Robust Complete Path Planning in the Plane

Victor Milenkovic, Elisha Sacks, and Steven Trac

***Abstract*—We present a complete path planning algorithm for a plane robot with three degrees of freedom and a static obstacle. The part boundaries consist of n linear and circular edges. The algorithm constructs and searches a combinatorial representation of the robot free space. Its computational complexity is $\mathrm{O}((n^4+c_3)\log n)$ with $c_3 \in \mathrm{O}(n^6)$ the number of configurations with three simultaneous contacts between robot and obstacle edges. The algorithm is implemented robustly using our adaptive-precision controlled perturbation library. The program is fast and memory efficient, is provably accurate, and handles degenerate input.**

***Index Terms*—Path planning, robust geometry.**

I. Introduction

We present a complete path planning algorithm for a plane robot with three degrees of freedom and a static obstacle. The part boundaries consist of n linear and circular edges. The algorithm constructs and searches a combinatorial representation of the robot free space. Complete path planning has been deemed impractical because the free space complexity is $\mathrm{O}(n^d)$ for an input of size n with d degrees of freedom. However, mild input restrictions reduce the complexity to $\mathrm{O}(n)$ [1]. Our algorithm is practical for this class of inputs because it is sensitive to the reduced complexity.

Complete path planning has also been deemed impractical because it employs computational geometry algorithms that are hard to implement robustly, meaning accurately and efficiently. We implement our algorithm using our adaptive-precision controlled perturbation robustness library. The program is fast and memory efficient, is provably accurate, and handles degenerate input.

Complete path planning solves the narrow passage problem of sample-based planning. Sample-based planning algorithms [2] build and search a graph whose vertices and edges are points and line segments in free space. As the sample size grows, the probability of finding a path converges to one. The outstanding problem is *narrow passages* where the robot clearance, ε, is small. The sample size that ensures a fixed probability of finding a path is $\Omega(\varepsilon^{-d})$ for a robot with d degrees of freedom. Planning experiments confirm that narrow passages require large sample sizes in practice. Our planner is correct for any ε. We demonstrate that it is fast for $\varepsilon = 10^{-8}$, a value that far exceeds application requirements.

Free space construction supports mechanical design [3] and part layout [4] algorithms by characterizing the space of potential robot configurations, whereas sample-based algorithms cannot provide this information.

Victor Milenkovic is at the University of Miami, vjm@miami.edu
Elisha Sacks is at Purdue University, elisha.sacks@gmail.com
Steven Trac is at Epic Systems Corporation, Madison, WI, strac@epic.com

II. Prior Work

Avnaim *et al* [5] present a free space construction algorithm for polygonal parts. Our algorithm has the same complexity as theirs, yet handles circular edges, which increases the algebraic degree of the free space and complicates its combinatorial structure. Circular edges enable one to model curved parts to a given accuracy with many fewer edges and permit one to work with level sets and rotational sweeps without approximation. Sacks [6] presents a precursor to our algorithm that computes type 3 criticalities (Sec. III) approximately, is not output sensitive, and is not robust. Stappen *et al* [7] develop efficient path planning algorithms for a translating robot with mild input restrictions. Sacks [3] computes the free space of two curved parts, in 2D or 3D, each of which rotates around or translates along a fixed axis.

Complete path planning has been implemented robustly via exact computation (Sec. VI) for translating polygons [8], translating polyhedra [9], and polygons with translation along an axis and rotation [10].

There is extensive research on sample-based planning with narrow passages. The approach closest to ours is a hybrid algorithm [11] that approximates the free space with an octree comprised of free, blocked, and mixed cells, builds a graph of free configurations for each mixed cell, and links the graphs into a global approximation of free space. In practice, this method is restricted to $d = 3$ because its computational complexity is r^{-d} with r the octree resolution.

III. Overview

The configuration space is $C = \Re^2 \times S$. Let $\theta M + a$ denote a robot, M, rotated by angle θ around the origin then translated by a. The free space of M with respect to an obstacle, F, is $\{(a,\theta) \in C | (\theta M + a) \cap F = \emptyset\}$. Define $-M = \{-m | m \in M\}$. When the robot translates at a fixed θ value, its free space is defined by the convex convolution [12] of $\theta(-M)$ and F. The convolution is the set of points, t, such that $\theta M + t$ has a local contact with F. It subdivides the plane into open regions: some regions comprise the free space and the others comprise the blocked space. Fig. 1a depicts an oval robot inside a dome-shaped room. The obstacle is the complement of the room. The convolution edges appear above. For $t \in c \oplus -a$, denoted $c - a$, $a + t$ contacts c, and similarly for b and d. The free space is the circular segment bounded by subsets of $c - a$ and $d - b$.

The subdivision is a smooth function of θ, except at a discrete set of critical angles where its structure changes. There are three types of criticality (Sec. IV), a change in: 1) the set of convolution edges, 2) the set of intersections among edges, or 3) the order of intersections on each edge.

Fig. 1b illustrates a type 1 criticality at which there exists t such that $a + t$ and d can be tangent at an endpoint (although

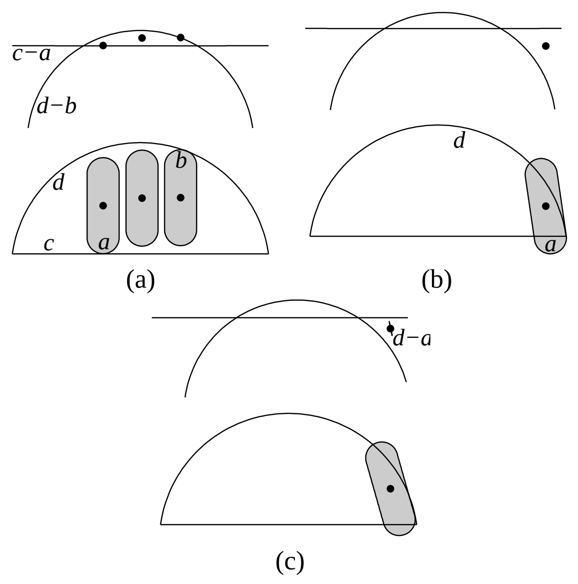


Fig. 1. Type 1 criticality: before (a), at critical angle (b), after (c).

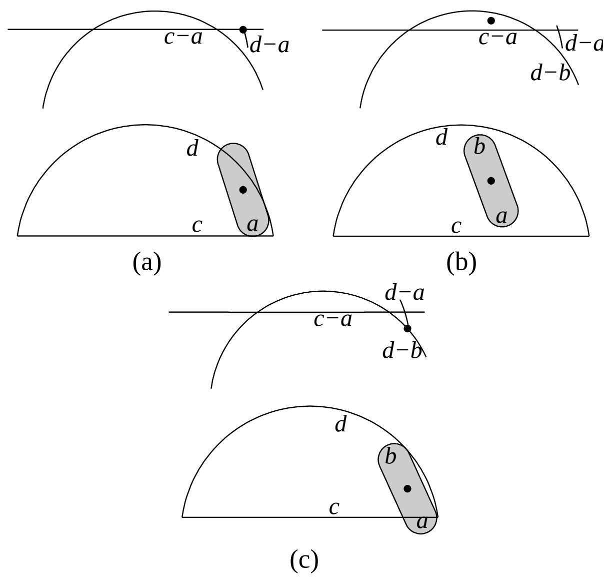


Fig. 2. Type 2 criticalities: at critical angle (a), between (b), at critical angle (c).

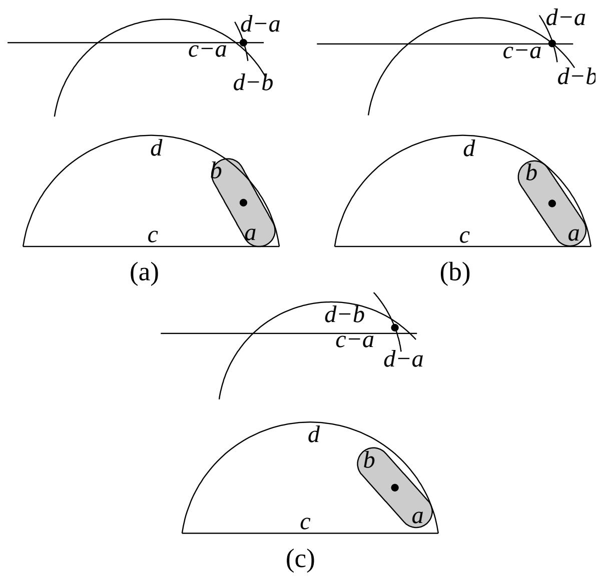


Fig. 3. Type 3 criticality: before (a), at critical angle (b), after (c).

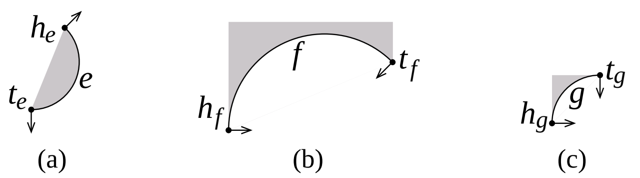


Fig. 4. Circular edges: convex e (a), concave f (b), concave $g = e \oplus f$ (c).

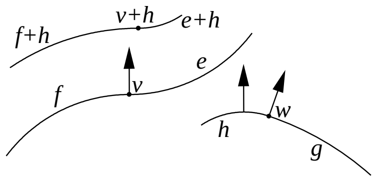
t is in blocked space). Beyond this criticality, the edge $d - a$ appears (Fig. 1c). Figs. 2a and 2c depict the type 2 criticalities at which $d - a$ first intersects $c - a$ and $d - a$ first intersects $d - b$. Fig. 3b depicts a type 3 criticality in which $c - a$, $d - a$, and $d - b$ are coincident and a triple contact is possible. The triangle formed by $c - a$, $d - a$, and $d - b$ flips (Fig. 3a and Fig. 3c), and the boundary of the free space now contains a subset of $d - a$.

By sweeping a plane of constant θ, we compute a vertical decomposition of configuration space along the θ axis and extract the subset that comprises the free space (Sec. V and Fig. 12b). We describe the robust implementation in Sec. VI and validate it in Sec. VII.

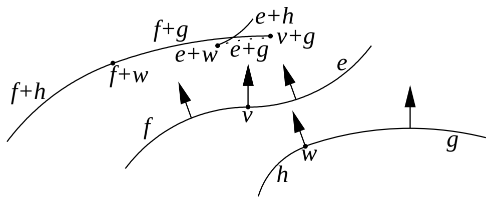
IV. Criticality Computation

A part is a plane region with a boundary comprised of vertices and edges. A vertex, v, is a point, p_v, and an outward normal, n_v. An edge, e, is an open line segment or circular arc with tail vertex $t = t_e$ and head vertex $h = h_e$ such that $n_t \times n_h \geq 0$ (Fig. 4). A linear edge has normal $n_e = n_t$ $(= n_h)$, normal interval $[n_e, n_e]$, and curvature $s_e = 0$. A circular edge has normal interval (n_t, n_h), angular extent at most π, center m_e, signed radius r_e, and curvature $s_e = 1/r_e$. It is convex if $r_e > 0$ and is concave otherwise. The part interior lies to the left when a linear or convex edge is traversed from t to h, or when a concave edge is traversed from h to t.

The convex convolution [12] of F and $\theta(-M)$ consists of sum vertices and sum edges. A sum vertex, $w = v \oplus e$, is the sum of a vertex, v, on one part and a point, a, on an edge, e, of the other part such that n_v equals the normal of e at a. If $v \in F$, $\theta e \in \theta(-M)$ and $w = v \oplus \theta e$, then $p_w = p_v + \theta m_e + r_e n_v$ and $n_w = n_v$. If $\theta v \in \theta(-M)$, $e \in F$ and $w = \theta v \oplus e$, then $p_w = \theta p_v + m_e + r_e \theta n_v$ and $n_w = \theta n_v$. A sum edge, $g = e \oplus \theta f$, is the sum of edges $e \in F$ and $\theta f \in \theta(-M)$ with $s_e + s_f > 0$ whose normal intervals intersect. The curvature condition implies that F and $\theta M + t$ are in contact without local overlap for every $t \in g$, which is necessary for g to contribute to the free space boundary. Edge g is the set of sums, $p + q$, of points $p \in e$ and $q \in \theta f$ with equal normals. If e and f are circular, g is circular with $m_g = m_e + \theta m_f$, $r_g = r_e + r_f$, $t_g = t_e \oplus \theta f$ or $t_g = \theta t_f \oplus e$, and $h_g = h_e \oplus \theta f$ or $h_g = \theta h_f \oplus e$ (Fig. 4). If one edge is linear, the other is circular by the convexity condition, so g is the offset of the linear edge by the circle. If e is circular, $n_g = \theta n_f$, $t_g = \theta t_f + m_e + r_e \theta n_f$ and $h_g = \theta h_f + m_e + r_e \theta n_f$; otherwise, $n_g = n_e$, $t_g = t_e + \theta m_f + r_f n_e$ and $h_g = h_e + \theta m_f + r_f n_e$.



(a)



(b)

Fig. 5. Type 1 criticality: $\theta < 0$ (a), $\theta > 0$ (b).

Let edges e and f in counterclockwise order around the boundary of one part meet at vertices v and w, so $p_v = p_w$. If $n_v \times n_w > 0$, p_v forms sum edges with the compatible edges of the other part. These sum edges can be derived as above by introducing an artificial circular edge with tail v, head w, and radius zero [13].

A. Type 1 Criticality

A type 1 criticality occurs when vertices $v \in F$ and $\theta w \in \theta(-M)$ have equal normals. This criticality, denoted (v, w), occurs at $\theta = n_v/n_w$. Here and throughout the paper, angles are equated with unit vectors and division denotes the complex quotient, so n_v/n_w is n_v rotated clockwise by the angle of n_w. The sum vertex $v \oplus \theta f$ with $f \in -M$ enters and exits the convolution at the (v, h_f) and (v, t_f) criticalities; $\theta w \oplus e$ with $e \in F$ enters and exits at the (t_e, w) and (h_e, w) criticalities. If $e \in F$ is circular and $f \in -M$ is linear, the sum edge $g = e \oplus \theta f$ enters and exits at the (t_e, n_f) and (h_e, n_f) criticalities (we use n_f instead of t_f or h_f because they have equal normals). If e is linear and f is circular, g enters and exits at the (n_e, h_f) and (n_e, t_f) criticalities. If both are circular, g enters and exits at the (t_e, h_f) and (h_e, t_f) criticalities. At the (t_e, t_f) criticality, t_g switches from $t_e \oplus \theta f$ to $\theta t_f \oplus e$. At the (h_e, h_f) criticality, h_g switches from $\theta h_f \oplus e$ to $h_e \oplus \theta f$.

Fig. 5 shows edges $e, f \in F$ that meet at v and $g, h \in \theta(-M)$ that meet at w. The (v, w) critical angle is $\theta = 0$, sum vertex $v \oplus h$ exits, sum vertices $v \oplus g$, $w \oplus e$, and $w \oplus f$ enter, and sum edge $f \oplus g$ enters. There is no $e \oplus g$ edge (drawn dashed) because $s_e + s_g < 0$. The normal vector out of v is vertical. The point on g with a parallel normal vector sums with v to form $v \oplus g$. Similarly, for $w \oplus e$, *etc.*

B. Type 2 Criticality

A type 2 criticality occurs when two sum edges are tangent (Fig. 6) or when a sum vertex hits a sum edge (Fig. 7). At a tangency, two vertices enter or exit the subdivision. At a hit, each incident edge gains or loses a vertex. A candidate tangency occurs when the lines or circles of two edges are

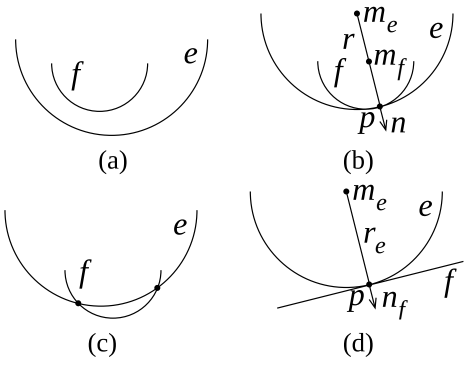


Fig. 6. Circle/circle (a–c) and circle/line (d) tangencies.

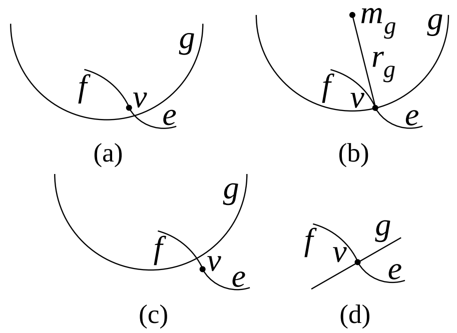


Fig. 7. Circle/circle (a–c) and circle/line (d) hits.

tangent. It is a criticality when the point of tangency lies on both edges. Likewise for hits.

The tangency equations for circular edges e and f with $|r_e| > |r_f|$ (Fig. 6b) are $||m_f - m_e|| = r$ with $r = |r_e| \pm |r_f|$. Let $m_e = a + \theta b$, $m_f = c + \theta d$, $u = c - a$, and $v = d - b$. Let b_i be the intersection points of the circle with center $(0, 0)$ and radius r, and the circle with center u and radius $||v||$. The e normals at the tangent points are $n_i = b_i/r_e$. The f normals for $r = |r_e| \pm |r_f|$ are $\mp\text{sign}(r_e r_f) n_i$. The critical angles are $\theta_i = n_i/(v/||v||)$. The critical points are $p_i = a + \theta_i b + r_e n_i$.

If f is linear, the distance from m_e to the f line equals $|r_e|$ (Fig. 6d). Define a linear trigonometric expression (LTE) as $k_1 \sin\theta + k_2 \cos\theta + k_3$ with the k_i constants. The tangency equations are LTE's: $n_f \cdot m_e + d = \pm r_e$ or $\theta n_f \cdot m_e + d = \pm r_e$ with $m_e = \theta_a + b$ and with d an LTE. We can solve for θ then compute p as before. A tangency between linear edges is degenerate.

The hit equations for sum vertex v and circular edge g are $||m_g - p_v|| = |r_g|$ (Fig. 7b). If g is linear, the equations are $n_g \cdot p_v + d = 0$ or $\theta n_g \cdot p_v + d = 0$ with d an LTE (Fig. 7d). The solutions are the same as for the tangency equations.

C. Type 1 Criticality with Intersection

A type 1 criticality that coincides with an edge hit is not counted as a type 2 criticality. Suppose edges e and f share vertex v on F and edges g and h share vertex w on $-M$, with e preceding f in a traversal with the inside on the left and similarly g and h. Type 1 events involving the normals at v and w can either add or remove an intersection. We will start with the case that e, f and g, h are arcs meeting smoothly at v and w. Next we will consider if some of e, f, g, h

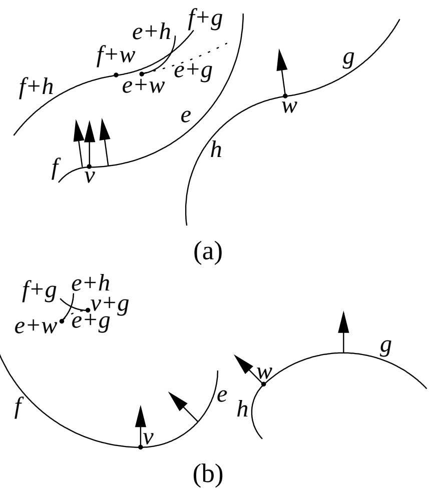


Fig. 8. Simultaneous type 1 and 2: Case 2 (a), Case 3 (b).

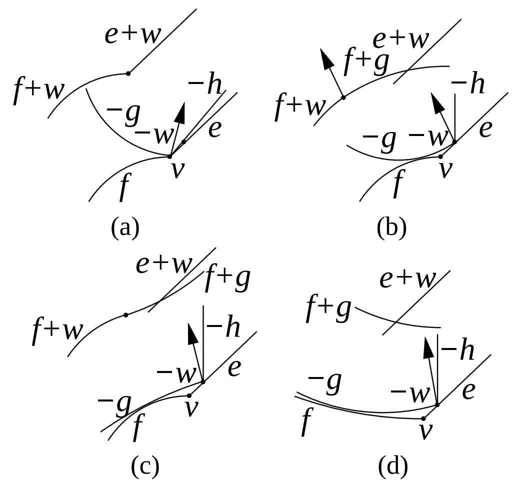


Fig. 9. Concave/convex vertex before (a) and after (b) a type 1 event with $r_f > 0$ and $r_g > 0$. After event with $r_f > 0$ and $r_g < 0$ (c) and $r_f < 0$ and $r_g > 0$ (d).

are line segments. After that, we will address non-smooth v and/or w. Finally, we will show that these are the only ways a type 1 event can introduce an intersection up to the following symmetries: a) exchange F and $-M$ hence e with g and f with h and b) take a mirror image and exchange e with f and g with h.

1) Smoothly joined arcs.: For two pairs of edges meeting smoothly, there are three cases based on the signed radii of the incident edges. Let edges $e, f \in F$ meet smoothly at v with outward normal n_v and $g, h \in -M$ meet smoothly at w with outward normal n_w. The analysis assumes the critical angle is $\theta = 0$ and hence $n_v \times n_w > 0$ after the type 1 event.

Case 1 is $r_e < 0$, $r_f, r_g, r_h > 0$, $r_e + r_g > 0$, $r_e + r_h < 0$ (Fig. 5b). For the coordinate system in which n_v is the y axis, edge $f \oplus g$ is above $e \oplus g$ (drawn dashed) because they are tangent and $r_f + r_g > r_g > r_e + r_g$. Since $r_e + r_g > 0$, $e \oplus g$ is concave downward and hence $e \oplus w$ with normal n_w is to the left of $v \oplus g$ with normal n_v. Edge $e \oplus h$ is above $e \oplus g$ because it is tangent to $e \oplus g$ at $e \oplus w$ and it is concave upward ($r_e + r_h < 0$). Hence, $f \oplus g$ intersects $e \oplus h$ for all sufficiently small $\theta > 0$.

Case 2 is $r_e, r_g < 0$, $r_f, r_h > 0$, $r_f + r_g < r_e + r_h < 0$ (Fig. 8a). For the coordinate system in which n_w is the y axis, $f \oplus w$ and $e \oplus w$ are the highest and lowest points of f and e, respectively, added to w. Hence, $e \oplus w$ is below and to the right of $f \oplus w$. But $f \oplus w$ is also a lowest point of $f \oplus g$, hence $e \oplus w$ is below $f \oplus g$. For all sufficiently small $\theta > 0$ hence $e \oplus w$ sufficiently close to $f \oplus w$, $e \oplus h$ intersects $f \oplus g$ because $|r_e + r_h| < |r_f + r_g|$.

Case 3, $r_e, r_f < 0$, $r_g, r_h > 0$, $r_e + r_h < 0$, $r_e + r_g > 0$, $r_f + r_g < 0$ (Fig. 8b), is similar to Case 1. Edges $f \oplus g$ and $e \oplus h$ are both externally tangent to $e \oplus g$ (dashed), hence they intersect for all sufficiently small $\theta > 0$.

2) Line segment edges.: For the purposes of detecting intersections, we can treat line segments as arcs with $r = -\infty$. Case 1 cannot have an intersection if e is a line because that implies $r_e + r_h < 0$. Case 2 has an intersection if g is a line segment and $r_e + r_h < 0$. Case 3 has an intersection if f is a line segment and the other inequalities hold.

3) Non-smooth vertices.: Non-smooth vertices are treated as zero radius arcs whose "sign" is positive or negative if the vertex is convex or concave. Under this interpretation, Case 1 and Case 2 have an intersection if f or h are convex zero radius arcs. Non-smooth vertices also introduce three new ways in which a type 1 event can introduce or remove an intersection. A convex vertex and concave vertex can cause an intersection only if the convex angle is smaller than the concave angle. In Fig. 9a, M can fit into F because the convex angle between g and h at w is smaller than the concave angle between e and f at v. Vector n_g (shown) is the outward normal to g at w, which is to the right of the vertical normal n_f to f at v (not shown). Locally, the convolution is a translated copy of e, f. In this case $e \oplus w$ and $f \oplus w$ are edges, not vertices, because w is a (zero-radius) arc with an interval of normal angles.

As n_g sweeps past vertical in Fig. 9b, edge $f \oplus g$ appears, intersecting $e \oplus w$. This is clear because M can be put into two-point contact with F, as shown. This happens if f and g are convexly compatible (Fig. 9b-d). The radii r_e and r_h do not matter. Here e and h are shown as line segments, but even if they are arcs, they approximate line segments close to v and w.

4) Completeness: We need to show that these are the only ways intersection can occur. Starting with the smooth case, we observe that only $e \oplus h$ and $f \oplus g$ can intersect in a neighborhood of the type 1 event. The pair $e \oplus g$ and $e \oplus h$ cannot intersect because their circles are tangent at their common endpoint, and similarly $f \oplus g$ and $f \oplus h$. An intersection corresponds to a double contact between F and M. Since $-g$ is to the left of $-h$ in M, $-g$ cannot contact e if $-h$ is contacting f, hence $e \oplus g$ cannot intersect $f \oplus h$.

If e, f, g, h are all convex, then $e \oplus h$ and $f \oplus g$ are joined smoothly by $e \oplus g$ or $f \oplus h$ and hence do not intersect. Otherwise, we cannot have $r_e, r_h < 0$ or $r_f, r_g < 0$ because both e/h and f/g must be convexly compatible. Apply the symmetries to make $r_e < 0$ hence $r_h > 0$. The three cases are therefore $r_f, r_g > 0$ (1), $r_f > 0$ and $r_g < 0$ (2), or $r_f < 0$ and $r_g > 0$ (3). For Case 2, if $n_v \times n_w < 0$, exchange F

and $-M$. For Case 3, if $n_v \times n_w < 0$, take the mirror image. Aside from the requirement that e/h and f/g be compatible, Case 1 requires $n_v \times n_w > 0$ and $r_e + r_g > 0$, Case 2 requires that $r_f + r_g < r_e + r_h$, and Case 3 requires that $r_e + r_g > 0$. We need to show that these are necessary for intersection.

Case 1 (Fig. 5): As Fig. 5a shows, $f \oplus g$ does not exist if $n_v \times n_w < 0$. If $n_v \times n_w > 0$ but $r_e + r_g < 0$, then $f \oplus g$ and $e \oplus h$ smoothly join $e \oplus g$ at its endpoints, hence do not intersect.

Case 2 (Fig. 8a): If $r_e + r_h < r_f + r_g$ (< 0), then $f \oplus g$ does not intersect $e \oplus h$ because $f \oplus g$ starts below and to the right of the minimum of $e \oplus h$ and has a larger magnitude radius.

Case 3 (Fig. 8b): As in Case 1, if $r_e + r_g < 0$, then $e \oplus g$ is a sum edge and smoothly connects $e \oplus h$ and $f \oplus g$, which therefore cannot intersect.

Now consider the non-smooth case. If the convex angle is smaller than the concave angle yet M cannot be seated inside F with $-w$ in contact with v (Fig. 9a), then the situation must be as shown in Figs. 9b-d or their mirror images. The double contacts hence intersections do not occur if v and w are both convex: if e has downward slope, $-w$ cannot contact e in a vicinity of v. Finally, if both v and w are concave, then $-w$ cannot come into contact with v.

D. Type 3 Criticality

A type 3 criticality occurs when three edges intersect at a point. For circular edges e, f, and g (Fig. 10a), let $\phi_{ef} = \angle m_e p m_f$ and $d_{ef} = m_f - m_e = u + \theta v$. By the law of cosines and the identity $(\theta u) \cdot (\theta v) = u \cdot v$,

$$\cos\phi_{ef} = \frac{r_e^2 + r_f^2 - d_{ef} \cdot d_{ef}}{2 r_e r_f}, \quad (1)$$

$$= \frac{r_e^2 + r_f^2 - u \cdot u - v \cdot v - 2u \cdot \theta v}{2 r_e r_f}. \quad (2)$$

Define ϕ_{fg} and ϕ_{ge} likewise. The equation

$$\cos^2\phi_{ef} + \cos^2\phi_{fg} + \cos^2\phi_{ge} - 2\cos\phi_{ef}\cos\phi_{fg}\cos\phi_{ge} = 1$$

follows from $\phi_{ef} + \phi_{fg} + \phi_{ge} = 2\pi$, which implies $\cos\phi_{ef} = \cos(\phi_{fg} + \phi_{ge})$, and from the identity $\cos(x+y) = \cos x \cos y - \sin x \sin y$. Replace $\cos\phi_{ef}$ with the rightmost expression in Eq. 2 and replace $\cos\phi_{fg}$ and $\cos\phi_{ge}$ likewise to obtain the criticality equation, a cubic in $\cos\theta$ and $\sin\theta$.

If g is linear (Fig. 10b), let $\phi_{eg} = \angle p m_e q$ with q the projection of m_e onto g. Since $\cos\phi_{eg} = d/r_e$ with $d = n_g \cdot (m_e - t_g)$ the distance from m_e to g, it is an LTE (Sec. IV-B). Define ϕ_{fg} likewise, define ϕ_{ef} as before, and use $\phi_{eg} + \phi_{fg} = \phi_{ef}$ to obtain a cubic. If f and g are linear (Fig. 10c), let ϕ_{ef} and ϕ_{eg} be the angles between pm_e and the projections onto f and g, let ϕ_{fg} be the angle between f and g, hence $\cos\phi_{fg} = n_f \times n_g$, and use $\phi_{ef} + \phi_{eg} = \phi_{fg}$ to obtain a cubic. If e, f, and g are linear (Fig. 10d), their line equations are $n_e \cdot (x, y) + d_e = 0$ with n_e and d_e LTE's and likewise for f and g. Because $(\theta u) \times (\theta v) = u \times v$, the criticality equation,

$$d_e(n_f \times n_g) + d_f(n_g \times n_e) + d_g(n_e \times n_f) = 0,$$

is quadratic in $\sin\theta$ and $\cos\theta$.

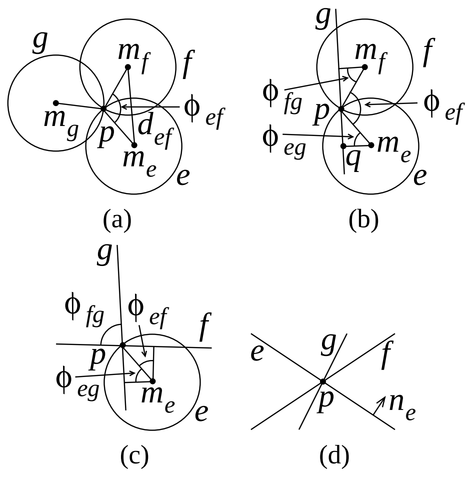


Fig. 10. Type 3 criticality involving 3 (a), 2 (b), 1 (c), and 0 (d) circular edges.

Given θ, the edge intersection point, p, is the solution of two linear equations: $e - f = 0$ and $e - g = 0$ for three circles, $e - f = 0$ and $g = 0$ for two circles and a line, and $f = 0$ and $g = 0$ otherwise. A criticality occurs if p lies on the three edges.

V. Free Space Construction

The free space construction algorithm consists of two parts. Part 1 is a plane sweep that computes the sum edges and their intersection points as θ increases from 0 to 2π (Sec. 1.5.1). Part 2 computes the free space boundary (Sec. 1.5.3). Part 1 dominates the computational complexity and the actual running time. We prove that it is output sensitive (Sec. 1.5.2).

A. Plane Sweep

Step 1 of the first part calculates the convolution edges and their intersections for $\theta = 0$. Each edge has a list of its endpoints and its intersections with other edges, ordered from tail to head. Each intersection is an ordered pair, (e, f), of sum edges, denoting e crosses f from left to right, to distinguish it from the (f, e) intersection. Step 2 initializes a priority queue with all the type 1 and type 2 criticalities, and with the type 3 criticalities of the initial candidate triangles. A candidate is three sum edges each of whose lists contains adjacent intersection points with the other two. Step 3 handles the criticalities in increasing θ order. The output is the the initial set of lists plus the edit that occurs at each criticality.

A type 1 criticality is handled by adding or removing vertices and edges, and by updating intersection lists. When a linear edge is added or removed, its intersection points with the other edges at the critical angle are added to or removed from the appropriate lists. When a circular edge with a coincident hit is added (Figs. 5b, 8a, 8b), a point is appended to one end of its list. A type 2 criticality is handled by updating the two intersection lists. For a tangency, two points are inserted or removed in each intersection list. For a hit, a point is appended to one end of the list or is removed. A type 3 criticality is handled by swapping three pairs of incident points on the lists of the three sum edges of the triangle. After each update, newly created candidate triangles are checked for type 3 criticalities which are added to the priority queue.

B. Analysis

The sweep algorithm is correct if every change in the structure of the subdivision is one of the three criticalities. This condition holds when every criticality equation has simple roots and every criticality occurs at a unique θ value. Correctness follows by verifying the criticality handling.

There are $m \in \mathrm{O}(n^2)$ sum vertices and sum edges with n the input size. The complexity of the $\theta = 0$ slice is $c_0 \in \mathrm{O}(m^2)$. There are $c_i \in \mathrm{O}(n^{2i})$ type i criticalities. Hence, the complexity of the output is $N \in \mathrm{O}(c_0 + c_1 m + c_2 + c_3)$ because each criticality adds $\mathrm{O}(1)$ complexity, except for type 1 criticalities that add or remove a line and hence add $\mathrm{O}(m)$ complexity. The number of candidate triangles is $\mathrm{O}(N)$ because each newly adjacent pair of intersections in a list defines a single candidate.

We can perform step 1 in $\mathrm{O}(c_0 \log n)$ time with a sweep algorithm. Computing the type 1 criticalities takes $\mathrm{O}(c_1)$ time. Computing the type 2 criticalities takes $\mathrm{O}(n^4)$ time, so step 2 takes $\mathrm{O}(n^4 \log n)$ time. Step 3 takes $\mathrm{O}(N \log n)$ time using binary search on edge lists to handle criticalities. Since $N \in \mathrm{O}(n^4 + c_3)$, the algorithm running time is $\mathrm{O}((n^4 + c_3) \log n)$. The algorithm is output sensitive in that the dominant cost is proportional to c_3, which is the number of configurations with simultaneous convex contacts between three pairs of moving/fixed part edges.

C. Faces, Shells, Cells, and Path Planning

We define a *subedge* (edge in the translational subdivision) to be an adjacent pair of elements (in a particular order) in the list of a sum edge. This pair is adjacent for a θ interval. The subedge plus its interval corresponds to a *face* in configuration space. For example, in Fig. 3a, sum edge $d - a$ ($d \oplus \theta(-a)$) is split into three subedges. The top subedge was created at the Fig. 2a criticality and the others at the Fig. 2c criticality. All three subedges end at the Fig. 3b criticality where the intersection points on $d - a$ swap.

Two faces are *horizontal neighbors* if they share a sum edge endpoint or if they share an edge intersection point and are on the sides of the outward normals of the intersecting edges. Hence, in Fig. 1, the middle subedges of $c - a$ and $d - b$ are neighbors because the normals point upward from $c - a$ and downward from $d - b$. In Fig. 3a, the intersection of $c - a$ and $d - a$ is in blocked space, but the subedges to the left and above the intersection are neighbors because the outward normal to $d - a$ points to the left.

Two faces are *vertical neighbors* if the same criticality ends one and starts the other, they belong to the same subedge, and they share an endpoint. Hence the lower subedge of $d - a$ in Fig. 2b is a vertical neighbor of the trivial subedge $d - a$ in 1c because they share the tail of $d - a$. The former is a neighbor of the middle subedge of $d - a$ in 3a because they share the intersection of $c - a$ and $d - a$ as an upper endpoint. Two linear subedges can also be neighbors if they are subsets of *neighboring sum edges* (defined below) and they

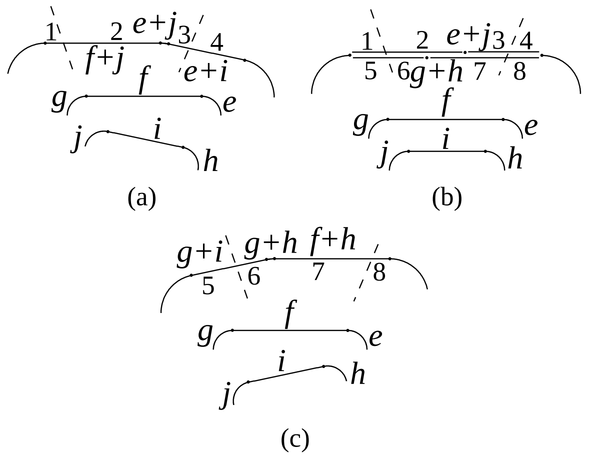


Fig. 11. Sums of linear edges before (a), at (b), and after (c) a criticality.

share a *corresponding list element* at the same endpoint or if one contains an endpoint of the other at the criticality. List elements of linear edges e and f correspond if they are both intersections with the same third sum edge g or if they are a *corresponding endpoint* (defined below).

If the same type 1 event (u, v) ends one linear sum edge and starts another, and either u or v is *not* the endpoint of a linear edge, then the two sum edges are neighbors and their tail and heads correspond. Otherwise, the situation is as depicted in Fig. 11. Before the criticality, linear sum edges $e + i$ and $f + j$ are joined by circular sum edge $e + j$. After, circular $g + h$ joins linear $f + h$ and $g + i$. Sum edges $e + j$ and $g + h$ shrink to points at the criticality. Linear sum edges $f + j$ and $e + i$ are neighbors of $g + i$ and $f + h$, and the tails of $e + i$ and $f + h$ (right endpoints) and the heads of $f + j$ and $g + i$ (left endpoints) correspond. Suppose another sum edge (dashed) splits $f + j$ into subedges 1 and 2 and $g + i$ into 5 and 6 and suppose yet another splits $e + i$ into 3 and 4 and $f + j$ into 7 and 8. Subedges 1, 2, 3, and 4 are neighbors of 5, 6, 7, and 8, respectively, because of corresponding list element endpoints. In addition 2 is a neighbor of 7 because 2 contains $g + h$ (also 7 contains $e + j$) at the criticality. If one or more of the arcs e, g, h, j are concave then some of the linear sum edges might be missing. If, for example, $e + i$ is missing, then the tail of $f + h$ has no corresponding endpoint.

We group the faces into connected components via graph traversal. The *shells* are the components where each face has a neighbor at every boundary point. Specifically, it should have two vertical neighbors. If it is linear, it should have both horizontal neighbors. If it contains a corresponding endpoint, as do subedges 2 and 7 in Fig. 11, it should have two neighbors at that criticality

We select a sample point t on each shell and discard the shell if $M + t$ intersects F except at the point of tangency. Next, we compute the shell nesting order by ray casting. We pick a ray orthogonal to the θ axis, so ray/face intersection reduces to ray/edge intersection. The result is a boundary representation of the cells (connected components) of the free space.

The path planner reports failure if the start and goal configurations are in different cells; otherwise it finds a path with a bug algorithm [14].

VI. Robustness

The free space construction algorithm is formulated in the real-RAM model where real arithmetic has unit cost. The control logic is expressed in terms of predicates: polynomials in the input parameters whose signs are interpreted as truth values. The robustness problem is how to implement real-RAM algorithms accurately and efficiently. We wish to use floating point arithmetic, and a numerical solver for the criticality equations, because these are accurate, fast, and memory efficient. But even a tiny computation error can cause a predicate to be assigned the wrong sign, which can create a large error in the algorithm output. We wish to handle all inputs, whereas the real-RAM algorithm requires simple criticalities with unique θ values.

A. Prior Work

The mainstream robustness method is to evaluate predicates exactly via algebraic computation [15]. Algebraic computation increases bit complexity, hence running time. Floating point filtering techniques somewhat reduce this cost [16]. Exact evaluation cannot assign a sign to a *degenerate* predicate whose exact value is zero. Degeneracy is common due to design constraints and to symmetry.

The other popular robustness method, controlled perturbation (CP) [17], [18], evaluates predicates with floating point arithmetic. The computed sign of $f(a)$ is correct, and the predicate is called safe, when the computed magnitude satisfies $|f(a)| > \varepsilon$ with ε a function of f and a. The numerical input to the algorithm is perturbed randomly by up to δ and the algorithm is executed. If every predicate is safe, the output is returned. Otherwise, the algorithm is rerun with a different δ. The final δ bounds the error due to replacing the true input with a verifiable input. This error is inconsequential when δ is less than the required accuracy of the application that provides the input, such as the tolerances in path planning.

CP is faster than exact computation because all computations are in floating point. Whereas degenerate predicates defeat exact computation, CP handles them just like non-degenerate predicates. Their perturbed values are of order δ, so they are verifiable for reasonable δ values. However, CP performs poorly on *singular* predicates whose value and gradient are both zero. A singular predicate of degree d requires δ of order $\sqrt[d]{\varepsilon}$. This error is unacceptable with the reported ε values and is marginal even with ε equal to the rounding unit, $\mu \approx 10^{-16}$.

Melhorn *et al* [19] handle singular predicates by rerunning the algorithm with extended precision arithmetic, which uses much more time and space than floating point arithmetic. We [20] identify the cases where the predicates in a Minkowski sum algorithm can be singular and replace them by non-degenerate predicates in defined parameters. This approach does not generalize.

B. Adaptive-Precision Controlled Perturbation

We have developed an extension of CP, adaptive-precision controlled perturbation (ACP), that handles singular predicates by selectively increasing the arithmetic precision and that supports *implicit parameters* that denote the roots of polynomials.

The user specifies the required accuracy, δ. ACP replaces each input parameter, x, by $x + d$ with d uniform in $[-\delta, \delta]$. The user defines explicit parameters using arithmetic operators on prior parameters. ACP assigns them values using interval arithmetic. ACP computes predicates using the interval filter: evaluate the predicate polynomial in interval arithmetic to obtain $[l, u]$, return -1 if $u < 0$, return 1 if $l > 0$, and fail otherwise. If a predicate fails, the program reevaluates it after increasing the precision of its arguments. Each geometric object has pointers to the objects on which it depends. For example, a circle/line hit criticality (Fig. 7) points to the circle and the line. The program traverses these pointers and increases the precision recursively. The initial precision is double float and higher precision is implemented using MPFR [21].

An implicit parameter, x, is a root of a polynomial, f. ACP isolates the roots of f on an interval, $[L, U]$, using a straightforward recursive algorithm that is $\mathrm{O}(d^2)$ in the degree, d, of f (there are faster algorithms in the literature). We describe how to implement this algorithm in interval arithmetic. If $d \leq 2$, use the explicit roots. If $d > 2$, let $x_1, \ldots, x_n$ be the roots of f' with $x_0 = L$ and $x_{n+1} = U$. For each pair, x_{i-1}, x_i, calculate $\mathrm{sign}(f(x_{i-1})), \mathrm{sign}(f(x_i))$. If either ACP sign test fails, restart with higher precision. If $\mathrm{sign}(f(x_{i-1})) \neq \mathrm{sign}(f(x_i))$, shrink the root isolating interval $[x_{i-1}, x_i]$ using interval Newton's method. ACP increases the precision of x by further shrinking its interval after increasing the precision of f.

We use ACP to implement the free space construction algorithm. The input parameters specify the part boundary geometry. The type 3 critical angles are implicit parameters. Their polynomials are obtained by substituting the appropriate charts of the rational parameterization of the unit circle into the bivariate criticality equations in Sec. IV-D. The rest of the implementation follows the real-RAM algorithm. The implementation handles any input and generates an output that is correct for a δ-perturbation of the input.

VII. Validation

Source code for our validation is available at http://www.cs.miami.edu/~vjm/robust. We validate the path planning algorithm by constructing a maximal clearance path for a given robot, obstacle, and start/goal configurations. We find the largest number, s, for which the algorithm finds a path for s-offsets of the parts. The s-offset of a part is its Minkowski sum with an s-disk centered at the origin. A path for the offset parts is a $2s$-clearance path for the original parts. We compute the maximal s by bisection search on $[0, 1]$ to the floating point resolution, which takes 53 iterations. The clearance is $s - \mathrm{O}(\delta)$ because the $2s$-path is correct for a δ-perturbation of the input parameters, which causes an $\mathrm{O}(\delta)$ perturbation of the parts.

The bisection algorithm tests how our program handles degenerate input. A zero-clearance path is degenerate because the robot has multiple simultaneous contacts with the obstacle. Hence, the final iterations of the algorithm are nearly

TABLE I
RESULTS: TEST i, ITERATION k, m SUM EDGES, e EDGES IN THE SUBDIVISION, f FREE FACES, t RUNNING TIME IN SECONDS FOR ONE CORE OF AN INTEL CORE 2 DUO, t_{12} TIME COMPUTING TYPE 1 AND 2 CRITICALITIES, $c = 10^6 t/(e \lg e)$, $c_{12} = 10^3 t_{12}/m$, r_f AND r_e NUMBER OF POLYNOMIALS WITH FLOATING POINT AND EXTENDED PRECISION ROOT ISOLATION, AND p_f AND p_e NUMBER OF FLOATING POINT AND EXTENDED PRECISION PREDICATE EVALUATIONS.

i	k	m	e	f	e/f	t	t_{12}	c	c_{12}	r_f	r_e	p_f	p_e
1	1	84	2,017	424	4.8	0.2	0.1	6.8	1.1	14	0	1.1e5	36,000
1	53	84	2,027	424	4.8	0.2	0.1	6.7	1.1	12	0	1.1e5	35,000
2	1	400	26,000	1,000	26.0	0.4	0.3	1.0	0.8	600	0	2.7e6	0
2	53	400	36,000	2,000	18.0	2.6	0.3	4.8	0.8	1,700	400	3.7e6	3.2e5
3	1	700	89,000	3,000	29.7	1.4	0.9	1.0	1.3	3,200	0	8.9e6	0
3	53	700	1.0e5	4,000	25.0	1.8	1.0	1.1	1.4	5,600	5	1.1e7	0
4	1	3,000	48,000	3,000	16.0	1.4	0.9	1.9	0.3	5,300	60	1.2e7	45,000
4	35	2,500	1.0e5	3,000	33.3	2.6	1.0	1.6	0.4	3,300	200	1.5e7	1.7e5
5	1	4,400	1.4e5	15,000	9.3	3.1	1.8	1.3	0.4	3,500	36	2.7e7	68,000
5	22	3,900	4.1e5	10,000	41.0	9.7	2.1	1.3	0.5	24,000	335	4.6e7	6.9e5
6	1	8,000	2.2e5	20,000	11.0	6.1	3.7	1.6	0.5	2,800	300	5.4e7	2.1e5
6	34	6,800	5.1e5	11,000	46.4	13	3.9	1.3	0.6	21,000	900	7.0e7	9.0e5
7	1	13,000	5.5e5	31,000	17.7	14	8.9	1.3	0.7	12,000	200	1.4e8	2.1e5
7	27	12,000	1.4e6	28,000	50.0	33	10	1.2	0.8	81,000	1,700	2.1e8	1.7e6
8	1	18,000	7.1e5	75,000	9.5	22	16	1.6	0.9	12,000	300	2.3e8	2.6e5
8	24	15,000	1.5e6	55,000	27.3	55	14	1.8	0.9	72,000	3,600	2.6e8	5.0e6

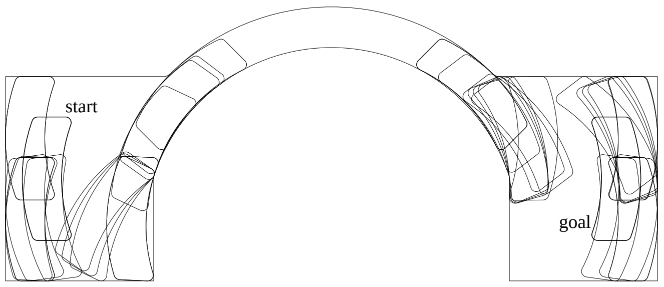


Fig. 12. Tightest clearance path $s = 0.27865$ in 2D.

degenerate. In every test, we obtain a correct output and the running time increases modestly from the first iteration to the last iteration. We use $\delta = 10^{-8}$ and extended precision of $p = 250$ binary digits. Table I shows the results.

Test 1 is an 8-edge robot and an 18-edge obstacle with two chambers connected by a narrow passage (Fig. 12). The start and goal configurations are in the top and bottom chambers. We ensure that the robot simultaneously rotates and translates for large s values, which forces the planner to search a large fraction of the configuration space, by making the passage boundary arcs concentric. Fig. 12 shows one sample configuration per face visited in the tightest clearance path, and Fig. 13 shows the 3D free space boundary for a smaller s so the narrow passage is visible.

Test 2 is a 5-pointed star with 15 edges inside a 6-sided container with 42 edges (Fig. 14a–b). The start and goal positions are identical, the start angle is $0°$, and the goal angle is $72°$ ($1/5$ of a turn). The solution is for the star to pivot about each of its tips in turn. Fig. 14b shows it part way through a pivot. Sliding contacts occur at 1, 2, 3, and 4. The other two tips are close but not in contact with the obstacle. This motion is degenerate because three contacts should fix the configuration. The input perturbation eliminates the degeneracy and no extended precision arithmetic is required ($r_e = 0$ and $p_e = 0$). The motion is doubly degenerate at the maximum offset, $s = 0.1$, because four points touch the container throughout each pivot. The input perturbation still

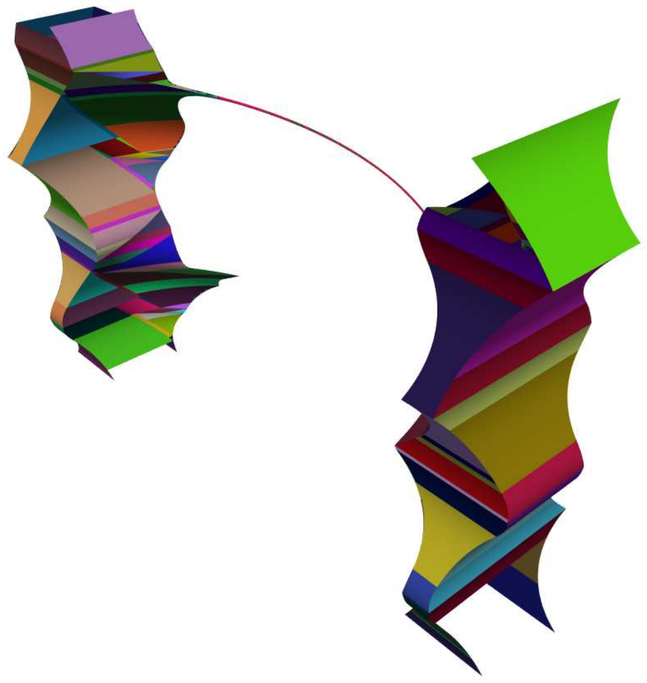


Fig. 13. View of free space boundary from blocked space with randomly colored faces for $s = 0.265625$.

eliminates the degeneracy, but significant extended precision arithmetic occurs ($r_e = 400$ and $p_e = 320,000$). Thanks to adaptive precision, running time only increases by a factor of six.

Test 3 is a 7-pointed star inside an 8-sided container. Unlike test 2, there is no sliding contact, so the input is not degenerate, the $s = 0.1$ offset is simply degenerate, and there is little extended precision arithmetic.

Tests 4–8 are an n-pin wheel inside a channel with $n + 1$ segments for $n = 3, \ldots, 7$. Figs. 14c and d show $n = 5$ and $n = 7$ and the points of contact during a pivot. The bisection search ends after $k < 53$ iterations because the start and goal become δ-close to blocked space. At minimum clearance

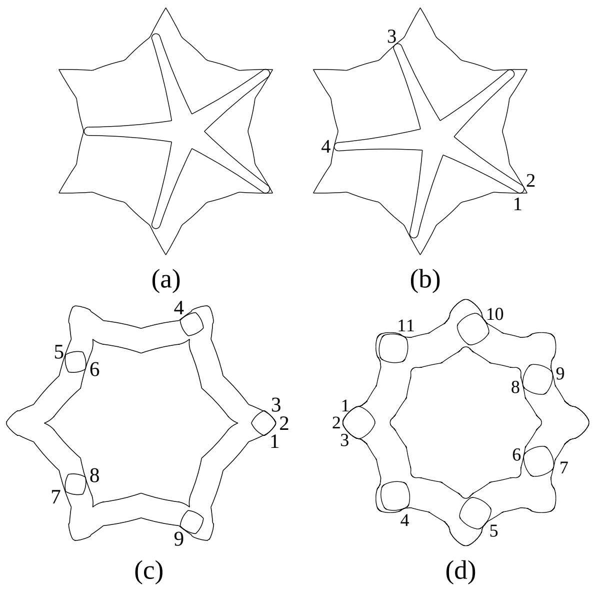


Fig. 14. Star in container: start configuration (a) and pivoting (b); wheel in channel (c), (d).

(maximum k), symmetry causes multiple pin/channel contacts. Hence, there is significant extended precision arithmetic (r_e) for large k. Even so, the time increases by at most a factor of three.

As m increases, the time, t_{12}, for computing type 1 and 2 criticalities is a smaller fraction of the total. This is the $n^4 = m^2$ component of the predicted $\mathrm{O}((n^4 + c_3)\log n)$ running time. Although type 2 criticality computation is $\mathrm{O}(m^2)$, we reduce the actual running time to $\mathrm{O}(m)$ using kd-trees, bounding boxes on the area swept by a sum edge over its θ interval, and a geometric constraint on the relative sizes of the constituent robot and obstacle edges of intersecting sum edges. For the geometric constraint, define

$$\mathrm{mind}(e, f) = \min_{p\in e, q\in f} |p - q|, \quad (3)$$

$$\mathrm{maxd}(e, f) = \max_{p\in e, q\in f} |p - q|. \quad (4)$$

If $[\mathrm{mind}(e, f), \mathrm{maxd}(e, f)] \cap [\mathrm{mind}(g, h), \mathrm{maxd}(g, h)] = \emptyset$ then a simultaneous f/g contact and e/h contact cannot occur and $f \oplus g$ will never intersect $e \oplus h$. Hence, $c_{12} = 10^3 t_{12}/m \approx 1$, and $\mathrm{O}(c_3 \log n)$ more accurately models the time. Looking at the value $c = 10^6 t/(e \lg e)$, we see that $c < 2$ except an anomalous values of 6.8 and 4.8 for small problems. Since, c_3 and e are proportional, these values agree with the model. The actual output size, f, is much smaller than e because most faces do not appear on the free space boundary. The ratio e/f ranges from 4.4 to 50. To this extent, the algorithm is not truly output sensitive.

The free space construction program does not handle non-smooth convex vertices or linear edges. The first restriction eliminates the three types of criticality discussed in Sec. 3.4.3.3 and the second eliminates the complicated type 1 criticalities shown in Fig. 11. The path planning program is unaffected by the first restriction because offsets of parts cannot have non-smooth convex vertices. The second restriction requires us to approximate linear edges by circular edges with large radii. We use this approximation in test 1 for the chamber walls. The penalty is the high percentage of extended precision predicate evaluations, p_e, due to the high numerical condition of the equations involving these edges. Reducing the radius from 10^8 to 10^6 decreases p_e from 36,158 to 10,048, and reduces the running time from 0.15 to 0.06. For radius 10^4, p_e is 1942 and the time is 0.031.

VIII. Discussion

We have presented a complete path planning algorithm for a planar robot with three degrees of freedom where the robot and the obstacle boundaries consist of circular and linear edges. We have implemented the algorithm using our ACP robustness library. The program is complete in that for any input the output is correct for a δ-deformation of the input. By setting $\delta = 10^{-8}$, we make the deformed input indistinguishable from the actual input. We have demonstrated that the program is fast and output sensitive on problems that contain narrow passages. The running time is only slightly dependent on the passage width, ε, whereas the cost of sample-based planning is ε^{-d} with d the configuration space dimension.

We are developing a complete, output sensitive path planning algorithm for a polyhedral robot that translates freely and rotates around a fixed axis ($d = 4$). Our plane sweep generalizes to a volume sweep. The challenge is to derive the criticality equations and the subdivision update rules. Our next goal is to develop a hybrid algorithm for a polyhedral robot with six degrees of freedom whose complexity is $\mathrm{O}(1/\varepsilon^2)$ and that performs well on narrow channels. We will sample the configuration space with $\mathrm{O}(1/\varepsilon^2)$ $d = 4$ subspaces, construct their free spaces, and link them into a path planning graph.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. 0904707 (Milenkovic) and 0904832 (Sacks). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Fig. 13 created by Justin Stoecker.

References

[1] A. F. van der Stappen, D. Halperin, and M. H. Overmars, “The complexity of the free space for a robot moving amidst fat obstacles,” *Computational Geometry Theory and Applications*, vol. 3, pp. 353–373, 1993.

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

[3] E. Sacks and L. Joskowicz, *The configuration space method for kinematic design of mechanical systems*. MIT Press, 2010.

[4] V. Milenkovic and K. Daniels, “Translational polygon containment and minimal enclosure using mathematical programming,” *International Transactions in Operational Research*, vol. 6, pp. 525–554, 1999.

[5] F. Avnaim and J. D. Boissonnat, “Polygon placement under translation and rotation,” *Informatique Théorique et Applications*, vol. 31, no. 1, pp. 5–28, 1989.

[6] E. Sacks, “Practical sliced configuration spaces for curved planar pairs,” *International Journal of Robotics Research*, vol. 18, no. 1, pp. 59–63, Jan. 1999.

[7] A. F. van der Stappen, M. H. Overmars, M. de Berg, and J. Vleugels, “Motion planning in environments with low obstacle density,” *Discrete and Computational Geometry*, vol. 20, no. 4, pp. 561–587, 1998.

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

[9] P. Hachenberger, "Exact minkowski sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces," *Algorithmica*, vol. 55, pp. 329–345, 2009.

[10] O. Salzman, M. Hemmer, B. Raveh, and D. Halperin, "Motion planning via manifold samples," in *Proceedings of the 19th European Symposium on Algorithms*, 2011.

[11] L. Zhang, Y. J. Kim, and D. Manocha, "A hybrid approach for complete motion planning," in *IEEE/RSJ International Conference On Intelligent Robots and Systems*, 2007, pp. 7–14.

[12] A. Kaul, M. A. O'Connor, and V. Srinivasan, "Computing Minkowski sums of regular polygons," in *Proceedings of the Third Canadian Conference on Computational Geometry*, 1991, pp. 74–77.

[13] V. Milenkovic and E. Sacks, "Two approximate minkowski sum algorithms," *International Journal of Computational Geometry and Applications*, vol. 20, no. 4, pp. 485–509, 2010.

[14] V. J. Lumelski and A. A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment," *IEEE Transactions on Control*, vol. AC-31, no. 11, pp. 1058–1063, 1986.

[15] C. Yap, "Robust geometric computation," in *Handbook of discrete and computational geometry*, 2nd ed., J. E. Goodman and J. O'Rourke, Eds. Boca Raton, FL: CRC Press, 2004, ch. 41, pp. 927–952.

[16] "Exact computational geometry," http://cs.nyu.edu/exact.

[17] D. Halperin and E. Leiserowitz, "Controlled perturbation for arrangements of circles," *International Journal of Computational Geometry and Applications*, vol. 14, no. 4–5, pp. 277–310, 2004.

[18] S. Funke, C. Klein, K. Mehlhorn, and S. Schmitt, "Controlled perturbation for delaunay triangulations," in *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, ACM. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005, pp. 1047–1056.

[19] K. Mehlhornlow, R. Osbilda, and M. Sagraloffa, "A general approach to the analysis of controlled perturbation algorithms," *Computational Geometry*, vol. 44, no. 9, pp. 507–528, 2011.

[20] E. Sacks, V. Milenkovic, and M.-H. Kyung, "Controlled linear perturbation," *Computer-Aided Design*, vol. 43, no. 10, pp. 1250–1257, 2011.

[21] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "MPFR: A multiple precision binary floating point library with correct rounding," *ACM Transactions on Mathematical Software*, vol. 33, 2007.