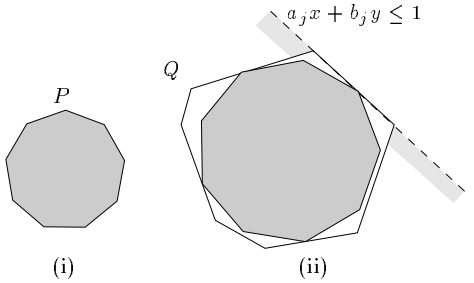

Figure 1: (i) The polygon P; (ii) The polygon Q and a largest copy of P inside Q

The geometric setup of the problem is as follows. We observe, following Baird [4], that similar placements of P can be parameterized nicely by referring to an arbitrarily chosen reference point $p \in P$. A placement is represented by a quadruple (s,t,u,v), where (u,v) is a translation of p in the plane, and $s = \rho\cos\theta$, $t = \rho\sin\theta$, where P is rotated by θ and scaled by ρ, around p. The standard placement puts p at the origin, with $\rho = 1$, $\theta = u = v = 0$. Thus if (x,y) is a vertex of P in the standard placement, its position at the placement (s,t,u,v) is $(sx - ty + u, tx + sy + v)$. Such a placement of P lies fully within Q if and only if every vertex (x_i, y_i) of P lies in every halfspace $a_j x + b_j y \le 1$ containing Q and bounded by the line supporting an edge of Q (here we assume, without loss of generality, that Q contains the origin). That is, the placement (s,t,u,v) must satisfy the following system of mn linear inequalities:

$$a_j(sx_i - ty_i + u) + b_j(tx_i + sy_i + v) \le 1$$

or

$$L_{i,j} : (a_j x_i + b_j y_i)s + (-a_j y_i + b_j x_i)t + a_j u + b_j v \le 1 .$$

In other words, the space $\mathcal{C}$ of all similar placements of P inside Q is a 4-dimensional convex polyhedron formed by the intersection of mn halfspaces (it is also easily seen to be bounded). This already implies that the combinatorial complexity of $\mathcal{C}$ is $O(m^2n^2)$, and that it can be constructed in $O(m^2n^2)$ time [18]. However, we will improve this bound in what follows, exploiting the fact that $\mathcal{C}$ is highly degenerate. The main results of this section are:

Theorem 1 *The vertices of the projection of $\mathcal{C}$ onto the st-plane can be computed in time $O(mn^2\log n)$.*

Theorem 2 *The total number of vertices of $\mathcal{C}$ is $O(mn^2)$, and they can be computed in time $O(mn^2\log n)$.*

Remark. Although Theorem 1 follows immediately from Theorem 2, we give a direct proof of Theorem 1, which is somewhat simpler and provides more geometric insight into the structure of the problem.

Proof of Theorem 1. We prove both theorems by applying the standard duality transform that maps a point $(\xi_1,\xi_2,\xi_3,\xi_4)$ to the hyperplane $\xi_1 s + \xi_2 t + \xi_3 u + \xi_4 v = 1$ and vice versa. We denote the coordinates in the dual space by s^*, t^*, u^*, v^*. The vertices of the polytope $\mathcal{D}$ dual to $\mathcal{C}$ are thus

$$w_{i,j} = (a_j x_i + b_j y_i,\ -a_j y_i + b_j x_i,\ a_j\ ,\ b_j)\ ,$$

for $i = 1,\ldots,m$ and $j = 1,\ldots,n$. It is easy to verify that all the points $w_{i,j}$ are indeed extreme points of $\mathcal{D}$

(or, equivalently, that all the hyperplanes bounding the halfspaces $L_{i,j}$ contain facets of $\mathcal{C}$). Note that, for each fixed j (corresponding to an edge of Q), the convex hull G_j of $\{w_{i,j}\}_{i=1}^m$ is a similar copy of P that lies in the 2-plane $\pi_j : u^* = a_j, v^* = b_j$. The dual polytope $\mathcal{D}$, then, is the convex hull of n similar copies of P, placed in parallel 2-planes in $\mathbb{R}^4$.

We exploit the well-known fact that projection in the primal is slicing in the dual. In more detail, let $\mathcal{C}_2$ denote the projection of $\mathcal{C}$ onto the st-plane $u = 0$, $v = 0$, as effected by the mapping $(s,t,u,v) \mapsto (s,t,0,0)$. Then a line $\alpha s + \beta t = 1$ in the st-plane is a supporting line of $\mathcal{C}_2$ if and only if the hyperplane $\alpha s + \beta t = 1$ is a supporting hyperplane of $\mathcal{C}$ in $\mathbb{R}^4$. This is equivalent, in the dual, to having the point $(\alpha, \beta, 0, 0)$ belong to the boundary of $\mathcal{D}$. Thus, computing $\mathcal{C}_2$ is equivalent to computing the cross section $\mathcal{D}_2$ of $\mathcal{D}$ with the 2-plane $u^* = 0$, $v^* = 0$.

Our strategy for computing $\mathcal{D}_2$ is first to compute $\mathcal{D}_3$, the cross section of $\mathcal{D}$ with the hyperplane $u^* = 0$, and then to slice $\mathcal{D}_3$ with the plane $v^* = 0$. Since it is trivial to intersect a three-dimensional polytope with a plane in time proportional to the complexity of the polytope, we only consider the construction of $\mathcal{D}_3$.

Without loss of generality, we can assume that none of the a_j's is 0. Then each of the polygons G_j lies outside the hyperplane $u^* = 0$. Hence, any vertex w of $\mathcal{D}_3$ must be an intersection of $u^* = 0$ with an edge of $\mathcal{D}$, connecting two vertices of a pair of distinct polygons, G_i and G_j, where G_i lies above $u^* = 0$ and G_j lies below. Moreover, w must also be a vertex of the intersection of the convex hull of $G_i \cup G_j$ with $u^* = 0$. So we can construct $\mathcal{D}_3$ by taking the convex hull, in $\mathbb{R}^4$, of every pair of polygons G_i, G_j, intersecting all of these sub-hulls with $u^* = 0$, and then taking the convex hull of the resulting intersections.

Let us consider the geometry of one such sub-hull. The two parallel 2-planes $u^* = a_i, v^* = b_i$ and $u^* = a_j, v^* = b_j$ lie in the common 3-plane $F_{i,j}$ defined by

$$(b_j - b_i)u^* + (a_i - a_j)v^* + (b_i a_j - b_j a_i) = 0$$

and so does the sub-hull determined by G_i, G_j. The three-dimensional geometry of $conv(G_i \cup G_j)$ in $F_{i,j}$ is as shown in Figure 2.

The intersection of $F_{i,j}$ with $u^* = 0$ is the 2-plane $u^* = 0, \quad v^* = (b_i a_j - b_j a_i)/(a_i - a_j)$, which is also parallel to the two polygons G_i, G_j. By slicing the convex hull of the two parallel polygons with a parallel plane, we get a third parallel polygon $G_{i,j}$ which is the Minkowski sum of appropriately scaled copies of G_i and G_j. This polygon has at most $2m$ vertices, and it is easy to compute directly from the vertices of G_i and G_j. Note that $G_{i,j}$ lies in both $F_{i,j}$ and in $u^* = 0$.

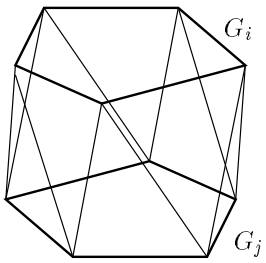

Figure 2: Convex hull of parallel polygons

The 3-polytope $\mathcal{D}_3$ in $u^* = 0$ is the convex hull of all these polygons $G_{i,j}$. There are $O(n^2)$ such polygons, each with at most $2m$ vertices, so the total complexity of $\mathcal{D}_3$ is $O(mn^2)$ (which of course is also a consequence of the bound for the overall complexity of $\mathcal{D}$, as asserted in Theorem 2 and proven below).

The algorithm is simply to form the polygons $G_{i,j}$, take their three-dimensional convex hull, and intersect it with $v^* = 0$. Since the Minkowski sum of two convex polygons can be computed in linear time [10], we spend $O(mn^2)$ time in computing the polygons $G_{i,j}$. Their convex hull can be computed in $O(mn^2 \log n)$ time, using the divide-and-conquer algorithm of [17] (which has now only $O(\log n)$ recursive levels, because we start with the already available polygons $G_{i,j}$). Hence, the total running time is $O(mn^2 \log n)$. □

Note that in practical terms, the implementation of this algorithm is a straightforward setup followed by a convex hull computation, which can be performed efficiently with publicly available software.

Proof of Theorem 2. We first consider the facets of $\mathcal{D}$ whose supporting hyperplanes are parallel to the 2-plane $u^* = 0, v^* = 0$. The equation of such a hyperplane h_F of a facet F has the form $\beta u^* + \gamma v^* + \delta = 0$. Hence, if h_F contains a vertex of some G_j, it must contain the entire polygon G_j. It then follows that F must be the convex hull of the union of two polygons G_i, G_j (as in the proof of Theorem 1 given above). The facet F is dual to the placement of P in which it is shrunk to a point and all its vertices are incident to the vertex of Q where edge i meets edge j (so that these two edges must be consecutive edges of Q). The number of such placements is n, and the complexity of each of the corresponding facets is $O(m)$, since it is the 3-dimensional convex hull of $2m$ points. (It is easily verified that each of these hulls is indeed a facet of $\mathcal{D}$.) It follows that the overall complexity of these facets of $\mathcal{D}$ is $O(mn)$. Constructing all these facets is easy to do in $O(mn)$ time.

Next, consider the facets of $\mathcal{D}$ whose supporting hyperplanes are not parallel to the 2-plane $u^* = 0$, $v^* = 0$. Let F be such a facet of $\mathcal{D}$, so that the equation of its containing hyperplane h can be written as $t^* = \alpha s^* + \beta u^* + \gamma v^* + \delta$. Then, for each $j = 1, \ldots, n$, the line ℓ_j of intersection between h and the 2-plane π_j containing G_j either touches G_j or is disjoint from G_j. The equation of ℓ_j is $t^* = \alpha s^* + \beta a_j + \gamma b_j + \delta$, $u^* = a_j$, $v^* = b_j$, Note that the coefficient α uniquely determines the vertex of G_j nearest to ℓ_j, for every j, unless α is a 'critical' value equal to the slope of an edge of some G_j. There are $\nu = mn$ such critical slopes α, corresponding to the orientations at which an edge of P is parallel to an edge of Q, and it is easy to compute them, in order, in time $O(mn \log n)$. Let $\alpha_1 < \alpha_2 < \cdots < \alpha_\nu$ be these critical slopes.

Let K be an open interval of α-coefficients between two successive critical slopes. Then, for each $j = 1, \ldots, n$, there exists a unique vertex $w_{i(K),j}$ of G_j, such that if h is any supporting hyperplane of $\mathcal{D}$, whose α-coefficient lies in K, then h can touch G_j, if at all, only at $w_{i(K),j}$. In other words, such an h is also a supporting hyperplane of $S_K = \{w_{i(K),j}\}_{j=1}^n$ (h must of course touch at least one of these vertices, and at least four if it contains a facet of $\mathcal{D}$). For two adjacent intervals K and K', the set $S_{K'}$ is obtained from S_K by replacing one vertex w by another vertex w' (both being adjacent vertices of some G_j). It easily follows that every facet F of $\mathcal{D}$ not parallel to $u^* = 0$, $v^* = 0$ is either a facet of $conv(S_K)$, for some interval K, or, if the α-coefficient of F is a critical value, a facet of $conv(S_K \cup S_{K'})$, for some pair of consecutive intervals K and K'. If the vertices of P and Q are in general position, these latter facets correspond to placements in which an edge of P is incident to an edge of Q. In fact, we can prove the following stronger claim. Assuming $\alpha_0 = -\infty$ and $\alpha_{\nu+1} = +\infty$, let K_i be the open interval (α_i, α_{i+1}) for $0 \le i \le \nu$, and let $w_i = S_{K_i} \setminus S_{K_{i-1}}$ for $1 \le i \le \nu$.

Lemma 3 *Every facet F of $\mathcal{D}$ that is not parallel to $u^* = 0$, $v^* = 0$ is either a facet of the convex hull $conv(S_{K_0})$ or a facet of the convex hull $conv(S_{K_{i-1}} \cup \{w_i\})$ incident to w_i, for some $1 \le i \le \nu$.*

Proof: Let F be a facet of $\mathcal{D}$ that is not parallel to $u^* = 0, v^* = 0$ and that is not a facet of $conv(S_0)$. Let W be the set of vertices of F, and let $i \le \nu$ be the index such that the α-coefficient of the hyperplane supporting F lies in the (semi-open) interval $(\alpha_{i-1}, \alpha_i]$. Then, by the above argument, $W \subseteq S_{i-1} \cup \{\xi_i\}$. Suppose $j \le i$ is the largest index such that $\xi_j \in W$ (i.e., S_j is obtained from S_{j-1} by inserting one of the points of W and deleting a point of S_{j-1}.) Then it is easily seen that $W \subseteq S_{j-1} \cup \{\xi_j\}$. Hence, F is a facet of $conv(S_{j-1} \cup \{\xi_j\})$ incident to ξ_j, as asserted. □

This lemma suggests that we should compute $conv(S_{K_0})$ and, for each $1 \le i \le \nu$, we compute the facets of $conv(S_{K_{i-1}} \cup \{w_i\})$ incident to w_i. Since the hyperplanes containing the facets of $conv(S_{K_{i-1}} \cup \{w_i\})$ incident to w_i have only three degrees of freedom, this problem can be formulated as a three-dimensional convex hull problem, and can be solved in $O(n \log n)$ time; the number of these facets, as well as their overall complexity, is $O(n)$. Notice that the set S_{K_0} and the vertices w_i for $1 \le i \le \nu$ can be computed in $O(mn \log n)$. Repeating this algorithm for all $1 \le i \le \nu$ and computing $conv(S_{K_0})$, the algorithm produces a total of $O(mn^2)$ facets, of $O(mn^2)$ overall complexity, in time $O(mn^2 \log n)$.

These arguments already prove that the total number of facets of $\mathcal{D}$ is $O(mn^2)$, and that their overall complexity, and hence the overall complexity of $\mathcal{C}$, is $O(mn^2)$. Unfortunately, the algorithm might produce additional *spurious* facets, which are not facets of $\mathcal{D}$. Indeed, a facet F of $conv(S_{i-1} \cup \{\xi_i\})$ corresponds to a placement π of P such that there are at least 4 vertex-edge incidences between the vertices of P_π and the edges of Q, and F is spurious if $P_\pi \not\subseteq Q$. If the α-coefficient of F lies in the interval $K_{i-1} \cup K_i$, then it follows by definition that F cannot be spurious. However, if this α-coefficient lies in another interval K_j, for some $j \notin \{i-1, i\}$, then F may be spurious, because P_π may violate a constraint $L_{u,v}$ corresponding to some vertex $w_{u,v} \in S_j \setminus (S_{i-1} \cup S_i)$. An example of such a spurious facet is given in the full version of the paper.

Hence, to complete our algorithm, we need to detect and discard the facets of the hulls $conv(S_K)$ which are not facets of $\mathcal{D}$. This is accomplished as follows. We triangulate each computed facet F into $O(|F|)$ tetrahedra, using the bottom-vertex triangulation scheme described in [7]. Let $\mathbf{\Delta}$ denote the set of resulting tetrahedra; $|\mathbf{\Delta}| = O(mn^2)$. Let $\mathcal{D}^*$ be the bottom-vertex triangulation of the boundary of $\mathcal{D}$. We want to discard those tetrahedra of $\mathbf{\Delta}$ that are not facets of $\mathcal{D}^*$. For a vertex w, let $\mathbf{\Delta}_w \subseteq \mathbf{\Delta}$ be the subset of tetrahedra incident to w, and let V_w be the set of vertices of the tetrahedra in $\mathbf{\Delta}_w$. It is easily verified that a tetrahedron $\Delta \in \mathbf{\Delta}_w$ is a facet of $\mathcal{D}^*$ if and only if Δ is a tetrahedron in the bottom-vertex triangulation of the boundary of $conv(V_w)$, which is necessarily incident to w. We therefore compute the facets of $conv(V_w)$ that are incident to w, by the reduction, noted above, to a 3-dimensional convex hull construction, and then compute the bottom-vertex triangulation of each such facet. Note that these facets can be computed in $O(|V_w| \log n)$ time, since

the vertices of V_w lie on only n 2-planes, so that the convex hull computation requires only $O(\log n)$ recursive levels; we omit the easy details. We can now discard those tetrahedra in $\mathbf{\Delta}_w$ that do not lie on the boundary of $conv(V_w)$. Repeating this procedure for all vertices w of $\mathcal{D}$ gets rid of all spurious facets computed by the algorithm.

The running time of this step is $\sum_w O(|V_w| \log n)$, where the sum extends over all vertices w of $\mathcal{D}$. Since $\sum_w |V_w| = 4|\mathbf{\Delta}| = O(mn^2)$, the total time spent is $O(mn^2 \log n)$. This completes the proof of Theorem 2. □

An immediate corollary of Theorems 1 and 2 is the following.

Corollary 4 *The largest similar copy of P inside Q can be computed in $O(mn^2 \log n)$ time.*

We conclude this section by constructing a pair of polygons P and Q, with m and n vertices, respectively, such that there are $\Omega(mn^2)$ placements of P inside Q, each of which induces four incidences of the form (p, e), where p is a vertex of P and e is an edge of Q. This implies that the combinatorial bound of Theorem 2 is tight in the worst case.

The construction is depicted in Figure 3. Let n be of the form $2l + 2$, for some positive integer l, m an even integer, and o the origin. The first $n/2$ vertices $q_1, \ldots, q_{n/2}$ of Q are evenly distributed along the arc of the unit-radius circle, centered at o, which goes from $-\pi/6$ to $\pi/6$ (in counterclockwise direction). The vertices $q_{n/2+1} \ldots q_n$ are evenly distributed along a tiny arc of a larger circle, say the circle with radius $10+\varepsilon$ and center $(10, 0)$, and we let the tiny arc span the orientations between $\pi - \frac{\varepsilon}{2(10+\varepsilon)}$ and $\pi + \frac{\varepsilon}{2(10+\varepsilon)}$, so that its arc length is ε. The value of ε will be chosen sufficiently small, in a manner to be detailed in a moment.

We place one vertex p_m of P at the origin o and the remaining $m - 1$ vertices, equally spaced, on a circular arc of radius $1/4$, centered at $(3/4, 0)$, that spans the orientations between $-\frac{\pi}{40l}$ and $+\frac{\pi}{40l}$.

Claim: *If ε is chosen sufficiently small then the following holds. For every triple $n/2 + 1 \le i \le n$, $1 \le j < n/2$, and $1 \le k \le m - 2$, there is a placement of P inside Q, using translation, rotation, and scaling, such that the vertex p_m of P coincides with the vertex q_i of Q, and such that the edge $p_k p_{k+1}$ of P coincides with the edge $q_j q_{j+1}$ of Q.*

Notice that every such placement of P induces four vertex-edge incidences between P and Q, and is thus a vertex of $\mathcal{C}$.

Proof: We consider the scaling, rotation, and translation of P that places $p_k p_{k+1}$ on the line ℓ supporting $q_j q_{j+1}$ and also places p_m at q_i.

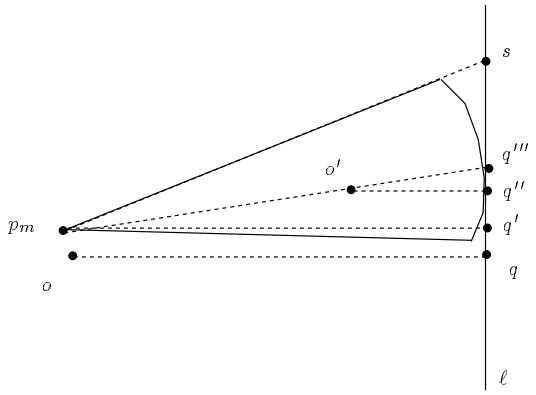

Figure 4: Proof of claim

As in Figure 4, let q be the center of edge $q_j q_{j+1}$; q is also the orthogonal projection of the origin o onto the line ℓ supporting $q_j q_{j+1}$. Let q' be the projection of p_m, which is placed at q_i, onto ℓ. Let q'' be the projection onto ℓ of o', the center of the small circle whose boundary contains the points $p_1, \ldots, p_{m-1}$, which is appropriately shifted with P. Let q''' be the intersection of the line from $p_m = q_i$ through o' with ℓ. Finally, let s be the intersection of the line supporting $p_m p_{m-1}$ (at this placement of P) with ℓ.

The distance from q to q' is at most ε. The angle $q'''p_m q'$ is the same as the angle $q'''o'q''$, which, by the construction of P, is at most $\frac{\pi}{40l}$. The angle $sp_m q'''$ is exactly $\frac{\pi}{40l}$. Since the distance from p_m to q' is at most $1 + \varepsilon$, the distance from q to s is

$$d(q, s) \le \varepsilon + (1 + \varepsilon) \tan \frac{\pi}{20l}.$$

Since the distance from q to q_{j+1} is $\sin \frac{\pi}{6l}$, ε can be chosen small enough so that

$$\varepsilon + (1 + \varepsilon) \tan \frac{\pi}{20l} < \sin \frac{\pi}{6l},$$

which then implies that this placement of P fully lies below the segment $p_m q_{j+1}$. An analogous argument shows that P lies above the segment $p_m q_j$, so P lies inside Q, as claimed.

We therefore obtain the following result.

Theorem 5 *There exist a convex m-gon P and another convex n-gon Q such that there are $\Omega(mn^2)$ placements of similar copies of P inside Q, each of which induces four vertex-edge incidences between P and Q.*

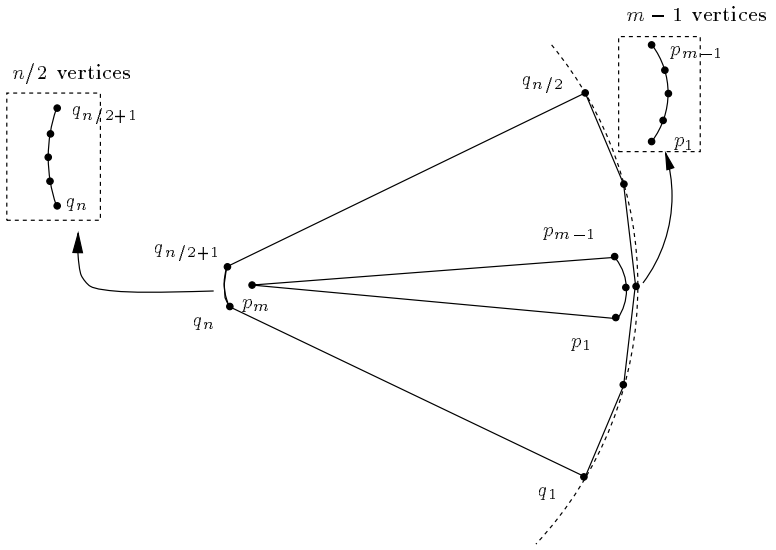

Figure 3: Polygons P and Q for which there exist $\Omega(mn^2)$ similar placements of P in Q with four vertex-edge incidences per placement

3 Placing a Convex Polygon in a General Polygonal Environment

We next consider the case where P is a convex m-gon translating and rotating in a general polygonal environment Q bounded by n edges (no scaling is allowed for the time being). As discussed in the introduction, the combinatorial complexity of the space $\mathcal{C}$ of free placements of P is $O(mn\lambda_6(mn))$, i.e., near-quadratic in mn, but there is no published algorithm that computes correctly the entire $\mathcal{C}$ in time that is close to this bound.

In this section, we propose a rather simple randomized technique for constructing the entire $\mathcal{C}$, with $O(mn\lambda_6(mn)\log^2 mn)$ expected running time. A somewhat more complicated randomized algorithm can compute $\mathcal{C}$ in expected time $O(mn\lambda_6(mn)\log mn)$. As it is usual for this type of algorithms, the expectation is over the random choices made by the algorithm, for any fixed input, and not over any assumed distribution of the input data.

In closing these introductory remarks, we note that the nice linear structure of the constraints defining $\mathcal{C}$ in the previous section does not exist here. Intuitively, this is because here we have a mixture of two types of constraints — those induced by contacts of vertices of P with edges of Q (as in the previous section) and those induced by contacts of edges of P with vertices of Q (which were absent in the previous analysis). One can choose coordinate frames in which either of these two types of constraints is linear, but then the other type is necessarily nonlinear.

Constructing $\mathcal{C}$. The space of all placements of P is three-dimensional, and can be parameterized by (x, y, θ) (or, preferably, by $(x, y, \tan\frac{\theta}{2})$), where (x, y) is the position of a reference point of P and θ is the angle by which P is rotated from some fixed reference orientation. For convenience, we make an assumption that simplifies our analysis. We triangulate the complement of Q, and from now on assume that it is the union of a set of n pairwise openly disjoint *triangular obstacles* (some of which may be unbounded); note that the new n is larger than the original n by a constant factor.

For each obstacle Δ, let $K(\Delta)$ denote the set of 'forbidden' placements of P at which it intersects the interior of Δ. These are open sets, and $\mathcal{C}$ is the complement of their union, so it suffices to compute (the boundary of) the union $K = \bigcup_\Delta K(\Delta)$. For each obstacle Δ_0, we compute the faces of ∂K that lie in $\partial K(\Delta_0)$, and then patch these faces together to construct K. This leads to the following simple high-level description of our algorithm: Fix an obstacle Δ_0, and compute the the portion of ∂K that is contained in $\partial K(\Delta_0)$, which is the complement of $\bigcup_{\Delta\neq\Delta_0}(K(\Delta)\cap\partial K(\Delta_0))$. Hence, after applying this procedure to all obstacles Δ_0, we can 'glue' together these portions of ∂K to obtain (an appropriate discrete representation of) the entire boundary of K. We omit the details concerning the gluing process,

as they are essentially the same as in the preceding algorithms [1, 3, 11].

Note that $\partial K(\Delta_0)$ consists of all (free or non-free) placements of P at which its boundary makes contact with $\partial\Delta_0$ and P and Δ_0 are openly disjoint. To simplify the algorithm, we partition $\partial K(\Delta_0)$ into $O(m)$ patches, each of which is the locus $\pi_{e,v}$ of all placements of P at which some fixed vertex v of P touches some fixed edge e of Δ_0, or some fixed edge e of P touches some fixed vertex v of Δ_0, and the interiors of P and Δ_0 are disjoint. If any of the patches is not xy-monotone, we further partition it into a constant number of xy-monotone patches. Such a partition is easy to obtain in $O(m)$ time. We refer to the patches $\pi_{e,v}$ as *contact surfaces*.

We thus obtain a collection of the $O(mn)$ 2-dimensional contact surfaces $\pi_{e,v}$. For each such surface π, we compute the intersections $\Delta_\pi = \pi \cap K(\Delta)$, for all obstacles Δ, and construct, $K_\pi = \pi \setminus (\bigcup_\Delta \Delta_\pi)$, the complement of their union within π. K_π corresponds to placements at which v is in contact with e and P does not intersect the interior of any obstacle. Gluing these complements together will give us ∂K, as above. We refer to the sets Δ_π as *virtual π-obstacles*.

Let $\pi = \pi_{e,v}$ be a fixed contact surface. We can parametrize π by $(\rho, \tan\frac{\theta}{2})$, where ρ measures the displacement along e of its contact with v, and where θ is the orientation of P. For an obstacle Δ, constructing Δ_π is easy: Note that, for any fixed θ, the locus of placements contained in Δ_π with orientations θ is a line segment. (Indeed, the only motion available for P in this set is translation parallel to e; the set of such translations at which the two convex polygons P and Δ intersect is a line segment.) The critical θ's at which the combinatorial nature of an endpoint of this segment changes are such that the line parallel to e through some vertex of P passes through some vertex of Δ. There are $O(m)$ such orientations, and Δ_π can easily be obtained by sorting and processing these orientations in increasing order. Hence, $\partial\Delta_\pi$ consists of $O(m)$ arcs. As shown in [20], each such arc is a section of an algebaric curve of degree at most 4. Δ_π can be computed in $O(m \log m)$ time. The total time needed to produce the sets Δ_π, over all Δ, is thus $n \times O(m \log m) = O(mn \log m)$.

We compute $K_\pi = \pi \setminus (\bigcup_\Delta \Delta_\pi)$ using a randomized divide-and-conquer approach. We randomly divide the set of virtual π-obstacles into two equal subsets (so that every such partition occurs with equal probability), recursively compute the complement of their two unions in π, denoted by K_1, K_2, and compute $K_\pi = K_1 \cap K_2$ using a standard sweep-line technique. We assume, as is standard, an appropriate model of computation, in which various basic operations on the arcs forming the boundaries of the virtual obstacles (such as intersecting a pair of such arcs) can be done in $O(1)$ time. Since every intersection point of an edge of K_1 with an edge of K_2 is a vertex of K_π, the total time spent in dividing and in the merge step is $O((|K_\pi| + |K_1| + |K_2|) \log mn)$, where $|K_\pi|$, $|K_1|$ and $|K_2|$ are the number of vertices of these respective sets. If we let κ_π denote the total number of vertices in all the intermediate unions (of all recursive subproblems) produced by the algorithm, then the total running time of the algorithm (applied to a fixed π), including the time spent in computing the virtual π-obstacles, is $O((mn + \kappa_\pi) \log mn)$.

Applying this procedure to each of the $O(mn)$ contact surfaces independently and gluing the results together, we construct ∂K in time $O((m^2n^2 + \sum_\pi \kappa_\pi) \log mn)$, where the summation is taken over all contact surfaces. We will prove below that the expected value of $\sum_\pi \kappa_\pi$ is $O(mn\lambda_6(mn) \log mn)$, which implies that the expected running time of the overall algorithm is $O(mn\lambda_6(mn) \log^2 mn)$. Hence, we can conclude:

Theorem 6 *Given a convex polygon P with m edges and a polygonal environment Q with a total of n edges, we can compute the entire free configuration space $\mathcal{C}$ by a randomized algorithm in expected time $O(mn\lambda_6(mn) \log^2 mn)$.*

Note that there is an alternative, randomized incremental approach to construct each K_π, in which we add the Δ_π's one at a time, in a random order, and maintain a 'trapezoidal' decomposition of the complement of their union; see [1, 3, 9]. The analysis of this technique is fairly standard, so we omit it here. The expected running time of this approach is only $O(mn\lambda_6(mn) \log mn)$, so this technique is slightly faster, but somewhat more complicated.

Bounding the expected value of $\sum_\pi \kappa_\pi$. In the remainder of this section, we establish the upper bound on the expected value of $\sum_\pi \kappa_\pi$, as stated above. For simplicity, assume that n, the total number of (triangular) obstacles, is of the form $2^h + 1$ for some integer h. Any vertex ζ that can appear on an intermediate union U produced by the algorithm, while computing K_π for some contact surface π, is an intersection of the boundaries of some pair of virtual π-obstacles (ignoring vertices of individual virtual π-obstacles, whose global number has already been shown to be $O(m^2n^2)$). Therefore ζ represents a placement of P at which (a) P makes three simultaneous contacts with the obstacle boundaries, and (b) P is openly disjoint from the union of the obstacles Δ whose corresponding virtual π-obstacles par-

ticipate in U, and from Δ_0, the obstacle for which π is a portion of $\partial K(\Delta_0)$.

A *triple-contact vertex* is a (not necessarily free) placement of P at which ∂P makes three simultaneous (vertex-edge or edge-vertex) contacts with ∂Q. We say that any triple-contact vertex ζ has *level* k (with respect to the full collection of obstacles) if removal of some k other obstacles (excluding the at most three that participate in the triple contact) causes ζ to become a free placement, relative to the remaining obstacles, and no set of fewer than k obstacles has this property. Note that level-0 vertices are exactly the vertices of $\mathcal{C}$. Let F_k denote the number of level-k vertices for the given P and Q, and let $G(r)$ denote the expected number of level-0 vertices for P in an environment obtained by picking a random sample of r of the n obstacles, where any subset of r obstacles is chosen with equal probability. Fix a level-k vertex and let p_k denote the expected number of recursive subproblems of size r which contain it in the output. Then the expected value of $\sum_\pi \kappa_\pi$ is easily seen to be

$$\mathrm{E}\left[\sum_\pi \kappa_\pi\right] = \sum_{k=0}^{n-3} F_k \cdot p_k .$$

We first obtain a bound on p_k. Note that, throughout the execution of the algorithm, it encounters sets of virtual obstacles of cardinality 2^i, for $i = 0, \ldots, h$. Fix one such i. Consider any three obstacles, and fix a triple-contact vertex ζ of the free configuration space defined when only these three obstacles are present. What is the probability that ζ occurs during the execution of the algorithm, for any contact surface π, while processing subproblems involving $r = 2^i$ obstacles? The previous discussion implies that ζ lies at the intersection of three contact surfaces. Fix one of these contact surfaces π. Suppose ζ is a level-k vertex, with respect to the full set of obstacles. Then ζ appears in some fixed subproblem involving r obstacles in the construction carried out within π if and only if these r obstacles include the other two contact obstacles and do not include any of the k obstacles that "cover" ζ. Since every set of r obstacles (excluding the obstacle inducing π) has the same probability of being the set of input obstacles to our fixed subproblem, the probability of ζ appearing in the output of the subproblem is $\frac{\binom{n-3-k}{r-2}}{\binom{n-1}{r}}$. (Recall that we ignore vertices that are determined by fewer than three obstacles; these vertices show up as vertices of some virtual π-obstacle, so we already have a bound on their number, as above.) Hence,

$$p_k \le \sum_{i=0}^{h} 3 \cdot 2^{h-i} \frac{\binom{n-3-k}{2^i-2}}{\binom{n-1}{2^i}} .$$

Here we used the fact that ζ may appear in the construction in the three different contact surfaces that define π, and that, in any fixed recursive construction within π, there are 2^{h-i} subproblems involving 2^i obstacles each. Hence,

$$\begin{aligned} \mathrm{E}\left[\sum_\pi \kappa_\pi\right] &\le \sum_{k=0}^{n-3} \left(F_k \sum_{i=0}^{h} 3 \cdot 2^{h-i} \frac{\binom{n-3-k}{2^i-2}}{\binom{n-1}{2^i}} \right) \\ &= \sum_{i=0}^{h} 3 \cdot 2^{h-i} \sum_{k=0}^{n-3} \frac{\binom{n-3-k}{2^i-2}}{\binom{n-1}{2^i}} F_k . \quad (1) \end{aligned}$$

To bound this sum, we express $G(r)$ in terms of F_k. What is the probability that a level-k vertex ζ defined by three contacts, as above, is counted in $G(r)$? In other words, what is the probability that it corresponds to a vertex of the free configuration space, in the environment defined by r randomly selected obstacles? It is defined by three obstacles and "covered" by k other obstacles, so the probability is $\frac{\binom{n-3-k}{r-3}}{\binom{n}{r}}$. Thus, the expected number of free vertices (each defined by three obstacles) arising in the r-sample is

$$G(r) = \sum_{k=0}^{n-3} \frac{\binom{n-3-k}{r-3}}{\binom{n}{r}} F_k .$$

Putting $r = 2^i + 1$, we obtain

$$\begin{aligned} G(2^i+1) &= \sum_{k=0}^{n-3} \frac{\binom{n-3-k}{2^i-2}}{\binom{n}{2^i+1}} F_k \\ &= \frac{2^i+1}{n} \sum_{k=0}^{n-3} \frac{\binom{n-3-k}{2^i-2}}{\binom{n-1}{2^i}} F_k . \quad (2) \end{aligned}$$

Substituting (2) into (1), we obtain

$$\begin{aligned} \mathrm{E}\left[\sum_\pi \kappa_\pi\right] &\le \sum_{i=0}^{h} 3 \cdot 2^{h-i} \frac{n}{2^i+1} G(2^i+1) \\ &= O(n^2) \cdot \sum_{i=0}^{h} \frac{G(2^i+1)}{2^i(2^i+1)} . \end{aligned}$$

Notice that $G(2^i+1)$ is bounded by the combinatorial complexity of $\mathcal{C}$ for P moving amidst 2^i+1 obstacles, which, as noted above, is known to be $O(2^i m \lambda_6(2^i m))$, so the total expected output size of all subproblems is $O(mn\lambda_6(mn)\log n)$, as claimed.

Motion-planning queries for P. In order to answer reachability queries of the form "given two placements I and F of P, determine whether there is a collision-free path of P (inside Q) from I to F," we need to preprocess $\mathcal{C}$ into a data structure so that we can determine whether two query points lie in the same connected component of $\mathcal{C}$.

We first compute a refinement $\mathcal{C}^*$ of $\mathcal{C}$ and then preprocess $\mathcal{C}^*$ for point-location queries. We consider the collection of all boundary edges of the contact surfaces, and of the loci of points on these surfaces with vertical tangency (in the z-direction, where z is the parameter $\tan\frac{\theta}{2}$). For each of these curves γ, we draw a vertical segment from every point on γ in both the $(+z)$- and the $(-z)$-directions, until it intersects another contact surface. That is, we erect a vertical wall through γ within the cell (3-dimensional face) of $\mathcal{C}$ that contains γ. This gives the desired refinement $\mathcal{C}^*$. It can be shown that $\mathcal{C}^*$ is vertically convex (i.e., every line intersects a 3-dimensional face of $\mathcal{C}^*$ in a connected interval), and that every two-dimensional face of the cross section of $\mathcal{C}^*$ with a plane parallel to the xz-plane is x-monotone. Following an argument similar to that in [1], one can show that the complexity of $\mathcal{C}^*$ is $O(mn\lambda_{10}(mn))$. This follows by showing that any pair of contact surfaces intersect at most 8 times within a fixed vertical wall. More details are given in the full version of the paper.

Two cells $f_1, f_2 \in \mathcal{C}^*$ lie in the same connected component of $\mathcal{C}$ if one can reach f_2 from f_1 by crossing only vertical walls of $\mathcal{C}^*$. We can now identify the faces of $\mathcal{C}^*$ that lie within the same connected component of $\mathcal{C}$ by a simple graph traversal of the edges of $\mathcal{C}^*$. The total time spent in this step is $O(mn\lambda_{10}(mn))$. Finally, we preprocess $\mathcal{C}^*$ for point-location queries, using the algorithm of Preparata and Tamassia, as described in [1]. Using this data structure, we can determine in $O(\log^2 mn)$ time whether two given placements lie in the same connnected component of $\mathcal{C}$. Hence, we can conclude:

Theorem 7 *Given a convex polygon P with m edges and a polygonal environment Q with a total of n edges, we can preprocess $\mathcal{C}$ in randomized expected time $O(mn\lambda_6(mn)\log^2 mn)$ into a data structure so that, for any two placements I and F of P, we can determine, in $O(\log^2 mn)$ time, whether there exists a collision-free motion of P from I to F.*

Finding the largest placement of P. As mentioned in the beginning of this section, we use the parametric-searching technique of Megiddo [16] to compute a largest collision-free similar placement of P inside Q. The parametric searching requires an 'oracle' procedure to determine, for a given scalin factor of P, whether the corresponding $\mathcal{C}$ is nonempty. Using Theorem 6, we can obtain an oracle that performs this task in expected time $O(mn\lambda_6(mn)\log^2 mn)$. An efficient implementation of the parametric searching, however, also requires a parallel algorithm for the oracle (in Valiant's comparisons model [24]). The only part of the above randomized algorithm that is difficult to parallelize is the sweep-line procedure used in the merge step, because a sweep-line algorithm is inherently sequential. We therefore perform the merge step in the parallel version using a different approach, based on segment trees, such as the one used in [2]. Omitting all further details from this version, we show that one can compute $\mathcal{C}$ in $O(\log^2 mn)$ parallel steps, using $O(mn\lambda_6(mn)\log mn)$ expected number of processors, in Valiant's comparison model. Megiddo showed that if the sequential algorithm for the oracle runs in time T_s and the parallel algorithm runs in time T_p using Π processors, then the parametric searching takes $O(T_p\Pi + T_s T_P \log\Pi)$ time. Hence, putting everything together, we can conclude:

Theorem 8 *Given a convex polygon P with m edges and a polygonal environment Q with a total of n edges, we can compute a largest free placement of P inside Q in randomized expected time $O(mn\lambda_6(mn)\log^5 mn)$.*

Open problems. We conclude this section by mentioning two open problems:

(i) What is the combinatorial complexity of the four-dimensional configuration space of all free placements of P in Q, when scaling is also allowed? Is it also near-quadratic in mn? (See the previous section for the case where Q is convex.)

(ii) How fast can one answer 'real' motion planning queries, where the output to a query should be a collision-free path connecting the two given placements (when such a path exists)? Can this be done in time that has a polylogarithmic overhead plus a cost that depends on the actual complexity of the path? The approach described above does not seem to yield such a performance.

References

[1] P. Agarwal, B. Aronov, and M. Sharir, Computing envelopes in four dimensions with applications, *Proc. 10th ACM Symp. on Computationl Geometry* (1994), pp. 348–358.

[2] P. K. Agarwal, M. Sharir, and S. Toledo, Applications of parametric searching in geometric optimization, *J. Algorithms* 17 (1994), 292–318.

[3] B. Aronov and M. Sharir, The union of convex polyhedra in three dimensions, *Proc. 34th IEEE Symp. Found. Comput. Sci.* (1993), pp. 518–527.

[4] H. S. Baird, *Model-Based Image Matching Using Location*, Distinguished Dissertation Series, MIT Press, Cambridge, MA, 1984.

[5] R. Basri and D. Jacobs, Recognition using region correspondences, *Proc. 5th Int. Conf. Comput. Vision*, 1985, pp. 8-13.

[6] B. Chazelle, The polygon containment problem, in *Advances in Computing Research, Vol. 1: Computational Geometry* (F. P. Preparata, Ed.), JAI Press, London, England, 1983, pp. 1–33.

[7] B. Chazelle and J. Friedman, A deterministic view of random sampling and its use in geometry, *Combinatorica* 10 (1990), 229–249.

[8] L.P. Chew and K. Kedem, A convex polygon among polygonal obstacles: placement and high-clearance motion, *Comput. Geom. Theory Appls.* 3(2) (1993), 59–89.

[9] M. de Berg, K. Dobrindt, and O. Schwarzkopf, On lazy randomized incremental construction, *Discrete Comput. Geom.* 14 (1995), 261–286.

[10] L. Guibas, L. Ramshaw, and J. Stolfi, A kinetic framework for computational geometry, *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, 1983, pp. 100–111.

[11] K. Kedem and M. Sharir, An efficient motion planning algorithm for a convex rigid polygonal object in 2-dimensional polygonal space, *Discrete Comput. Geom.* 5 (1990), 43–75.

[12] K. Kedem, M. Sharir and S. Toledo, On critical orientations in the Kedem-Sharir motion planning algorithm for a convex polygon in the plane, *Proc. 5th Canadian Conference on Computational Geometry* (1993), 204–209.

[13] D. Leven and M. Sharir, An efficient and simple motion planning algorithm for a ladder moving in two-dimensional space amidst polygonal barriers, *J. Algorithms* 8 (1987), 192–215.

[14] D. Leven and M. Sharir, On the number of critical free contacts of a convex polygonal object moving in two-dimensional polygonal space, *Discrete Comput. Geom.* 2 (1987), 255–270.

[15] D. Leven and M. Sharir, Planning a purely translational motion for a convex object in two–dimensional space using generalized Voronoi diagrams, *Discrete Comput. Geom.* 2 (1987), 9–31.

[16] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM* 30 (1983), 852–865.

[17] F. Preparata and S. Hong, Convex hulls of finite sets of points in two and three dimensions, *Commun. ACM* 20 (1977), 87–93.

[18] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[19] F.P. Preparata and R. Tamassia, Efficient point location in a convex spatial cell-complex, *SIAM J. Comput.* 21 (1992), 267–280.

[20] J.T. Schwartz and M. Sharir, On the Piano Movers' problem: I. The case of a rigid polygonal body moving amidst polygonal barriers, *Comm. Pure and Appl. Math.* 36 (1983), 345–398.

[21] M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, 1995.

[22] M. Sharir and S. Toledo, Extremal polygon containment problems, *Comput. Geom. Theory Appls.* 4 (1994), 99–118.

[23] S. Sifrony and M. Sharir, A new efficient motion planning algorithm for a rod in two-dimensional polygonal space, *Algorithmica* 2 (1987), 367–402.

[24] L. Valiant, Parallelism in comparison problems, *SIAM J. Comput.* 4(3) (1975), 348–355.