

Soft Subdivision Search in Motion Planning, II: Axiomatics

Chee K. Yap^(✉)

Department of Computer Science, Courant Institute of Mathematical Sciences,
New York University, New York, NY 10012, USA
yap@cs.nyu.edu

Abstract. We propose to design motion planning algorithms with a strong form of resolution completeness, called **resolution-exactness**. Such planners can be implemented using **soft predicates** within the subdivision paradigm. The advantage of softness is that we avoid the Zero problem and other issues of exact computation. **Soft Subdivision Search** (SSS) is an algorithmic framework for such planners. There are many parallels between our framework and the well-known Probabilistic Road Map (PRM) framework. Both frameworks lead to algorithms that are practical, flexible, extensible, with adaptive and local complexity. Our several recent papers have demonstrated these favorable properties on various non-trivial motion planning problems. In this paper, we provide a general axiomatic theory underlying these results. We also address the issue of subdivision in non-Euclidean configuration spaces, and how exact algorithms can be recovered using soft methods.

1 Introduction

Motion planning has been studied for over 30 years, and remains a central problem in robotics. Path planning is the most basic form of motion planning in which we only consider kinematics, ignoring issues of timing, dynamics, non-holonomic constraints, sensing and mapping. In the algorithmic study of path planning, the problem is reduced to connectivity or reachability in some configuration space. There are three main approaches here: Exact, Sampling and Subdivision. Divergent paths have been taken: theoreticians favor the Exact Approach [2], but practical roboticists prefer the Sampling and Subdivision Approaches [9, 11]. For two decades, the Sampling Approach has dominated the field. According to Choset et al. [9, p.201], “*PRM, EST, RRT, SRT, and their variants have changed the way path planning is performed for high-dimensional robots. They have also paved the way for the development of planners for problems beyond basic path planning.*” The premise of this paper is that subdivision has many merits over sampling, and this power has not been fully exploited. But to open

C.K. Yap—Plenary Talk at the 9th Int’l. Frontiers of Algorithmics Workshop (FAW 2015) in Guilin, China, July 3–5. This work is supported by NSF Grants CCF-0917093 and CCF-1423228.

up this exploitation, we need to give it a sound foundation. This paper will provide one such foundation. We formulate the **Soft Subdivision Search** or SSS to unify and generalize our several recent papers [12, 13, 20, 21] in which we designed and implemented subdivision planners for several classes of robots. These SSS planners are relatively easy to design and implement. In our experiments, they outperform random sampling methods.

To introduce our approach, we compare the notion of correctness according to the three approaches. In the path planning problem, the robot R_0 is fixed, and each input instance is (Ω, α, β) where $\Omega \subseteq \mathbb{R}^k$ ($k = 2, 3$) is a description of the obstacles, and $\alpha, \beta \in \mathcal{Cspace}(R_0)$ are the start and goal configurations. In exact algorithms, the planner must return a path if one exists, and must return **NO-PATH** otherwise. In sampling, the input has an extra parameter N that bounds the maximum number of samples; the planner is said to be “sampling complete” if the planner returns a path with high probability when one exists and N is sufficiently large. In subdivision, the input has an extra resolution parameter $\varepsilon > 0$, and the planner is “resolution complete” if the planner returns a path when the ε is small enough. Thus sampling and (current) subdivision planners are similar in that their behaviors are only prescribed when there is a path. If there is no path, nothing is prescribed. In computability, such one-sided prescription of algorithmic behavior is well-known and is called “partial completeness”. To make the completeness “total”, we [20] introduce the concept of **resolution-exact** planners. Such a planner has an **accuracy constant** $K > 1$ (independent of input) such that:

- (P) If there is a path of clearance $K\varepsilon$, it returns a path.
- (N) If there is no path of clearance ε/K , it returns **NO-PATH**.

Thus the **NO-PATH** output guarantees that there is no path of clearance $K\varepsilon$. But the true innovation is the gap between the clearance bounds $K\varepsilon$ and ε/K : our planner could either return a path or **NO-PATH** when the optimal clearance lies in this gap. This “indeterminacy”, unavoidable in some sense [20], has a big payoff — resolution-exact planners can be implemented with purely numerical approximations. As all the standard fundamental constants¹ of Physics are known to no more than 8 digits of accuracy, and no robot dimension, actuator control, sensors or environment is known to nearly such accuracy, we should not see this indeterminacy as a limitation.

Our paper [25] is a companion to the present paper, providing background and other motivations. It presents SSS alongside PRM [10] as two general algorithmic “frameworks” based on a small number of subroutines and data structures. We get specific algorithms by instantiating these subroutines and data structures. As framework, “PRM” can cover many of its known variants. These two frameworks share many favorable properties, all lacking in exact algorithms. But we claim one advantage of SSS over PRM: *PRM has a halting problem which SSS does not have*. We clarify this remark: under the usual idea that **NO-PATH**

¹ Except speed of light which is exactly known, by definition.

means “non-existence of paths”, PRM cannot halt when there is no path. But suppose PRM adopts our viewpoint that NO-PATH means “no path of sufficient clearance”. Now, PRM *could* halt² but this amounts to exhaustive (exponential) search. In effect, exponential search amounts to non-halting. But our subdivision approach need not suffer from exponential behavior because we are able to eliminate large regions of the configuration space with a single test. Conceivably, there are adaptive search strategies that guarantee polynomial size search trees. For example, such results are known in our subdivision work on root isolation [6, 18, 19]: here, the worst-case subdivision tree sizes is provably linear (resp., quadratic) in terms of tree depth for real (resp., complex) root isolation.

1. Overview: In Sect. 2, we describe the SSS Framework. In Sect. 3, we provide the abstract elements of SSS: configuration spaces are replaced by metric spaces and Non-Euclidean spaces are subdivided via charts and atlases. Section 4 proves properties of SSS planners that satisfy some general axioms. Section 5 shows that exact algorithms can be recovered with SSS planners. We conclude in Sect. 6. For reasons of space, some proofs are deferred to the full paper. Figures 1 and 2 are in color.

2 The SSS Framework

What sets Subdivision Search apart from sampling or grid methods is that its predicates are not point-based but region-based. Suppose each $\gamma \in \mathcal{Cspace}$ has a classification as FREE, STUCK, or MIXED. Write $C(\gamma)$ for the classification of γ . We extend the classification to a set (or region) $B \subseteq \mathcal{Cspace}$ as follows: define $C(B) = \text{FREE}$ (resp., $= \text{STUCK}$) iff each $\gamma \in B$ is FREE (resp., STUCK); otherwise $C(B) = \text{MIXED}$. A classification function \tilde{C} is a **soft predicate** (relative to C) if it is conservative (i.e., $\tilde{C}(B) \neq \text{MIXED}$ implies $C(B) = \tilde{C}(B)$) and convergent (i.e., if $\lim_{i \rightarrow \infty} B_i \rightarrow \gamma \in \mathcal{Cspace}$ then $\tilde{C}(B_i) = C(\gamma)$ for i large enough). Here we write $\lim_{i \rightarrow \infty} B_i \rightarrow \gamma$ to mean that $\{B_i : i \geq 0\}$ is a monotone decreasing sequence of sets that converge to γ .

Let us now use soft predicates for path planning. Fixed robot R_0 . The motion planning input is $(\Omega, \alpha, \beta, \varepsilon)$ as above. It is standard (and without much loss) to also specify an initial box $B_0 \subseteq \mathcal{Cspace}$ to confine our sought-for path. Our main data structure is a subdivision tree, \mathcal{T} . It is useful to initially imagine $\mathcal{Cspace} \subseteq \mathbb{R}^d$, and \mathcal{T} as the standard multidimensional version of quadtrees, rooted at B_0 . But bear in mind our goal of extending \mathcal{Cspace} to non-Euclidean spaces, and B to non-box geometries. The SSS planner amounts to a loop that “grows” \mathcal{T} in each iteration by expanding some leaf until we find a path or conclude NO-PATH. There are two supporting data structures and three key routines:

- (Priority Queue) Q is a priority queue comprising those MIXED-leaves with length $\ell(B)$ (defined below) is at least ε .

² To do this, it would have to detect (probabilistically) that the sampling is dense enough, a non-trivial extension of the current PRM formulations.

- (Union-Find) D is a union-find data structure to maintain the connected components of the **FREE** boxes. As soon as we find a new **FREE** box, we form its union with the other adjacent **FREE** boxes. Boxes B, B' are **adjacent** if $B \cap B'$ is a $d - 1$ dimensional set.
- (Classifier) The routine \tilde{C} is a soft predicate that classifies each node in \mathcal{T} as **FREE**/**STUCK**/**MIXED**.
- (Search Strategy) This is represented by the queue's $Q.\text{getNext}()$ that returns a box in Q of highest priority.
- (Expander) The subroutine $\text{Expand}(B)$ subdivides B into two or more sub-boxes. These subboxes become the children of B in \mathcal{T} . In general, $\text{Expand}(B)$ represents a splitting strategy because it may have to choose from one or more alternative expansions.
- For $\gamma \in \mathcal{Cspace}$, let $\text{Box}(\gamma)$ denote any leaf in \mathcal{T} that contains γ . Also, $\text{Find}(\gamma)$ denote the box returned by the find operation of D when it is given $\text{Box}(\gamma)$. Thus, a path is found as soon as we discover $\text{Find}(\alpha) = \text{Find}(\beta)$.

Putting them together, we get our SSS framework:

```

SSS FRAMEWORK
1.  ▷ Initialization.
    While ( $\tilde{C}(\text{Box}(\alpha)) \neq \text{FREE}$ )
      If  $\text{Box}(\alpha)$  has length  $< \varepsilon$ , Return (NO-PATH)
      Else  $\text{Expand}(\text{Box}(\alpha))$ 
    While ( $\tilde{C}(\text{Box}(\beta)) \neq \text{FREE}$ )
      ... do the same for  $\beta$  ...
2.  ▷ Main Loop:
    While ( $\text{Find}(\alpha) \neq \text{Find}(\beta)$ )
      If  $Q$  is empty, Return(NO-PATH)
       $B \leftarrow Q.\text{getNext}()$ 
       $\text{Expand}(B)$ 
3.  Compute a FREE channel  $P$  from  $\text{Box}(\alpha)$  to  $\text{Box}(\beta)$ 
    Generate and return the “canonical path”  $\bar{P}$  inside  $P$ .

```

This framework has been used successfully to implement our disc and triangle planners [20], and our 2-link planner [12] including an interesting variant where self-crossing is not allowed [13]. Illustrating the power of subdivision and softness, we can easily generalize all these examples by fattening the robots and/or the polygonal obstacles. Notice that such extensions would be difficult for exact methods (to our knowledge, exact algorithms are unknown for such extensions). Of course many variants of this framework has appeared in the subdivision literature; conversely, some of these algorithms can be recaptured within SSS. E.g., the hierarchical search of Zhu and Latombe [28], Barbehenn and Hutchinson [1], or Zhang, Kim and Manocha (2008) [27]. One major difference is that these papers expand along a “mixed channels” (i.e., path comprising **FREE** or **MIXED** boxes). We could modify our `getNext` to achieve similar behavior; one advantage of this approach is that **NO-PATH** could be detected before emptying the queue. This abstract description hides an important feature of our technique: our computation of \tilde{C} is

deeply intertwined with the expansion of \mathcal{T} (see [8]). Steve LaValle (insightfully) described this as “opening up the blackbox” of collision testing.

3 Generalized Setting for SSS

Once the SSS framework has been instantiated with specific routines, we have an SSS planner. How do we know that the planner is resolution-exact? Our goal is to prove this under general “axiomatic” conditions. Designing a short list of such axioms is very useful: first, it gives us a uniform way to check that any proposed SSS algorithm is resolution-exact, just by checking the axioms. We could for instance apply this to our previous planners [12, 13, 20]. Second, because planning is a complex task, and we expect that SSS will suffer many variants, we must know the boundaries of the variations. The axioms serve as boundary markers.

The starting point is to replace \mathcal{Cspace} by a metric space X , and replace \mathcal{Cfree} by an open set $Y \subseteq X$. Points in the boundary ∂Y of Y are said to be **semi-free**. Let $C_Y : X \rightarrow \{+1, 0, -1\}$ denote the (exact) **classifier** for Y : $C_Y(\gamma) := +1/0/-1$ iff γ belongs to $Y/\partial(Y)/X \setminus \bar{Y}$ where \bar{Y} is the closure of Y . Note that we have performed a simple (non-essential) translation in our classification values: $\text{FREE} \rightarrow +1$, $\text{MIXED} \rightarrow 0$, and $\text{STUCK} \rightarrow -1$.

We extend the classification of points to classification of sets. There are two general ways to extend any function to a function on sets: let $f : S \rightarrow T$ be a function. The **set extension** of f (still denoted f) is the function $f : 2^S \rightarrow 2^T$ such that for $B \subseteq S$, $f(B) = \{f(b) : b \in B\}$. Here 2^S denotes the power set of S . Another general method applies to any geometric³ predicate $g : S \rightarrow \{+1, 0, -1\}$. The **set extension** of g (still denoted g) is the geometric predicate $g : 2^S \rightarrow \{+1, 0, -1\}$ such that for any definite value $v \in \{+1, -1\}$, $g(B) = v$ iff $g(b) = v$ for all $b \in B$; otherwise $g(B) = 0$.

Although the set extension of the classifier $C_Y : X \rightarrow \{+1, 0, -1\}$ is applicable to any subset $B \subseteq X$, in practice, we need B to be “nice” in order to carry out our algorithm: B must be able to support subdivision, $C_Y(B)$ must be (softly) computable, and we should be able to discuss the limits of such sets, $\lim_{i \rightarrow \infty} B_i$. We next capture these properties using “test cells”.

2. Test Cells and Subdivision Trees: Consider an Euclidean set $B \subseteq \mathbb{R}^d$. It is called a **test cell** if it is a full-dimensional, compact and convex polytope. For $d = 1$ ($d = 2$), test cells are intervals (convex polygons). Our subdivision of the metric space X will be carried out using such test cells.

Let the **width** $w(B)$ (resp., **length** $\ell(B)$) refer to the minimum (resp., maximum) length of an edge of B . The unique smallest ball containing B is called the **circumball** of B ; its center and radius are denoted $c(B)$ and $r(B)$. Note that $c(B)$ need not lie in the interior of B . The **inner radius** $r_0(B)$ of B is the

³ A **geometric predicate** is a 3-valued function, with a distinguished value 0 called the **indefinite value**. The others are called **definite values**. This is in contrast to a **logical predicate** which is 2-valued.

largest radius of a ball contained in B . Let $ic(B)$ comprises the centers of balls of radius $r_0(B)$ that are contained in B . E.g., if B is a rectangle, then $ic(B)$ is a line segment. Clearly, $ic(B)$ is convex. Then $c(ic(B))$ is called the **inner center** of B , denoted $c_0(B)$. Unlike $c(B)$, we now have $c_0(B)$ in the interior of B . We use $c_0(B)$ as follows: for any $\alpha > 0$, αB will mean scaling B by a factor α relative to the center $c_0(B)$. If $\alpha > 1$ (< 1) this amounts to growing (shrinking) B . The inverse operation is denoted B/α . Thus $(\alpha B)/\alpha = B$. The **aspect ratio** of B is $\rho(B) := r(B)/r_0(B) > 1$.

By a **subdivision** of a test cell B , we mean any finite set of test cells $\{B_1, \dots, B_m\}$ such that $B = \bigcup_{i=1}^m B_i$ and $\dim(B_i \cap B_j) < d$ for all $i \neq j$. We denote the subdivision relationship as $B = B_1 \uplus B_2 \uplus \dots \uplus B_m$.

Let $\square\mathbb{R}^d$ denote some set of test cells. For instance, $\square\mathbb{R}^d$ may be the set of all boxes, or the set of all simplices. Let the function $\mathbf{Expand} : \square\mathbb{R}^d \rightarrow 2^{\square\mathbb{R}^d}$ return a subdivision $\mathbf{Expand}(B)$ of B . In general, \mathbf{Expand} is a non-deterministic function⁴ and we may call it an “expansion scheme”. Using an expansion scheme, we can grow subdivision trees rooted in any $B \in \square\mathbb{R}^d$, by repeated expansion at any chosen leaf. We note some concrete schemes:

- **Longest Edge Bisection:** let $\square\mathbb{R}^d$ be simplices and $\mathbf{Expand}(B)$ returns a subdivision of B into two simplices by bisecting the longest edge in B (see [17]).
- **Box Subdivision Scheme:** let $\square\mathbb{R}^d$ be the set of all (axes-parallel) boxes and $\mathbf{Expand}(B)$ return a set of 2^i congruent boxes (for some $i = 1, \dots, d$). This set is defined by introducing i axes-parallel hyperplanes through the center of B . There are $\binom{d}{i}$ ways to choose these hyperplanes. So there are $2^d - 1$ possible expansions.
- **Dyadic Schemes:** We call a scheme is **dyadic** if, for any test cell B , each vertex of a subcell $B' \in \mathbf{Expand}(B)$ is either a vertex of B or the midpoint of an edge of B . The previous two examples are dyadic schemes. The significance of such schemes is that they can be exactly and efficiently computed: recall that a **dyadic number** (or **BigFloat**) is a rational number of the form $m2^n$ ($m, n \in \mathbb{Z}$). The operations $+$, $-$, \times on dyadic numbers are very efficient and division by 2 is exact. Vertices of test cells in a dyadic subdivision tree have the form $\sum_{i=1}^k c_i v_i$ where c_i are dyadic numbers and v_1, \dots, v_k are the vertices of the root. The bit size of the c_i ’s grows linearly with the depth, not exponentially.

3. Subdivision Atlases for Non – euclidean Spaces: Note that if we have a point or ball robot in Euclidean space, then the resolution-exactness of SSS algorithms is indeed trivial. But configuration spaces are rarely Euclidean. Subdivision in non-Euclidean spaces is a nontrivial problem. Likewise, sampling in such spaces is also a research issue (Yershova et al. [26]). Our approach is to borrow the language of charts and atlases from differential geometry. Suppose

⁴ We use the notation in, e.g., [3]. This means there is a set, denoted $\mathbf{set-Expand}(B)$, of subdivisions of B , and $\mathbf{Expand}(B)$ denotes (non-deterministically) any element of this set. We assume $\mathbf{set-Expand}(B)$ is non-empty so that $\mathbf{Expand}(B)$ is a total function.

the metric space X has the property $X = X_1 \cup X_2 \cup \dots \cup X_m$ such that for each X_t , we have an onto homeomorphism $\mu_t : B_t \rightarrow X_t$ where B_t is a test cell, and $\dim(\mu_t^{-1}(X_t \cap X_s)) < d$ for all $t \neq s$. We call each μ_t a **chart** and the set $\{\mu_t : t = 1, \dots, m\}$ is called an **subdivision atlas** for X .

The subdivision of X is thus reduced to subdivision in each X_t , carried out vicariously, via the chart μ_t . More precisely, let $\text{Expand}_t : \square B_t \rightarrow 2^{\square B_t}$ be an expansion scheme where $\square B_t \subseteq 2^{B_t}$ is a set of test cells. Call $\mu_t(B) := \{\mu_t(\gamma) : \gamma \in B\}$ ($B \in \square B_t$) a test cell **induced** by μ_t . Let $\square X$ denote the set of induced test cells. Finally, let $\square X$ denote the disjoint union of the $\square X_t$'s (for all $t = 1, \dots, m$) and let $\text{Expand}_X : \square X \rightarrow 2^{\square X}$ denote the induced expansion defined by $\text{Expand}_X(\mu_t(B)) = \mu_t(\text{Expand}_t(B))$. We have thus achieved subdivision in X . In the following, we might say “ B/α ” (scaling), “ $c(B)$ ” (center), etc. But it should be understood that we mean $\mu(B'/\alpha)$, $\mu(c(B'))$, etc., where $B = \mu(B')$ for some test cell B' .

Call the intersection $X_t \cap X_s$ ($s \neq t = 1, \dots, m$) an **atlas transition** if $\dim(X_t \cap X_s) = d - 1$. For motion planning, recall that two cells are adjacent if they share a face of codimension 1. Thus atlas transitions yield adjacencies between cells in $\square X_s$ and in $\square X_t$. Thus we have two kinds of adjacencies: those that arise from the subdivision of test cells, and from atlas transitions.

4. Subdivision Atlases for S^2 and $SO(3)$: We now give consider two non-Euclidean metric spaces, S^2 and $SO(3)$. We will identify $SO(3)$ with the unit quaternions, $q = (a, b, c, d) = a + \mathbf{i}b + \mathbf{j}c + \mathbf{k}d$ with $a^2 + b^2 + c^2 + d^2 = 1$. Then $SO(3)$ is a metric space with a metric $d(\cdot, \cdot)$ given by the angle $d(q, q') := \cos^{-1}(|q \cdot q'|)$ between two unit quaternions q, q' (see [26]). Likewise, we can treat S^2 as a metric space with the great circle distance.

We are interested in the 2-sphere S^2 because the configuration spaces of several simple rigid robots living in \mathbb{R}^3 is given by $\mathbb{R}^3 \times S^2$: a rod (1D), a cone or bullet (3D), a disc (2D) and a ring (1D). See Fig. 1(a). The ring is interesting because it is the simplest rigid robot that is not simply-connected. Despite the simplicity of their configuration spaces (being 5-DOF), it seems that no complete exact planners have been designed for them. The reason seems to be related to the difficulties of exact algorithms for the “Voronoi Quest” [22]. We are currently designing and implementing a resolution-exact planner for a rod [21]. It would test the practicality of our theory. We can make the rod, ring and disc into **thick robots** by taking their Minkowski sum with a 3D-ball. But we expect that any SSS planner for thin robots will extend relatively easily to thick analogues (similar to the situation in the plane [12]).

Note that S^2 is not a subgroup, but a quotient group of $SO(3)$ (this is clear from the Hopf fibration of $SO(3)$ [26]). To create a subdivision atlas for S^2 , let $I^3 = I \times I \times I$ be the 3-cube where $I = [-1, 1]$. Its boundary ∂I^3 can be subdivided into 6 squares denoted $S_{\pm\delta}$ where $\delta \in \{x, y, z\}$. See Fig. 1(b). For instance, $S_{+z} = \{(x, y, 1) : x, y \in I\}$ and $S_{-z} = \{(x, y, -1) : x, y \in I\}$. We obtain a subdivision chart of S^2 by using 6 charts: $\mu_{\pm\delta} : S_{\pm\delta} \rightarrow S^2$ where $\mu_{\pm\delta}(q) = q/\|q\|$ where $\|q\|$ is the Euclidean norm. Note that $\mu_{\pm\delta}$ does not depend

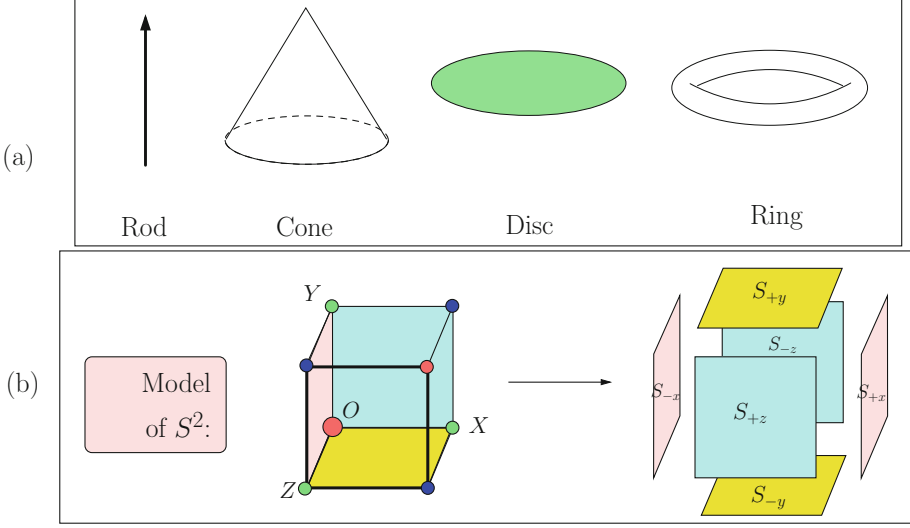


Fig. 1. 3D rigid robots with 5-DOF (Color figure online)

on $\pm\delta$ and so there is really only one function $\mu(q)$ for all the charts. The inverse map $\mu^{-1} : S^2 \rightarrow \partial I^3$ is also easy: $\mu^{-1}(\gamma) = \gamma/\|\gamma\|_\infty$ where $\|q\|_\infty$ is the infinity norm.

Call this construction the **cubic atlas** for S^2 . We now construct a similar cubic atlas for $SO(3)$ (it was mentioned in Nowakiewicz [14]).

Begin with the 4-cube I^4 : it has eight 3-dimensional cubes as faces. After identifying the opposite faces, we have four faces denoted $C_w^3, C_x^3, C_y^3, C_z^3$ (see Fig. 2). We define the chart: $\mu_t : C_t^3 \rightarrow SO(3)$ given by $\mu_t(q) = q/\|q\|$ (where $t = w, x, y, z$). As noted above, we must keep track of the adjacencies that arise from our atlas. In our case, this arise from the identification of antipodal points, $q \sim -q$ in S^3 . In our cubic model, this information is transferred to identification of 2-dimensional faces among of C_t^3 .

A chart $\mu : B_t \rightarrow X_t$ is **good** if there exists a **chart constant** $C_0 > 0$ such that for all $q, q' \in B_t$, $1/C_0 \leq \frac{d_X(\mu(q), \mu(q'))}{\|q - q'\|} \leq C_0$ where $d_X(\cdot, \cdot)$ is the metric in X_t . The subdivision atlas is **good** if there is an **atlas constant** C_0 that is common to its charts. Note that good atlases can be used to produce nice sampling sequences: since our test cells are Euclidean sets, we can exploit sampling of Euclidean sets. Alternatively, we can produce a “uniform” subdivision into sufficiently test cells, and pick the center of each test cell as sample point. The following is immediate:

Lemma 1. *The cubic subdivision atlases for S^2 and $SO(3)$ are good.*

5. Soft Predicates: We define soft predicates in the space X . Let $Y \subseteq X$. We call $\tilde{C} : \Box X \rightarrow \{+1, 0, -1\}$ a **soft classifier** of Y if it satisfies two properties:

- (conservative) for all $B \in \Box X$, $\tilde{C}(B) \neq 0$ implies $\tilde{C}(B) = C_Y(\mu(B))$.

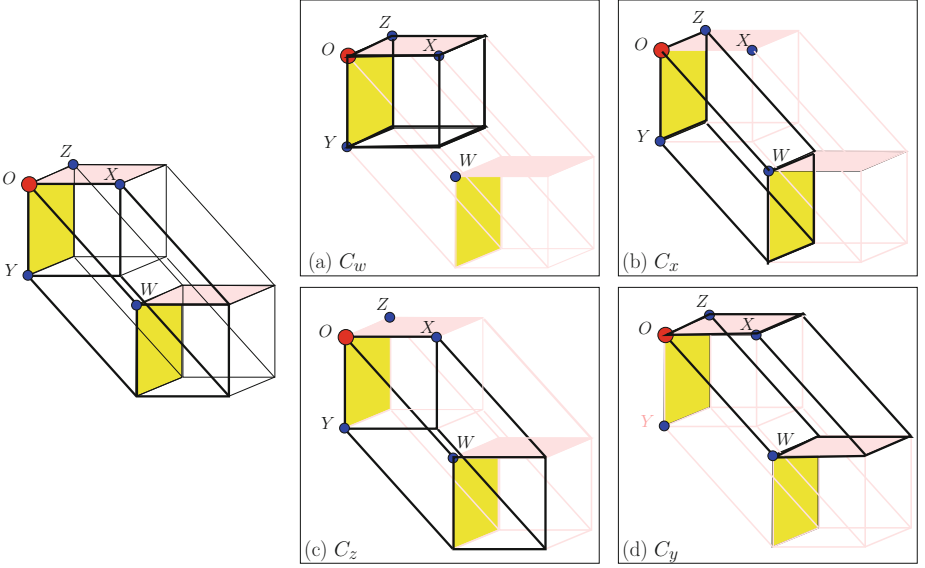


Fig. 2. The Cubic Atlas for $SO(3)$ (Color figure online)

– (convergent) if $q = \lim_{i \rightarrow \infty} B_i$ then $\tilde{C}(B_i) = C_Y(\mu(q))$ for i large enough.

For resolution-exactness, we need another property: a soft classifier \tilde{C} is **effective** if there is an **effectivity factor** $\sigma > 1$ such that if $\tilde{C}(B) = +1$ then $\tilde{C}(B/\sigma) = +1$. For instance, we see that effectivity of \tilde{C} implies it is convergent. Note we do not require $C_Y(B) = -1$ to imply $C_Y(B/\sigma) = -1$.

Given $\alpha, \beta \in X$ and $Y \subseteq X$, the exact planning problem is finding a path from α to β in Y if they belong to the same connected component of Y , and **NO-PATH** otherwise. The resolution-exact version will require a connection between the metric in configuration space X and the metric in physical space \mathbb{R}^k . For this purpose, recall the concepts of footprint and separation of Euclidean sets (see [20, 25]): Our robot R_0 lives in physical space \mathbb{R}^k ($k = 2$ or $k = 3$) amidst an obstacle set $\Omega \subseteq \mathbb{R}^k$. The **footprint map** is $Fprint : Cspace \rightarrow 2^{\mathbb{R}^k}$ where $Cspace = Cspace(R_0)$ is the configuration space. Intuitively, $Fprint(\gamma) \subseteq \mathbb{R}^k$ is the physical space occupied by robot R_0 in configuration γ . The **clearance function**, $Cl : Cspace \rightarrow \mathbb{R}_{\geq 0}$ is given by $Cl(\gamma) := \text{Sep}(Fprint(\gamma), \Omega)$, where $\text{Sep}(A, B) := \inf \{\|a - b\| : a \in A, b \in B\}$ is the **separation** between two Euclidean sets in \mathbb{R}^k . We say γ is **free** if $Cl(\gamma) > 0$. A **motion** is a continuous function $\pi : [0, 1] \rightarrow Cspace$; its **clearance** is $\inf \{Cl(\pi(t)) : t \in [0, 1]\}$. Call π a **path** if it has positive clearance.

In our abstract formulation, we postulate the existence of a continuous function $Cl : X \rightarrow \mathbb{R}$ without reference to the underlying footprint or Ω . Moreover, this is called a **generalized clearance function** because we now allow negative clearance, interpreted as “penetration depth” (e.g., [8, 27]). Call Cl a **clearance**

function for Y if $Y = \{\gamma \in X : C\ell(\gamma) > 0\}$. We then consider interval functions of the form

$$\Box C\ell : \Box X \rightarrow \Box \mathbb{R}.$$

(Recall that $\Box \mathbb{R}$ is a set of intervals.) We call $\Box C\ell$ a **conservative approximation** of $C\ell$ if $\Box C\ell(B) \neq 0$ implies $\Box C\ell(B) = C\ell(B)$ for all $B \in \Box X$. We say $\Box C\ell$ **converges to** $C\ell$ if whenever $\gamma = \lim_{i \rightarrow \infty} B_i$, then $\Box C\ell(B_i) = C\ell(\gamma)$ for i large enough. Finally, $\Box C\ell$ is called a **box function** for $C\ell$ if it is conservative and convergent relative to $C\ell$.

Note that $\Box C\ell$ defines a classification function $\tilde{C} : \Box X \rightarrow \{+1, 0, -1\}$ where $\tilde{C}(B) = 0$ iff $0 \in \Box C\ell(B)$; otherwise, $\tilde{C}(B) = \text{sign}(\Box C\ell(B))$ (either $+1$ or -1). The following is immediate:

Lemma 2. *Let $C\ell : X \rightarrow \mathbb{R}$ be a clearance function for $Y \subseteq X$, and suppose $\Box C\ell : \Box X \rightarrow \Box \mathbb{R}$ is a box function for $C\ell$.*

Then the classification function $\tilde{C} : \Box X \rightarrow \{+1, 0, -1\}$ defined by $\Box C\ell$ is a soft classifier of Y .

6. Soft Predicates for Complex Robots: An example of a robot with complex geometry is the gear robot of Zhang et al. [27]. Such robots pose difficulties for exact algorithms. We show that soft predicates for complex robots can be decomposed. Let $G_0 \subseteq \mathbb{R}^2$ be the gear robot. We write it as a union $G_0 = \cup_{j=1}^m T_j$ of triangles T_j . The free space of G_0 can be written as the intersections of the freespaces of T_j , provided the T_j 's are expressed in a common coordinate system. This proviso requires a slight generalization of the soft predicate for triangles in [8]. The next theorem shows how to obtain a soft predicate for G_0 from those of the T_j 's:

Theorem 1 (Decomposability of Soft Predicates). *Suppose $Y = Y_1 \cap \dots \cap Y_m$. If $\tilde{C}_i : \Box X \rightarrow \{+1, 0, -1\}$ is a soft classifier for Y_i , then the following is a soft classifier for Y :*

$$\tilde{C}(B) := \begin{cases} +1 & \text{if } (\forall j)[\tilde{C}_j(B) = +1] \\ -1 & \text{if } (\exists i)[\tilde{C}_i(B) = -1], \\ 0 & \text{else.} \end{cases}$$

If each \tilde{C}_j 's has effectivity factor σ , then $\tilde{C}(B)$ has effectivity factor σ .

Proof. We easily check that $\tilde{C}(B)$ is safe. To show convergence, suppose that $B_i \rightarrow p$. If $p \in Y$, then $p \in Y_j$ for each j . That means $\tilde{C}_j(B_i) = 1$ for i large enough. I.e., $\tilde{C}(B_i) = 1$ for i large enough. This proves $\lim_{i \geq 0} \tilde{C}(B_i) = +1 = C(p)$. If $p \in X \setminus \bar{Y}$, then $p \in X \setminus \bar{Y}_j$ for some Y_j . This means $\tilde{C}_j(B_i) = -1$ for i large enough, and therefore $\tilde{C}(B_i) = -1$ for i large enough. Again, $\lim_{i \geq 0} \tilde{C}(B_i) = -1 = C(p)$. Suppose $p \in \partial Y$. Then $p \in \partial Y_j$ for some j and for all $k \neq j$, $p \in \bar{Y}_k$. That implies that $\tilde{C}_j(B_i) \in \{+1, 0\}$ and Again, $\lim_{i \geq 0} \tilde{C}(B_i) = 0 = C(p)$. This proves the softness of the predicate $\tilde{C}(B)$.

Assume each \tilde{C}_j has an effectivity factor $\sigma > 1$. Let $C_j(B)$ be the exact box predicate for Y_j . Suppose $C(B)$ is free. This means each $C_j(B)$ is free. By definition of effectivity, each $\tilde{C}_j(B/\sigma)$ is free. Hence $\tilde{C}(B/\sigma)$ is free. **Q.E.D.**

4 Axiomatic Properties of SSS

We prove general properties of SSS planners using basic assumptions which we call **axioms**. The proofs are instructive because they reveal how these axioms and properties of SSS are used. We introduce 5 axioms, beginning with these four:

- **(A0: Softness)**
 \tilde{C} is a soft classifier for $\mathcal{C}free = \{\gamma \in X : C\ell(\gamma) > 0\}$.
- **(A1: Bounded dyadic expansion)**
 The expansion scheme is dyadic, and there is a constant $D_0 > 2$ such that $\text{Expand}(B)$ splits B into at most D_0 subcells, each with at most D_0 vertices, with the ratio $\ell(B)/w(B)$ at most D_0 .
- **(A2: Clearance is Lipschitz)**
 There is a constant $L_0 > 0$ such that for all $\gamma, \gamma' \in \mathcal{C}free$, $|C\ell(\gamma) - C\ell(\gamma')| < L_0 \cdot d_X(\gamma, \gamma')$ where $d_X(\cdot, \cdot)$ is the metric on X .
- **(A3: Good Atlas)**
 The subdivision atlas has a atlas constant $C_0 \geq 1$.

Note that these axioms concern about the clearance $C\ell : X \rightarrow \mathbb{R}$, classification $\tilde{C} : \square X \rightarrow \square \mathbb{R}$ and the **Expand** scheme. We have no axioms about **getNext** because the needed properties are embedded in the SSS framework, namely **getNext** returns a MIXED-leaf with length $\ell(B) \geq \varepsilon$ if any exist. Recall that in general, $B \in \square X$ is induced via our charts μ_t , and so the metrics such as $\ell(B)$ and $w(B)$ are induced from the Euclidean sets B' where $\mu_t(B') = B$, i.e., $\ell(B)$ refers to $\ell(\mu_t^{-1}(B)) = \ell(B')$, etc. Note that (A1) does not bound the aspect ratio $r(B)/r_0(B)$ and these may be unbounded (slivers can arise). (A2) relates clearance to the metric on X : this is a non-trivial axiom in non-Euclidean spaces. (A3) is necessary since subdivision of X is done via charts $\{\mu_t : t = 1, \dots, m\}$.

Theorem 2 (Halting). *Every SSS planner halts. When a path is output, it is valid.*

Proof. In any infinite path $\{B_i : i \geq 0\}$, (A1) implies $\lim_i \ell(B_i) \rightarrow 0$. Since we do not subdivide a box if “ $\ell(B) < \varepsilon$ ”, halting is assured. At termination, we either report a path or output NO-PATH. If we report a path, it meant we found a “free channel” from $B(\alpha)$ to $B(\beta)$. We check that SSS ensures that the channel is truly free: the dyadic scheme (A1) ensures that test cells are computed exactly, and thus adjacencies are computed without error. Each cell in the channel is classified as **FREE**, and this truly free because (A0) ensures a conservative classifier \tilde{C} . Finally, output paths are valid as they are contained in free channels. **Q.E.D.**

This theorem depends only on (A0) and (A1). Although our goal in (A0) is soft classifiers, it is a useful preliminary to consider the case where \tilde{C} is the exact classifier. In this case, we say our planner is **exact**. This preliminary step is captured in the next result:

Theorem 3 (Exact SSS). *Assuming an exact SSS planner:*

- (a) *If there is no path, the planner outputs NO-PATH.*
- (b) *If there is a path with clearance $\geq 2C_0D_0\varepsilon L_0$, the planner outputs a path.*

Proof. Part(a) is essentially the contrapositive of the above Halting theorem. For part(b), let \mathcal{T} be the subdivision tree at termination. The nodes of \mathcal{T} are induced cells of $\square X$. Each $B \in \square X$ comes from an Euclidean test cell $\mu^{-1}(B) \in \square \mathbb{R}^d$ for some chart μ . Euclidean distance $\|\cdot\|_2$ in $\mu^{-1}(B)$ and the metric $d_X(\cdot, \cdot)$ of X are related via the chart constant C_0 . Let $\pi : [0, 1] \rightarrow X$ be a path from α to β with clearance $2C_0D_0\varepsilon L_0$. By way of contradiction, suppose SSS outputs NO-PATH. This implies that every mixed leaf satisfies $\ell(B) < \varepsilon$. Consider the set \mathcal{A} of leaves of \mathcal{T} that intersect $\pi[0, 1]$ (the range of π). If $B \in \mathcal{A}$, there exists $t \in [0, 1]$ such that $\pi(t) \in B$. This implies B is either free or mixed. We claim that B is free. If B is mixed, then $\ell(\mu^{-1}(B)) < \varepsilon$ and there is a point $p' \in B$ that is semi-free. Thus $\|p - q\|_2 < D_0\varepsilon$ for any two Euclidean points p, q in $\mu^{-1}(B)$. Using the chart μ , we conclude that $d_X(\mu(p), \mu(q)) < C_0D_0\varepsilon$. Therefore $d_X(\pi(t), p') \leq d_X(\pi(t), c(B)) + d_X(c(B), p') < 2C_0D_0\varepsilon$. By (A2), $|C\ell(\pi(t)) - C\ell(p')| < 2C_0D_0\varepsilon L_0$. Thus $C\ell(p') > C\ell(\pi(t)) - 2C_0L_0\varepsilon L_0 \geq 0$, i.e., p' is free. This contradicts the assumption that p' is semi-free; so B must be free. Thus we obtain a channel of free cells from α to β using cells in \mathcal{A} . The existence of such a channel implies that our union-find data structure in SSS would surely detect a path. **Q.E.D.**

Our goal is not to produce the sharpest constants but to reveal their roles in our framework. Notice that this theorem has a gap: if the optimal clearance lies in $(0, 2C_0D_0\varepsilon L_0)$, the exact Planner may output either a path or NO-PATH.

7. Three Desiderata: The literature invariably assumes exactness in its analysis, such as in Theorem 3. But there are three desiderata beyond such a result. The first is to remove the exactness assumption. Second, we would like to strength the hypothesis of Theorem 3(a) to “*if there is no path with clearance ε/K* ” for some input-independent $K > 1$. In other words, NO-PATH ought to mean no path of “sufficient clearance”, a reasonable idea in view of the inherent uncertainty of physical devices. Third, we may want to strengthen the conclusion of Theorem 3(b) so that the output path has clearance $\geq \varepsilon/K$.

The first desiderata calls for soft predicates. We say that the SSS planner is **effective** if the soft predicate \tilde{C} has an effectivity constant $\sigma > 1$. In applications, it is useful to assume that \tilde{C} is **isotone**⁵ i.e., $\tilde{C}(B) \neq 0$ and $B' \subseteq B$ implies $\tilde{C}(B') \neq 0$. The proof of part(b) in the previous theorem can be extended to show:

⁵ This term is from the interval literature. Though not strictly necessary, but it simplifies some arguments.

Theorem 4 (Effective SSS). *Assume an SSS planner with effectivity $\sigma > 1$.*

(a) *If there is no path, the planner outputs NO-PATH.*

(b) *If there is a path with clearance $\geq C_0 D_0 \varepsilon (1 + \sigma) L_0$, the planner outputs a path.*

The indeterminacy gap is slightly widened to $(0, C_0 D_0 \varepsilon (1 + \sigma) L_0)$ by the soft predicate.

The second desiderata amounts to asking for a resolution-exact planner. As defined in the Introduction, such planners has an accuracy constant $K > 1$. So we seek to narrow indeterminacy gap by raising the gap lower bound from 0 to ε/K . The fundamental issue is to infer a lower bound on the clearance of a path inside a free channel. This requires a new axiom:

– **(A4: Translational Cells)**

There is a constant $K_0 > 0$ such that if $B \in \square X$ is free, then its inner center $c_0 = c_0(B)$ has clearance $C\ell(c_0) \geq K_0 \cdot r_0(B)$. Such cells are said to be **translational**.

Like (A2), axiom (A4) relates the clearance to the metric space (via the chart μ). The “translational” terminology is based on the analogy that if X is purely translational, then (A4) is true. But in fact, it will be true in all the common motion planning scenarios.

Theorem 5 (Resolution-Exact SSS). *Assuming (A0–A4), SSS planners are resolution-exact.*

This proof is more involved and will appear in the full paper. The third desiderata requires that we strengthen condition (P) in the definition of resolution-exactness as follows:

(P’) If there is a path of clearance $K\varepsilon$, then return a path of clearance ε/K .

See [20, 25] where (P’) is used. The combination of (P) and (N) implies that whenever a path is output, we are assured that *there exists a path of clearance ε/K* . So (P’) attempts to turn this existential guarantee into a constructive guarantee. Unfortunately, this requires additional effort as in [20, 25]. We will not attempt an axiomatic treatment to achieve (P’) here.

5 What About Exactness?

Can the SSS framework produce⁶ exact algorithms? The answer is yes, but as always, only in the algebraic case. First, here is a non-solution: *using an exact SSS planner with the resolution parameter $\varepsilon = 0$* . Using an exact SSS re-introduces the need for algebraic computation. By setting $\varepsilon = 0$, indeterminacy

⁶ We are indebted to Steve LaValle for asking this question at the IROS 2011 Workshop in San Francisco.

is removed, but at a high price: if there is no path, then SSS will not halt. Even if there is a path, SSS may not halt; but this could be fixed by imposing a “generalized BFS” property on `getNext`. For these reasons, our normal formulation of SSS requires $\varepsilon > 0$ and $K > 1$. We now present a solution within the SSS framework using an effective soft predicate. The idea is to exploit the theory of constructive zero bounds [24].

Proposition 3. *If R_0, Ω are semi-algebraic sets, and the parameters α, β are algebraic, then there is an effectively computable number $\delta = \delta(R_0, \Omega, \alpha, \beta) > 0$ such that: if there is a path from α to β , then there is one with clearance δ .*

One way to derive such a δ is to use the general retraction theory in [15, 16, 23]: there is a “retract” $V \subseteq \mathcal{C}free = \mathcal{C}free(R_0, \Omega)$ and a retraction map $Im : \mathcal{C}free \rightarrow V$ with this property: for all $\alpha, \beta \in \mathcal{C}free$, we have that α, β are path-connected in $\mathcal{C}free$ iff $Im(\alpha), Im(\beta)$ are path-connected in V . Here V is a Voronoi diagram and we can subdivide V into semi-algebraic Voronoi cells. The minimum clearance on V serves as δ , and this can be lower bounded using the degree and height of the semi-algebraic sets [5]. The upshot is this:

Theorem 6. *Suppose we have a resolution-exact planner with accuracy constant $K > 1$. If we choose ε to be $\delta(R_0, \Omega, \alpha, \beta)/K$, then our SSS planner is exact: it outputs NO-PATH iff there is no path.*

6 Conclusion

In this paper, we described the SSS framework for designing resolution-exact algorithms. We argued [25] that it shares many of the attractive properties of the successful PRM framework. Subdivision algorithms are as old as the history of path planning [4]. But to our knowledge, the simple properties of soft classifiers have never been isolated, nor have concepts of resolution-limited computation been carefully scrutinized. We believe focus on these “simple ideas” will open up new classes of algorithms that are practical *and* theoretically sound. This has implications beyond motion planning. Our work in SSS is not just abstract, as we have validated these ideas in several non-trivial planners [12, 13, 20].

There are many open questions concerning SSS framework. Perhaps the biggest challenge for SSS is the conventional wisdom that PRM can provide practical solutions for problems with high degrees-of-freedom, while resolution methods can only reach medium DOF, generally regarded as 5–8 DOF (Choset et al. [9, p. 202]). Likewise, in Nowakiewicz [14], “[subdivision methods] are not suitable for 6-DOF rigid body motion planning due to the large expected number of cells ... We believe that in high-dimensional spaces it has little practical value.”

The other major challenge is a theoretical one: how to do complexity analysis of adaptive subdivision in Motion Planning (cf. [18]). Here are some other topics:

- The current SSS framework detects NO-PATH by exhaustion. It is a challenge to design efficient techniques (related to maintaining homology) to allow fast detection of NO-PATH. A promising new work by Kerber and Cabello [7] shows how to do this when $\mathcal{C}space = \mathbb{R}^2$.

- Beyond kinematic spaces, subdivision in state spaces for kinodynamic planning seems quite challenging.
- Design and analysis of good adaptive search strategies, including the Voronoi heuristic [23], or randomized or hybrid ones. E.g., efficient updates for dynamic A-star search [1] seems open.

Acknowledgments. I am indebted to Yi-Jen Chiang, Danny Halperin, Steve LaValle, and Vikram Sharma for many helpful discussions.

References

1. Barbehenn, M., Hutchinson, S.: Toward an exact incremental geometric robot motion planner. In: *Proceedings of Intelligent Robots and Systems 1995*, vol. 3, pp. 39–44 (1995). 1995 IEEE, RSJ International Conference, Pittsburgh. PA, USA, pp. 5–9, August 1995
2. Basu, S., Pollack, R., Roy, M.-F.: *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics, vol. 10, 2nd edn. Springer, Heidelberg (2006)
3. Beyersdorff, O., Köbler, J., Messner, J.: Nondeterministic functions and the existence of optimal proof systems. *Theor. Comput. Sci.* **410**(38–40), 3839–3855 (2009)
4. Brooks, R.A., Lozano-Perez, T.: A subdivision algorithm in configuration space for findpath with rotation. In: *Proceedings of the 8th IJCAI*, San Francisco, CA, USA, vol. 2, pp. 799–806. Morgan Kaufmann Publishers Inc. (1983)
5. Brownawell, W.D., Yap, C.K.: Lower bounds for zero-dimensional projections. In: *2009 34th International Symposium on Symbolic and Algebraic Computation (ISSAC 2009)*, pp. 79–86. KIAS, Seoul, Korea, 28–31 July 2009
6. Burr, M., Krahmer, F.: SqFreeEVAL: an (almost) optimal real-root isolation algorithm. *J. Symb. Comput.* **47**(2), 153–166 (2012)
7. Cabello, S., Kerber, M.: Semi-dynamic connectivity in the plane. In: *Algorithms and Data Structure Symposium (WADS 2015)* (to appear, 2015). [arXiv:1502.03690](https://arxiv.org/abs/1502.03690)
8. Chiang, Y.-J., Yap, C.: Numerical subdivision methods in motion planning. 2011 Poster, IROS Workshop on Progress and Open Problems in Motion Planning, San Francisco, 30 September 2011
9. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston (2005)
10. Kavraki, L., Švestka, P., Latombe, C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **12**(4), 566–580 (1996)
11. LaValle, S.M.: *Planning Algorithms*. Cambridge University Press, Cambridge (2006)
12. Luo, Z., Chiang, Y.-J., Lien, J.-M., Yap, C.: Resolution exact algorithms for link robots. In: *2014 Proceedings of the 11th WAFR*, Boğaziçi University, Istanbul, Turkey, 3–5 August 2014. (to appear in a Springer Tracts in Advanced Robotics (STAR))
13. Luo, Z., Yap, C.: Resolution exact planner for non-crossing 2-link robot (2015, Submitted)

14. Nowakiewicz, M.: MST-based method for 6DOF rigid body motion planning in narrow passages. In: 2010 Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, pp. 5380–5385, 18–22 October 2010
15. Ó'Dúnlaing, C., Sharir, M., Yap, C.K.: Retraction: a new approach to motion-planning. *ACM Symp. Theor. Comput.* **15**, 207–220 (1983)
16. Ó'Dúnlaing, C., Yap, C.K.: A “retraction” method for planning the motion of a disc. *J. Algorithms* **6**, 104–111 (1985)
17. Rivara, M.-C.: Lepp-bisection algorithms, applications and mathematical properties. *Appl. Numer. Math.* **59**(9), 2218–2235 (2009)
18. Sagraloff, M., Yap, C.K.: A simple but exact and efficient algorithm for complex root isolation. In: Emiris, I.Z. (ed.) 36th International Symposium on Symbolic and Algebraic Computation, San Jose, California, pp. 353–360, 8–11 June 2011
19. Sharma, V., Yap, C.: Near optimal tree size bounds on a simple real root isolation algorithm. In: 2012 37th International Symposium on Symbolic and Algebraic Computation (ISSAC 2012), Grenoble, France, pp. 319–326, 22–25 July 2012
20. Wang, C., Chiang, Y.-J., Yap, C.: On soft predicates in subdivision motion planning. In: 2014 Computational Geometry: Theory and Applications, Special Issue for SoCG, Rio de Janeiro, Brazil, 17–20 June 2013
21. Wei, Z., Yap, C.: Soft subdivision planner for a rod (2015. in preparation)
22. Yap, C., Sharma, V., Lien, J.-M.: Towards exact numerical voronoi diagrams. In: 2012 9th Proceedings of the International Symposium of Voronoi Diagrams in Science and Engineering (ISVD), Rutgers University, NJ, pp. 2–16. IEEE, 27–29 June 2012. Invited Talk
23. Yap, C.K.: Algorithmic motion planning. In: Schwartz, J., Yap, C. (eds.) *Advances in Robotics. Algorithmic and Geometric Issues*, vol. 1, pp. 95–143. Lawrence Erlbaum Associates, Hillsdale (1987)
24. Yap, C.K.: Robust geometric computation. In: Goodman, J.E., O'Rourke, J. (eds.) *Handbook of Discrete and Computational Geometry*, 2nd edn, pp. 927–952. Chapman & Hall/CRC, Boca Raton (2004)
25. Yap, C.K.: Soft subdivision search in motion planning. In: Aladren, A., et al. (eds.) In: Proceedings of 1st Workshop on Robotics Challenge and Vision (RCV 2013), A Computing Community Consortium (CCC) Best Paper Award, Robotics Science and Systems Conference (RSS 2013), Berlin (2013). [arXiv:1402.3213](https://arxiv.org/abs/1402.3213)
26. Yershova, A., Jain, S., LaValle, S.M., Mitchell, J.C.: Generating uniform incremental grids on $SO(3)$ using the Hopf fibration. *IJRR* **29**(7), 801–812 (2010)
27. Zhang, L., Kim, Y.J., Manocha, D.: Efficient cell labeling and path non-existence computation using C-obstacle query. *Int. J. Robot. Res.* **27**(11–12), 1325–1349 (2008)
28. Zhu, D., Latombe, J.-C.: New heuristic algorithms for efficient hierarchical path planning. *IEEE Trans. Robot. Autom.* **7**, 9–20 (1991)

Frontiers in Algorithmics

9th International Workshop, FAW 2015, Guilin, China,

July 3-5, 2015, Proceedings

Wang, J.; Yap, C.K. (Eds.)

2015, XI, 335 p. 54 illus., Softcover

ISBN: 978-3-319-19646-6