Rods and Rings: Soft Subdivision Planner for $\mathbb{R}^3 \times S^2$

Ching-Hsiang Hsu[1], **Yi-Jen Chiang**[2], **and Chee Yap**[3]

1 **Department of Computer Science, New York University, New York, NY, USA;** chhsu@nyu.edu.
2 **Department of Computer Science and Engineering, New York University, Brooklyn, NY, USA;** chiang@nyu.edu.
3 **Department of Computer Science, New York University, New York, NY, USA;** yap@cs.nyu.edu.

Abstract

We consider path planning for a rigid spatial robot moving amidst polyhedral obstacles. Our robot is either a rod or a ring. Being axially-symmetric, their configuration space is $\mathbb{R}^3 \times S^2$ with 5 degrees of freedom (DOF). Correct, complete and practical path planning for such robots is at the cutting edge of current robotics. While the rod is one of the most widely studied spatial robots in path planning, the ring seems to be new, and a rare example of a non-simply-connected robot. This work provides rigorous and complete algorithms for these robots with theoretical guarantees. We implemented the algorithms in our open-source Core Library. Experiments show that they are practical, achieving near real-time performance.

Our subdivision path planner is based on the twin foundations of ε-exactness and soft predicates. Correct implementation is relatively easy. The technical innovations include subdivision atlases for S^2, introduction of Σ_2 representations for footprints, and extensions of our feature-based technique for "opening up the blackbox of collision detection".

Keywords and phrases Algorithmic Motion Planning; Subdivision Methods; Resolution-Exact Algorithms; Soft Predicates; Rod Robots; Axial-Symmetric Robots.

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Motion planning [18, 6] is a fundamental topic in robotics because the typical robot is capable of movement. Such algorithms are increasingly relevant with the current surge of interest in inexpensive commercial mobile robots, from domestic robots that vacuum the floor to drones that deliver packages. We focus on what is called **path planning** which, in its elemental form, asks for a collision-free path from a start to a goal position, assuming a known environment. Path planning is based on robot kinematics and collision-detection only, and the variety of such problems are surveyed in [15]. The output of a "path planner" is either a path or a `NO-PATH`, signifying that no path exists. Remarkably, the single bit of information encoded by `NO-PATH` is often missing in discussions. The standard definitions of correctness for path planners (**resolution completeness** and **probabilistic completeness**) omit this bit [33]. The last 30 years have seen a flowering of practical path planning algorithms. The dominant algorithmic paradigm of these planners has been variants of the **Sampling Approach** such as PRM, EST, RRT, SRT, etc (see [6, p. 201]). Because this bit of information is not built into the specification of such algorithms, it has led to non-termination issues and a large literature addressing the "narrow passage problem" (e.g., [23, 9]). Our present paper is based on the **Subdivision Approach**. This approach has a venerable history in robotics – see [4, 41] for early planners based on subdivision.

Exact path planning has many issues [37], including a serious gap between theory and implementability. In [33, 38], we introduced a theoretical framework based on subdivision to close this gap. This paper demonstrates that our framework is able to achieve rigorous state-of-the-art planners in 3D by designing and implementing planners for rod and ring robots. Figure 1 shows our rod robot in an environment with 100 random tetrahedra. Figure 5 (in Appendix A) shows our ring robot in an environment with 2 L-shaped posts. See a video demo from http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html.

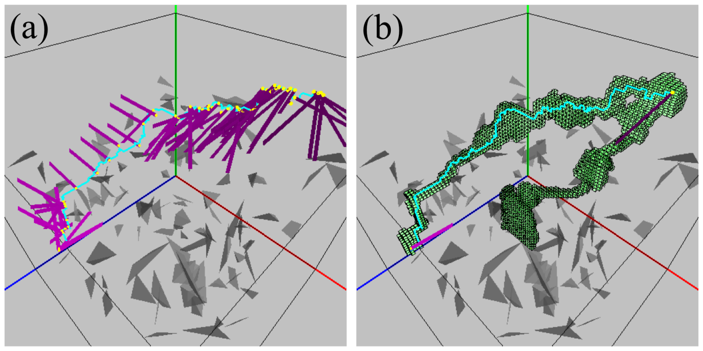

Figure 1 Rod robot amidst 100 random tetrahedra: (a) trace of a found path; (b) subdivision of translational boxes on the path.

In this paper, we consider a rigid spatial robot R_0 that has an axis of symmetry. See Figure 2(a) for several possibilities for R_0: rod ("ladder"), cone ("space shuttle"), disc ("frisbee") and ring ("space station"). Our techniques easily allow these shapes to be "thickened". The **configuration space** may be taken to be $C_{space} = \mathbb{R}^3 \times S^2$ where S^2 is the unit 2-sphere. We identify R_0 with a closed subset of $\mathbb{R}^3$, called its "canonical footprint". E.g., if R_0 is a rod (resp., ring), then the canonical footprint is a line segment (resp., circle) in $\mathbb{R}^3$. Each configuration $\gamma \in C_{space}$ corresponds to a rotated translated copy of the canonical footprint, which we denote by $Fp(\gamma)$. Path planning involves another input, the **obstacle set** $\Omega \subseteq \mathbb{R}^3$ that the robot must avoid. We assume that Ω is a closed polyhedral set. Say γ is **free** if $Fp(\gamma) \cap \Omega$ is empty. The **free space** comprising all the free configurations is an open set by our assumptions, and is denoted $C_{free} = C_{free}(\Omega)$. A parametrized continuous curve $\mu : [0,1] \to C_{space}$ is called a **path** if the range of μ is in C_{free}. Path planning amounts to finding such paths. Following [41], we need to classify boxes $B \subseteq C_{space}$ into one of three types: `FREE`, `STUCK` or `MIXED`. Let $C(B)$ denote the classification of B: $C(B) =$ `FREE` if $B \subseteq C_{free}$, and $C(B) =$ `STUCK` if B is in the interior of $C_{space} \setminus C_{free}$. Otherwise, $C(B) =$ `MIXED`. One of our goals is to introduce classifications $\widetilde{C}(B)$ that are "soft versions" of $C(B)$ (see Appendix B).

We present four desiderata in path planning:
(G0) the planner must be mathematical rigorous and complete;
(G1) it must have correct implementations which are also
(G2) relatively easy to achieve and
(G3) practically efficient.
In (G0), we use the standard Computer Science notion of an algorithm being **complete** if (a) it is partially complete[1] and (b) it halts. The notion of "resolution completeness" and

[1] Partial completeness means the algorithm produces a correct output *provided* it halts.

"probabilistic completeness" in robotics has requirement (a) but not (b). But desideratum (G0) is only the base line. Having a (G0)-planner may not be worth much in a practical area like robotics unless it also has implementations with properties (G1-G3). E.g., the usual exact algorithms satisfy (G0) but their typical implementations fail (G1). With proper techniques [31], it is possible to satisfy (G1); thus Halperin et al [14] describe such solutions in 2D based on the `CGAL` library. Both (G0) and (G1) can be formalized (see next), but (G2) and (G3) are informal. The robotics community has developed various criteria to evaluate (G2) and (G3). The accepted practice here is an implementation (proving (G2)) that achieves "real time" performance on a suite of non-trivial instances (proving (G3)).

The main contribution of this paper is the design of planners for spatial robots with 5 DOFs that have the "good" properties (G0-G3). This seems to be the first for such robots. To achieve our results, we introduce theoretical innovations and algorithmic techniques that we believe will prove to be more widely applicable than in subdivision methods.

In path planning and in Computational Geometry, there is a widely accepted interpretation of desideratum (G0): it is usually simply called "exact algorithms". But to stress that there could be alternative notions, we refer to this standard notion as "exact (unqualified)". Planners that are exact (unqualified) are first shown in [27]; this can be viewed as a fundamental theorem on decidability of connectivity in semi-algebraic sets [2]. The curse of exact (unqualified) algorithms is that you must detect any degeneracies in the input and handle them explicitly in the algorithm. But **unconditional** exact (unqualified) algorithms are rare, mainly because degeneracies are numerous and hard to analyze: the usual expedient is to require nice (non-degenerate) inputs. Such gaps in exact (unqualified) algorithms are not an issue as long as they are not implemented. For non-linear problems beyond 2D, complete degeneracy analysis is largely non-existent. This is vividly seen in the fact that, despite long-time interest, there is still no exact (unqualified) algorithm for the Euclidean Voronoi diagram of a polyhedral set (see [16, 13, 12, 35]). For similar reasons, unconditional exact (unqualified) planners in 3D are unavailable.

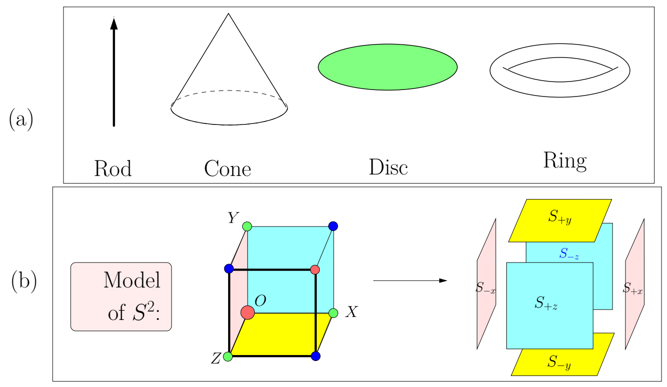

Figure 2 (a) Spatial rigid robots with $C_{space} = \mathbb{R}^3 \times S^2$. (b) Subdivision atlas for S^2 via $\widehat{S^2}$.

We now address (G1-G3). The typical implementation is based on machine arithmetic (the IEEE standard). Such implementations may satisfy (G2) but almost certainly not (G1). We regard this as a (G1-G2) tradeoff. As a matter of fact, our implementations here as well as in our previous papers [33, 21, 34] are such machine implementations. This follows the practice in robotics community, in order to have a fair comparison against other implementations. Below, we shall expand on our claims about (G1-G3) including how to achieve theoretically correct implementation (G1). What makes this possible is our replacement of "exact (unqualified)" planners by "exact (up to resolution)" planners, defined as follows:

Resolution-Exact Path Planning for robot R_0:
Input: $(\alpha, \beta, \Omega; B_0, \varepsilon)$
where $\alpha, \beta \in C_{space}(R_0)$ is the start and goal, $\Omega \subseteq \mathbb{R}^3$ the obstacle set, $B_0 \subseteq C_{space}(R_0)$ is a box, and $\varepsilon > 0$.
Output: Halt with either an Ω-free path from α to β in B_0, or `NO-PATH` with the signification (P) and (N) below.

The **resolution-exact planner** (or, ε-exact planner) has an **accuracy constant** $K > 1$ (independent of input) such that its output satisfies two conditions:

- (P) If there is a path (from α to β in B_0) of clearance $K\varepsilon$, the output *must* be[2] a path.
- (N) If there is no path in B_0 of clearance ε/K, the output *must* be `NO-PATH`.

Here, clearance of a path is the minimum separation of the obstacle set Ω from the robot's footprint on the path. Note that the preconditions for (P) and (N) are not exhaustive: in case the input fails both preconditions, our planner may either output a path or `NO-PATH`. This indeterminacy is essential to escape from exact computation (and arguably justified for robotics [38]). But resolution-exactness is just a definition. How do we design such algorithms? We propose to use subdivision, as noted above, but we couple this with the idea of **soft predicates** to exploit resolution-exactness. We replace the classification $C(B)$ by a soft version $\widetilde{C}(B)$ [33]. This leads to a general resolution-exact planner which we call **Soft Subdivision Search** (SSS) [38, 39] that shares many of the favorable properties of sampling planners (and unlike exact planners). We demonstrated in [33, 21, 34] that for planar robots with up to 4 DOFs, our planners can consistently outperform state-of-the-art sampling-based planners.

1.1 Contributions of this Paper

We design ε-exact planners for rods and rings, with implementations. One major technical difference from our previous work on planar robots [33, 21, 34] is that we gave up the notion of "forbidden orientations" (this is quite complicated for spatial robots) but opted for an alternative approach to bound the footprint of boxes using what we call Σ_2-sets (Sec. **??**). Such sets generalize the concept of Π_1-sets in our previous work. One benefit of Σ_2-sets is that they are very flexible, allowing us to easily extend the planners to "thick" versions of a rod or ring. The trade-off in using Σ_2-sets is a modest increase in the accuracy constant K. Such Σ_2-sets were critical for achieving our practical implementation of a ring robot.

Overview of the Paper

Section 2 is a brief literature review. Section 3 explains an essential preliminary to doing subdivision in S^2. Sections 4–6 describe our technique for computing approximate footprints of rods and rings. We discuss efficiency and experimental results in Section 7. We conclude in Section 8. Appendices A-G contain background materials and proofs.

[2] For simplicity, we do not require the output path to have any particular clearance, but we could require clearance $\geq \varepsilon/K$ as in [33].

2 Literature Review

Halperin et al [15] gave a general survey of path planning. An early survey is [36] where two universal approaches to exact path planning were described: cell-decomposition [26] and retraction [25, 24, 5]. Since exact path planning is a semi-algebraic problem [27], it is reducible to general (double-exponential) cylindrical algebraic decomposition techniques [2]. But exploiting path planning as a connectivity problem yields singly-exponential time (e.g, [11]). The case of a planar rod (called "ladder") was first studied in [26] using cell-decomposition. More efficient (quadratic time) methods based on the retraction method were introduced in [29, 30]. On-line versions for a planar rod are also available [8, 7].

Spatial rods were first treated in [28]. The combinatorial complexity of its free space is $\Omega(n^4)$ in the worst case but this can be closely matched by an $O(n^{4+\epsilon})$ time algorithm [17]. The most detailed published planner for a spatial rod is Lee and Choset [19]. They use a retraction approach. The paper expose many useful and interesting details of their computational primitives (see its appendices). In particular, they follow a Voronoi edge by a numerical path tracking; but like most numerical code, there is no a priori guarantee of correctness. Although their goal is an exact path planner, degeneracies are not fully discussed. The two accompanying videos of their implementation have no timing or experimental data.

One of the few papers to address the non-existence of paths is Zhang et al [40]. Their implementation work is perhaps the closest to our current work, using subdivision. They noted that "no good implementations are known for general robots with higher than three DOFs". They achieved planners with 3 and 4 DOFs (one of which is a spatial robot). Although their planners can detect `NO-PATH`, they do not guarantee detection.

3 Subdivision Charts and Atlas for S^2

Terminology. We fix some terminology for the rest of the paper. The fundamental **footprint map** Fp from configuration space $C_{space} = C_{space}(R_0)$ to subsets of $\mathbb{R}^3$ was introduced above. If $B \subseteq C_{space}$ is any set of configurations, we define $Fp(B)$ as the union of $Fp(\gamma)$ as γ ranges over B. Typically, B is a "box" of C_{space} (see below for its meaning in non-Euclidean space S^2). We may assume $\Omega \subseteq \mathbb{R}^3$ is regular (i.e., equal to the closure of its interior). Although Ω need not be bounded (e.g., it may be the complement of a box), we assume its boundary $\partial(\Omega)$ is a bounded set. Then $\partial(\Omega)$ is partitioned into a set of **(boundary) features**: **corners** (points), **edges** (relatively open line segments), or **walls** (relatively open triangles). Let $\Phi(\Omega)$ denote the set of features of Ω. The (minimal) set of corners and edges is uniquely defined by Ω, but walls depend on a triangulation of $\partial\Omega$. If $A, B \subseteq \mathbb{R}^3$, define their **separation** $\text{Sep}(A, B) := \inf\{\|a - b\| : a \in A, b \in B\}$ where $\|a\|$ is the Euclidean norm. The **clearance** of γ is $\text{Sep}(Fp(\gamma), \Omega)$. Say γ is **Ω-free** (or simply **free**) if it has positive clearance. Let $C_{free} = C_{free}(\Omega)$ be the set of Ω-free configurations. The **clearance** of a path $\mu : [0, 1] \to C_{space}$ is the minimum clearance attained by $\mu(t)$ as t ranges over $[0, 1]$.

Subdivision in Non-Euclidean Spaces. Our C_{space} has an Euclidean part ($\mathbb{R}^3$) and a non-Euclidean part (S^2). We know how to do subdivision in $\mathbb{R}^3$ but it is less clear for S^2. Non-Euclidean spaces can be represented either (1) as a submanifold of $\mathbb{R}^m$ for some m (e.g., $SO(3) \subseteq \mathbb{R}^9$ viewed as orthogonal matrices) or (2) as a subset of $\mathbb{R}^m$ subject to identification (in the sense of quotient topology [22]). A common representation of S^2 (e.g., [19]) uses a pair of angles (i.e., spherical polar coordinates) $(\theta, \phi) \in [0, 2\pi] \times [-\pi/2, \pi/2]$ with the identification $(\theta, \phi) \equiv (\theta', \phi')$ iff $\{\theta, \theta'\} = \{0, 2\pi\}$ or $\phi = \phi' = \pi/2$ (North Pole) or $\phi = \phi' = -\pi/2$ (South Pole). Thus an entire circle of values θ is identified with each pole, causing severe distortions near the poles which are singularities. So the numerical primitives described in [19, Appendix F] have severe numerical instabilities.

To obtain a representation of S^2 without singularities, we use the following map [39]

$$q \in \mathbb{R}^3 \mapsto \widehat{q} := q/\|q\|_\infty$$

whose range is the boundary of a 3D cube $\widehat{S^2} := \partial([-1,1]^3)$. This map is a bijection when its domain is restricted to S^2, with inverse map $q \in \widehat{S^2} \mapsto \overline{q} := q/\|q\|_2 \in S^2$. Thus $\overline{\widehat{q}}$ is the identity for $q \in S^2$. We call $\widehat{S^2}$ the **square model** of S^2. We view S^2 and $\widehat{S^2}$ as metric spaces: S^2 has a natural metric whose geodesics are arcs of great circles. The geodesics on S^2 are mapped to corresponding polygonal geodesic paths on $\widehat{S^2}$ by the map $q \mapsto \widehat{q}$. Define the constant

$$C_0 := \sup_{p \neq q \in S^2} \left\{ \max \left\{ \frac{d_2(p,q)}{\widehat{d_2}(\widehat{p},\widehat{q})}, \frac{\widehat{d_2}(\widehat{p},\widehat{q})}{d_2(p,q)} \right\} \right\}$$

where d_2 and $\widehat{d_2}$ are the metrics on S^2 and $\widehat{S^2}$ respectively. Clearly $C_0 \geq 1$. Intuitively, C_0 is the largest distortion factor produced by the map $q \mapsto \widehat{q}$ (by definition the inverse map has the same factor).

▶ **Lemma 1.** $C_0 = \sqrt{3}$.

The proof in Appendix C (C.1) also shows that the worst distortion is near the center of the faces of $\widehat{S^2}$. The constant C_0 is one of the constants (there are 3 others) that go into the ultimate accuracy constant K in the definition of ε-exactness (see [39] for details).

It is obvious how to do subdivision in $\widehat{S^2}$: the first subdivision is into its 6 faces. This is illustrated in Figure 2(b). Subsequent subdivision is just the usual quadtree subdivision of each face. We interpret the subdivision of $\widehat{S^2}$ as a corresponding subdivision of S^2. In [39], we give a general description of this procedure using the concept of **subdivision charts and atlases** (borrowing terms from manifold theory).

4 Let us design soft predicates for $\mathbb{R}^3 \times S^2$

We focus on soft predicates because, in principle, once we have designed and implemented such a predicate, we already have a rigorous and complete planner within the **Soft Subdivision Search** (SSS) framework [33, 39]. For convenience, the SSS framework is summarized in the Appendix B. As noted in the introduction, our soft predicate $\widetilde{C}$ classifies any input box $B \subseteq C_{space}$ into one 3 possible values. A key idea of our 2-link robot work [21, 34] is the notion of "forbidden orientations" (of a box B, in the presence of Ω). The same concept may be attempted for $\mathbb{R}^3 \times S^2$, except that the details are more difficult to analyze and implement. Instead, this paper focuses on direct approximations of the footprint of a box $Fp(B) = \bigcup\{Fp(\gamma) : \gamma \in B\}$. Let $\widetilde{Fp}(B) \subseteq \mathbb{R}^3$ denote an **approximate footprint** of $Fp(B)$. This section is abstract, in order to expose the mathematical structure of what is needed to achieve resolution-exactness for our planners. The reader might peak at the next two sections to see the instantiations of these concepts for the rod/ring robot.

To understand what is needed of this approximation, recall that our approach to soft predicates is based on the "method of features" [33]. The idea is to maintain a set $\widetilde{\phi}(B)$ of features for each box B. We classify B as `MIXED` when the feature set is non-empty; otherwise, we can classify the box as `FREE` or `STUCK` with relative ease. This idea is quite obvious in 2D, but in 3D it is slightly more involved and is detailed in Appendix C (C.2).

Another basic technique to achieve efficiency is "inheritance of features": any feature of a box is also a feature of its parent.

We now show what this computational scheme demands of our approximate footprint. Define the exact and approximate **feature sets** of box B as $\phi(B) := \{f \in \Phi(\Omega) : f \cap Fp(B) \neq \emptyset\}$ and $\widetilde{\phi}(B) := \left\{f \in \Phi(\Omega) : f \cap \widetilde{Fp}(B) \neq \emptyset\right\}$. We need the fundamental inclusions

$$\widetilde{Fp}(B/\sigma) \subseteq Fp(B) \subseteq \widetilde{Fp}(B) \tag{1}$$

for some global constant $\sigma > 1$ in order to have conservative and σ-effective convergent soft predicates. Second, if B is a child of B' in the subdivision, we need

$$\widetilde{Fp}(B) \subseteq \widetilde{Fp}(B'). \tag{2}$$

in order to justify our method of feature inheritance.

Geometric Notations. We will be using planar concepts like circles, squares, etc, for sets that lie in some plane of $\mathbb{R}^3$. We shall call them **embedded** circles, squares, etc. By definition, if X is an embedded object then it defines a unique plane $Plane(X)$ (unless X lies in a line). Let $Ball(r, c) \subseteq \mathbb{R}^3$ denote a ball of radius r centered at c. If c is the origin, we simply write $Ball(r)$. Suppose $X \subseteq \mathbb{R}^3$ is any non-empty set. Let $Ball(X)$ denote the **circumscribing ball** of X, defined as the smallest ball containing X. Next, if $c \notin X$ then $Cone(c, X)$ denotes the union of all the rays from c through points in X, called the **cone** of X with **apex** c. We consider two cases of X in this cone definition: if X is a ball, then $Cone(c, X)$ is called a **round cone**. If the radius of ball X is r and the distance from the center of $Ball(X)$ to c is $h \geq r$, then call $\arcsin(r/h)$ the **half-angle** of the cone; note that the angle at the the apex is twice this half-angle. If X is an embedded square, we call $Cone(c, X)$ a **square cone**, and the ray from c through the center of the square is called the **axis** of the square cone. If P is any plane that intersects the axis of a square cone $Cone(c, X)$, then $P \cap Cone(c, X)$ is a square iff P is parallel to square X. A **ring** (resp., **cylinder**) is the Minkowski sum of an embedded circle (resp., a line) with a ball. Finally consider a box $B = B^t \times B^r \subseteq \mathbb{R}^3 \times \widehat{S^2}$ where B^r is a subsquare of a face of $\widehat{S^2}$. The **cone of** B, denoted $Cone(B)$, is the round cone $Cone(m_B, Ball(m_B + B^r))$. If the center of square $m_B + B^r$ is c and width of B^r is w, then $Cone(B)$ is just $Cone(m_B, Ball(c, w/\sqrt{2}))$.

4.1 On Σ_2-sets.

Besides the above correctness properties of $\widetilde{Fp}(B)$, we also need algorithms to decide if $\widetilde{Fp}(B)$ intersects a given feature f. We want such intersection algorithms to be easy to implement (G2) and efficient (G3). Such properties makes $\widetilde{Fp}(B)$ "nice". We now formalize and generalize some ideas of niceness that had been implicit in previous work [33, 21, 34].

An **elementary set** (in $\mathbb{R}^3$) is defined to be one of the following sets or their complements: half space, ball, ring, cone or cylinder. Let $\mathcal{E}$ (or $\mathcal{E}_3$) denote the set of elementary sets in $\mathbb{R}^3$; they are evidently "nice" in the preceding sense. (The niceness of a ring has some subtleties in Sec. 6.) In $\mathbb{R}^2$, we have a similar notion of elementary sets $\mathcal{E}_2$ comprising half-planes, discs or their complements. Note that all these elementary sets are defined by a single polynomial inequality – so technically, they are all "algebraic half-spaces".

Next recall a well-known[3] construction of an infinite hierarchy of sets, starting from some initial collection of sets. If Δ is any collection of sets, let $\Pi(\Delta)$ denote the collection of finite

[3] From mathematical analysis, constructive set theory and complexity theory.

intersection of sets in Δ; similarly, $\Sigma(\Delta)$ denotes the collection of finite union of sets in Δ. Then, starting with any collection Δ_0 of sets, define the infinite hierarchy of sets:

$$\Sigma_i, \Pi_i, \Delta_i \qquad (i \geq 1) \tag{3}$$

where $\Sigma_i := \Sigma(\Delta_{i-1})$, $\Pi_i := \Pi(\Delta_{i-1})$, and $\Delta_i := \Sigma_i \cup \Pi_i$. We thus have the inclusions

$$\Delta_{i-1} \subseteq \Sigma_i \cup \Pi_i \subseteq \Delta_i, \qquad i \geq 1.$$

An element of Σ_i or Π_i is simply called a Σ_i-set or a Π_i-set.

In our previous work, Δ_0 was the collection of elementary sets in $\mathbb{R}^2$, and predicates are defined with the help of "swept areas" which turns out to be Π_1-sets. In this paper, we take Δ_0 to be $\mathcal{E}$ and we shall be interested mainly in Σ_2. Clearly, a Σ_2-set is a finite union of a finite intersection of elementary sets.

Our goal is to define $\widetilde{Fp}(B)$ as a Σ_2-set. Since $\Pi_1 \subsetneq \Sigma_2$, this can be regarded as a generalization of Π_1-sets. Why do we need this generalization? Good approximations of footprints are harder to do accurately in 3D, and the extra power of Σ_2 is very useful. To see the power of Σ_2-sets, note that in 2D, when R_0 is a disc, the footprint of a box can only be approximated by a Π_1-set (we use a disc as approximation), but this footprint has an *exact* description as a Σ_2-set: it is the union of 4 discs and an octagon. The same is true when R_0 is a planar rod. However, there is a complexity-accuracy trade-off between using approximate Π_1-set and exact Σ_2-set. This tradeoff should be studied empirically as the "practically efficient" criteria of (G3). Note that a Σ_2-set S has a description of the form

$$S = \bigcup_{i=1}^{n} \bigcap_{j=1}^{m_i} S_{ij} \tag{4}$$

for some n and m_i's, and $S_{i,j}$ are elementary sets. We call (4) a **Σ_2-decomposition** of S. Note that this decomposition is by no means unique, but in the simple cases that arise in our "simple robots", there is often an obvious optimal description. Moreover, n and m_i's are small constants. We can construct new sets by manipulating such a decomposition: For instance, replace each S_{ij} by its τ-expansion i.e., $S_{ij} \oplus B(\tau)$ which is remains elementary. Under certain conditions, the corresponding set is a reasonable approximation to $S \oplus B(\tau)$. If so, we can extend generalize the corresponding soft predicate to robots with thickness τ.

Once we have a Σ_2-description of $\widetilde{Fp}(B)$, we can implement the intersection test with relative ease (G2) and quite efficiently (G3). For instance we can test intersection of the set S in (4) with a feature f by writing a doubly nested loop. At the beginning of the inner loop, we can initialize a set f_0 to f. Then the inner loop amounts to the update "$f_0 \leftarrow f_0 \cap S_{ij}$". If ever f_0 becomes empty, we know that the set $S_i = \bigcap_j^m S_{ij}$ has empty intersection with f. The possibility of such representations is by no means automatic but the next two sections verify that they can be achieved for our rod and ring robots. These sections makes our planners fully "explicit" for an implementation.

5 Soft Predicates for Rod Robot

In this section, R_0 is a rod with length r_0. Let $B = B^t \times B^r \subseteq \mathbb{R}^3 \times \widehat{S^2}$ be a box. Our main goal is to define the set $\widetilde{Fp}(B)$, approximate footprint of B, and to prove the basic inclusions in Eqs. (1) and (2). This turns out to be a Π_1-set (we also indicate a more accurate Σ_2-set.)

It is useful to define the **inner footprint** of B, denoted $Fp_0(B) := Fp(m_B \times B^r)$.

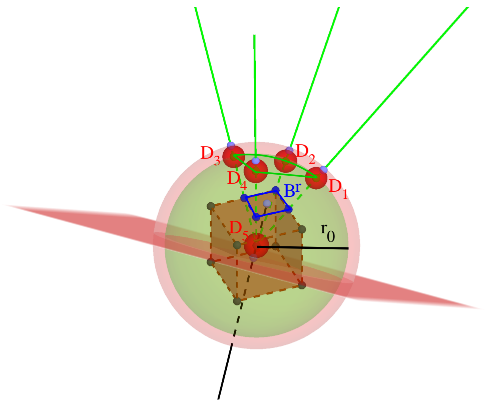

Figure 3 Rod Robot: $Fp(B) = Fp_0(B) \oplus (B^t - m_B)$ where $Fp_0(B)$ is indicated by the four green rays.

This set is the intersection of a ball and a square cone:

$$Ball(r_0, m_B) \cap Cone(m_B, B^r + m_B) \tag{5}$$

The edges of this square cone is shown as green lines in Figure 3; furthermore, the brown box is a copy of $\widehat{S^2} + m_B$ (translation of $\widehat{S^2}$ so that it is centered at m_B). Notice that the box footprint $Fp(B)$ is the Minkowski sum of $Fp_0(B)$ with $B^t - m_B$, the translation of B^t so that it becomes centered at the origin. It is immediate that

$$Fp_0(B) \subseteq Cone(B).$$

Thus we may write $Cone(m_B, B^r + m_B)$ as the intersection of four half spaces H_i ($i = 1, \ldots, 4$). Let $Cone^{(+r_B)}(m_B, B^r + m_B)$ denote the intersection of the expanded half-spaces, $H_i \oplus Ball(r_B)$ ($i = 1, \ldots, 4$). In general, $Cone^{(+r_B)}(m_B, B^r + m_B)$ is not a cone (it may not have a unique "apex"). Similarly we "expand" the inner footprint of (5) into

$$\text{“}\widetilde{Fp}(B)\text{”} := Ball(r_0 + r_B, m_B) \cap Cone^{(+r_B)}(m_B, B^r + m_B). \tag{6}$$

We use quotes for "$\widetilde{Fp}(B)$" in (6) because we view it as a candidate for an approximate footprint of B. Certainly, it has the desired property of containing the exact footprint $Fp(B)$. Unfortunately, this is not good enough. To see this, let θ be the half-angle of the round cone $Cone(B) = Cone(m_B, Ball(B^r + m_B))$. Then Hausdorff distance of "$\widetilde{Fp}(B)$" from $Fp(B)$ can be arbitrarily big as θ becomes arbitrarily small. Indeed θ can arbitrarily as small because it can be proportional to the input resolution ε. We conclude that such a planner is not resolution-exact.

To fix this problem this, we finally define

$$\widetilde{Fp}(B) := \text{“}\widetilde{Fp}(B)\text{”} \cap H_0$$

where H_0 is another half space. A natural choice for H_0 is the half-space "above" the pink-color plane of Figure 3, defined as the plane normal to the axis of cone $Cone(B)$ and at distance r_B "below" m_B. Unfortunately, it is harder to prove inclusion properties with this choice of H_0. Instead of the pink plane, we use the "horizontal" plane that is parallel to B^r and containing the "lower" face of B^t.

This completes the description of $\widetilde{Fp}(B)$. It should be clear that checking if $\widetilde{Fp}(B)$ intersects any feature f is relatively easy (since it is even a Π_1-set). The two lemmas below show the inclusion properties in Eqs. (1) and (2) (see Appendix D for proofs).

▶ **Lemma 2.** *(Inheritance)*
Let B_1 be a child of B. Then $\widetilde{Fp}(B_1) \subseteq \widetilde{Fp}(B)$. (This means that we can obtain $\widetilde{\phi}(B_1)$ by distributing the features from the parent feature set $\widetilde{\phi}(B)$ to the children.)

▶ **Lemma 3.** *There exists some fixed constant $\sigma > 1$ such that $\widetilde{Fp}(B/\sigma) \subseteq Fp(B)$.*

6 Soft Predicates for Ring Robot

Let R_0 be a ring robot. Its canonical footprint is an embedded circle with radius r_0. First we show how to compute $\mathrm{Sep}(C, f)$, the separation of an embedded circle C from a feature f. This was treated in detail by Eberly [10]. This is easy when f is a point or a plane. When f is a line, Eberly gave two formulations, and suggests the need to do root solving for a system of 2 quadratic equations in 2 variables which reduces to a quartic equation. We derive these equations in Appendix E (E.1). The predicate "Does f intersect $C \oplus Ball(r')$, a ring of thickness r'?" is needed later; it reduces to "Is $\mathrm{Sep}(C, f) \le r'$?".

Our next task is to describe an approximate footprint, First recall the round cone of box B defined in the previous section: $Cone(B) = Cone(m_B, Ball(m_B + B^r))$. Let $\theta = \theta(B)$ be the half-angle of this cone, and c the center of B^r. Here, we think of c as a point of $\widehat{S^2}$, and define $\gamma(B) := m_B \times c$ viewed as an element of $\mathbb{R}^3 \times \widehat{S^2}$. Call $\gamma(B)$ the **central configuration** of box B. Let $Ray(B)$ be the axis (a ray from m_B through $m_B + c$). If $Plane(B)$ is the plane through m_B and normal to $Ray(B)$, then clearly the footprint $Fp(\gamma(B))$ is an embedded circle lying in $Plane(B)$. Again we define the **inner footprint** of B as $Fp_0(B) := Fp(m_B \times B^r)$. Unfortunately, it is hard to work with $Fp_0(B)$. Since map $q \mapsto \overline{q}$ is the inverse of $q \mapsto \widehat{q}$, and $c \in \widehat{S^2}$, we have $\overline{c} \in S^2$. Instead consider the set $D(B)$ of all points in S^2 whose distance[4] from $\overline{c}$ is at most $\theta(B)$. Clearly $D(B)$ is the intersection of S^2 with a round cone with axis from the origin to c. We then define $Fp_1(B)$ as

$$Fp_1(B) := Fp(m_B \times D(B)).$$

It is clear that $Fp_0(B) \subseteq Fp_1(B)$. Our main computational interest is the approximate

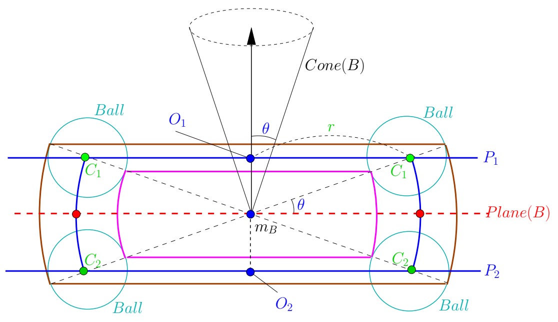

Figure 4 Ring Robot: central cross-section of $Fp_1(B)$ appears as two blue arcs. $\widetilde{Fp}(B)$ equals the union of two "thick rings" and a "truncated annulus". The axis of $Cone(B)$ is shown as a vertical ray. Each $Ball$ has radius r_B.

footprint of B defined as

$$\widetilde{Fp}(B) := Fp_1(B) \oplus Ball(r_B).$$

[4] Recall that S^2 is a metric space whose geodesics are arcs of great circles.

Note that $Fp_1(B)$ has a simple geometric description. We illustrate this in Figure 4 using a central cross-section with a plane through m_B containing the axis of $Cone(B)$ (the axis of $Cone(B)$ is drawn vertically). The footprint of $\gamma(B)$ is a circle that appears as two red dots in the horizontal line (i.e., $Plane(B)$). Let $S^2(m_B, r_0)$ denote the 2-sphere centered at m_B with radius r_0. Then $Fp_1(B)$ is the intersection of $S^2(m_B, r_0)$ with a slab (i.e., intersection of two half-spaces whose bounding planes P_1 and P_2 are parallel to $Plane(B)$). These planes are seen as two horizontal blue lines in Figure 4. In the cross section, $Fp_1(B)$ appears as two blue circular arcs. For $i = 1, 2$, let $C_i = P_i \cap S^2(m_B, r_0)$; it is an embedded circle that appears as a pair of green points in Figure 4. Each C_i is centered at O_i, with radius $r = r_0 \cos\theta$; see Figure 4.

We can now describe a Σ_2-decomposition of $\widetilde{Fp}(B)$: it can be written as the union of two ”thick rings”, $C_1 \oplus Ball(r_B)$ and $C_2 \oplus Ball(r_B)$ (each with thickness radius r_B), and a shape $Ann(B)$ which we call a **truncated annulus**. First of all, the region bounded between the spheres $S^2(m_B, r_0 + r_B)$ (the brown arcs in the figure) and $S^2(m_B, r_0 - r_B)$ (the magenta arcs) is called a (solid) annulus. Let C_i^* denote the embedded disc whose relative boundary is C_i. Then we have two round cones, $Cone(m_B, C_1^*)$ and $Cone(m_B, C_2^*)$. Together, they form a *double cone* that is actually a simpler object for computation! Finally, define $Ann(B)$ to be the intersection of the annulus with the complements of the double cone.

Note that for each thick ring $C_i \oplus Ball(r_B)$, we need to decide “Does f intersect $C_i \oplus Ball(r_B)$?” for a feature f, which is equivalent to “Is $\mathrm{Sep}(C_i, f) \le r_B$?” as mentioned at the beginning of this section. In Appendix E (E.1) we discuss how to compute $\mathrm{Sep}(C_i, f)$, and in Appendix E (E.2) we prove:

▶ **Theorem 4.** *$\widetilde{Fp}(B)$ satisfies the inclusion properties in Eqs. (1) and (2). Namely:*
(a) (Inheritance) Let B_1 be a child of B. Then $\widetilde{Fp}(B_1) \subseteq \widetilde{Fp}(B)$.
(b) There exists some fixed constant $\sigma > 1$ such that $\widetilde{Fp}(B/\sigma) \subseteq Fp(B)$.

7 Practical Efficiency of Correct Implementations

We have developed ε-exact planners for rod and ring robots. We have explicitly exposed all the details necessary for a correct implementation, i.e., criterion (G1). The careful design of the approximate footprints of boxes as Σ_2-sets ensures (G2), i.e., it would be relatively easy to implement. We now address (G3) or practical efficiency. For robots with 5 or more DOFs, it becomes extremely critical that some good search strategies are deployed. In this paper, we have found that some form of the Voronoi heuristic is extremely effective. The basic idea is to try to move along the Voronoi curves, and we could use our Voronoi algorithms based on our method of features [35, 3]. There are difficulties including the inability to guarantee the curves' existence ([19, 29, 30]), but we do not rely on Voronoi diagrams for our correctness, and various expedients are available. To recognize Voronoi curves, we maintain (in addition to the collision-detection feature set $\widetilde{\phi}(B)$), the **Voronoi feature set** $\widetilde{\phi}_V(B)$. These two sets have some connection but there are no obvious inclusion relationships.

Our current implementation achieves near real-time performance (see video http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html). Table 1 summarizes some experiments. The three environments: Rand100, Rand40 (100 and 40 random tetrahedra), and 2 Posts (two L-shaped posts), are shown in Figs. 1, 6 and 5 (the latter two in Appendix A). In the final paper, we expect to report improved timings with better search strategies.

Rod Robot							
Environment	length	epsilon	Start configuration ($\mathbb{R}^3 \times \widehat{S^2}$)	Goal configuration	Path	Time (s)	#Box
Rand100	120	16	(400, 480, 80, -0.5, -1.0, 0.5)	(50, 80, 450, 0.5, 0.5, -1.0)	Y	38.92	67.55k
Rand40	80	16	(320, 200, 400, 1.0, 0.0, -0.5)	(80, 450, 100, 0.2, -1.0, 0.2)	Y	6.89	27.34k
2 Posts	60	16	(200, 480, 190, 0.0, 0.1,-1.0)	(390, 180, 320, 1.0, -0.1, 0.0)	Y	0.217	1.6k
Ring Robot							
Environment	radius	epsilon	Start configuration	Goal configuration	Path	Time (s)	#Box
Rand100	40	16	(400, 480, 80, -0.5, -1.0, 0.5)	(50, 80, 450, 0.5, 0.5, -1.0)	Y	0.304	1.30k
Rand40	60	16	(320, 200, 400, 1.0, 0.0, -0.5)	(80, 450, 100, 0.2, -1.0, 0.2)	Y	5.301	14.23k
2 Posts	60	32	(200, 380, 190, 0.0, 0.1, -1.0)	(390, 380, 190, 1.0, -0.1, 0.0)	Y	8.405	14.97k

Table 1 Rod and Ring Experiments.

Our implementation uses `C++` and OpenGL on the Qt platform. Our code, data and experiments are distributed[5] with our open source `Core Library`. We ran our experiments on a MacBook Pro under Mac OS X 10.10.5 with a 2.5 GHz Intel Core i7 processor, 16GB DDR3-1600 MHz RAM and 500GB Flash Storage. Details about these experiments are found in a folder in `Core Library` for this paper; a `Makefile` there can automatically run all the experiments. Thus these results are reproducible from the data there.

Correct Implementation of Soft Exact Algorithms
We have provided an "exact" description of planners for a rod and a ring, albeit a "soft kind". We claim that all our computations can be guaranteed in the soft sense. This is possible because all the inequalities in our algorithms are "one-sided" in the sense that we do not assume that the failure of an inequality test implies the complementary condition (as in exact (unqualified) computation). More detailed discussions are in Appendix F.

8 Conclusion

The world of computing with real numbers is a severe challenge for theoretical computer science which is extremely good with discrete algorithms but has largely left "continuum computing" to numerical analysts. Our work offers a small window into this larger challenge.

Our 5-DOF spatial robots are pushing current limits for subdivision methods. But how good is "5"? To our knowledge there is no similar subdivision algorithm with comparable rigor. (The spatial robot in [40] had 4 DOFs.) But how does "5" compare to sampling methods? Conventional wisdom says that sampling methods can achieve higher DOFs than subdivision. By an estimate of Choset et al [6, p. 202], state-of-the-art sampling methods are limited to $5-12$ DOFs. If we take the lower limit of this estimate, we may be catching up. We believe that it should be possible for us to reach 6 DOFs for spatial robots.

Since resolution-exactness delivers much stronger guarantees than probabilistic-completeness, we might expect a performance hit compared to sampling methods. Our experimental work so far [33, 38, 39, 21] has seen no such tradeoffs: we consistently outperform state-of-the-art sampling methods (such as the publicly available OMPL [32]), often by two orders of magnitude. We have no reason to believe that subdivision is inherently inferior to sampling because we can also do random subdivision with good global control on the process.

[5] http://cs.nyu.edu/exact/core/download/core/.

References

1 J. Basch, L.J. Guibas, D. Hsu, and A. Nguyen. Disconnection proofs for motion planning. In *IEEE Int'l Conf. on Robotics Animation*, pages 1765–1772, 2001.

2 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer, 2nd edition, 2006.

3 Huxley Bennett, Evanthia Papadopoulou, and Chee Yap. Planar minimization diagrams via subdivision with applications to anisotropic Voronoi diagrams. *Eurographics Symposium on Geometric Processing*, 35(5), 2016. SGP 2016, Berlin, Germany. June 20-24, 2016.

4 Rodney A. Brooks and Tomas Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. 8th Intl. Joint Conf. on Artificial intelligence - Volume 2*, pages 799–806, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.

5 John Canny. Computing roadmaps of general semi-algebraic sets. *The Computer Journal*, 36(5):504–514, 1993.

6 H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, 2005.

7 Howie Choset, Brian Mirtich, and Joel Burdick. Sensor based planning for a planar rod robot: Incremental construction of the planar Rod-HGVG. In *IEEE Intl. Conf. on Robotics and Automation (ICRA'97)*, pages 3427–3434, 1997.

8 James Cox and Chee K. Yap. On-line motion planning: case of a planar rod. *Annals of Mathematics and Artificial Intelligence*, 3:1–20, 1991. Special journal issue. Also: NYU-Courant Institute, Robotics Lab., No.187, 1988.

9 Jory Denny, Kensen Shi, and Nancy M. Amato. Lazy Toggle PRM: a Single Query approach to motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2407–2414, 2013. Karlsrube, Germany. May 2013.

10 David Eberly. Distance to circles in 3D, May 31 2015. Downloaded from `https://www.geometrictools.com/Documentation/Documentation.html`.

11 Mohab Safey el Din and Eric Schost. A baby steps/giant steps probabilistic algorithm for computing roadmaps in smooth bounded real hypersurface. *Discrete and Comp. Geom.*, 45(1):181–220, 2011.

12 Hazel Everett, Christian Gillot, Daniel Lazard, Sylvain Lazard, and Marc Pouget. The Voronoi diagram of three arbitrary lines in $\mathbb{R}^3$. In *25th European Workshop on Computational Geometry (EuroCG'09)*, 2009. Mar 2009, Bruxelles, Belgium.

13 Hazel Everett, Daniel Lazard, Sylvain Lazard, and Mohab Safey el Din. The Voronoi diagram of three lines. *Discrete and Comp. Geom.*, 42(1):94–130, 2009. See also 23rd SoCG, 2007. pp.255–264.

14 Dan Halperin, Efi Fogel, and Ron Wein. *CGAL Arrangements and Their Applications*. Springer-Verlag, Berlin and Heidelberg, 2012.

15 Dan Halperin, Oren Salzman, and Micha Sharir. Algorithmic motion planning. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017. To appear, expanded from second edition.

16 Michael Hemmer, Ophir Setter, and Dan Halperin. Constructing the exact Voronoi diagram of arbitrary lines in three-dimensional space. In *Algorithms – ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 398–409. Springer Berlin / Heidelberg, 2010.

17 V. Koltun. Pianos are not flat: rigid motion planning in three dimensions. In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, pages 505–514, 2005.

18 Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006.

19 Ji Yeong Lee and Howie Choset. Sensor-based planning for a rod-shaped robot in 3 dimensions: Piecewise retracts of r3 x s2. *Int'l. J. Robotics Research*, 24(5):343–383, 2005.

20 Long Lin, Chee Yap, and Jihun Yu. Non-local isotopic approximation of nonsingular surfaces. *Computer-Aided Design*, 45(2):451–462, October 2012. Symp. on Solid and Physical Modeling (SPM). U. of Burgundy, Dijon, France, Oct 29-31, 2012.

21 Zhongdi Luo, Yi-Jen Chiang, Jyh-Ming Lien, and Chee Yap. Resolution exact algorithms for link robots. In *Proc. 11th Intl. Workshop on Algorithmic Foundations of Robotics (WAFR '14)*, volume 107 of *Springer Tracts in Advanced Robotics (STAR)*, pages 353–370, 2015. 3-5 Aug 2014, Boğazici University, Istanbul, Turkey.

22 James R. Munkres. *Topology*. Prentice-Hall, Inc, second edition, 2000.

23 Michal Nowakiewicz. MST-Based method for 6DOF rigid body motion planning in narrow passages. In *Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems*, pages 5380–5385, 2010. Oct 18–22, 2010. Taipei, Taiwan.

24 Colm Ó'Dúnlaing, Micha Sharir, and Chee K. Yap. Retraction: a new approach to motion-planning. *ACM Symp. Theory of Comput.*, 15:207–220, 1983.

25 Colm Ó'Dúnlaing and Chee K. Yap. A "retraction" method for planning the motion of a disc. *J. Algorithms*, 6:104–111, 1985. Also, Chapter 6 in *Planning, Geometry, and Complexity*, eds. Schwartz, Sharir and Hopcroft, Ablex Pub. Corp., Norwood, NJ. 1987.

26 J. T. Schwartz and M. Sharir. On the piano movers' problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

27 Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Appl. Math.*, 4:298–351, 1983.

28 Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: V. the case of a rod moving in three-dimensional space amidst polyhedral obstacles. *Comm. Pure and Applied Math.*, 37(6):815–848, 1984. `doi:DOI:10.1002/cpa.3160370605`.

29 M. Sharir, C. O'D'únlaing, and C. Yap. Generalized Voronoi diagrams for moving a ladder I: topological analysis. *Communications in Pure and Applied Math.*, XXXIX:423–483, 1986. Also: NYU-Courant Institute, Robotics Lab., No. 32, Oct 1984.

30 M. Sharir, C. O'D'únlaing, and C. Yap. Generalized Voronoi diagrams for moving a ladder II: efficient computation of the diagram. *Algorithmica*, 2:27–59, 1987. Also: NYU-Courant Institute, Robotics Lab., No. 33, Oct 1984.

31 Vikram Sharma and Chee K. Yap. Robust geometric computation. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 46, pages 927–952. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017 (to appear). Revised and expanded from 2004 version.

32 I.A. Şucan, M. Moll, and L.E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012. `http://ompl.kavrakilab.org`. `doi:10.1109/MRA.2012.2205651`.

33 Cong Wang, Yi-Jen Chiang, and Chee Yap. On Soft Predicates in Subdivision Motion Planning. *Comput. Geometry: Theory and Appl. (Special Issue for SoCG'13)*, 48(8):589–605, September 2015.

34 Chee Yap, Zhongdi Luo, and Ching-Hsiang Hsu. Resolution-exact planner for thick non-crossing 2-link robots. In *Proc. 12th Intl. Workshop on Algorithmic Foundations of Robotics (WAFR '16)*, 2016. 13-16 Dec 2016, San Francisco. The appendix in the full paper (and arXiv from `http://cs.nyu.edu/exact/` (and `arXiv:1704.05123 [cs.CG]`) contains proofs and additional experimental data.

35 Chee Yap, Vikram Sharma, and Jyh-Ming Lien. Towards Exact Numerical Voronoi diagrams. In *9th Int'l Symp. of Voronoi Diagrams in Science and Engineering (ISVD).*, pages 2–16. IEEE, 2012. Invited Talk. June 27-29, 2012, Rutgers University, NJ.

36 Chee K. Yap. Algorithmic motion planning. In J.T. Schwartz and C.K. Yap, editors, *Advances in Robotics, Vol. 1: Algorithmic and geometric issues*, volume 1, pages 95–143. Lawrence Erlbaum Associates, 1987.

37 Chee K. Yap. In praise of numerical computation. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, volume 5760 of *Lect. Notes in C.S.*, pages 308–407. Springer-Verlag, 2009.

38 Chee K. Yap. Soft Subdivision Search in Motion Planning. In A. Aladren et al., editor, *Proceedings, 1st Workshop on Robotics Challenge and Vision (RCV 2013)*, 2013. A Computing Community Consortium (CCC) **Best Paper Award**, Robotics Science and Systems Conference (RSS 2013), Berlin. In arXiv:1402.3213.

39 Chee K. Yap. Soft Subdivision Search and Motion Planning, II: Axiomatics. In *Frontiers in Algorithmics*, volume 9130 of *Lecture Notes in Comp.Sci.*, pages 7–22. Springer, 2015. Plenary Talk at 9th FAW. Guilin, China. Aug 3-5, 2015.

40 Liangjun Zhang, Young J. Kim, and Dinesh Manocha. Efficient cell labeling and path non-existence computation using C-obstacle query. *Int'l. J. Robotics Research*, 27(11–12):1246–1257, 2008.

41 D.J. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991.

A Appendix: Additional Images from Experiments

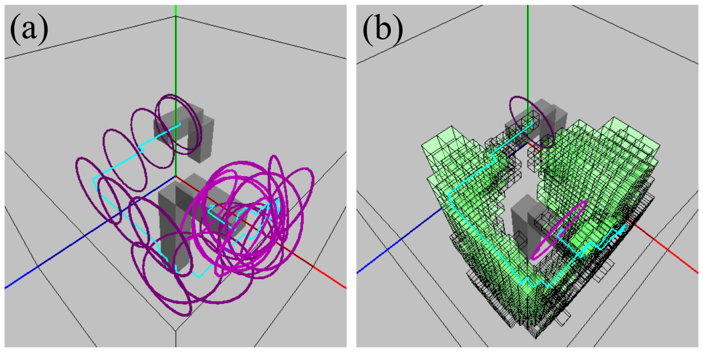

Figure 5 Ring robot amidst two posts: (a) trace of a found path; (b) subdivision of translational boxes on the path.

B Appendix: Elements of Soft Subdivision Search

This section is only for the convenience of the reviewers, and will be omitted in the final paper.

We review the the notion of soft predicates and how it is used in the SSS Framework. See [33, 38, 21] for more details.

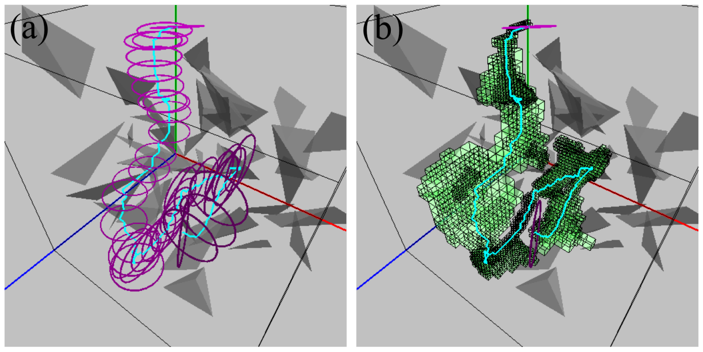

Figure 6 Ring robot amidst 40 random tetrahedra: (a) trace of a found path; (b) subdivision of translational boxes on the path.

B.1 Soft Predicates

The concept of a "soft predicate" is relative to some exact predicate. Define the exact predicate $C : C_{space} \to \{0, +1, -1\}$ where $C(x) = 0/+1/-1$ (resp.) if configuration x is semi-free/free/stuck. The semi-free configurations are those on the boundary of C_{free}. Call $+1$ and -1 the **definite values**, and 0 the **indefinite value**. Extend the definition to any set $B \subseteq C_{space}$: for a definite value v, define $C(B) = v$ iff $C(x) = v$ for all x. Otherwise, $C(B) = 0$. Let $\square(C_{space})$ denote the set of d-dimensional boxes in C_{space}. A predicate $\widetilde{C} : \square(C_{space}) \to \{0, +1, -1\}$ is a **soft version of** C if it is conservative and convergent. **Conservative** means that if $\widetilde{C}(B)$ is a definite value, then $\widetilde{C}(B) = C(B)$. **Convergent** means that if for any sequence $(B_1, B_2, \ldots)$ of boxes, if $B_i \to p \in C_{space}$ as $i \to \infty$, then $\widetilde{C}(B_i) = C(p)$ for i large enough. To achieve resolution-exact algorithms, we must ensure $\widetilde{C}$ converges quickly in this sense: say $\widetilde{C}$ is **effective** if there is a constant $\sigma > 1$ such if $C(B)$ is definite, then $\widetilde{C}(B/\sigma)$ is definite.

B.2 The Soft Subdivision Search Framework

An SSS algorithm maintains a subdivision tree $\mathcal{T} = \mathcal{T}(B_0)$ rooted at a given box B_0. Each tree node is a subbox of B_0. We assume a procedure $\mathtt{S}plit(B)$ that subdivides a given leaf box B into a bounded number of subboxes which becomes the children of B in $\mathcal{T}$. Thus B is "expanded" and no longer a leaf. For example, $\mathtt{S}plit(B)$ might create 2^d congruent subboxes as children. Initially $\mathcal{T}$ has just the root B_0; we grow $\mathcal{T}$ by repeatedly expanding its leaves. The set of leaves of $\mathcal{T}$ at any moment constitute a subdivision of B_0. Each node $B \in \mathcal{T}$ is classified using a soft predicate $\widetilde{C}$ as $\widetilde{C}(B) \in \{\mathtt{MIXED}, \mathtt{FREE}, \mathtt{STUCK}\} = \{0, +1, -1\}$. Only MIXED leaves with radius $\geq \varepsilon$ are candidates for expansion. We need to maintain three auxiliary data structures:

- A priority queue Q which contains all candidate boxes. Let $Q.\mathtt{GetNext}()$ remove the box of highest priority from Q. The tree $\mathcal{T}$ grows by splitting $Q.\mathtt{GetNext}()$.
- A **connectivity graph** G whose nodes are the FREE leaves in $\mathcal{T}$, and whose edges connect pairs of boxes that are adjacent, i.e., that share a $(d-1)$-face.
- A Union-Find data structure for connected components of G. After each $\mathtt{S}plit(B)$, we update G and insert new FREE boxes into the Union-Find data structure and perform unions of new pairs of adjacent FREE boxes.

Let $Box_{\mathcal{T}}(\alpha)$ denote the leaf box containing α (similarly for $Box_{\mathcal{T}}(\alpha)$). The SSS Algorithm has three WHILE-loops. The first WHILE-loop will keep splitting $Box_{\mathcal{T}}(\alpha)$ until it becomes `FREE`, or declare NO-PATH when $Box_{\mathcal{T}}(\alpha)$ has radius less than ε. The second WHILE-loop does the same for $Box_{\mathcal{T}}(\beta)$. The third WHILE-loop is the main one: it will keep splitting Q.`GetNext`() until a path is detected or Q is empty. If Q is empty, it returns NO-PATH. Paths are detected when the Union-Find data structure tells us that $Box_{\mathcal{T}}(\alpha)$ and $Box_{\mathcal{T}}(\beta)$ are in the same connected component. It is then easy to construct a path. Thus we get:

```
SSS Framework:
    Input:  Configurations $\alpha, \beta$, tolerance $\varepsilon > 0$, box $B_0 \in C_{space}$.
    Output:  Path from $\alpha$ to $\beta$ in $Fp(R_0, \Omega) \cap B_0$.
      Initialize a subdivision tree $\mathcal{T}$ with root $B_0$.
      Initialize $Q, G$ and union-find data structure.
      While ($Box_{\mathcal{T}}(\alpha) \neq$ FREE)
         If radius of $Box_{\mathcal{T}}(\alpha)$) is $< \varepsilon$, Return(NO-PATH)
         Else Split($Box_{\mathcal{T}}(\alpha)$)
      While ($Box_{\mathcal{T}}(\beta) \neq$ FREE)
         If radius of $Box_{\mathcal{T}}(\beta)$) is $< \varepsilon$, Return(NO-PATH)
         Else Split($Box_{\mathcal{T}}(\beta)$)
      ▷ MAIN LOOP:
      While ($Find(Box_{\mathcal{T}}(\alpha)) \neq Find(Box_{\mathcal{T}}(\beta))$)
         If $Q_{\mathcal{T}}$ is empty, Return(NO-PATH)
         $B \leftarrow Q_{\mathcal{T}}$.GetNext()
         Split($B$)
      Generate and return a path from $\alpha$ to $\beta$ using $G$.
```

The correctness of our algorithm does not depend on how the priority of Q is designed. See [38] for the correctness of this framework under very general conditions. But for our implementations, especially in 3-D, it is imperative to some clever techniques for Q.`GetNext`(). In particular, some form of the Voronoi heuristic seems to be the most effective.

C Appendix: Properties of Square Models and Classifying a Box

C.1 Proof: Properties of Square Models

Lemma 1. $C_0 = \sqrt{3}$.

Proof. Let B be the ball whose boundary is S^2 and $C = [-1, 1]^3$. Then $B/\sqrt{3} \subseteq C \subseteq B$. From any geodesic α of S^2, we obtain a corresponding geodesic α' on the surface of $B/\sqrt{3}$, and a geodesic $\widehat{\alpha}$ of $\widehat{S^2} = \partial(C)$. Observe that $|\alpha'| \leq |\widehat{\alpha}| \leq |\alpha|$ where $|\cdot|$ is the length of a geodesic. But $|\alpha| = \sqrt{3}|\alpha'|$. This proves our bound. This bound is tight because for geodesic arcs in arbitrary small neighborhoods of the centers of the 6 faces of $\widehat{S^2}$, the bound is arbitrarily close to $\sqrt{3}$. **Q.E.D.**

C.2 Classifying a Box

In Sec. 4 we mentioned using soft predicates based on the "method of features" [33] to classify a box B. Recall that we classify B as `MIXED` when the feature set is non-empty; otherwise, we classify B as `FREE` or `STUCK`. Now we discuss how to classify B as `FREE` or `STUCK` when its feature set is empty. Suppose Ω is given as the union of a set of polyhedra that may overlap (this situation arises in Sec. 7). Let B' be the parent of B, then the feature set $\widetilde{\phi}(B')$ is non-empty. For each obstacle polyhedron P in $\widetilde{\phi}(B')$, we find the feature $f \subseteq \partial P$ closest to m_B and use f to decide whether m_B is outside P. Then m_B is outside Ω (and B is FREE) iff m_B is outside all such polyhedra P.

To find the feature $f \subseteq \partial P$ closest to m_B, we first find among the corners of P the one f_c that is the closest. Then among the edges of P incident on f_c, we check if there exist edges e that are even closer (i.e., $\mathrm{Sep}(e, m_B) < ||f_c - m_B||$; $\mathrm{Sep}(e, m_B)$ is defined by a point interior to e) and if so pick the closest one f_e. Finally, if f_e exists, we repeat the process for faces of P incident on f_e and pick the closest one f_w (if it exists). The closest feature f is set to f_c then updated to f_e and to f_w accordingly if f_e (resp. f_w) exists.

Given the feature $f \subseteq \partial P$ closest to m_B, we can easily determine if m_B is interior or exterior of P when f is a wall or an edge. When f is a corner, it is slightly more involved. We will classify a corner f to be **pseudo-convex** (resp., **pseudo-concave**) if there exists a closed half space H such that (1) $f \in \partial H$, and (2) for any small enough ball Δ centered at f, we have that $(H \cap P \cap \Delta) = f$ (resp., $H \cap \Delta \subseteq P \cap \Delta$). Note that if f is locally convex (resp., locally concave) then it is pseudo-convex (resp., pseudo-concave). We call a corner f an **essential corner** if for all balls Δ centered at f, $\Delta \cap \partial P$ is not a planar set. We may assume that our corners are essential; as consequence, no corner can be both pseudo-convex and pseudo-concave. However, it is possible that a corner is neither pseudo-convex nor pseudo-concave; we call such corners **mixed**. The lemma below enables us to avoid the difficulty of mixed corners.

▶ **Lemma 5.** *Let $q \notin \partial P$ and C a corner of P. If C is the point in ∂P closest to q, i.e., $\mathrm{Sep}_{\partial P}(q) = \|q - C\|$, then C is either pseudo-convex or pseudo-concave. Hence C cannot be a mixed corner. Moreover, $q \in P$ iff C is pseudo-concave.*

Proof. Let Δ be the ball centered at q with radius $\|q - C\|$. Since $\mathrm{Sep}_{\partial P}(q) = \|q - C\|$, we have $\Delta \cap \partial P = \{C\}$. Let H be the closed half-space such that ∂H is tangential to Δ at the point C, and $q \notin H$. This H is a witness to either the pseudo-convexity or pseudo-concavity of C. In particular, C is pseudo-concave iff $q \in P$. **Q.E.D.**

D Appendix: Soft Predicate for a Rod — Proof of Properties

Lemma 2. *(Inheritance)*
Let B_1 be a child of B. Then $\widetilde{Fp}(B_1) \subseteq \widetilde{Fp}(B)$. (This means that we can obtain $\widetilde{\phi}(B_1)$ by distributing the features from the parent feature set $\widetilde{\phi}(B)$ to the children.)

Proof. Let $B = B^t \times B^r$ and $B_1 = B_1^t \times B_1^r$. We consider two cases: (1) B_1^t is a child of B^t but their rotational boxes are the same, and (2) B_1^r is a child of B^r but their translational boxes are the same. For (1), their inner footprints $Fp_0(B)$ and $Fp_0(B_1)$ (each a green square cone intersected by the green ball in Fig. 3) are of the same shape and orientation; only the positions are shifted: each has the apex at its own translational box center. The apices (so are all other corresponding corners where red balls D_i in Fig. 3 are centered) are apart by a distance of r_{B_1}; the red balls D_i of B_1 are of radius r_{B_1}, but for the parent B the red-ball radius is twice as large ($r_B = 2r_{B_1}$). Thus each red ball of the child is entirely contained in the corresponding red ball of the parent, and thus $\widetilde{Fp}(B_1) \subseteq \widetilde{Fp}(B)$.

For (2), now their inner footprints have the same apex at m_B, and their red balls D_i have the same radius r_B. Since the child angular range B_1^r is a subset of the parent angular range, the green square cone (Fig. 3) of the child is entirely contained in the green square cone of the parent. Thus for "$\widetilde{Fp}(B)$", i.e., before bounding by H_0 to get $\widetilde{Fp}(B)$ (recall that $\widetilde{Fp}(B) :=$ "$\widetilde{Fp}(B)$" $\cap H_0$; see also (6)), the desired containment property holds. The bottom part bounded by H_0 has no problem either, since their horizontal planes H_0 are the same. (Note that if we replace H_0 by the pink plane in Fig. 3 then it will be tilted differently since the square cone axis is oriented differently for B_1 and B.) **Q.E.D.**

Lemma 3. *There exists some fixed constant $\sigma > 1$ such that $\widetilde{Fp}(B/\sigma) \subseteq Fp(B)$.*

Proof. The idea is to first use a "nice" shape to contain $\widetilde{Fp}(B)$, and then show that we can shrink this nice shape by a factor of some fixed constant $\sigma > 1$ such that it is contained in $Fp(B)$. Let c be the center of B^r. Clearly the round cone $Cone_{round} := Cone(m_B, Ball(m_B + B^r)$ contains the square cone $Cone_{square} := Cone(m_B, B^r + m_B)$, and thus $V := Cone_{round} \cap Ball(r_o, m_B)$ contains $Cone_{square} \cap Ball(r_o, m_B) = Fp_0(B)$. Recall that $\widetilde{Fp}(B) =$ "$\widetilde{Fp}$(B)$'' \cap H_0$. Consider the point q on H_0 that is cut by "$\widetilde{Fp}(B)$" and is farthest from m_B. The distance between q and m_B depends on the orientation of the square/round cone axis (going through m_B and c). The maximum happens when the axis goes from the center to the corner of the brown box in Fig. 3, making an angle of $arcsin(1/\sqrt{3})$. Since the distance between m_B and H_0 is r_B, this maximum distance between q and m_B is $\sqrt{3}r_B$. Therefore $\widetilde{Fp}(B)$ is contained in $V_{final} := V \oplus Ball(\sqrt{3}r_B)$. Also, $Cone_{round}/\sqrt{2}$ is contained in $Cone_{square}$. Note that $Fp_0(B) \oplus (B^t - m_B) = Fp(B)$, where $(B^t - m_B)$ contains $Ball(r_B/\sqrt{3})$. Now consider $V_{final}/3$: $V/3$ is contained in $Fp_0(B)$ and $Ball(\sqrt{3}r_B)/3 = Ball(r_B/\sqrt{3})$ is contained in $(B^t - m_B)$, and thus $V_{final}/3 \subseteq Fp(B)$. Overall, we have $\widetilde{Fp}(B/3)/ \subseteq V_{final}/3 \subseteq Fp(B)$. **Q.E.D.**

Note that the existence of such a constant σ is all we need to guarantee that our algorithm is resolution-exact; we do not need to know this constant in implementations.

E Appendix: Soft Predicate for a Ring

E.1 Computing the Separation Between a Circle and a Feature

As mentioned in Sec. 6, our soft predicates for the ring robot need to compute the separation of an embedded circle C from f, i.e., Sep(C, f), where f is a point, line or a plane.

In the following, let C be a circle of radius r centered at O, and lying in a plane P_C with normal vector n. Also let u be a vector along the direction of line L. Note that r, n, O, u are all given constants.

Simple Filtering

Before actually computing Sep(C, f), we can first perform a simple filtering. Recall from Sec. 6 that the purpose of Sep(C, f) is to decide "Is Sep$(C, f) \le r_B$?". If we have a simple way to know that Sep$(C, f) > r_B$ then there is no need to compute Sep(C, f). Here is how. Suppose f is a line or a plane. We can easily compute the separation d from the circle center O to f, i.e., $d =$ Sep(O, f). If $d > r + r_B$, then Sep$(C, f) \ge d - r > r_B$ and we are done. Only when $d \le r + r_B$ do we need to compute Sep(C, f), which can be much more complicated (see below).

Computing the Separation Sep(C, f)

The case where f is a point is trivial, and involves solving a quadratic equation. The case f is a plane is a rational problem: if f is parallel to P_C, then Sep(C, f) is just the separation between the two planes. Otherwise, let L' be the intersection of the two planes. Let $p \in C$ be the closest point in C to L', and q the projection of p to the plane f. Then $\mathrm{Sep}(C, f) = \|p - q\|$. (Note: if L' intersects C, then p is just any point in $L' \cap C$ and $p = q$ in this case.)

Finally, we address the most interesting case, where f is a line L defined by an obstacle edge. But before showing the exact computation of $\mathrm{Sep}(C, L)$, we show a relatively easy way to compute an upper bound, denoted $\mathrm{Sep}'(C, L)$, on $\mathrm{Sep}(C, L)$. We project the two edge endpoints p_1, p_2 onto the plane P_C to get p'_1, p'_2. First, assume $p'_1 \neq p'_2$ (non-degenerate case). Then any point in this projected line L' is expressed by $p'_1 + t(p'_2 - p'_1)$ with parameter t. Let p' be the point in L' closest to C; recall that O is the circle center. The corresponding point $p \in L$ that projects to p' has the same t as p. Then we compute $\mathrm{Sep}(C, p') := d$ from the radius and the distance between p' and O. Suppose q is the point on C closest to p'. Then define $\mathrm{Sep}'(C, L) := ||p - q||$. We can obtain $||p - q||$ without solving q, by the fact that q, p', p form a right triangle with leg lengths d and $||p - p'||$. We return to the degenerate case where $p'_1 = p'_2$. This means L is perpendicular to P_C, and $\mathrm{Sep}(C, L)$ is easily obtained. But numerically, whenever $\|p'_1 - p'_2\|$ is small, we ought to use this particular approximation. Since this is just a filter, we will not dwell on this.

Reduction of $\mathrm{Sep}(C, f)$ to Root-Finding

We now show how to reduce computing $\mathrm{Sep}(C, L)$ to solving quartic equations. Let p, q be the two points with $p \in C$ and $q \in L$ such that $\mathrm{Sep}(C, L) = \|p - q\|$. We can view $p = p(x, y, z)$ and $q = q(t)$ where x, y, z, t are variables to be solved.

We obtain four equations by the following conditions.

(A) The point p lies in the sphere centered at O of radius r:

$$\|p - O\| = r^2. \tag{7}$$

Explicitly, $(x - O_x)^2 + (y - O_y)^2 + (z - O_z)^2 = r^2$.

(B) The plane Opq is perpendicular to the plane of C:

$$((p - O) \times (q - O)) \cdot n = 0. \tag{8}$$

This equation is multilinear in t and in $\{x, y, z\}$. It has the form $tA(x, y, z) + B(x, y, z, t) + C = 0$ where A, B are linear in the indicated variables, and C is a constant.

(C) The line pq is perpendicular to L:

$$(p - q) \cdot u = 0. \tag{9}$$

This is a linear function in x, y, z, t.

(D) The (radius) line Op is perpendicular to n:

$$(p - O) \cdot n = 0. \tag{10}$$

This is a linear function in x, y, z.

Using Condition (D), we can express z as a linear function in x, y and plug into Eqs. of (A), (B), (C) to eliminate z without changing the nature of these equations (i.e., Eq. of (A) remains quadratic and Eq. of (B) remains multilinear). By using Condition (C) we can eliminate t from Eq. of (B) and turn it into a quadratic equation in x, y. So we now have a system of two quadratic equations in x, y:

$$\left.\begin{array}{rcl} ax^2 + bx + c & = & 0 \\ a'x^2 + b'x + c' & = & 0 \end{array}\right\} \tag{11}$$

where a, b, c (resp., a', b', c') are polynomials in y of degrees $0, 1, 2$ respectively. We obtain $x = \frac{-b\pm\sqrt{\Delta}}{2a} = \frac{-b'\pm\sqrt{\Delta'}}{2a'}$ where $\Delta = b^2 - 4ac$ and Δ' similarly. Thus

$$\begin{aligned}
a'(-b \pm \sqrt{\Delta}) &= a(-b' \pm \sqrt{\Delta'}) \\
A \pm a'\sqrt{\Delta} &= \pm a\sqrt{\Delta'} \qquad\qquad \text{where} \quad A = \det\begin{bmatrix} a & b \\ a' & b' \end{bmatrix} \\
\left(A \pm a'\sqrt{\Delta}\right)^2 &= a^2\Delta' \\
\pm 2a'A\sqrt{\Delta} &= a^2\Delta' - A^2 - (a')^2\Delta \\
(2a'A)^2\Delta &= \left(a^2\Delta' - A^2 - (a')^2\Delta\right)^2.
\end{aligned} \tag{12}$$

We summarize by restating the last equation:

$$(2a'A)^2\Delta = \left(a^2\Delta' - A^2 - (a')^2\Delta\right)^2. \tag{13}$$

This is a quartic equation in y, as claimed.

E.2 Proof of Properties

Theorem 4. *$\widetilde{Fp}(B)$ satisfies the inclusion properties in Eqs. (1) and (2). Namely:*
(a) (Inheritance) Let B_1 be a child of B. Then $\widetilde{Fp}(B_1) \subseteq \widetilde{Fp}(B)$.
(b) There exists some fixed constant $\sigma > 1$ such that $\widetilde{Fp}(B/\sigma) \subseteq Fp(B)$.
Proof.
Part (a). We would like to have the feature-set inheritance property from parent to children. But it seems not easy to prove it directly. Instead, we *enforce* this property *algorithmically.* For clarity, let us re-name $\widetilde{Fp}(B)$ defined in Sec. 6 as $\widetilde{Fp}'(B)$. As given, B_1 is a child of B. In computing the child feature set, we always take the parent feature set and test against $\widetilde{Fp}'(B_1)$ for the child B_1. This is equivalent to setting $\widetilde{Fp}(B_1) = \widetilde{Fp}'(B_1) \cap \widetilde{Fp}(B)$. The question is whether $\widetilde{Fp}(B_1)$ is still a superset of $Fp(B_1)$ as required in Eq. (1).
Claim: $Fp(B_1) \subseteq \widetilde{Fp}(B_1)$.
Proof of Claim: By construction $\widetilde{Fp}'(B_1)$ is a superset of $Fp(B_1)$. It suffices to show that $\widetilde{Fp}(B)$ is a superset of $Fp(B_1)$. But $\widetilde{Fp}(B)$ is a superset of $Fp(B)$ (initially and inductively), which in term is a superset of $Fp(B_1)$. This completes the proof of Claim. □

Part (b). When trying to prove $\widetilde{Fp}(B/\sigma) \subseteq Fp(B)$ for some fixed constant $\sigma > 1$, we can just use $\widetilde{Fp}'(B)$ and try to prove $\widetilde{Fp}'(B/\sigma) \subseteq Fp(B)$ without worrying about its intersection with $\widetilde{Fp}$(parent) since the intersection is a smaller set. Therefore, we only need to consider $\widetilde{Fp}'(B) = Fp_1(B) \oplus Ball(r_B)$. For $Fp(B)$, it is the Minkowski sum of $Fp_0(B)$ and a cube of radius r_B. The difference between $Fp_0(B)$ and $Fp_1(B)$ is the orientation of the cone axis, with the maximum difference happening when the axis goes from the cube center to a cube corner, making a factor of $\sqrt{3}$. For the other part of the Minkowsky sum, $Ball(r_B/\sqrt{3})$ is contained in a cube of radius r_B. Overall, the statement is true with $\sigma = \sqrt{3}$. **Q.E.D.**

F Appendix: Correct Implementation of Soft Exact Algorithms

As mentioned at the end of Sec. 7, the earlier sections provide an "exact" description of planners for a rod and a ring, albeit a "soft kind" that admits a user-controlled amount of numerical indeterminacy. The reader may have noticed that we formulated precise mathematical relations and exact geometric shapes for which various inclusions must be verified for correctness. Purely numerical computations (even with arbitrary precision) cannot "exactly determine" such relations in general. Nevertheless, we claim that all our computations can be guaranteed in the soft sense. The basic idea is that for each box B, all the computations associated with B is computed to some absolute error bound that at most r_B/K^* where r_B is the box radius and K^* is a constant depending on the algorithm only. Thus, as boxes become smaller, we need higher precision (but the resolution ε ensures termination). Moreover, the needed precision requires no special programming effort.

This is possible because all the inequalities in our algorithms are "one-sided" in the sense that we do not assume that the failure of an inequality test implies the complementary condition (as in exact (unqualified) computation). We can define a **weak feature set** denoted $\widehat{\phi}(B)$ with this property:

$$\widehat{\phi}(B/\sigma) \subseteq \widetilde{\phi}(B) \subseteq \widehat{\phi}(B)$$

for some $\sigma > 1$. The "weak" $\widehat{\phi}(B)$ is not uniquely determined (i.e., $\widehat{\phi}(B)$ can be *any* set that satisfies the inequalities). In contrast, the set $\widetilde{\phi}(B)$ is mathematically precise and unique. If we use $\widehat{\phi}(B)$ instead of $\widetilde{\phi}(B)$, the correctness of our planner remains intact. Moreover, the weak set $\widehat{\phi}(B)$ can be achieved as using numerical approximation (note: we do not need "correct rounding" from our bigFloats, so `GMP` suffice).

We stress that these ideas have not been implemented, partly because there is no pressing need for this at present.

G Appendix: Counter Example for Ring Heuristic

We show that the use of $\text{Sep}'(C, f)$ (Appendix E) can lead to a wrong classification of a box B. Recall that $\text{Sep}'(C, f)$ is an upper bound on $\text{Sep}(C, f)$, and is an equality in case f is a corner or a triangle.

Assume that the footprint of configuration m_B is a unit circle C centered at the origin lying in the horizontal $z = 0$ plane.

We consider the polyhedral set $F \subseteq \mathbb{R}^3$ such that the intersection of F with any horizontal plane $H : \{z = z_0\}$ (for any z_0) is the L-shape $[-10, 10]^2 \setminus (2, 10]^2$ when projected to the (x, y)-plane. See Figure 7.

Let f_0 be the boundary feature of F that is closest to circle C. Clearly, f_0 is the vertical line $\langle x = 2, y = 2\rangle$. Moreover, $\text{Sep}(C, f_0) = 2\sqrt{2} - 1 < 1.82$. Now, slightly perturb F so that f_0 is slightly non-vertical, but it's projection onto the (x, y)-plane is the line $y = 2$ (in Figure 7, f_0 is the red dot, and $y = 2$ is the green line). We also verify that $\text{Sep}'(C, f_0) = \sqrt{5} \simeq 2.36$.

It is also important to see that all the other boundary features $f \neq f_0$ of F, we have $\text{Sep}'(C, f) > 2$. To see this, there are 2 possibilities for f: if f is an edge, this is clear. If f is a face, this is also clear unless the face is bounded by f_0 (there are two such faces). In this case, our algorithm sets $\text{Sep}'(C, f)$ to $\text{Sep}'(C, f_0)$ which is > 2.23. Note that F does not have any corner features.

Now construct any convex polyhedron $G \subseteq \mathbb{R}^3$ that is disjoint from F such that boundary feature of G that is closest to C is a corner $g_0 = (2.1, 2.1, 0)$. It is easy to construct such a G. Moreover, we see that $\text{Sep}(C, g_0) = \text{Sep}'(C, g_0) = \sqrt{2(2.1)^2} - 1 \simeq 1.97$.

Suppose $\Omega = F \cup G$ and the translational and rotational parts of B are given by $B^t = [-1/2, 1/2]^2$ and $B^r = [-1/8, 1/8, 1]$. We may assume that $\widetilde{\phi}(B)$ is empty. To classify B,

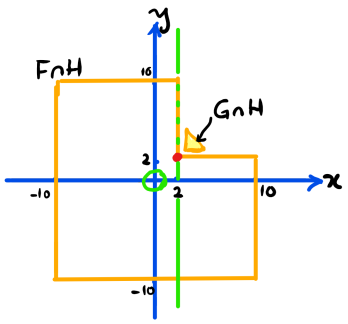

Figure 7 Counter Example

we look at the set $\widetilde{\phi}(parent(B))$. Say the translational and rotational parts of $parent(B)$ are $[-1/2, 3/2]^2$ and $[-1/8, 3/8, 1]$, respectively. In this case $\widetilde{\phi}(parent(B))$ contains any g_0 (and possibly f_0). In any case, g_0 would be regarded as the closest feature in $\widetilde{\phi}(parent(B))$ because we use $\mathrm{Sep}'(C, f)$ for comparison. Based on g_0, our algorithm would decide that B is `FREE` when in fact B is `STUCK`.