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

Ching-Hsiang Hsu  
Department of Computer Science, Courant Institute, New York University, New York, NY, USA 
chhsu@nyu.edu

Yi-Jen Chiang  
Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA 
chiang@nyu.edu

Chee Yap  
Department of Computer Science, Courant Institute, 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 a long standing challenge in 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. We compared our planner to state-of-the-art sampling planners in OMPL [31].

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

2012 ACM Subject Classification  Theory of computation → Computational geometry; Computing methodologies → Robotic planning

Keywords and phrases Algorithmic Motion Planning, Subdivision Methods, Resolution-Exact Algorithms, Soft Predicates, Spatial Rod Robots, Spatial Ring Robots

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.43


Funding Supported in part by NSF Grants #CCF-1423228 and #CCF-1563942. 
Ching-Hsiang Hsu: Supported by NSF Grant #CCF-1423228.
Chee Yap: Supported in part by NSF Grants #CCF-1423228 and #CCF-1563942.

1 Introduction

Motion planning [18, 5] 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 [14]. 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 [32]. 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 [5, 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., [22, 8]). Our present paper is based on the Subdivision Approach. This approach has a venerable history in robotics – see [3, 40] for early planners based on subdivision.

Exact path planning has many issues including a serious gap between theory and implementability. In [32, 36], we introduced a theoretical framework based on subdivision to close this gap. This paper demonstrates for the first time that our framework is able to achieve rigorous state-of-the-art planners in 3D. Figure 1 shows our rod robot in an environment with 100 random tetrahedra. Figure 6 shows our ring robot in an environment with pillars and L-shaped posts. See a video demo from http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html.

![Figure 1](image1.png)

**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 robots to be “thickened” by Minkowski sum with balls (see [33]). 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 $\Omega$ is a closed polyhedral set. Say $\gamma$ 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 $\mu$ is in $C_{free}$. Path planning amounts to finding such paths. Following [40], 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 $\tilde{C}(B)$ that are “soft versions” of $C(B)$ (see the full paper [16] Appendix A).
We present four desiderata in path planning:

(G0) the planner must be mathematically 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 notions of resolution completeness and probabilistic completeness in robotics have requirement (a) but not (b). In probabilistic-complete algorithms, halting with NO-PATH is achieved heuristically by putting limits on time and/or number of samples. But such limits are not intrinsic to the input instance. In resolution-complete algorithms, NO-PATH halting is based on width \(w\) of subdivision box being small enough (say \(w < \varepsilon\)). One issue is that the width of a box is a direct measure of clearance (but there is a nontrivial correlation); secondly, box predicates are numerical and “accurate enough” (\(\sigma\)-effective in our theory). These issues are exacerbated when algorithms do not use box predicates, but perform sampling at grid points of the subdivision. In contrast, our NO-PATH guarantees an intrinsic property: there is no path of clearance \(K \varepsilon\) (see below).

But desideratum (G0) is only the base line. 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 methods [30], it is possible to satisfy (G1); Halperin et al [13] give such solutions in 2D using CGAL. 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 is having 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 may prove to be more widely applicable.

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 our interest in alternative notions of exactness, we refer to the standard notion as exact (unqualified). Planners that are exact (unqualified) are first shown in [26]; this can be viewed as a fundamental result on decidability of connectivity in semi-algebraic sets [1]. The curse of

---

\(^1\) Partial completeness means the algorithm produces a correct output provided it halts.
exact (unqualified) algorithms is that the algorithm must detect any degeneracies in the input and handle them explicitly. But exact (unqualified) algorithms are rare, mainly because degeneracies are numerous and hard to analyze: the usual expedient is to assume “nice” (non-degenerate) inputs. So the typical exact (unqualified) algorithms in the literature are conditional algorithms, i.e., its correctness is conditioned on niceness assumptions. 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 [15, 12, 11, 34]). For similar reasons, unconditional exact (unqualified) path planners in 3D are unknown.

We now address (G1-G3). The typical implementation is based on machine arithmetic (the IEEE standard), which may satisfy (G2) but almost certainly not (G1). We regard this as a (G1-G2) trade-off. In fact, our implementations here as well as in our previous papers [32, 20, 33] are such machine implementations. This follows the practice in the 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 below:

**Resolution-Exact Path Planning** for robot \( R_0 \):

**Input:** \((\alpha, \beta; \Omega; B_0, \varepsilon)\)

where \( \alpha, \beta \in C_{\text{space}}(R_0) \) is the start and goal, \( \Omega \subseteq \mathbb{R}^3 \) the obstacle set, \( B_0 \subseteq C_{\text{space}}(R_0) \) is a box, and \( \varepsilon > 0 \).

**Output:** Halt with either an \( \Omega \)-free path from \( \alpha \) to \( \beta \) in \( B_0 \), or \texttt{NO-PATH} satisfying the conditions (P) and (N) below.

- (P) If there is a path (from \( \alpha \) to \( \beta \) in \( B_0 \)) of clearance \( K \varepsilon \), the output must be a path.
- (N) If there is no path in \( B_0 \) of clearance \( \varepsilon/K \), the output must be \texttt{NO-PATH}.

Here, clearance of a path is the minimum separation of the obstacle set \( \Omega \) 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 \texttt{NO-PATH}. This indeterminacy is essential to escape from exact computation (and arguably justified for robotics [36]). The constant \( K > 1 \) is treated in more detail in [32, 37]. But resolution-exactness is just a definition. How do we design such algorithms? We propose to use subdivision, and couple with soft predicates to exploit resolution-exactness. We replace the classification \( C(B) \) by a soft version \( \tilde{C}(B) \) [32]. This leads to a general resolution-exact planner which we call **Soft Subdivision Search** (SSS) [36, 37] that shares many of the favorable properties of sampling planners (and unlike exact planners). We demonstrated in [32, 20, 33] that for planar robots with up to 4 DOFs, our planners can consistently outperform state-of-the-art sampling planners.

### 1.1 What is New: Contributions of This Paper

In this work, we design \( \varepsilon \)-exact planners for rods and rings, with accompanying implementation that addresses the desiderata (G0-G3). This fulfills a long-time challenge in robotics. We are able to do this because of the twin foundations of resolution-exactness and soft-predicates.

\[ \text{For simplicity, we do not require the output path to have any particular clearance, but we could require clearance } \geq \varepsilon/K \text{ as in [32].} \]
Although we had already used this foundation to implement a variety of planar robots [32, 20, 33, 39] that can match or surpass state-of-the-art sampling methods, it was by no means assured that we can extend this success to 3D robots. Indeed, the present work required a series of technical innovations: (I) One major technical difference from our previous work on planar robots is that we had to give up the notion of “forbidden orientations” (which seems “forbidding” for 3D robots). We introduced an alternative approach based on the “safe-and-effective” approximation of footprint of boxes. We then show how to achieve such approximations for the rod and ring robots separately. (II) The approximated footprints of boxes are represented by what we call $\Sigma_2$-sets (Sec. 4.1); this representation supports desideratum (G2) for easy implementation. One side benefit of $\Sigma_2$-sets is that they are very flexible; thus, we can now easily extend our planners to “thick” versions of the rod or ring. In contrast, the forbidden orientation approach requires non-trivial analysis to justify the “thick” version [33]. The trade-off in using $\Sigma_2$-sets is a modest increase in the accuracy constant $K$. (III) We also need good representations of the 5-DOF configuration space. Here we introduce the square model of $S^2$ to avoid the singularities in the usual spherical polar coordinates [19], and also to support subdivision in non-Euclidean spaces. (IV) Not only is the geometry in 3D more involved, but the increased degree of freedom requires new techniques to further improve efficiency. Here, the search heuristic based on Voronoi diagrams becomes critical to achieve real-time performance (desideratum (G3)).

**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 techniques for computing approximate footprints of rods and rings. We discuss efficiency and experimental results in Section 7. We conclude in Section 8. All proofs and some background are given in the full paper [16] (in appendices).

## 2 Literature Review

Halperin et al [14] gave a general survey of path planning. An early survey is [35] where two universal approaches to exact path planning were described: cell-decomposition [25] and retraction [24, 23, 4]. Since exact path planning is a semi-algebraic problem [26], it is reducible to general (double-exponential) cylindrical algebraic decomposition techniques [1]. But exploiting path planning as a connectivity problem yields singly-exponential time (e.g., [10]). The case of a planar rod (called “ladder”) was first studied in [25] using cell-decomposition. More efficient (quadratic time) methods based on the retraction method were introduced in [28, 29]. On-line versions for a planar rod are also available [7, 6].

Spatial rods were first treated in [27]. The combinatorial complexity of its free space is $\Omega(n^4)$ in the worst case and this can be closely matched by an $O(n^{4+\epsilon})$ time algorithm [17]. The most detailed published planner for a 3D rod is Lee and Choset [19]. They use a retraction approach. The paper exposes 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. Though the goal is an exact path planner, degeneracies are not fully discussed. Their two accompanying videos have no timing or experimental data.

One of the few papers to address the non-existence of paths is Zhang et al [38]. 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 (this is impossible without exact computation).
3 Subdivision Charts and Atlas for $S^2$

Terminology. We fix some terminology for the rest of the paper. The fundamental footprint map $F_p$ 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 $F_p(B)$ as the union of $F_p(\gamma)$ as $\gamma$ 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 $\Omega$ 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 $\Omega$. The (minimal) set of corners and edges is uniquely defined by $\Omega$, but walls depend on a triangulation of $\partial \Omega$. If $A, B \subseteq \mathbb{R}^3$, define their separation $Sep(A, B) := \inf \{\|a - b\| : a \in A, b \in B\}$ where $\|a\|$ is the Euclidean norm. The clearance of $\gamma$ is $Sep(F_p(\gamma), \Omega)$. Say $\gamma$ is $\Omega$-free (or simply free) if it has positive clearance. Let $C_{free} = C_{free}(\Omega)$ be the set of $\Omega$-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., $SO(3) \subseteq \mathbb{R}^3$ viewed as orthogonal matrices) or (2) as a subset of $\mathbb{R}^m$ subject to identification (in the sense of quotient topology [21]). 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 $\theta$ is identified with each pole, causing severe distortions near the poles which are singularities. So the numerical primitives in [19, Appendix F] have severe numerical instabilities.

To obtain a representation of $S^2$ without singularities, we use the map [37]

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

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

$$C_0 := \sup_{p \neq q \in S^2} \left\{ \max \left\{ \frac{d_2(p, q)}{d_2(p, \tilde{q})}, \frac{\tilde{d}_2(p, q)}{\tilde{d}_2(p, \tilde{q})} \right\} \right\}$$

where $d_2$ and $\tilde{d}_2$ are the metrics on $S^2$ and $\tilde{S}^2$ respectively. Clearly $C_0 \geq 1$. Intuitively, $C_0$ is the largest distortion factor produced by the map $q \mapsto \tilde{q}$ (by definition the inverse map has the same factor).

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

The proof in the full paper [16] Appendix B.1 also shows that the worst distortion is near the corners of $\tilde{S}^2$. The constant $C_0$ is one of the 4 constants that go into the ultimate accuracy constant $K$ in the definition of $\varepsilon$-exactness (see [37] for details).
It is obvious how to do subdivision in $\hat{S}^2$. This is illustrated in Figure 2(b). After the first subdivision of $\hat{S}^2$ into 6 faces, subsequent subdivision is just the usual quadtree subdivision of each face. We interpret the subdivision of $\hat{S}^2$ as a corresponding subdivision of $S^2$. In [37], we give the general framework using the notion of subdivision charts and atlases (borrowing terms from manifold theory).

4 Approximate Footprints for Boxes in $\mathbb{R}^3 \times S^2$

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 [32, 37]. (The SSS framework is summarized in the full paper [16] Appendix A.) As noted in the introduction, our soft predicate $\tilde{\phi}$ classifies any input box $B \subseteq C_{\text{space}}$ into one 3 possible values. A key idea of our 2-link robot work [20, 33] is the notion of “forbidden orientations” (of a box $B$, in the presence of $\Omega$). The same concept may be attempted for $\mathbb{R}^3 \times S^2$, except that the details seem to be formidable to analyze and to implement. Instead, this paper introduces a direct approximation of the footprint of a box, $\tilde{Fp}(B) := \bigcup \{ \tilde{Fp}(\gamma) : \gamma \in B \}$. We now introduce $\tilde{Fp}(B) \subseteq \mathbb{R}^3$ as the approximate footprint, and discuss its properties. 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 peek 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” [32]. The idea is to maintain a set $\tilde{\phi}(B)$ of approximate features for each box $B$. We softly classify $B$ as $\tilde{C}(B) =$ MIXED as long as $\tilde{\phi}(B)$ is non-empty; otherwise, we can decide whether $\tilde{C}(B) = C(B) =$ FREE or STUCK. This decision is relatively easy in 2D, but is more involved in 3D and detailed in the full paper [16] Appendix B.2. For correctness of this procedure, we require

$$\tilde{\phi}(B/\sigma) \subseteq \phi(B) \subseteq \tilde{\phi}(B).$$

(1)

Here $\sigma > 1$ is some global constant and “$B/\sigma$” denotes the box $B$ shrunk by factor $\sigma$. Basically, (1) guarantees that our soft predicate $\tilde{C}(B)$ is conservative and $\sigma$-effective (i.e., if $B$ is free then $\tilde{C}(B/\sigma) =$ FREE). For computational efficiency, we want the approximate feature sets to have inheritance property, i.e.,

$$\tilde{\phi}(B) \subseteq \tilde{\phi}(\text{parent}(B)).$$

(2)

We now show what this computational scheme demands of our approximate footprint. Define the exact feature set of box $B$ as usual: $\phi(B) := \{ f \in \Phi(\Omega) : f \cap Fp(B) \neq \emptyset \}$ and (tentatively) the approximate feature set of box $B$ as

$$\tilde{\phi}(B) := \{ f \in \Phi(\Omega) : f \cap \tilde{Fp}(B) \neq \emptyset \}.$$  

(3)

The important point is that $\tilde{Fp}(B)$ is defined prior to $\tilde{\phi}(B)$. We need the fundamental inclusions

$$\tilde{Fp}(B/\sigma) \subseteq Fp(B) \subseteq \tilde{Fp}(B).$$

(4)

Note that this immediately implies (1). Unfortunately, (3) and (4) together do not guarantee inheritance, i.e., (2). Instead, we define $\tilde{\phi}(B)$ recursively as follows:

$$\tilde{\phi}(B) := \begin{cases} 
\{ f \in \Phi(\Omega) : f \cap \tilde{Fp}(B) \neq \emptyset \} & \text{if } B \text{ is the root,} \\
\{ f \in \tilde{\phi}(\text{parent}(B)) : f \cap \tilde{Fp}(B) \neq \emptyset \} & \text{else.}
\end{cases}$$

(5)
Notice that this only defines \( \tilde{\varphi}'(B) \) when \( B \) is an aligned box (i.e., obtained by recursive subdivision of the root box). But \( B/\sigma \) is never aligned when \( B \) is aligned, and thus \( \tilde{\varphi}'(B/\sigma) \) is not captured by (5). Therefore we introduce a parallel definition:

\[
\tilde{\varphi}'(B/\sigma) := \begin{cases} 
\{ f \in \Phi(E) : f \cap \tilde{Fp}(B/\sigma) \neq \emptyset \} & \text{if } B \text{ is the root,} \\
\{ f \in \tilde{\varphi}(\text{parent}(B)/\sigma) : f \cap \tilde{Fp}(B/\sigma) \neq \emptyset \} & \text{else.}
\end{cases}
\] (6)

Now, \( \tilde{\varphi}'(B) \) satisfies (2). But does it satisfy (1), which is necessary for correctness? This is answered affirmatively by the following lemma (proved in the full paper [16] Appendix B.3):

\begin{lemma}
If the approximate footprint \( \tilde{Fp}(B) \) satisfies Eq. (4), then \( \tilde{\varphi}'(B) \) satisfies Eq. (1), i.e.,

\[ \tilde{\varphi}'(B/\sigma) \subseteq \phi(\sigma) \subseteq \tilde{\varphi}(B). \]
\end{lemma}

Since \( \tilde{\varphi}'(B) \) has all the properties we need, we have no further use for the definition of \( \tilde{\varphi}(B) \) given in (3). Henceforth, we simply write “\( \tilde{\varphi}(B) \)” to refer to the set \( \tilde{\varphi}'(B) \) defined in (5) and (6).

**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 \( X \) in \( r \) and the distance from the center of \( Ball(X) \) to \( c \) is \( h \geq r \), then call \( \text{arcsin}(r/h) \) 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 \( B = B^t \times B^r \subseteq \mathbb{R}^3 \times \mathbb{S}^2 \) where \( B^t \) and \( B^r \) are the translational and rotational components of \( B \), and \( B^r \) is either \( \mathbb{S}^2 \) or a subspace of a face of \( \mathbb{S}^2 \). We let \( m_B \) and \( r_B \) denote the center and radius (distance from the center to any corner) of \( B^t \). 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 \( \Sigma_2 \)-Sets

Besides the above inclusion properties of \( \tilde{Fp}(B) \), we also need to decide if \( \tilde{Fp}(B) \) intersects a given feature \( f \). We say \( \tilde{Fp}(B) \) is “nice” if there are intersection algorithms that are easy to implement (desideratum G2) and practically efficient (desideratum G3). We now formalize and generalize some “niceness” properties of \( \tilde{Fp}(B) \) that were implicit in our previous work ([32, 20, 33], especially [39]).

An elementary set (in \( \mathbb{R}^2 \)) 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 \( \mathbb{R}^3 \). In \( \mathbb{R}^2 \), we have a similar notion of elementary sets \( \mathcal{E}_2 \) comprising half-planes, discs or their complements. All these elementary sets are defined by a single polynomial inequality – so
technically, they are all “algebraic half-spaces”. The sets in $\mathcal{E}$ are evidently “nice” (niceness of a ring has some subtleties – see Sec. 6). We next extend our collection of nice sets: define a $\Pi_1$-set to be a finite intersection of elementary sets. We regard a $\Pi_1$-set $S = \bigcap_{i=1}^{n} S_i$ to be “nice” because we can easily check if a feature $f$ intersects $S$ by a simple while-loop (see below). Notice that $\Pi_1$ contains all convex polytopes in $\mathbb{R}^3$. Our definitions of $\tilde{F}_p(B)$ in [32, 20, 33] are all $\Pi_1$-sets. But in [39], we make a further extension: define a $\Sigma_2$-set to be a finite union of the $\Pi_1$-sets, i.e., each $\Sigma_2$-set $S$ has the form

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

where $S_{ij}$’s are elementary sets. We still say such an $S$ is “nice” since checking if a feature $f$ intersects $S$ can be written in a doubly-nested loop (see below). Although this intersection is more expensive to check than with a $\Pi_1$-set, it may result in fewer subdivisions and better efficiency in the overall algorithm. Thus, there is an accuracy-efficiency trade-off. Good approximations of footprints are harder to do accurately in 3D, and the extra power of $\Sigma_2$ seems critical.

We can put all these in the framework of a well-known construction of an infinite hierarchy of sets, starting from some initial collection of sets. If $\Delta$ is any collection of sets, let $\Pi(\Delta)$ denote the collection of finite intersections of sets in $\Delta$; similarly, $\Sigma(\Delta)$ denotes the collection of finite unions of sets in $\Delta$. Then, starting with any collection $\Delta_1$ of sets, define the infinite hierarchy of sets:

$$\Sigma_i, \Pi_i, \Delta_i \quad (i \geq 1)$$

where $\Sigma_i := \Sigma(\Delta_i)$, $\Pi_i := \Pi(\Delta_i)$, and $\Delta_{i+1} := \Sigma_i \cup \Pi_i$. An element of $\Sigma_i$ or $\Pi_i$ is simply called a $\Sigma_i$-set or a $\Pi_i$-set.

We call (7) a $\Sigma_2$-decomposition of $S$, where $\Delta_1 := \mathcal{E}$. Note that this decomposition may not be unique, but in the cases arising from 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, e.g., replacing each $S_{ij}$ by its $\tau$-expansion, i.e., $S_{ij} \oplus \text{Ball}(\tau)$ (where $\oplus$ denotes the Minkowski sum), which remains elementary. Under certain conditions, the corresponding set is a reasonable approximation to $S \oplus \text{Ball}(\tau)$. If so, we can generalize the corresponding soft predicate to robots with thickness $\tau$.

Once we have a $\Sigma_2$-decomposition of $\tilde{F}_p(B)$, 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 (7) 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 “$f_0 \leftarrow f_0 \cap S_{ij}$” for $j = 1, \ldots, m_i$. If ever $f_0$ becomes empty, we know that the set $S_i = \bigcap_{j=1}^{m_i} S_{ij}$ has empty intersection with $f$. The possibility of such representations is by no means automatic but in the next two sections we verify that they can be achieved for our rod and ring robots. These sections make our planners fully “explicit” for an implementation.

5 Soft Predicates for a Rod Robot

In this section, $R_0$ is a rod with length $r_0$; we choose one endpoint of the rod as the rotation center. Let $B = B' \times B' \subseteq \mathbb{R}^3 \times S^2$ be a box. Our main goal is to define approximate footprint $\tilde{F}_p(B)$, and to prove the basic inclusions in Eqs. (4) and (1). This turns out to be a $\Pi_1$-set (we also indicate a more accurate $\Sigma_2$-set.)

---

3 From mathematical analysis, constructive set theory and complexity theory.
It is useful to define the inner footprint of $B$, $Fp_0(B)$, as $Fp(m_B \times B^r)$. This set is the intersection of a ball and a square cone:

$$Fp_0(B) = Ball(r_0, m_B) \cap Cone(m_B, B^r + m_B). \tag{9}$$

The edges of this square cone is shown as green lines in Figure 3; furthermore, the brown box is $S^2 + m_B$ (translation of $S^2$ so that it is centered at $m_B$). Note that the box footprint $Fp(B)$ is the Minkowski sum of $Fp_0(B)$ with $B^t - m_B$ (the translation of $B^t$ to make it 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 $Cone(\theta + \epsilon)(m_B, B^r + m_B)$ denote the intersection of the expanded half-spaces, $H_i \oplus Ball(r_B)$ ($i = 1, \ldots, 4$). In general, $Cone(\theta + \epsilon)(m_B, B^r + m_B)$ is not a cone (it may not have a unique “apex”). Similarly we “expand” the inner footprint of (9) into

$$\widetilde{Fp}(B) := Ball(r_0 + r_B, m_B) \cap Cone(\theta + \epsilon)(m_B, B^r + m_B). \tag{10}$$

We use quotes for “$\widetilde{Fp}(B)$” in (10) 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 $\theta$ 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 $\theta$ becomes arbitrarily small. Indeed $\theta$ can be arbitrarily small because it can be proportional to the input resolution $\epsilon$. We conclude that such a planner is not resolution-exact. To fix this problem, we finally define

$$\hat{Fp}(B) := \widetilde{Fp}(B) \cap H_0 \tag{11}$$

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” $m_B$. We can also use the “horizontal” plane that is parallel to $B^t$ and containing the “lower” face of $B^t$. We adopt this latter $H_0$ to have a simpler geometric structure.

This completes the description of $\hat{Fp}(B)$. It should be clear that checking if $\hat{Fp}(B)$ intersects any feature $f$ is relatively easy (since it is even a $\Pi_1$-set). In the full paper [16] Appendix C we prove the following theorem:

> **Theorem 3.** The approximate footprint $\hat{Fp}(B)$ as defined for a rod robot satisfies Eq. (4), i.e., there exists some fixed constant $\sigma > 1$ such that $\hat{Fp}(B/\sigma) \subseteq Fp(B) \subseteq \hat{Fp}(B)$. 

![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.](image-url)
6 Soft Predicates for a Ring Robot

Let $R_0$ be a ring robot. Its footprint is an embedded circle of radius $r_0$. First we show how to compute $\text{Sep}(C, f)$, the separation of an embedded circle $C$ from a feature $f$. This was treated in detail by Eberly [9]. This is easy when $f$ is a point or a plane. When $f$ is a line, Eberly gave two formulations: they reduce to solving a system of 2 quadratic equations in 2 variables, and hence to solving a quartic equation; see the full paper [16] Appendix D.1 for more details. The predicate “Does $f$ intersect $C \oplus \text{Ball}(r')$, a ring of thickness $r'$?” is needed later; it reduces to “Is $\text{Sep}(C, f) \leq r'$?”.

Our next task is to describe an approximate footprint. First recall the round cone of box $B$ defined in the previous section: $\text{Cone}(B) = \text{Cone}(m_B, \text{Ball}(m_B + B'))$. Let $\theta = \theta(B)$ be the half-angle of this cone, and $c$ the center of $B'$. Here, we think of $c$ as a point of $\mathbb{S}^2$, and define $\gamma(B) := m_B \times c$ viewed as an element of $\mathbb{R}^3 \times \mathbb{S}^2$. Call $\gamma(B)$ the central configuration of box $B$. Let $\text{Ray}(B)$ be the ray from $m_B$ through $m_B + c$. If $\text{Plane}(B)$ is the plane through $m_B$ and normal to $\text{Ray}(B)$, then the footprint $\text{Fp}(\gamma(B))$ is an embedded circle lying in $\text{Plane}(B)$. We define the inner footprint of $B$ as $\text{Fp}_0(B) := \text{Fp}(m_B \times B')$. The map $q \mapsto \overline{q}$ is the inverse of $q \mapsto \widetilde{q}$, taking $c \in \mathbb{S}^2$ to $\overline{c} \in \mathbb{S}^2$. It is hard to work with $\text{Fp}_0(B)$. Instead consider the set $D(B)$ of all points in $\mathbb{S}^2$ whose distance from $\overline{c}$ is at most $\theta(B)$. So $D(B)$ is the intersection of $\mathbb{S}^2$ with a round cone with ray from the origin to $c$. Then we have $\text{Fp}_0(B) \subseteq \text{Fp}_1(B)$ where

$$\text{Fp}_1(B) := \text{Fp}(m_B \times D(B)).$$

Our main computational interest is the approximate footprint of $B$ defined as

$$\overline{\text{Fp}}(B) := \text{Fp}_1(B) \oplus \text{Ball}(r_B).$$

Note that $\text{Fp}_1(B)$ 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 $\text{Cone}(B)$ (the axis of $\text{Cone}(B)$ is shown as a vertical ray. Each $\text{Ball}$ has radius $r_B$.

![Figure 4](image)

**Figure 4** Ring Robot: central cross-section of $\text{Fp}_1(B)$ appears as two blue arcs. $\overline{\text{Fp}}(B)$ equals the union of two “thick rings” and a “truncated annulus”. The axis of $\text{Cone}(B)$ is shown as a vertical ray. Each $\text{Ball}$ has radius $r_B$.

4 Recall that $\mathbb{S}^2$ is a metric space whose geodesics are arcs of great circles.
appear as two horizontal blue lines in Figure 4. In the cross section, $F_{p1}(B)$ are seen 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 $\Sigma_2$-decomposition of $\tilde{F}_p(B)$: it is the union of two “thick rings”, $C_1 \oplus \text{Ball}(r_B)$ and $C_2 \oplus \text{Ball}(r_B)$ (both of thickness $r_B$), and a shape $\text{Ann}(B)$ 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, $\text{Cone}(m_B, C^*_1)$ and $\text{Cone}(m_B, C^*_2)$. Together, they form a double cone that is actually a simpler object for computation! Finally, define $\text{Ann}(B)$ to be the intersection of the annulus with the complements of the double cone.

For each thick ring $C_i \oplus \text{Ball}(r_B)$, deciding “Does a feature $f$ intersect $C_i \oplus \text{Ball}(r_B)$?” is equivalent to “Is Sep($C_i, f$) $\leq r_B$?” (see beginning of this section). In the full paper [16], Appendix D.1 discusses this computation and Appendix D.2 proves the following theorem:

**Theorem 4.** The approximate footprint $\tilde{F}_p(B)$ as defined for a ring robot satisfies Eq. (4), i.e., there exists some fixed constant $\sigma > 1$ such that $\tilde{F}_p(B/\sigma) \subseteq F_p(B) \subseteq \tilde{F}_p(B)$.

7 Practical Efficiency of Correct Implementations

We have developed $\varepsilon$-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 $\Sigma_2$-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 good search strategies are deployed. In this paper, we have found that some form of Voronoi heuristic is extremely effective: the idea is to find paths along Voronoi curves [in the sense of [24, 28]], and exploit subdivision Voronoi techniques based (again) on the method of features [34, 2]. There are subtleties necessitating the use of pseudo-Voronoi curves [19, 28, 29]. Since we do not rely on Voronoi heuristics for correctness, simple expedients are available. To recognize Voronoi curves, we maintain (in addition to the collision-detection feature set $\tilde{\phi}(B)$), the Voronoi feature set $\tilde{\phi}_V(B)$. These two sets have some connection but there are no obvious inclusion relationships.

![Figure 5](http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html) Ring robot amidst 40 random tetrahedra: (a) trace of a found path; (b) subdivision of translational boxes on the path.

Our current implementation achieves near real-time performance (see video [http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html](http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html)). Table 1 summarizes ex-
Figure 6 Ring robot amidst pillars and L-shaped posts (Posts): (a) trace of a found path; (b) subdivision of translational boxes on the path.

Figure 7 Ring robot amidst another set of pillars and posts (Posts2): (a) start and goal configurations (no path found); (b) subdivision of translational boxes during the search.

Table 1 Rod and Ring Experiments.

<table>
<thead>
<tr>
<th>Exp. #</th>
<th>Envic</th>
<th>Length</th>
<th>$\epsilon$</th>
<th>Start Conf.</th>
<th>Goal Conf.</th>
<th>Path</th>
<th>Time (s)</th>
<th>#Boxes (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>Rand100</td>
<td>120</td>
<td>16.8</td>
<td>(240, 120, 360, -0.5, -0.5, -1)</td>
<td>(220, 30, 80, 0.1, 0.8, 1)</td>
<td>Y/Y</td>
<td>1.05:2.82</td>
<td>8.122:1</td>
</tr>
<tr>
<td>3/4</td>
<td>Rand100</td>
<td>120</td>
<td>16.8</td>
<td>(400, 60, 380, -1, 0.0)</td>
<td>(200, 200, 240, 0, 1.0)</td>
<td>Y/Y</td>
<td>1.13:2.92</td>
<td>19.3:82.2</td>
</tr>
<tr>
<td>5/6</td>
<td>Rand40</td>
<td>160</td>
<td>16.8</td>
<td>(80, 32, 480, 0, 0, -1)</td>
<td>(240, 440, 200, 1, 0, 0)</td>
<td>Y/Y</td>
<td>16:1290.65</td>
<td>244.7/1138.2</td>
</tr>
<tr>
<td>7/8</td>
<td>Rand40</td>
<td>160</td>
<td>16.8</td>
<td>(400, 480, 80, 0, 0, 1.0)</td>
<td>(30, 80, 480, 0.5, 0.1, -1)</td>
<td>Y/Y</td>
<td>14.54:9.4</td>
<td>217.5/113.8</td>
</tr>
<tr>
<td>9/10</td>
<td>Posts</td>
<td>60</td>
<td>16.8</td>
<td>(160, 480, 190, 0, 0, -1)</td>
<td>(390, 60, 420, 1, 0, 0)</td>
<td>Y/Y</td>
<td>0.07:0.13</td>
<td>2.13:7</td>
</tr>
<tr>
<td>11/12</td>
<td>Posts</td>
<td>60</td>
<td>16.8</td>
<td>(320, 120, 320, 0, 0, 0)</td>
<td>(200, 360, 60, 0, -1, 0)</td>
<td>NN</td>
<td>1.77:242.7</td>
<td>25.6/390.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exp. #</th>
<th>Envic</th>
<th>Radius</th>
<th>$\epsilon$</th>
<th>Start Conf.</th>
<th>Goal Conf.</th>
<th>Path</th>
<th>Time (s)</th>
<th>#Boxes (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>Rand100</td>
<td>40</td>
<td>16.8</td>
<td>(240, 120, 360, -0.5, -0.5, -1)</td>
<td>(220, 30, 80, 0.1, 0.8, 1)</td>
<td>Y/Y</td>
<td>0.66:0.57</td>
<td>2.72:2.63</td>
</tr>
<tr>
<td>3/4</td>
<td>Rand100</td>
<td>40</td>
<td>16.8</td>
<td>(400, 60, 380, -1, 0.0)</td>
<td>(160, 240, 240, 0, 1.0)</td>
<td>Y/Y</td>
<td>0.25:0.24</td>
<td>0.92:0.92</td>
</tr>
<tr>
<td>5/6</td>
<td>Rand40</td>
<td>60</td>
<td>16.8</td>
<td>(80, 120, 480, 0, 0, -1)</td>
<td>(240, 440, 200, 1, 0, 0)</td>
<td>Y/Y</td>
<td>9.38:30.61</td>
<td>38.37:110.93</td>
</tr>
<tr>
<td>7/8</td>
<td>Rand40</td>
<td>60</td>
<td>16.8</td>
<td>(400, 480, 80, 0, 0, 1.0)</td>
<td>(190, 80, 480, 0.5, 0.1, -1)</td>
<td>Y/Y</td>
<td>2.07:3.76</td>
<td>10.61:13.90</td>
</tr>
<tr>
<td>9/10</td>
<td>Posts</td>
<td>60</td>
<td>16.8</td>
<td>(250, 350, 190, 0, 0, 1)</td>
<td>(390, 320, 320, 1, 0, 0)</td>
<td>Y/Y</td>
<td>35.68:69.7</td>
<td>114.3:139.3</td>
</tr>
<tr>
<td>11/12</td>
<td>Posts2</td>
<td>60</td>
<td>16.8</td>
<td>(410, 90, 190, 0, 0, -1)</td>
<td>(315, 220, 325, 0, 1, 0)</td>
<td>N/N</td>
<td>7.09:271.3</td>
<td>8.1539.1</td>
</tr>
</tbody>
</table>
experiments on our rod and ring robots. The environments Rand100, Rand40 (100 and 40 random tetrahedra), Posts and Posts2 are shown in Figs. 1, 5, 6 and 7. The dimensions of the environments are 512\textsuperscript{3}. Our implementation uses C++ and OpenGL on the Qt platform. Our code, data and experiments are distributed\textsuperscript{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.

Table 2 (correlated with Table 1 by the Exp #’s) compares our methods with various sampling-based planners in OMPL \cite{OMPL}, where we accepted the default parameters and each instance was run 10 times, with the “average time (in s)/standard deviation/success rate” reported. This comparison has various caveats: we simulated the rod and ring robots by polyhedral approximations. We usually outperform RRT in cases of PATH. In case of NO-PATH, we terminated in real time while all sampling methods timed out (300s).

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Comparison with Sampling Methods in OMPL (the best run-time is shown in bold).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exp</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

8 Conclusions

Path planning in 3D has many challenges. Our 5-DOF spatial robots have pushed the current limits of subdivision methods. To our knowledge there is no similar algorithm with comparable rigor or guarantees. Conventional wisdom says that sampling methods can achieve higher DOFs than subdivision. By an estimate of Choset et al \cite[p. 202]{OMPL}, sampling methods are limited to 5–12 DOFs. We believe our approach can reach 6-DOF spatial robots. Since resolution-exactness delivers stronger guarantees than probabilistic-completeness, we expect a performance hit compared to sampling methods. But for simple planar robots (up to 4 DOFs) \cite{OMPL} we observed no such trade-offs because we outperform state-of-the-art sampling methods (such as OMPL \cite{OMPL}) often by two orders of magnitude. But in the 5-DOF robots of this paper, we see that our performance is competitive with sampling methods. It is not clear to us that subdivision is inherently inferior to sampling (we can also do random subdivision). It is true that each additional degree of freedom is conquered only with effort and suitable techniques. This remark seems to cut across both subdivision and sampling approaches; but it hits subdivision harder because of our stronger guarantees.

\textsuperscript{5} http://cs.nyu.edu/exact/core/download/core/
References


