2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Mapping the Configuration Space of Polygons Using Reduced Convolution

Evan Behar and Jyh-Ming Lien

***Abstract*— Configuration space (C-space) plays an important role not only in motion planning but also in geometric modeling, shape and kinematic reasoning, and is fundamental to several basic geometric operations, such as continuous collision detection and generalized penetration depth estimation, that also find their applications in motion planning, animation and simulation. In this paper, we developed a new method for constructing the boundary of the C-space obstacles (C-obst) of polygons. This method is simpler to implement and is theoretically more efficient than the existing techniques. Our main idea is to devote the computation on the discontinuity in temporal and spatial coherence where the structure of the C-obst changes. We also developed a method for estimating the generalized penetration depth by computing the distance between the query point and the C-obst surface.**

I. Introduction

The Configuration space (C-space) of a movable object P is the enumeration of all configurations of P. A closed subset of the C-space that causes P to collide with obstacles Q is called C-space obstacle (C-obst).

It is well known that computing an explicit geometric representation of C-obst is intractable for objects with high degree of freedom [1], and researchers have been successfully solving difficult problems without computing the C-obst, e.g., using probabilistic motion planners (see [2]). However, an explicit representation of C-space remains important to many problems, including problems that require complete motion planners (e.g., assembly/disassembly), CAD (e.g., Caine's design of shape [3]), virtual prototyping, object placement [4] and containment [5]. In addition, C-space mapping is fundamental to basic geometric operations, such as continuous collision detection and generalized penetration depth estimation.

Since the early 1980s to the mid-1990s, many researchers have proposed several methods to compute and approximate various types of representation of the C-obst. However, not until more recently have newer developments (e.g. the idea of *configuration products* by Nelaturi and Shapiro [6]) been made toward improving and generalizing these methods. See the survey by Wise and Bowyer [7] for a complete review on these earlier works and see Section II for a brief overview of the related and recent works.

In this paper, we propose a new method for mapping 2-d polygons to their 3-d C-obst. Our method represents the boundary (∂C-obst) of C-obst as a set of *ruled surfaces*. The proposed method is simpler to implement than the existing methods in the literature [4], [8] and is often more efficient. These main advantages are provided by a new algorithm that allows us to map C-obst using the idea of *critical orientations*. As a warm-up, we will start our discussion using convex polygons (in Section IV-A). We then show that the ∂C-obst of non-convex polygons can be constructed by updating the M-sum at the critical orientations constructed from the reduced convolution (in Section IV-B). The time complexity of our method is $O(m^3n^3 + bT_{cd})$ for polygons with m and n vertices, where b is the number of boundaries of C-obst, and T_{cd} is the time for a single collision query. We also show that the resulting C-obst can be used for efficiently estimating the generalized penetration depth by computing the closest feature between the query point and the ruled surfaces (Section V-B).

Both authors are with the Department of Computer Science, George Mason University, Fairfax, Virginia, USA, 22030, {ebehar,jmlien}@gmu.edu

II. Related Work

The idea of C-space mapping was first proposed by Lozano-Pérez [9]. His original idea was to construct the C-obst by slicing it at a predefined resolution along the rotational axis, and computing the Minkowski sum (M-sum)

$$-P \oplus Q = \{-p + q \mid p \in P, q \in Q\} \tag{1}$$

of the robot P and the workspace obstacle Q at each slice. To connect two consecutive slices at θ and θ', Lozano-Pérez proposed to use the swept volume (area) of P rotating from θ to θ' to replace P in Eq. 1.

Since then, more techniques and representations (including grids [10], bounding shapes [11], analytical functions [4], [8], and semi-algebraic sets [1], [12]) have been proposed, mostly in the context of motion planning. Note that the mapping techniques for free-flying robots and fixed-base articulated (manipulator) robots are very different. Since this paper focuses on free-flying robots, for readers interested in the manipulator C-space mapping, please refer to the work by Branicky and Newman [13]; Hwang [14]; and Ward and Katupitiya [15] for more recent work on this topic.

Among all these techniques, the slicing-based strategy remains quite popular due to its simplicity. Notably, Zhu and Latombe [11] generalized the idea to use the outer and inner swept areas (which are the union and the intersection of the areas swept out by the robot) to bound ∂C-obst. Kavraki [10] proposed a method to construct C-obst by computing the convolution of two polygons (represented as pixels) using Fast Fourier Transform. Curto and Moreno [16] extended Kavraki's method to handle both free-flying and articulated robots. Later, Sacks and Bajaj [17] proposed to generate the slices for curved 2-d objects at fixed intervals. There

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

are two major drawbacks of the slicing-based approaches. First, computation can be wasted. There are significant performance improvements that can be gained by exploiting the temporal and spatial coherence. Second, the M-sums are normally separated by a *fixed rotational resolution*, which is defined empirically. A better approach is to identify "events" where the structure of the M-sum changes.

Thus, another line of research focused on the exact representations of either the boundary using analytic functions or the volume using semi-algebraic sets. For example, Donald [12] dealt with motion-planning problems with a 3-d free-flying robot amongst polyhedral obstacles. Both robot and obstacles are composed of a set of convex shapes; the C-obst are therefore a set of 6-d *contact surfaces*. The intersections of these contact surfaces are computed to help the robot move along the intersection or slide from one surface to another. Another example is the work done by Halperin et al. [18]. They studied the motion planning problem of a L-shaped robot (composed of two line segments) among point obstacles. In this setting, the C-obst *is* a set of ruled-surfaces. The simplicity of the problem allows them to construct the complete C-obst by identifying all critical orientations to determine the changes of the line segment arrangements. In these techniques, the methods proposed by Avnaim et al. [4] and Brost [8] are the ones closely related to our work.

Avnaim et al. [4] proposed to compute ∂C-obst using contact regions. A contact region is computed between a vertex of P and an edge of Q or vice versa. Their method computes sets of translations that result in contact between the polygons, while excluding regions which represent contacts that are in collision. This rotation/intersection space is computed as analytic functions that are put into one-to-one correspondence with the actual configuration space to compute the general contact regions for a given contact. Their method pre-computes a discrete set of contacts, and then computes the contact regions for each contact, which become the facets of the boundary. Though M-sums are not used in this method, a similar configuration space is produced. Their algorithm has time complexity $O(n^3m^3 \log nm)$ for polygons with n and m vertices.

Similar to Avnaim et al. [4], Brost [8] also considered all possible contacts. Local information between the contact vetex/edge pair is used to compute the contact regions. Each contact region is a ruled surface. For non-convex polygons, part of the contact region may belong to the interior of the C-obst. Therefore, all contact regions are tested for intersection. Finally, each contact region is trimmed around the boundary created by the intersections and the remaining area is part of the ∂C-obst.

Note that in both methods a contact region can have zero area on ∂C-obst. This means that the entire contact region is trimmed. In fact, even for simple shapes (for example the star shape shown in Fig. 2), many contact regions will not be on the surface of C-obst. As a result, significant computation is wasted on finding the intersections between contact regions (which is a computational expensive operation). Our method avoids this problem and accelerates the mapping process (1) by reducing the number of contact regions and (2) by detecting and resolving the intersection in the two dimensional space. Another disadvantage of these approaches is that no clear means for computing the distance (e.g., for penetration depth estimation) is presented or easily derived from the representations.

Recently, Varadhan and Manocha [19] also proposed an approach that generates polygonal meshes to approximate the ∂C-obst using a marching cube technique to extract the iso-surface from a signed distance field. They used an adaptive cell to improve the robustness and efficiency of their method. The construction of the non-directional backprojection for solving compliant motion-planning problems under uncertainty is also similar to the C-space mapping. Donald [20] and Briggs [21] have proposed ways to accelerate the construction by identifying the critical points of topological changes of the backprojections with respect to the visibility graph of the workspace. The main difference is that rays, instead of line segments, are used in these computations.

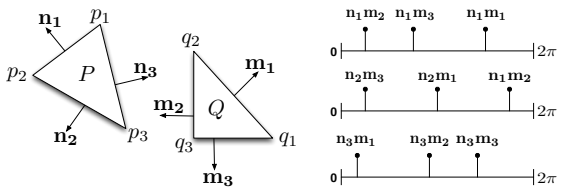
III. Preliminaries

In this section, we define the notations that are used throughout the paper. We assume that P is movable while Q is stationary. Both P and Q are simple polygons composed of n and m (counterclockwise) ordered vertices, respectively. Our approach is based on computing and updating the M-sums using convolution. The convolution of two shapes P and Q, denoted as $P \otimes Q$, is a set of line segments in 2-d that is generated by "combining" the segments of P and Q [22]. One can think of the convolution as the M-sum that involves only the boundary, i.e., $P \otimes Q = \partial P \oplus \partial Q$. It is known that the convolution forms a superset of their M-sum boundary [23], i.e., $\partial(P \oplus Q) \subset P \otimes Q$. If both P and Q are convex, $\partial(P \oplus Q) = P \otimes Q$. Otherwise, it is necessary to trim the line segments or the facets of the convolution to obtain the M-sum boundary. Recently, Wein [24] shows a robust and exact method based on convolution for non-convex polygons. To obtain the M-sum boundary from the convolution, his method computes the arrangement induced by the line segments of the convolution and keeps the cells with non-zero winding numbers. A more detailed review on the M-sum can be found in our previous work [25].

An edge $\overline{p_ip_{i+1}}$ of P and a vertex q_j of Q (or vice versa) form a segment of $P \otimes Q$ if $\overrightarrow{p_i\,p_{i+1}} \in [\overrightarrow{q_{j-1}\,q_j}, \overrightarrow{q_j\,q_{j+1}})$, and we say that $\overline{p_ip_{i+1}}$ and q_j are *compatible*. Equivalently, $\overline{p_ip_{i+1}}$ and q_j are compatible if the outward normal of $\overline{p_ip_{i+1}}$ lies between the normals of the incident edges of q_j. For example, in Fig. 1(a), $\overline{p_3p_1}$ and q_1 are compatible. When rotation is considered, the edges of P are rotated about a fixed center, c, and an edge/vertex pair is *alive* if they are compatible and *dead* otherwise. Without loss of generality, we assume that P rotates counterclockwise about c, and c is the world origin, so that $c_x = c_y = 0$.

IV. Our Methods

We will first discuss the case of convex polygons in Section IV-A and then extend the ideas the handle non-



(a) (b)

Fig. 1. (a) Two convex polygons P and Q shown with the edges outward normals. (b) Events for $\vec{n_1}$, $\vec{n_2}$, and $\vec{n_3}$ (from top to bottom) when P rotates counterclockwise from 0 to 2π. For example, when P rotates $\pi/4$, $\vec{n_1}$ and $\vec{m_2}$ (and $\vec{n_2}$ and $\vec{m_3}$) become aligned and two events are issued.

convex polygons in Section IV-B.

A. C-obst of Convex Polygons

Given two convex polygons P and Q (see Fig. 1(a)), an edge of their convolution is the sum of an edge $\overline{p_i p_{i+1}}$ of P and a vertex q_j of Q or vice versa. We let θ_0 be the orientation when an edge/vertex pair is born until it dies at θ_1 when P rotates counterclockwise, and each edge/vertex pair forms a *parameterizable ruled surface* (i.e. contact region). Let $p_i = (x_0, y_0)$ and $p_{i+1} = (x_1, y_1)$. We take a vector $\vec{v} = \overrightarrow{p_i\, p_{i+1}}$ and a vector $\vec{t} = \overrightarrow{O\, q_j}$, where O is the world origin. Then the surface defined by the pair ($\overline{p_i p_{i+1}}$ and q_j) is parameterized as:

$$S_R(r,\theta) = \begin{bmatrix} (x_0 + rv_x)\cos\theta - (y_0 + rv_y)\sin\theta + t_x \\ (x_0 + rv_x)\sin\theta + (y_0 + rv_y)\cos\theta + t_y \\ \theta \end{bmatrix}, \tag{2}$$

where $r \in [0,1]$, $\theta \in [\theta_0, \theta_1)$.

Surfaces are also formed by the edges of Q as P rotates. Similarly, we let $q_j = (x_0, y_0)$ and $q_{j+1} = (x_1, y_1)$. We take the vector $\vec{v} = \overrightarrow{q_j\, q_{j+1}}$ and a vector $\vec{t} = \overrightarrow{O\, p_i}$. The surface for the pair p_i and $\overline{q_j q_{j+1}}$ is parameterized as:

$$S_N(r,\theta) = \begin{bmatrix} y_0\cos\theta - x_0\sin\theta + t_x + rv_x \\ x_0\cos\theta + y_0\sin\theta + t_y + rv_y \\ \theta \end{bmatrix}. \tag{3}$$

In order to support operations like distance query and line intersection, each surface is stored as a tuple $(p, q, \theta_0, \theta_1)$, where p and q are the indices to the vertices and edges of P and Q, and θ_0 and θ_1 define the *birth and death* orientations of the surface. To construct the surfaces in this representation, we use a sweeping algorithm that updates the convolution at critical orientations (events). Fig. 1(b) shows all the events for each edge of P. To handle each event, we delete two segments from the convolution and create two new segments. For example, at event $n_3\vec{m}_1$ in Fig. 1(b), the pairs $\langle\overline{p_3p_1}, q_1\rangle$ and $\langle p_1, \overline{q_1q_2}\rangle$ both die and the pairs $\langle\overline{p_3p_1}, q_2\rangle$ and $\langle p_3, \overline{q_1q_2}\rangle$ are both born. Note that these changes are local, therefore each event can be handled in a constant time, and there can be at most mn events. Moreover the events for each edge of P is simply an offset copy of the normals of Q, so all (sorted) events can be built in linear time. Therefore, the entire computation takes only $\Theta(nm)$ time.

For two convex polygons, the surfaces trivially form ∂C-obst, as the surfaces will never penetrate into the interior of the C-obst. However, in the case where one or both of the inputs are non-convex, this is not guaranteed to be the case. This poses fundamental problems in computing the penetration depth on such a solid; for example, the closest point on a non-manifold hull to a query point inside the solid may consequently still be on the interior of the solid.

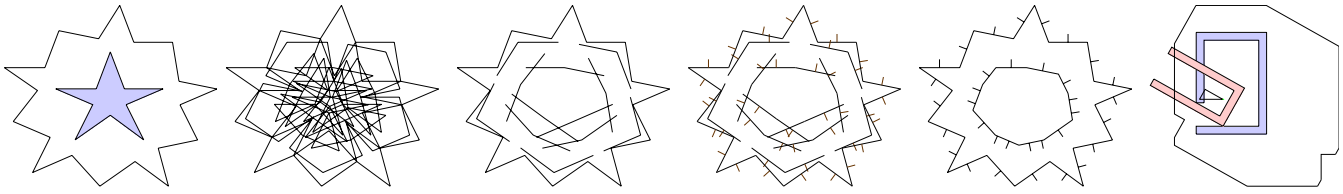
B. C-obst of Non-Convex Polygons

We now generalize the approach to consider non-convex polygons. Similar to the algorithm that we proposed for computing the ∂C-obst for convex shapes, the algorithm for the ∂C-obst for non-convex shapes also consists of $\Theta(mn)$ events for creating and deleting each contact patch. The main difference is that each patch is now generated by a segment (with varying length) in the **reduced convolution** [26]. A reduced convolution is a set of convolution segments $\overline{p_i p_{i+1}} \oplus q_j$ and $p_k \oplus \overline{q_l q_{l+1}}$ and q_j and p_k must be convex. In our previous work [26], we have shown a close relationship between M-sum and reduced convolution. To compute M-sum, we first identify orientable loops from the reduced convolution, in each of which all edges have normals pointing consistently inward or outward. These loops form potential boundaries of the M-sum and are further filtered by analyzing their nesting relationship. Finally, the remaining boundaries are filtered by checking the intersections between the input polygons placed at the configurations along these loops. Each of these steps is illustrated in Fig. 2.

In addition to these events, the intersection of line segments (from the reduced convolution) also changes during the rotation of P, and these changes can affect the topological structure of the M-sum. Therefore, the second type of event for a given segment s is a list of rotations $\{\theta_i\}$ where the intersection status of s changes (e.g., s starts to intersect or stops intersecting with a segment) when P rotates from 0 to 2π. There can be $O(C^2)$ such events, where C is the size of the reduced convolution.

The data structure that we use for representing the surface is also a tuple $(p, q, \theta_0, \theta_1, s_1, s_2)$, where p, q, θ_0, and θ_1 are the same as the convex case, and s_1 and s_2 are indices to the convolution segments intersecting with the segment between θ_0, and θ_1. We use the same sweeping algorithm to construct this data structure. To handle the events where a segment s is created (or deleted), we simply add (or remove) s to the reduced convolution and add (or remove) the intersections due to s. To handle the second type of event, intersections due to the events are updated.

When a new intersection occurs, s is split into two segments at the intersection point. One of the new segments resulting from this split may be degenerate in the case of a new intersection occuring at an endpoint of s, however this does not require any special case handling. The degenerate segment will expand with the movement of the intersecting segment into a proper line segment and generate the correct ruled surface. An additional benefit to this is that the result of the splitting operations guarantees us that at most two



(a) input/output (b) convolution (c) subset of (b) (d) segment normals (e) orientable loops (f) incorrect loop

Fig. 2. Steps for computing the M-sum of two simple polygons. In (a), the boundary of the M-sum of a star and a slightly rotated copy of it is shown.

line segments intersect each of the resulting segments of the convolution. This ensures that our data structure is always sufficient to represent a particular surface patch of the C-obst.

For both events, we check if the event site (i.e., the new or dead intersection) is locally orientable and manifold to decide if a loop (of constant size) should be created or deleted as described above. Note that we will skip the last two filters (i.e., the polygon nesting and the collision detection filters) during sweeping. Both filters will only be needed at the end of the sweep to reject the false 3-d hole boundaries. Similarly, we can show that only one (2-d) point is needed to verify each 3-d boundary.

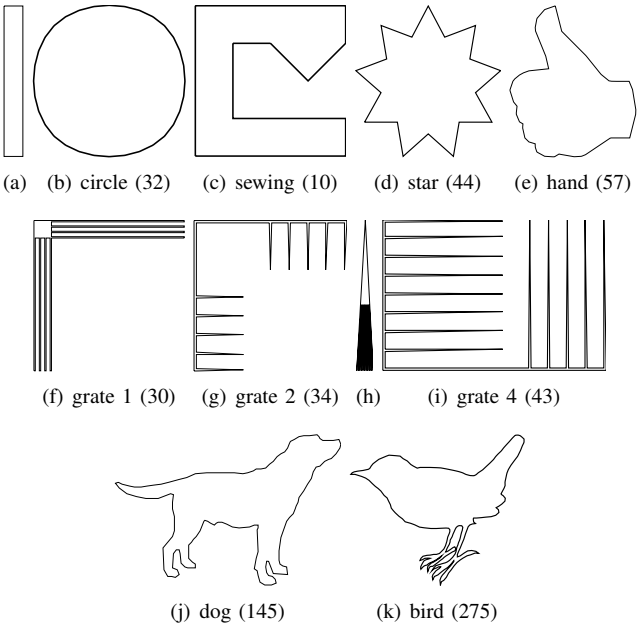
An example of the results generated by our method is shown in Fig. 4. From the figures, we can see that the interior part of the C-obst is hollow and all the extra parts of the reduced convolution are correctly removed.

In the rest of this section, we briefly discuss how to detect the second type of event. Each of these events can be found in constant time, though the computation requires us to classify the types of edges and surfaces, i.e., S_R and S_N in Eqs. 2 and 3, since they move (and rotate) in different ways.

Consider two rotating edges in the convolution e_1 and e_2 that may intersect at some θ. We let θ_0 be the first value of θ for which e_1 and e_2 are both alive. An edge e_i lies along a line L_i whose equation is $y_i(\theta) = m_i(\theta)x_i(\theta) + b_i(\theta)$. The intersection $(x(\theta), y(\theta))$ of L_i can be computed so that

$$x(\theta) = \frac{b_2(\theta_0) - b_1(\theta)}{m_1(\theta) - m_2(\theta_0)} \ ,$$

where $b_i(\theta) = x_0 \cos\theta + y_0 \sin\theta + t_y - m_i(\theta)(y_0 \cos\theta - x_0 \sin\theta + t_x)$ and $m_i(\theta) = \frac{1+m_i \tan\theta}{m_i - \tan\theta}$, and m_i is the initial slope of L_i. It is trivial to compute $y(\theta)$ from $x(\theta)$. Then from the intersection of the lines, we solve for θ such that the intersection $(x(\theta), y(\theta))$ will fall into the range of the line segments e_1 and e_2. This is done by classifying the segments into three cases that involving *rotating* and *non-rotating* edges. We say that the edges that create S_R surfaces are rotating edges and the edges that create S_N surfaces are non-rotating edges. Therefore e_1 and e_2 can be either (1) both rotating edges, (2) both non-rotating edges or (3) a rotating and non-rotating pair. In certain cases, the segments can be checked quickly to determine if they ever intersect. The details are shown in our technical report [27].



(a) (b) circle (32) (c) sewing (10) (d) star (44) (e) hand (57)

(f) grate 1 (30) (g) grate 2 (34) (h) (i) grate 4 (43)

(j) dog (145) (k) bird (275)

Fig. 3. Examples used in the experiments. The numbers in the parentheses are the size of the polygon. (a) bar (4), (h) grate 3 (34). Some of these models are inspired by those in [28], [24].

V. Experimental Results and Application

A. Results

We have implemented the proposed method in C++. In this section, we will show some of the results that we obtained from this implementation using the examples shown in Fig. 3. In these examples, there are two convex polygons and 9 non-convex polygons. The number of the vertices of each polygon is also shown. Some of these models are inspired by those in [28], [24]. All the experiments are performed on CPUs at 2.13 GHz with 4 GB RAM.

In Table I, we show the computation time for constructing ∂C-obst using the proposed method. The running times range from a fraction of a second to close to an hour. Since we have no other public implementation to compare to, it is important to look at these running times relatively. Therefore, we list the number of ruled surfaces before trimming (N_s), the number of ruled surfaces on the final ∂C-obst (n_s), and the number of C-obst (external and hole) boundaries (n_b). From the values of N_s, n_s, and n_b, it is clear all of them can affect the computation time. For example, both "star/star" and "grate 1/grate 2" take about the same time to compute,

TABLE I

EXPERIMENTAL RESULTS FOR C-SPACE MAPPING. HERE, N_s IS THE NUMBER OF RULED SURFACES BEFORE TRIMMING, n_s IS THE NUMBER OF RULED SURFACES ON ∂C-OBST, n_s IS THE NUMBER (BOTH EXTERNAL AND HOLE) C-OBST BOUNDARIES, AND t IS THE TOTAL COMPUTATION TIME IN SECONDS.

P/*Q*	bar/circle	bar/sewing	star/star	star/hand	grate 1/grate 2	bar/grate4	grate 3/grate 4	dog/bird
t	0.1	0.05	4.9	6.8	4.9	0.3	21.7	3350.6
N_s	256	68	2288	3286	1028	244	991	39145
n_s	256	82	3499	3034	4097	1027	1947	18500
n_b	1	1	1	1	126	17	39	1

but the number of ruled surface patches in "grate 1/grate 2" is half of that in "star/star." Therefore, it is the large n_b in "grate 1/grate 2" that increases the computation time. Moreover, it is clear that the reason that the "dog/bird" takes nearly an hour to finish is because of N_s, which is about 40 times the N_s of "grate 1/grate 2" and "grate 3/grate 4." One single example that we cannot explain directly from Table I is the time difference between "grate 1/grate 2" and "grate 3/grate 4." Both N_s and n_s are smaller and n_b is larger in "grate 1/grate 2." However if we look deeper into the the number of segments and the number of intersections in both reduced convolution, the reason of the time difference becomes clear. The pair "grate 1/grate 2" has 469 line segments but has only 1204 intersections, and the pair "grate 3/grate 4" has only 400 line segments but has 1544 intersections.

B. Application: Generalized Penetration Depth Estimation

The parameterizations in Eqs. 2 and 3 also yield distance functions in r and θ which can be used to find the minimum distance to a given facet relatively easily. Let p be a query point and let $f(r) = (x_0 + rv_x)$, $g(r) = (y_0 + rv_y)$, $F(r) = g(r)\cos\theta - f(r)\sin\theta$, and $G(r) = f(r)\cos\theta + g(r)\sin\theta$, then the square distance $d(r, \theta, p)$ for the rotating edges:

$$d(r,\theta,p) = (F(r)+t_x-p_x)^2+(G(r)+t_y-p_y)^2+w^2(\theta-p_z)^2$$

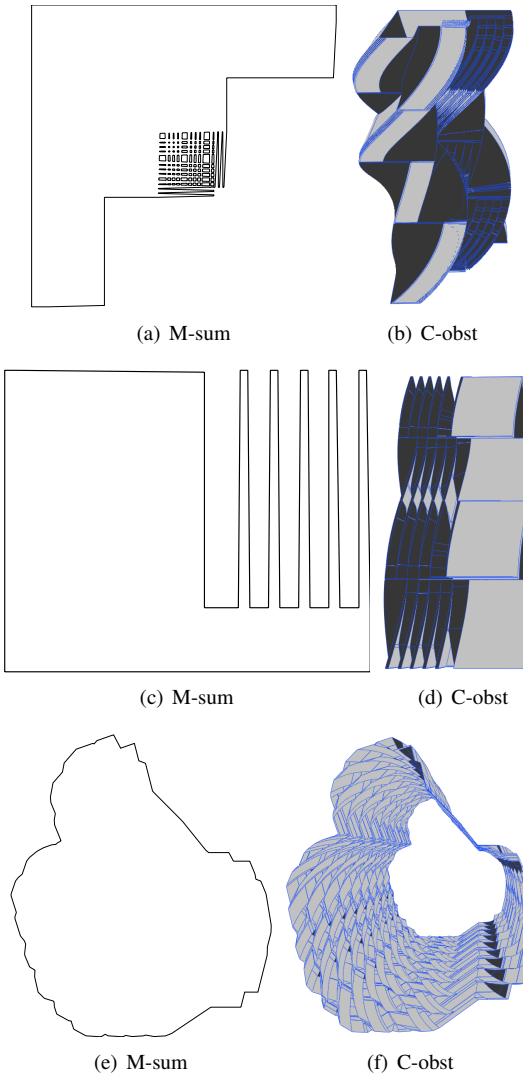
If we fix r, then d is a very well-behaved sinusoid, and while there does not seem to be a closed-form solution for the global minimum, it is easy to find the minimum using simple gradient descent. If by contrast we fix θ, then d is simply quadratic in r, and finding the global minimum on $[0, 1]$ is also quite easy.

Computing $d(r, \theta)$. In the case of S_R (see Fig. 5(a)), the regularity of the surface of the distance function allows us to easily calculate a global minimum by finding θ values for the global minimums at $r = 0$ and $r = 1$ by gradient descent, then finding the global minimums for r when we fix θ at the values found by fixing r initially. Picking the minimum of the yielded values gives us the global minimum of the distance function, as well as yield r and θ values which explicitly give us the closest point on the facet (see Fig. 5(c)). The distance function follows similarly for S_N (see Fig. 5(b)), except that because the r term is independent of the rotation, the surface is somewhat more regular. We still end up with no clear closed-form solution for the sinusoid however, so we solve for the minimum using gradient descent as above.

In the case of non-convex polygons, a surface may have a left-r-bound function $r_{min}(\theta)$ and a right-r-bound function $r_{max}(\theta)$ that describe how its non-manifold intersections move as θ changes, so that its associated facet is r-bounded at a given θ by $[\max\{0, r_{min}(\theta)\}, \min\{1, r_{max}(\theta)\}]$. These same r-bounds apply to the distance function. As a consequence, finding seed values for r and θ in the general case is more complicated. To deal with this issue, we choose to seed at regular intervals. Let segment e have its birth at θ_0 and death at θ_1, then seed values are taken for $\theta \in \{k\frac{|(\theta_0-\theta_1)|}{8} : k \in \mathbb{Z}, 0 \le k \le 8\}$. For each of these θ values, we seed at $\max\{0, r_{min}(\theta)\}$, $\min\{1, r_{max}(\theta)\}$, and $(\max\{0, r_{min}(\theta)\} + \min\{1, r_{max}(\theta)\})/2 \pm \epsilon$, just to the left and right of the medial axis of the r-bounds.

This gives us a total of 36 seeds per surface. We use so many seeds largely because the r-bounds are irregular enough that some descents may get caught along the boundary. This spread however provides good coverage. Because of the regularity of the surface itself, a particular iteration of the gradient descent tends to converge in a small number of iterations and so the total cost of the gradient descent is relatively low in any case.

Computing penetration depth. Given a configuration p of P, we would like to find the closest feature on ∂C-obst. This problem can be decomposed into two steps: (1) find the closest surface f to p and (2) find the closest point on f to p. We have already proposed a method for the second step. For finding the closest surface, ideally, we can precompute the *Voronoi tessellation* of the space using each surface as a site, and then find which cell q is in. However, both computing the tessellation and finding the enclosing cell seem to be difficult. The only properties that we know are that the boundaries of the tessellation are also ruled surfaces, and each cell forms a single connected component. Based on these properties, we propose a sampling-based approach. Initially, a set of uniformly distributed samples are taken, and the closest surface for each sample point is computed offline using a brute-force search (through all surfaces). Each query point is then categorized by its k nearest neighbors, and only the $n \le k$ surfaces associated with those neighbors are checked. For the results in Table II, we set $k = 10$ experimentally. For convex polygons, this approach yields a very high rate (98.3% for bar/circle) of identifying the actual closest facet and low average error values ($< 10^{-5}$) when a facet other than the closest is chosen for distance comparison. For non-convex polygons, this approach still yields very high rate ($> 95.7\%$) of identifying the actual closest facet and low average error values ($< 10^{-3}$). The accuracy and error

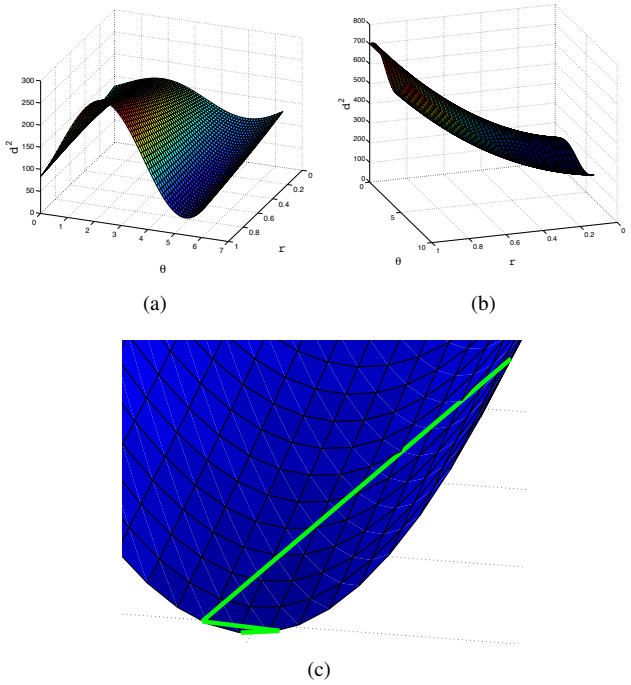


(a) M-sum (b) C-obst

(c) M-sum (d) C-obst

(e) M-sum (f) C-obst

Fig. 4. (a) and (b) are the M-sum and C-obst of grate 1 and grate 2 in Fig. 3, respectively. The darker (lighter) patches in (b) are S_R (S_N) surfaces. (c) and (d) are generated from bar and grate 3 in Fig. 3. (e) and (f) are generated from star and hand in Fig. 3.



(a) (b)

(c)

Fig. 5. (a) An example distance function for S_R, $c = (0,0), p = (10,-5,\pi)$, $v = (1,4)$, $x_0 = 1, y_0 = 1$. (b) An example distance function for S_N, same parameters as (a) except that v is elongated. (c) An example gradient descent for the sinusoid portion zoomed in on the gradient descent. The descent converges in just 5 iterations (two iterations in which only step-size is adjusted), and in this case finds not only the local minimum for the sinusoid at r_{max}, but also the global minimum.

TABLE II

RESULTS FOR PENETRATION DEPTH ESTIMATION. HERE ϵ IS THE AVG. DISTANCE ERROR, t IS THE AVG. QUERY TIME OVER 1000 QUERIES, AND T IS THE TIME TO PRE-COMPUTE THE DISTANCES FOR ALL SAMPLES.

P/Q	bar/circle	bar/sewing	star/star	grate 1/grate 2
ϵ	0.000004	0.00067	0.0004	0.0001
t	7.8ms	6.5ms	8.5ms	12.0ms
T	50.1s	80.1s	398.2s	862.2s

are estimated by comparing to the results of the brute force approach.

C. Complexity Analysis

We now analyze the time complexity of the proposed method. When P and Q have n and m vertices which include n' and m' reflex vertices, respectively, there will be $2mn$ segments in the complete convolution; in the reduced convolution there are at most $(m-m')n+(n-n')m$ segments. That is, the arrangement of the reduced convolution is at least 4 times less complex than that of the complete convolution when $n' = 1/2n$ and $m' = 1/2m$. Thus, the reduction will further reduce the complexity of the arrangement of 3-d rule-surface patches by at least 8 times. Note that this analysis is based on the assumption that a convex vertex is compatible with $\Theta(n)$ edges and in the worst case that each segment will intersect all the other segments. In the examples that we have above, the difference between the reduced and complete convolutions is more significant (e.g., "star/star" and "dog/bird"). The time complexity for computing the M-sum of P and Q is $O\left((mn+I)\log(mn+I) + \ell T_{cd}\right)$, where $I = O(m^2n^2)$ is the complexity of the arrangement of the reduced convolution, ℓ is the number of loops, and $T_{cd} = O(mn)$ is the collision detection time in our implementation. The time complexity for computing the C-obst of P and Q is $O\left((mn+I)\log(mn+I) + m^2n^2T_e + bT_{cd}\right)$, where b is the number of (hole) boundaries in the C-obst, and $T_e = O(mn)$ is the time for handling each event (i.e., finding all new/dead intersections and update the M-sum locally near the intersections).

VI. CONCLUSION AND FUTURE WORK

We proposed a new method for constructing ∂C-obst. Our methods takes $O(m^3n^3 + bT_{cd})$ for polygons with m and n vertices and C-obst with b boundaries, where T_{cd} is time spent on collision detection. We believe that this method

is easier to implement and more efficient than the existing methods. The main step in our method is the computation of the Minkowski sum (M-sum) which is based on the ideas of reduced convolution. Then the M-sum is updated at each critical orientation to construct ∂C-obst. The main efficiency gain is that there are significantly fewer segments and surfaces produced compared to the existing methods.

We consider this work as the first step toward a more interesting and challenging problem: computing the C-obst of 3-d polyhedra. Donald [12] and several others have studied the problems, but we believe that there is still room for significant improvement. For example, most of these methods depend on convex decomposition. As we have discussed above, computing C-obst for convex polyhedra is not difficult. However, it is unclear how to deal with non-convex polyhedra without using 3-d convex decomposition, which is notoriously slow. We hope to provide an answer to this by using the ideas from the recent development in M-sum and the ideas from this paper.

Computing the C-obst of *3-d polytopes* is similar but involves more steps in creating and handling events. An important observation is that the event list of each edge of P (see Fig. 1(b)) is simply an offset of Q's Gauss map! Similarly, given polytopes P and Q, the events for each facet of P can be constructed by rotating Q's Gauss map. In 3-d, the facets of the M-sum can only come from two sources: fv-facets, generated from a facet of P and a vertex of Q or vice versa, and ee-facets, generated from a pair of edges from P and Q, respectively. Similar to polygons, a M-sum facet (f_M) is valid if the orientations of f_M's primitives (i.e., a vertex-facet pair or an edge-edge pair) from P and Q match each other, and the events occur at these boundary conditions, where the M-sum structure changes. Finally, all the events can be enumerated by overlaying all the rotated Gauss maps (one for each P's facet).

The resulting data structure for representing the C-obst of polytopes is a list of facets. Each facet f_M is a tuple (p, q, r), where p and q are the indices to the vertices, edges or facets of P and Q, and r is a convex region (in the Gauss map) in which f_M remains valid. One can show that the complexity of this data structure is $O(n^2m + nm^2)$, where n and m are the complexities of P and Q. If a brute force method is used to enumerate all possible M-sums, it will take $O(n^3m^2 + n^2m^3)$.

VII. Acknowledgement

This work is supported in part by the National Science Foundation under grants IIS-096053 and CNS-1205260 and by the Air Force Office of Scientific Research FA995-12-1-0238.

References

[1] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[2] S. M. LaValle, *Planning Algorithms*, 6th ed. Cambridge University Press, 2006.

[3] M. Caine, "The design of shape interactions using motion constraints," in *1994 IEEE International Conference on Robotics and Automation, 1994. Proceedings.*, 1994, pp. 366–371.

[4] F. Avnaim and J. Boissonnat, "Polygon placement under translation and rotation," *STACS 88*, pp. 322–333, 1988.

[5] B. S. Baker, S. J. Fortune, and S. R. Mahaney, "Polygon containment under translation," *J. Algorithms*, vol. 7, pp. 532–548, 1986.

[6] S. Nelaturi and V. Shapiro, "Configuration products in geometric modeling," in *SPM '09: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, 2009, pp. 247–258.

[7] K. Wise and A. Bowyer, "A survey of global configuration-space mapping techniques for a single robot in a static environment," *The International Journal of Robotics Research*, vol. 19, no. 8, pp. 762–779, 2000.

[8] R. Brost, "Computing metric and topological properties of configuration-space obstacles," in *1989 IEEE International Conference on Robotics and Automation, 1989. Proceedings.*, 1989, pp. 170–176.

[9] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, pp. 108–120, 1983.

[10] L. Kavraki, "Computation of configuration-space obstacles using the fast fourier transform," *IEEE Trans. Robot. & Autom*, vol. 11, pp. 255–261, 1995.

[11] D. Zhu and J. Latombe, "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9–20, 1991.

[12] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artif. Intell.*, vol. 31, no. 3, pp. 295–353, 1987.

[13] M. Branicky and W. Newman, "Rapid computation of configuration space obstacles," in *Proc. IEEE Int. Conf. Rob. and Aut*, 1990, pp. 304–310.

[14] Y. Hwang, "Boundary equations of configuration obstacles for manipulators," in *1990 IEEE International Conference on Robotics and Automation, 1990. Proceedings.*, 1990, pp. 298–303.

[15] J. Ward and J. Katupitiya, "Free space mapping and motion planning in configuration space for mobile manipulators," in *2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 4981–4986.

[16] B. Curto and V. Moreno, "Mathematical formalism for the fast evaluation of the configuration space," in *cira*. Published by the IEEE Computer Society, 1997, p. 194.

[17] E. Sacks and C. Bajaj, "Sliced configuration spaces for curved planar bodies," *The International Journal of Robotics Research*, vol. 17, no. 6, p. 639, 1998.

[18] D. Halperin, M. H. Overmars, and M. Sharir, "Efficient motion planning for an L-shaped object," *SIAM J. Comput.*, vol. 21, no. 1, pp. 1–23, 1992.

[19] G. Varadhan and D. Manocha, "Accurate Minkowski sum approximation of polyhedral models," *Graph. Models*, vol. 68, no. 4, pp. 343–355, 2006.

[20] B. R. Donald, "The complexity of planar compliant motion planning under uncertainty," *Algorithmica*, vol. 5, pp. 353–382, 1990.

[21] A. Briggs, "An efficient algorithm for one-step planar compliant motion planning with uncertainty," *Algorithmica*, vol. 8, no. 1, pp. 195–208, 1992.

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

[23] P. K. Ghosh, "A unified computational framework for Minkowski operations," *Computers and Graphics*, vol. 17, no. 4, pp. 357–378, 1993.

[24] R. Wein, "Exact and efficient construction of planar Minkowski sums using the convolution method," in *Proc. 14th Annual European Symposium on Algorithms*, 2006, pp. 829–840.

[25] J.-M. Lien, "A simple method for computing Minkowski sum boundary in 3d using collision detection," in *The Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Guanajuato, Mexico, Dec 2008.

[26] E. Behar and J.-M. Lien, "Fast and robust 2d Minkowski sum using reduced convolution," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, San Francisco, CA, Sep. 2011.

[27] ——, "A new method for mapping the configuration space obstacles of polygons," George Mason University, Tech. Rep. GMU-CS-TR-2010-11, 2010.

[28] D. Halperin and M. Sharir, "Arrangements and their applications in robotics: Recent developments," in *Proc. Workshop on Algorithmic Foundations of Robotics*, 1995, pp. 495–511.