Computer-Aided Design 90 (2017) 18–26

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Robust free space construction for a polyhedron with planar motion☆

Elisha Sacks [a,*], Nabeel Butt [b], Victor Milenkovic [c]

[a] Computer Science Department, Purdue University, West Lafayette, IN 47907-2066, USA
[b] Autodesk, Pier 9 The Embarcadero, San Francisco, CA 94111, USA
[c] Department of Computer Science, University of Miami, Coral Gables, FL 33124-4245, USA

ARTICLE INFO

Keywords:
Configuration space
Robust computational geometry
Path planning

ABSTRACT

We present a free space construction algorithm for a polyhedron that translates in the *xy* plane and rotates around its *z* axis, relative to a stationary polyhedron. We employ the proven paradigm of constructing the configuration space subdivision defined by patches that comprise the configurations where the boundary features of the polyhedra are in contact. We implement the algorithm robustly and efficiently. The challenge is to detect degenerate predicates efficiently and to handle them correctly. We use our ACP (Adaptive Controlled Perturbation) robustness strategy to prevent degenerate predicates due to input in special position. The remaining cases are predicates that are identical to the zero polynomial because their arguments are derived from overlapping sets of input vertices. We detect and handle these cases with custom logic. We validate the implementation by computing maximum clearance paths.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

We present research in rigid body kinematics. We consider a polyhedron *A* that translates and rotates in a plane while avoiding a stationary polyhedron *B*. The polyhedra can have multiple components and need not be convex. These kinematic pairs model vehicles that travel on the ground while avoiding obstacles, layout of parts that rest on a base, and mechanical systems with planar motion. Fig. 1 shows a simple example in which a tetrahedron *A* navigates an obstacle *B*. Our task is to compute the configurations (positions and orientations) of *A* where it is disjoint from *B*. The analysis supports motion planning [1], part layout [2], and mechanical design [3].

In an appropriately chosen coordinate system, *A* translates in the *xy* plane and rotates by angle θ around its *z* axis. The manifold with coordinates (x, y, θ) is the *configuration space*. The constraint that *A* cannot intersect *B* restricts *A* to an open subset of configuration space, the *free space*. The boundary, the *contact space*, consists of the configurations where the boundaries of *A* and *B* intersect but the interiors are disjoint. Fig. 2 shows the contact space of the example. The free space is its exterior. The free configurations from Fig. 1 are drawn as green spheres. The tetrahedron can move in or out of the obstacle along the curve that connects these configurations.

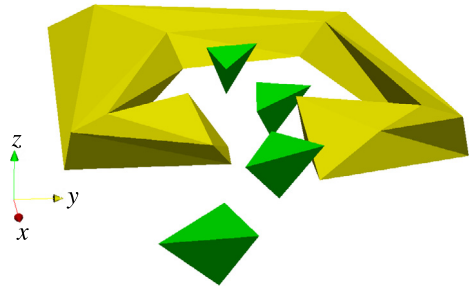

Fig. 1. Tetrahedron in four configurations and obstacle.

We present an algorithm for constructing a boundary representation of free space (Section 3). The key fact is that at a contact configuration either a vertex of one polyhedron lies on a facet of the other polyhedron or two edges share a point. The configurations where a vertex/facet or an edge/edge pair intersect form a surface called a *patch*. Fig. 3 shows the two types of patches for our example and Fig. 4 shows all the patches. The patches subdivide configuration space into open regions, called *cells*. The free space is a disjoint union of cells and the contact space is a subset of the union of the patches. Our algorithm constructs the subdivision and identifies the free space cells.

We develop the first robust implementation of the algorithm, using our ACP robustness strategy [4] (Section 4). We validate the implementation by computing maximum clearance paths (Section 5).

☆ This paper has been recommended for acceptance by Dr. Mario Botsch, Dr. Yongjie Jessica Zhang and Dr. Stefanie Hahmann.

* Corresponding author.
E-mail addresses: eps@purdue.edu (E. Sacks), nabeel.butt@autodesk.com (N. Butt), vjm@cs.miami.edu (V. Milenkovic).

http://dx.doi.org/10.1016/j.cad.2017.05.017
0010-4485/© 2017 Elsevier Ltd. All rights reserved.

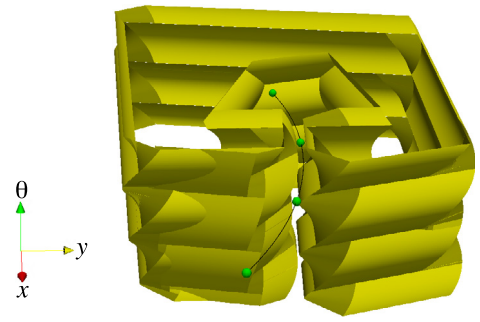

Fig. 2. Contact space of tetrahedron and obstacle.

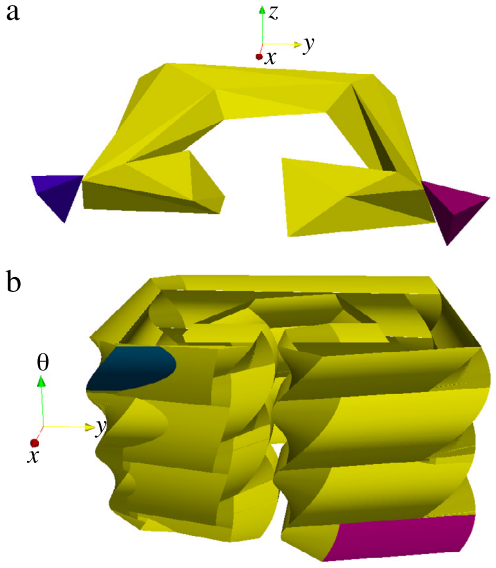

Fig. 3. Vertex/facet (left) and edge/edge (right) contacts (a) and corresponding patches in same colors (b).

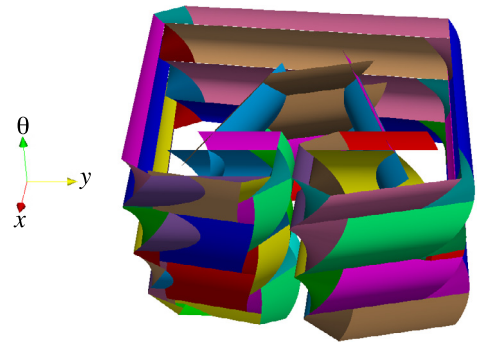

Fig. 4. Patches for tetrahedron and obstacle.

2. Prior work

The difficulty of free space construction grows sharply with configuration space dimension. Dimension two is implemented in the CGAL library [5]. For dimension three, we have devised state-of-the-art algorithms for a planar body [4] and for a translating polyhedron [6]. Both algorithms have the same structure as the current algorithm – construct the subdivision of the patches and identify the free cells – and are implemented robustly using ACP. Kim, Elber, and Kim [7] construct patch intersection curves for planar bodies bounded by spline curves. The only prior algorithm for the configuration spaces in this paper [8] is not robust and never appears to have been implemented. The best algorithm for configuration spaces of dimension four and up [9] has exponential complexity in dimension and has not been implemented.

The limited progress in free space construction has led to algorithms in which the configuration space is sampled and the free samples are linked into a graph via short paths in free space [10]. The hardest case is long narrow passages. One strategy for detecting them is local search near contact configurations [11]. Free space construction and probabilistic method can also be combined [12].

Wang, Chiang, and Yap [13] argue that exact predicate evaluation is too slow for free space construction. They approximate free spaces of planar bodies using a bounded-depth recursive axis-parallel subdivision.

The cost of the approximate approaches is proportional to ϵ^{-d} with ϵ the accuracy and d the dimension of the configuration space. This cost is prohibitive in robot path planning with narrow free space passages, in precision assembly planning, in mechanical design, and in part layout.

Free space construction is related to penetration depth computation. The penetration depth of a configuration in the complement of free space is the minimum distance to a configuration in contact space. In other words, it is the smallest motion that transforms a configuration in which the parts overlap to one in which they do not overlap. It appears that the only way to compute the penetration depth is to construct the free space. The complexity of this task motivates research on approximate and heuristic penetration depth computation. Tang and Kim [14] use a heuristic to find a contact configuration near the overlap configuration then locally minimize the distance to the overlap configuration. The error is large when the local minimum is far from the global minimum. Pan and Manocha [15] approximate the contact space with a support vector machine (SVM) that they construct from free and overlap configurations, collected using uniform sampling followed by active learning. Since an SVM defines a smooth surface, sharp features and narrow channel are poorly modeled. Kim, Manocha, and Kim [16] combine this algorithm with local refinement of the contact space. He, Pan, Li, and Manocha [17] approximate the contact space with a graph of configurations that they obtain via random sampling followed by local search around contact samples. They compute penetration depth via nearest-neighbor search and interpolation. The approach is more accurate than prior work, yet narrow passages remain problematic. None of this work provides error bounds.

3. Algorithm

The input to the free space construction algorithm is polyhedra A and B with manifold triangle mesh boundaries. The geometric part of the algorithm constructs the patches (Section 3.1), the intersection curves of two patches (Section 3.2), and the intersection points of three patches (Section 3.3). The combinatorial part of the algorithm constructs the subdivision of the patches and identifies the cells that comprise the free space (Section 3.4).

3.1. Patches

The *configuration* of A is $\mathcal{C} = (u, \theta)$ with $u = (x, y, 0)$ a translation vector and with θ a rotation angle around its z axis. A configuration $\mathcal{C}$ maps a point p to $u+\theta p$ where θp denotes p rotated by θ. The image of A is written as $A(\mathcal{C})$. A *contact* is a configuration $\mathcal{C}$ where (1) a vertex a of $A(\mathcal{C})$ lies on a facet bcd of B, (2) a vertex a of B lies on a facet bcd of $A(\mathcal{C})$, or (3) an edge ab of $A(\mathcal{C})$ shares a point with an edge cd of B. The contact is *compatible* if the interiors of $A(\mathcal{C})$ and B are disjoint in a neighborhood of the shared point (Fig. 6). The compatible contacts form a surface, called a *patch*. Fig. 5 shows

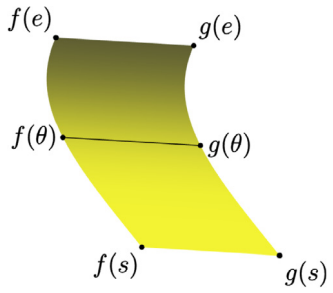

Fig. 5. Patch.

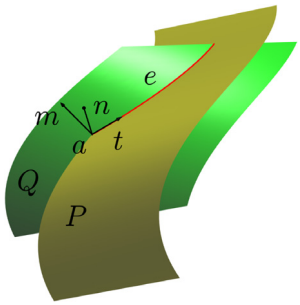

Fig. 7. PP-curve.

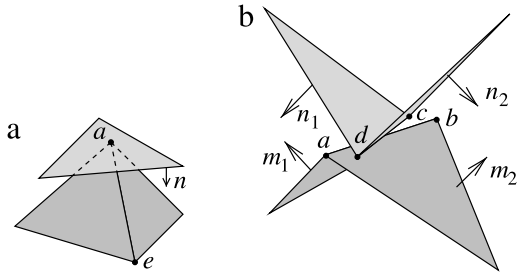

Fig. 6. Compatible vertex/facet (a) and edge/edge (b) contacts.

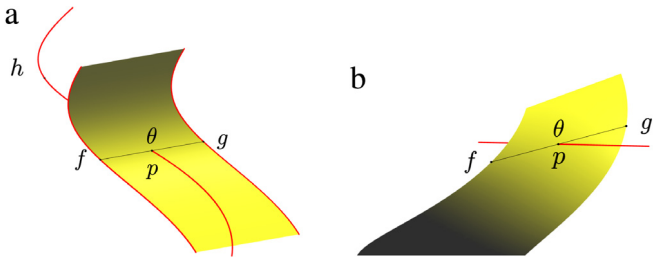

Fig. 8. EP-points of spiral (a) and linear (b) edges.

a typical patch. It is defined on an angle interval $[s, e]$, has spiral edges $f(\theta)$ and $g(\theta)$, has linear edges $[f(s), g(s)]$ and $[g(e), f(e)]$, and is ruled by the line segment $[f(\theta), g(\theta)]$.

The three types of patches lie on algebraic surfaces where a, b, c, and d are coplanar. The surfaces have the form $(a - d) \cdot [(b - d) \times (c - d)] = 0$ with $\cdot$ and $\times$ denoting the inner and outer vector products. The equations are obtained by transforming the vertices of A.

$$(\theta a + u - d) \cdot [(b - d) \times (c - d)] = 0 \quad (1)$$

$$(a - \theta d - u) \cdot [\theta(b - d) \times \theta(c - d)] = 0 \quad (2)$$

$$(\theta a + u - d) \cdot [(\theta b + u - d) \times (c - d)] = 0. \quad (3)$$

The surfaces are ruled by lines parallel to the xy plane. Two spirals delimit the segment of the ruling line where the contact occurs. Outside this segment, a is in contact with the support plane of bcd or the support lines of ab and cd are in contact. On the spirals, a vertex p of A lies on an edge qr of B or vice versa. Let $p' \in qr$ have $p'_z = p_z$. Either $u = p - \theta p'$ or $u = p' - \theta p$. We employ the rational parameterization of rotation

$$\theta p = \left(\frac{(1 - t^2)p_x - 2tp_y}{1 + t^2}, \frac{2tp_x + (1 - t^2)p_y}{1 + t^2}, p_z \right)$$

to express u ($f(\theta)$ or $g(\theta)$) as a rational function of t.

A patch is generated by convex features that are compatible over an interval of rotation angles that we derive by solving compatibility constraints. For a vertex a of A and a facet of B with normal n (Fig. 6a), $n \cdot \theta(e - a) > 0$ for every edge ae. For a vertex of B and a facet of A, $\theta n \cdot (e - a) > 0$. For edges ab of A and cd of B (Fig. 6b), let $v = b - a$ and $w = d - c$, let the incident facets have normals m_1 and n_1, let the facets incident on ba and dc have normals m_2 and n_2, and let s_i and t_i denote the signs of $\theta m_i \cdot w$ and $n_i \cdot \theta v$. The contact is compatible if $s_1 = -s_2 = -t_1 = t_2$.

The compatibility constraints have the form $\theta n \cdot v > 0$ or $n \cdot \theta v > 0$ with n a facet normal of one polyhedron and v an edge tangent of the other. The equations $\theta n \cdot v = 0$ and $n \cdot \theta v = 0$ are quadratic in the rational parameter t. Their roots are *EF-angles* where an edge parallels a facet. The constraints hold on angle intervals bounded by EF-angles. Edge ae of A parallels facet bcd of B if

$$(\theta e - \theta a) \cdot [(c - b) \times (d - b)] = 0.$$

For an EF-angle θ that solves this equation, the *EF-line* is the set of u satisfying Eq. (1). For an edge ae of B and a facet bcd of A, the EF-angle equation is

$$(e - a) \cdot [(\theta c - \theta b) \times (\theta d - \theta b)] = 0$$

and the EF-line is determined by Eq. (2).

The patch normal is n, $-\theta n$, or $s_1 \theta v \times w$. Motion in the normal direction causes A and B to separate in a neighborhood of the contact point, whereas motion in the opposite direction causes them to overlap. The patch boundary traversal $f(s), g(s), g(e), f(e)$ is counterclockwise with respect to the normal.

3.2. Patch intersection curves

The intersection curves of patches P and Q are called *PP-curves* (Fig. 7). There cannot be two PP-curves at a θ value because the patches intersect at the intersection point of the ruling line segments. A PP-curve separates a portion of P that lies on the normal side of Q, where A and B locally separate, from a portion where they intersect. The normal side is to the left of the PP-curve if, at every point a, $t \cdot (n \times m) > 0$ with t the PP-curve tangent, m the normal of P, and n the normal of Q.

The endpoints of a PP-curve, called *EP-points*, are intersection points of an edge of one patch with the other patch. A spiral edge h intersects a patch P with spiral edges f and g (Fig. 8a) if $(h(\theta) - f(\theta)) \times (f(\theta) - g(\theta)) = 0$, which is a quadratic equation in t, θ is in the angle intervals of h and of P, and $h(\theta)$ lies on the line segment $[f(\theta), g(\theta)]$. A linear edge h with EF-angle θ intersects P (Fig. 8b) if it intersects the line segment $[f(\theta), g(\theta)]$. The PP-curves are constructed by computing the EP-points, sorting them by θ to obtain $p_1, \ldots, p_m$, and forming PP-curves for $(p_1, p_2), (p_3, p_4), \ldots, (p_{m-1}, p_m)$.

Fig. 9. PPP-point.

Input: polyhedra A and B.

1. Construct the patches.
2. Subdivide the patches into faces.
3. Group the faces into surfaces and determine those that bound free space cells.

Output: Free space boundary.

Fig. 10. Free space construction algorithm.

3.3. Patch intersection points

An intersection point of three patches is called a *PPP-point* (Fig. 9). The ruling lines $k_i(\theta)x + l_i(\theta)y + m_i(\theta) = 0$ meet at a point, which yields the equation

$$\begin{vmatrix} k_1 & l_1 & m_1 \\ k_2 & l_2 & m_2 \\ k_3 & l_3 & m_3 \end{vmatrix} = 0.$$

We substitute k_i, l_i, m_i from Eqs. (1)–(3) to obtain a quartic in t. We compute its zeros using ACP (Section 4). At each zero, we intersect two of the ruling lines to obtain a candidate PPP-point and test if it lies on the three patches.

3.4. Algorithm

Fig. 10 summarizes the free space construction algorithm. We discuss each step in turn. The discussion is brief because the combinatorial algorithms are standard and are described elsewhere [6].

Step 1 of the algorithm constructs the patches. We enumerate the convex vertex/facet and convex edge/convex edge pairs, compute the compatible intervals, and construct the patch boundaries.

Step 2 subdivides the patches into faces. We enumerate the pairs of patches whose bounding boxes intersect, using an octree, and derive the PP-curves. The bounding box in x and y is determined by the projection of the spiral edges into the xy-plane, which is a pair of circular arcs. Fig. 11a shows an example with PP-curves ab, bc, and de. Each PP-curve defines a PP-edge in the orientation where the normal side of the other patch is to its left (Fig. 7). These edges are marked with arrows in Fig. 11b. The subdivision edges are constructed by splitting the patch edges at the EP-points (e.g. a, c, d, e) and splitting the PP-edges at the PPP-points (e.g. q). The face boundaries are constructed by forming and nesting the loops of subdivision edge. The only loop in our example is $abqef(s)g(s)$.

Step 3 completes the free space construction. We group the subdivision faces into surfaces via a graph traversal. A surface is part of contact space if it bounds a cell of free space. Select a face, a random θ in its angle interval, and a random point on its θ cross-section. Move a random distance into the cell in the direction perpendicular to the cross-section segment, stopping short of an intersection with another patch's cross-section segment. The cell is free if $A(C)$ and B are disjoint at the resulting configuration. We obtain the free space boundary representation by nesting these surfaces via ray casting.

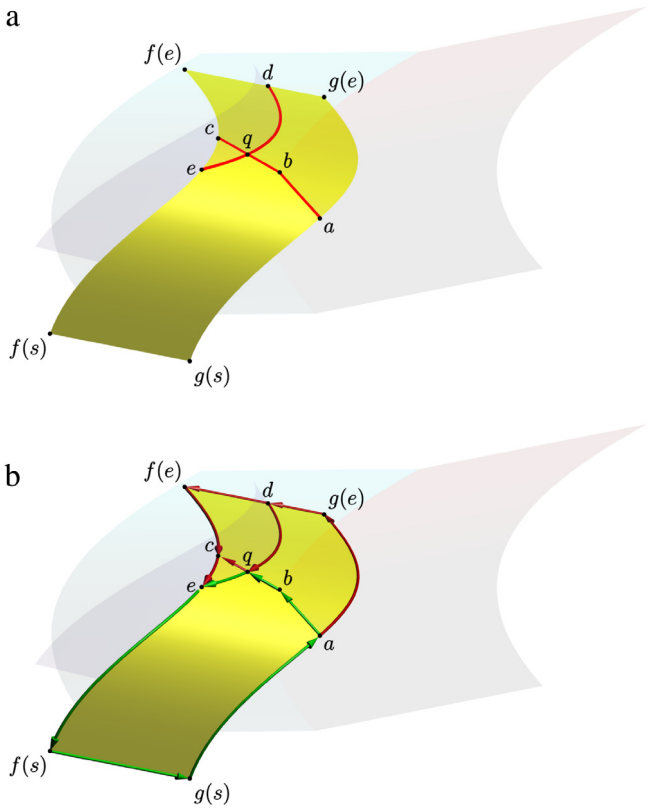

Fig. 11. Patch with PP-curves (a) and edges (b).

4. Robustness

We describe a robust and efficient implementation of the free space construction algorithm. An implementation of a computational geometry algorithm is robust if the output is correct for every input. It is efficient if the running time is consistent with the real-RAM complexity of the algorithm and the constant factor is modest. The challenge is to implement the control logic, which branches on the signs of predicates, correctly and efficiently. A predicate is the sign of a multivariate polynomial that is evaluated at the coordinates of geometric objects. It is *degenerate* if the sign is zero. The sign of a predicate can often be computed using floating point arithmetic and then verified using an error bound. If the verification fails, as is always the case for a degenerate predicate, exact evaluation is required. The CGAL library [18] supports this technique, but exact evaluation of predicates on algebraic numbers is costly. This section explains how we achieve robustness efficiently by avoiding algebraic computation.

4.1. Special position

One way for a predicate to be degenerate is that its arguments are inputs in special position. One example is coplanar facets of B, which generate coplanar patches, which makes the PP-curve predicates degenerate. Another example is a facet that is perpendicular to the z axis. We prevent input in special position with our ACP (Adaptive Controlled Perturbation) technique [4,6], which builds on a perturbation scheme due to Halperin [19].

ACP adds a random relative perturbation of at most $\delta = 10^{-8}$ to each input vertex coordinate. The perturbation is negligible relative to the measurement and manufacturing error in applications. Predicates are evaluated in double-float interval arithmetic, which verifies the sign unless the interval contains zero, in which case ACP switches to extended precision interval arithmetic and increases the precision until zero is excluded. Despite the use of extended precision arithmetic, predicate evaluation takes less than 20% of the running time. We attribute this low overhead to the fact that most degenerate predicates have non-degenerate derivatives, so their perturbed values are δ-far from zero, whereas the interval widths are small multiples of the double-float rounding unit $\mu \approx 10^{-16}$.

4.2. Identities

A second way for a predicate to be degenerate is that its arguments are related. For example, the signed area predicate for points a, b, q is degenerate when the points are collinear. If a, b, q are input points, this cannot occur because of the input perturbation. But if q is the intersection point of line segments ab and cd, it is collinear with a and b by construction. Substituting the definition of q in terms of a, b, c, d into the predicate yields the zero polynomial, so perturbing a, b, c, d does not remove the degeneracy.

We call a predicate that is identical to the zero polynomial an *identity*. We need to detect the identities in the free space construction algorithm and to extend the control logic with zero branches for those predicates.

Identities occur when geometric objects are derived from overlapping sets of vertices of A and B. We assign each object a unique label consisting of its vertex set and, if it is the root of a polynomial, its ordinal index. An EF-angle is labeled with five vertices, two from the edge and three from the facet, plus the ordinal index. A spiral is labeled with three vertices: one from A and two from an edge of B or vice versa (p, q, and r in Section 3.1). Patches, EP-points, PP-curves, and PPP-points are labeled similarly.

We describe the six types of identities and explain how they are detected and resolved. We write the patch generated by a vertex a of A and a facet bcd of B as $a + bcd$, and write $abc + d$ and $ab + cd$ for the other types of patches. Likewise, $a + bc$ and $ab + c$ denote spiral edges.

(1) The EF-angles α and β are equal.
Predicate: Is α less than β?
Detection: The angles α and β have equal labels.
Resolution: Return false.

(2) A spiral edge g overlaps a spiral edge h of patch P.
Predicate: Is the EP-point in the cross-section of P?
Detection: The spirals of g and h have equal labels.
Resolution: Return false.

For example, patches $a + bcd$ and $a + cbe$ with overlapping angle intervals both have edges on the $a + bc$ spiral.

(3) A linear edge g overlaps a linear edge h of a patch P.
Predicate: Is the EP-point in the angle interval of P?
Detection: The EF-angles of g and h have equal labels.
Resolution: Return false.

For example (Fig. 12), the top edge g of $a+cde$ and the bottom edge h of $ab+cd$ are on the EF-line of ab and cde. As A traverses this line, ab translates in the plane of cde from u_1 to u_4. For configurations in u_1u_3, ab contacts cd. For u_2u_4, a contacts cde. Hence, $g = u_1u_3$ and $h = u_2u_4$, which overlap on u_2u_3.

(4) A linear edge g intersects a spiral edge h of a patch P.
Predicate: Is the EP-point in the cross-section of P?
Detection: The label of the spiral of h is a subset of the label of the EF-angle of g.

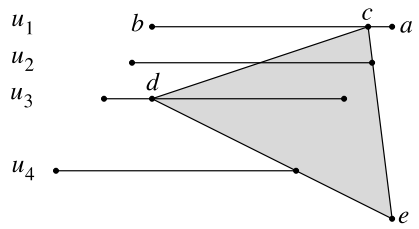

Fig. 12. Identity 3 example.

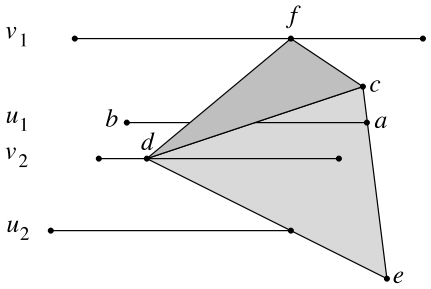

Fig. 13. Identity 4 example.

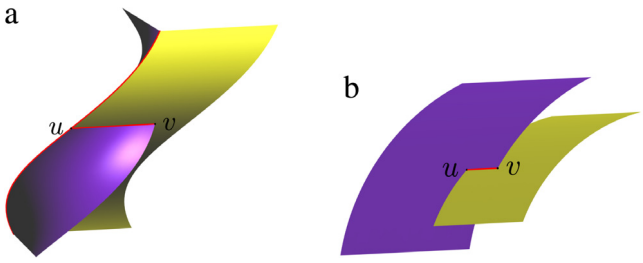

Fig. 14. Linear PP-curves with (a) and without (b) shared spiral.

Resolution: Return false.

For example (Fig. 13), the top edge $g = u_1u_2$ of $a + cde$ is on the EF-line of ab and cde. The cross-section of $ab + df$ at that EF-angle is v_1v_2 and v_2 puts ab in contact with d and plane cde, so v_2 lies on $g = u_1u_2$ and g intersects the spiral edge $h = ab + d$ at v_2. The angle interval of $ab+df$ is not constrained by the $ab+cde$ EF-angle, so v_2 is in the interior of the spiral edge h.

(5) Two patches intersect along an EF-line.
Predicate: Is the PP-curve increasing in θ?
Detection: The union of the labels consists of two vertices from one polyhedron and three vertices from the other.
Resolution: Assign both endpoints the EF-angle of the EF-line, so the PP-curve is linear. If one endpoint lies on a shared spiral edge (Fig. 14), the PP-curve is singular at this point, so we compute the PP-edge orientation at the other endpoint.

For example (Fig. 15), the patches $ab + cd$ and $ab + de$ intersect along the EF-line of ab and cde, which is defined even though cde is not a facet. The patches have a common spiral edge $ab + d$. If ab does not contain d when its support line passes through d, there is no common spiral edge.

(6) A linear PP-curve uv intersects a patch P.
Predicate: The PPP-point construction fails.
Detection: The angles of u and v have equal labels.
Resolution: Intersect uv with P as in Section 3.2 (Fig. 16).

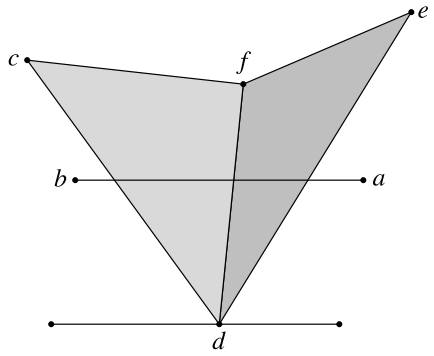

Fig. 15. Identity 5 example.

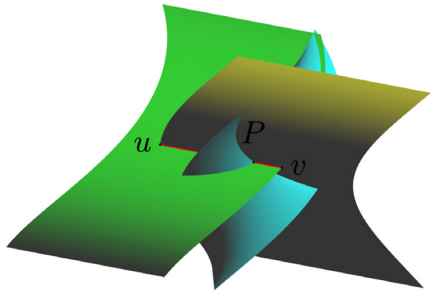

Fig. 16. Identity 6 example.

Table 1
Input complexity: vertices v, edges e, facets f, convex vertices, v_c, and convex edges e_c.

	v	e	f	v_c	e_c
A1	4	6	4	4	6
A2	9	21	14	9	18
A3	9	21	14	8	16
A4	18	48	32	16	36
A5	185	549	366	53	172
A6	187	555	370	70	199
B1	370	1104	736	112	437
B2	1318	3960	2640	838	2079
B3	2316	6942	4628	2036	4701
B4	9367	28101	18734	5886	14836
B5	4036	12102	8068	2154	6132
B6	7251	21765	14510	3728	10061

5. Validation

We validated the free space computation program on six robots A (Fig. 17) and six obstacles B (Fig. 18). Table 1 gives their complexity. The robots have general 3D shapes and the obstacles are 2.5D with fixed xy cross-sections. Since the obstacles are triangulated, they contain coplanar vertical triangles, which is a difficult degenerate case for a general 3D algorithm and challenges our ACP technique.

Table 2 describes the output complexity and the running time of free space computation for the 36 robot/obstacle pairs. The timings are for one core of an Intel 1.7 GHz i5-421U CPU with 4 GB of RAM. The output complexity c is linear in the input complexity, as measured by the combined number of A and B faces. The average and max ratios are 240 and 1080, whereas the worst-case is degree 6 in the input size. The average and maximum ratios of running time to output complexity are 0.003 and 0.018 with no correlation between the ratio and the input or output complexity. The running time is dominated by EP-point computation. On average, 13% of the running time is for computing the roots of polynomials.

We further validated the program by computing maximum clearance paths. The clearance of a free space path between configurations $\mathcal{C}_1$ and $\mathcal{C}_2$ is the minimum distance between A and B over the path. Maximum clearance paths are important in path planning

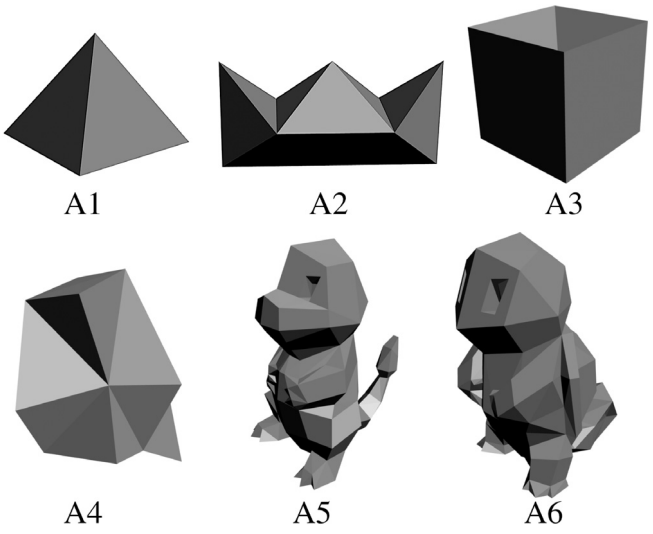

Fig. 17. Test robots.

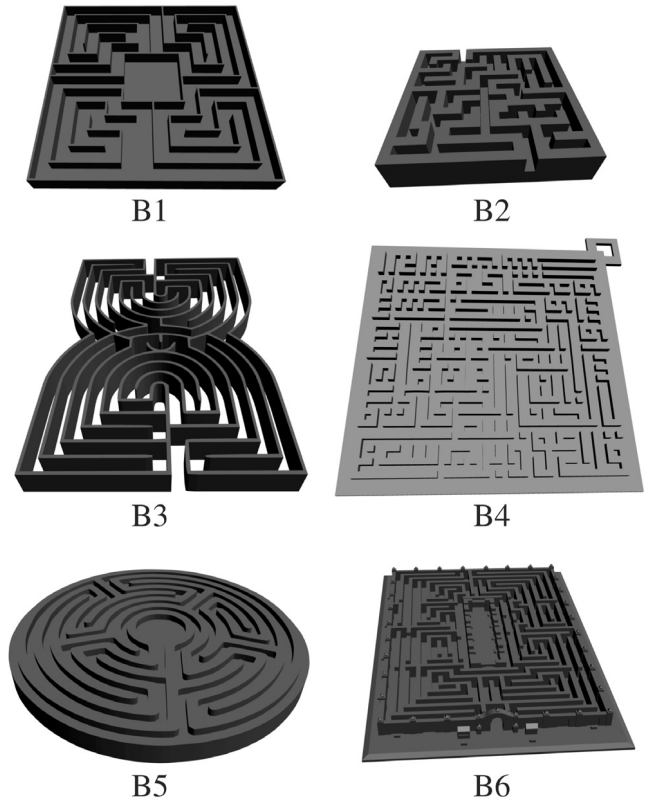

Fig. 18. Test obstacles.

Table 2
Free space construction: contact space vertices, edges, and facets c, EP-vertices ep, PPP-vertices ppp, PP-edges pp, spiral edges sp, and linear edges le, running time t, and polynomial root finding time rt.

		c	ep	ppp	pp	sp	le	t	rt
A1	B1	6894	12034	71	1759	1400	2468	5.44	1.23
A1	B2	362558	215619	310	12869	2640	4600	26.78	3.17
A1	B3	172603	94294	158	36224	3639	6209	21.53	5.67
A1	B4	97470	659476	192	23100	20619	35157	271.36	35.6
A1	B5	286812	867465	516	54597	18420	31194	365.15	47.21
A1	B6	29314	578964	423	48652	5943	10809	420.19	52.36
A2	B1	7584	32034	156	5789	2536	4580	16.27	3.54
A2	B2	762214	548729	549	35286	7216	12486	72.69	18.63
A2	B3	3548789	157896	476	69781	5146	8560	297.47	17.68
A2	B4	97470	659476	192	23100	20619	54846	331.21	35.26
A2	B5	568448	867465	1051	47865	21420	33260	568.17	48.59
A2	B6	61254	845896	672	48652	5943	10809	657.18	22.67
A3	B1	7584	29486	201	52498	3062	4580	19.44	2.74
A3	B2	712348	512346	432	36487	6986	12486	40.98	18.13
A3	B3	3125468	234685	621	65423	5548	9454	334.81	22.51
A3	B4	114566	678495	148	22986	1961	56236	437.28	53.26
A3	B5	632154	836547	654	51236	2840	31284	623.67	53.31
A3	B6	72365	945687	571	46875	6254	9954	768.87	23.65
A4	B1	15236	54236	287	11247	3960	6480	33.67	6.53
A4	B2	544233	752365	678	52031	12684	6864	55.73	10.25
A4	B3	5023146	2412546	595	145697	8950	13950	835.37	30.18
A4	B4	1035698	659476	248	25670	20619	54846	1926.87	49.75
A4	B5	6123548	987456	984	54879	39486	65280	4113.59	61.91
A4	B6	716484	912354	786	65464	7520	15340	6127.74	45.71
A5	B1	15236	54236	287	11247	3960	6480	63.48	7.53
A5	B2	544233	752365	678	52031	12684	6864	130.58	30.25
A5	B3	5023146	2412546	595	145697	8950	13950	1948.57	30.18
A5	B4	1035698	659476	248	25670	20619	54846	4984.11	49.75
A5	B5	6123548	987456	984	54879	39486	65280	7867.85	61.91
A5	B6	716484	912354	786	65464	7520	15340	9439.62	45.71
A6	B1	15236	54236	287	11247	3960	6480	72.36	7.53
A6	B2	544233	752365	678	52031	12684	6864	150.17	31.65
A6	B3	5023146	2412546	595	145697	8950	13950	2687.91	750.84
A6	B4	1035698	659476	248	25670	20619	54846	5781.98	1074.39
A6	B5	6123548	987456	984	54879	39486	65280	9951.84	2367.96
A6	B6	716484	912354	786	65464	7520	15340	12804.36	3191.23

Table 3
Maximum clearance experiments: input and output complexity for first iteration i_f and o_f, and for last iteration i_l and o_l, and time (s) for first and last iterations t_f and t_l.

		i_f	o_f	i_l	o_l	t_f	t_l	o_l/o_f	t_l/t_f
R1	O1	3924	6894	57468	43587	5.44	85.29	6.32	15.68
R1	O2	75798	362558	591265	2320376	26.78	518.05	6.40	19.35
R1	O3	159110	172603	1416079	1018358	21.53	488.32	5.90	22.69
R1	O4	228956	97470	1854543	738935	271.36	5874.944	7.58	21.65
R1	O5	396186	286812	3288343	1976902	365.15	8646.75	6.89	23.68
R1	O6	552384	29314	4916217	395739	420.19	12513.26	13.50	29.78
R2	O1	48830	75841	698269	320807	16.27	282.94	4.23	17.39
R2	O2	217166	762214	3018607	8278068	72.69	1743.11	10.87	23.98
R2	O3	431870	354879	5562053	2851959	297.47	8082.26	8.04	27.17
R2	O4	621452	269274	8924050	3680975	331.21	10360.25	13.67	31.28
R2	O5	1075362	498632	16108922	5861662	568.17	22567.71	11.76	39.72
R2	O6	1499328	845896	20479276	10462887	657.18	28626.76	12.37	43.56
R3	O1	52685	29486	853497	363946	19.44	479.19	12.35	24.65
R3	O2	227837	512346	3941580	6916671	40.98	1140.89	13.50	27.84
R3	O3	465965	234685	8014598	3184644	334.81	11313.23	13.57	33.79
R3	O4	670514	678495	12531906	9218982	437.28	16822.17	13.59	38.47
R3	O5	1160259	836547	23054346	14748323	623.67	26979.94	17.63	43.26
R3	O6	1617696	945687	32143619	16316883	768.87	37759.21	17.25	49.11

and in assembly planning because they are maximally robust to sensing and navigation error. We use the Minkowski sum of A with a sphere of radius r centered at the origin, called an *r-offset*. If the r-offset of A has a path from $\mathcal{C}_1$ to $\mathcal{C}_2$, A has a path with clearance r. We approximate the spheres with polyhedra and compute the r-offsets with our prior program [6]. A path exists when $\mathcal{C}_1$ and $\mathcal{C}_2$ are in the same free space cell. We use bisection search to compute the maximum r for which a path exists.

We computed maximum clearance paths for the robots A1, A2, and A3 and the six obstacles. The initial offset is $r = 1$ and the bisection search ends when the interval width reaches 0.025. Fig. 19 shows the setup for A3 and B2. Table 3 shows the results for the 18 tests. The output complexity ratio between the last and first iterations o_l/o_f is between 4 and 18 with a mean of 11. The running time ratio t_l/t_f is between 15 and 50 with a mean of 30. The ratio of t_l/t_f to o_l/o_f is between 2 and 4 with a mean of 3. We conclude

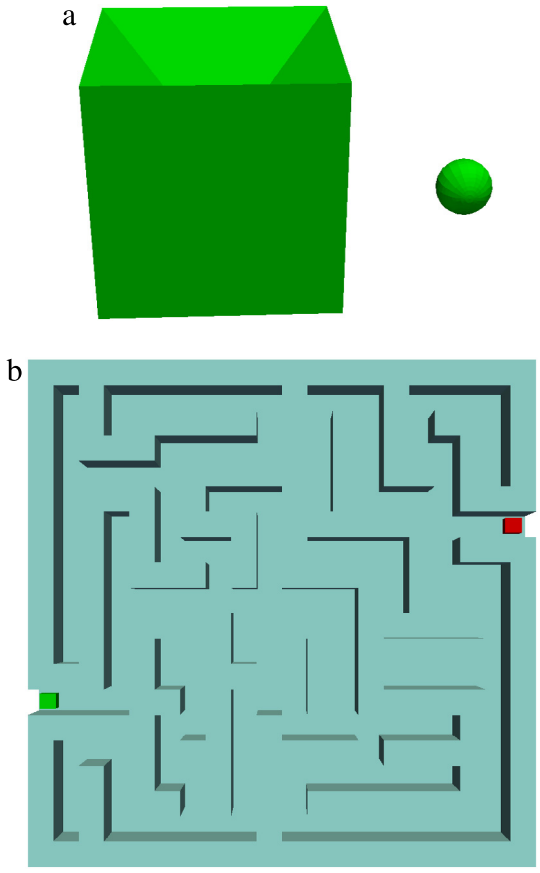

Fig. 19. Blowup of robot A3 and of $r = 1$ sphere (a) and start and end configurations for maximum clearance path in obstacle B2 (b).

that the running time increases three times faster than the output complexity. This is due mainly to an increase in the number of pairs of patches that are tested for intersection.

6. Discussion

We have presented a free space construction algorithm for a polyhedron that moves in a plane relative to a stationary polyhedron. We have demonstrated a robust implementation whose running time is linear in its output size, which in turn is linear in the input size, based on 36 test inputs. In order to detect the same connectivity, a probabilistic approach would have running time ϵ^{-3}, where ϵ is smaller than the clearance. In contrast, our free space construction has only a mild dependence on the clearance.

The algorithm uses the proven approach of generating feature contact patches, constructing their subdivision, and identifying the free cells. Our contribution is the specifics of patch construction and intersection. The robust implementation uses our proven ACP robustness technique to avoid degeneracy due to input in special position. Our contribution is logic for detecting and handling the degenerate predicates that are identically zero due to relations among derived geometric objects.

The bottleneck in our implementation is patch intersection. Although we use an octree to find the patches whose bounding boxes intersect, few of these patches intersect. In contrast, bounding boxes are highly effective for linear patches [6]. A research direction is to enclose patches in tighter containers that are still easy to intersect. The bottleneck in our algorithm is that it generates and discards the subdivision cells (EP-points, PP-curves, PPP-points, etc.) whose boundaries are disjoint from the contact space. A research direction is to generate the contact space incrementally, starting from a known contact configuration, as proved effective in Minkowski sum construction [6].

We have validated the identity detector via extensive testing. If we had missed even a single identity, ACP would have thrown an exception when it failed to resolve the sign of that polynomial. Even though empirical validation of software is the norm, we would prefer a rigorous identity detection scheme. More importantly, we cannot envision enumerating the identities in a four-dimensional configuration space. We are working on an automated alternative.

Acknowledgments

Sacks and Butt are supported by National Science Foundation grant CCF-1524455. Milenkovic is supported by National Science Foundation grant CCF-1526335.

References

[1] Choset H, Lynch KM, Hutchinson S, Kantor G, Burgard W, Kavraki LE, et al Principles of robot motion. MIT Press; 2005.
[2] Milenkovic V, Daniels K. Translational polygon containment and minimal enclosure using mathematical programming. Int Trans Oper Res 1999;6:525–54.
[3] Sacks E, Joskowicz L. The configuration space method for kinematic design of mechanical systems. MIT Press; 2010.
[4] Milenkovic V, Sacks E, Trac S. Robust free space computation for curved planar bodies. IEEE Trans Autom Sci Eng 2013;10(4):875–83.
[5] Fogel E, Halperin D, Wein R. CGAL arrangements and their applications: A step-by-step guide. Springer; 2012.
[6] Kyung M-H, Sacks E, Milenkovic V. Robust polyhedral minkowski sums with gpu implementation. Comput Aided Des 2015;67–68:48–57.
[7] Kim Y-J, Elber G, Kim M-S. Precise continuous contact motion for planar freeform geometric curves. Graph Models 2014;76(5):580–92.
[8] Wentink C, Schwartzkopf O. Motion planning for vacuum cleaner robots. In: Proceedings of the sixth Canadian conference on computational geometry. 1994. p. 51–6.
[9] Canny J. The complexity of robot motion planning. Cambridge, MA: MIT Press; 1988.
[10] LaValle SM. Planning algorithms. Cambridge University Press; 2006.
[11] Yershova A, LaValle SM. Improving motion-planning algorithms by efficient nearest-neighbor searching. IEEE Trans Robot 2007;23(1):151–7.
[12] Zhang L, Kim, YJ, Manocha D. A hybrid approach for complete motion planning, in: IEEE/RSJ international conference on intelligent robots and systems. 2007. p. 7–14.
[13] Wang C, Chiang Y-J, Yap C. On soft predicates in subdivision motion planning. Comput Geom 2015;48(8):589–605.
[14] Tang M, Kim YJ. Interactive generalized penetration depth computation for rigid and articulated models using object norm. ACM Trans Graph 2014;33(1):1:1–1:15.
[15] Pan J, Manocha D. Efficient configuration space construction and optimization for motion planning. Engineering 2015;1(1):046–57.
[16] Kim Y, Manocha D, Kim YJ. Hybrid penetration depth computation using local projection and machine learning. In 2015 IEEE/RSJ international conference on intelligent robots and systems. 2015. p. 4804–9.
[17] He L, Pan J, Li D, Manocha D. Efficient penetration depth computation between rigid models using contact space propagation sampling. IEEE Robot Autom Lett 2016;1(1):10–7.
[18] CGAL, Computational Geometry Algorithms Library, http://www.cgal.org.
[19] Halperin D. Controlled perturbation for certified geometric computing with fixed-precision arithmetic. In: ICMS. 2010. p. 92–5.

Elisha Sacks is a professor in the Department of Computer Science of Purdue University. His research interests are Computational Geometry, Computer Graphics, and Mechanical Design. Butt is a lead principal engineer at Autodesk. He received his Ph.D. from Purdue in 2017 under the supervision of Sacks. Milenkovic is a professor in the Department of Computer Science of the University of Miami. His interests are Computational Geometry, Packing and Nesting, Graphics, and Visualization.

Nabeel Butt

Victor Milenkovic