On Soft Predicates in Subdivision Motion Planning*

Yi-Jen Chiang† Chee Yap‡

Polytechnic Institute of NYU New York University

Abstract

We propose to design new algorithms for motion planning problems using the well-known Domain Subdivision paradigm, coupled with "soft" predicates. Unlike the traditional exact predicates in computational geometry, our primitives are only exact in the limit. We introduce the notion of **resolution-exact algorithms** in motion planning: such an algorithm has an "accuracy" constant $K \geq 1$, and takes an arbitrary input "resolution" parameter $\varepsilon > 0$ such that: if there is a path with clearance $K\varepsilon$, it will output a path with clearance ε/K; if there are no paths with clearance ε/K, it reports "no path". Besides the focus on soft predicates, our framework also admits a variety of global search strategies including forms of the A* search and also probabilistic search.

Our algorithms are practical, easy to implement, and have adaptive complexity. Our deterministic and probabilistic strategies can achieve if-and-only-if form of completeness, unlike probabilistic completeness. We will give examples of our approach, including problems that currently have no complete exact description, complex robot geometry and high-degree freedom robots. The approach is widely applicable to other related problems, illustrating what we call "numerical Computational Geometry".

*This work is supported by NSF Grant CCF-0917093 and DOE Grant DE-SC0004874.

†Chiang is with the Department of Computer Science and Engineering, Polytechnic Institute of NYU, yjc@poly.edu.

‡Yap is with the Department of Computer Science, NYU, New York, NY 10012, USA, yap@cs.nyu.edu.

1 Introduction

A central problem of robotics is motion planning [4, 22, 23, 10]. In the early 80's there was strong interest in this problem among computational geometers [18, 29]. This period saw the introduction of strong algorithmic techniques with complexity analysis, and the careful investigation of the algebraic C-space. We introduced the retraction method in [25, 30, 31, 35]. In a survey of algorithmic motion planning [36], we also established the universality of the retraction method. This method is now commonly known as the road map approach, popularized by Canny [7] who showed that its algebraic complexity is in single exponential time. Typical of algorithms in Computational Geometry, these exact motion planning algorithms assume a computational model in which exact primitives are available in constant time. Implementing these primitives exactly is non-trivial (certainly not constant time), involving computation with algebraic numbers. In the 90's, interest shifted back to more practical techniques, such as the well-known **probabilistic roadmap method** (PRM) [20]. The idea is to compute a partial road map by random sampling of the configuration space. PRM offers a computational framework for a class of algorithms. Moreover, many[1] variants of the basic framework have been developed. In an invited talk at the recent workshop[2] on open problems in motion planning, J.C. Latombe stated that the major open problem of such **Sampling Methods** is that they do not know how to terminate when there is no free path. In practice, one would simply time-out the algorithm, but this leads to issues such as the "Climber's Dilemma" [19, p. 4] that arose in the work of Bretl (2005). We call this the **halting problem** of PRM, viewed as the ultimate form of what is popularly known as the "Narrow Passage Problem" [10, p. 216]. Latombe's talk suggested promising approaches such as Lazy PRM [3]. An excellent account of the state of the art in motion planning is found in Choset et al [10].

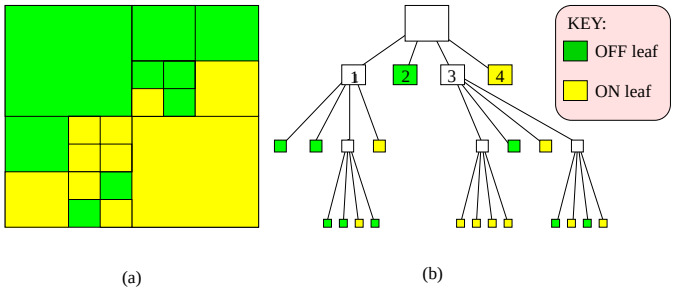

Figure 1: (a) Subdivision of a region (yellow). (b) Its Subdivision Tree

In this paper, we turn to another popular approach [42] for motion planning, which we call **Subdivision Methods**. The general idea is to subdivide some bounded domain B_0, typically a subset of $\mathbb{R}^d$. In motion planning, the domain is a subset of configuration space. In its simplest form, the subdivision of B_0 can be represented as a **subdivision tree**, which is a generalization of bisection search ($d = 1$) or quad-trees ($d = 2$), as illustrated in Figure 1. An early reference for this approach is Brooks and Lozano-Perez [5]. Some more recent subdivision references include [42, 2, 41, 12, 26]. Manocha's Group at UNC has been very active and highly successful in producing practical subdivision algorithms for a variety of tasks, many critical in motion planning [34, 33]. Although domain subdivisions are sometimes known as "cell decomposition" (e.g., [42]), we reserve "cell decomposition" for methods of partitioning configuration space into cells with finite combinatorial-algebraic description (e.g., [28, 36]). In contrast to such cells, the boxes in our subdivision need not have any direct correlation to combinatorial complexity of the problem. Such boxes are clearly related to "resolution". But more "guided subdivision" that takes into account combinatorial complexity are seen in [41, 42]. This "combinatorial-cum-resolution" complexity leads to new complexity analysis that is not possible in the traditional exact algorithms. Recent examples of such complexity analysis are seen in our related work [27, 13, 32, 6].

¶1. What is New in This Paper. The most distinctive contribution of this paper is the explicit introduction of **soft primitives** into motion planning (and more generally, into Subdivision Methods). Briefly, soft primitives are just suitable numerical approximations of exact (hard) primitives. Soft primitives are ideally suited in a subject like motion planning where the standard notion of exactness of computational geometry makes little sense for robot sensors and mechanical systems that have inherent accuracy limits. As we shall

[1] A partial list includes Expansive-Spaces Tree planner (EST), Rapidly-exploring Random Tree planner (RRT), and Sampling-Based Roadmap of Trees planner (SRT).

[2] IROS 2011 Workshop on Progress and Open Problems in Motion Planning, September 30, 2011, San Francisco. `http://www.cse.unr.edu/robotics/tc-apc/ws-iros2011`.

show, *the development of such primitives will lead to new theoretically sound algorithms that are highly practical*. Based on the above work [27, 13, 32, 6], we conjecture that these algorithms are also provably efficient. Currently, it seems that practitioners have to choose between sound algorithms that are not practical and heuristic methods that seems to work well in practice. We stress that Subdivision Methods using soft primitives are easy to implement *exactly*. Implementing a primitive exactly does not mean the primitive itself is exact; but it does means that our implemented algorithm will not suffer from "implementation gaps" that could render theoretical correctness/exactness properties worthless. This has been a bugbear of many algorithms of computational geometry. The basis for our general approach is discussed in [39, 40]. Our second main contribution is a systematic treatment of resolution completeness. The concept of "resolution completeness" is widely used in the motion planning literature [10], but rarely analyzed. We introduce the concept of **resolution-exact** (or ε**-exact**) **algorithms**, and analyze their properties. There are subtleties in this concept – for instance, we will prove that there is inherent indeterminacy in such algorithms.

2 On Numerical Subdivision Algorithms

Computational Geometry until now has almost exclusively concentrated on **Exact Methods**. The many attractive features of exact algorithms are well-known. But there is limitation in its computational model based on hard primitives. The inability of Exact Methods to have wider impact on robotics and those fields of Computational Sciences and Engineering (CS&E) where geometric reasoning is dominant ought to call for a re-examination of our assumptions. We argue that Subdivision Algorithms, when[3] combined with soft primitives, offer a pathway for Computational Geometers to design new algorithms that are theoretically sound and can have an impact on the above fields. Our soft primitives do not entail any error analysis in the style of numerical analysis; rather, we rely on some form interval method [24]. It is hard to ignore the many advantages in this alternative view of Computational Geometry [40]. We now outline these advantages for motion planning.

Subdivision Methods share with Sampling Methods many advantages over Exact Methods. They both lead to algorithms that are very easy to implement and modify. Implementability is a property that should be valued more. Modifiability is important in practice: imagine deploying such algorithms in physical robots where non-algorithmic considerations may need to be accounted for. Exact algorithms are not so flexible. A less obvious advantage is that both methods are very forgiving – you can implement the primitives of Sampling or Subdivision Methods imprecisely, and the algorithms would not necessarily suffer catastrophic breakdown (crashing, entering an infinite loop, etc) [21]. In contrast, exact algorithms are infamous for their sensitivity to numerical errors; this is the key motivation for the field of EGC [38]. Despite these similarities, the big difference between Sampling and Subdivision is that the latter can detect non-existence of free paths [41]. In view of Latombe's keynote talk above, this ability is significant. Some authors make sampling methods resolution-complete by ensuring a certainly sample density as in [9]. Nevertheless, the guarantees are much weaker than in Subdivision.

A practical advantage of Subdivision and Sampling Methods is that they allow the possibility of finding free paths *before* the entire free configuration space has been fully explored. In contrast, known Exact Methods have an expensive pre-processing phase which amounts to computing a complete description of the free configuration space (C_{free}) before any path searching can even begin.

Exact algorithms typically returns a combinatorial path defined by implicit constraints; this path must be converted into a trajectory (i.e., parametrized path in C_{free}) before a physical robot can follow the path. Thus, trajectory planning requires a subsequent numerical stage that is rarely discussed in exact algorithms. In Subdivision Methods, these 2 stages are integrated in a natural way.

One disadvantage of numerical primitives is that they are only complete in the limiting sense. But in robotics, where the data and control have limited precision, this is no serious restriction — numerical

[3] We note that Subdivision Algorithms could also be combined with hard primitives, as seen in Section III. But the full power of Subdivision Methods reveals itself when we exploit soft primitives.

methods can be certified up to any desired precision. But numerical methods are more general than exact approaches in the sense that they are applicable to analytic (non-algebraic) problems where exact solutions are generally unknown (see [8]).

3 Subdivision Motion Planning

In this section, we illustrate our approach with a basic motion planning problem. Fix a rigid robot $R_0 \subseteq \mathbb{R}^d$ and an obstacle set $\Omega \subseteq \mathbb{R}^d$. Both R_0 and Ω are closed sets. Initially we assume R_0 is a d-dimensional ball of radius $r_0 > 0$.

Suppose we want to compute a motion from an initial configuration α to some final configuration β. One of the best exact solution when R_0 is a ball is based on roadmaps (i.e., retraction approach). Historically, the case $d = 2$ was the first exact roadmap algorithm [25]. For polygonal Ω, the roadmap is efficiently computed as the Voronoi diagram of line segments [37, 17]. For $d = 3$, it is clear that a similar exact solution is possible. But here we see the severe barrier posed by exact solutions: we are not aware of any exact algorithm for the Voronoi diagram of polyhedral obstacles. The problem lies in working out details of the algebraic primitives and the possible degeneracies (e.g., in the style of [14]). Indeed, the details for the exact computation of the Voronoi diagram of three lines in $\mathbb{R}^3$ has only recently been worked out [16, 15].

The **configuration space** or C_{space} is $\mathbb{R}^d$ when R_0 is a ball. In general, we write $C_{space}(R_0)$ for the configuration of a robot R_0. Let $\alpha, \beta \in C_{space}$. The **footprint** of R_0 at α is the set $R_0[\alpha]$ comprising those points in $\mathbb{R}^d$ occupied by R_0 in configuration α. We say α is **free** if $R_0[\alpha] \cap \Omega$ is empty; it is **semi-free** if it is not free but $R_0[\alpha]$ does not intersect the interior of Ω. Thus α is semi-free if $R_0[\alpha]$ is just touching Ω without penetrating it. Finally α is **stuck** if it neither free nor semi-free. Thus, every configuration is classified as free, stuck or semi-free. We extend this classification to any set $B \subseteq C_{space}$: we say B is free (resp., stuck) if every $\alpha \in B$ is free (resp., stuck). Otherwise, B is **mixed**. We have thus defined the **classification predicate** $C : 2^{C_{space}} \to \{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$. In practice, the predicate $C(\cdot)$ is only computable for "nice" sets, e.g., when B is a box. Our goal in soft primitive design is to avoid this exact computation. This classification goes back to beginning of subdivision motion planning when Brooks and Perez [5] used the empty/full/mixed labels.

Let $C_{free} = C_{free}(R_0, \Omega) \subseteq C_{space}$ denote the set of free configurations. A **motion** from α to β is a continuous map $\mu : [0, 1] \to C_{space}$ with $\mu(0) = \alpha$ and $\mu(1) = \beta$. We say μ is a **free motion** if its range is contained in C_{free}

¶2. Subdivision Trees. Our main data structure is a subdivision tree $\mathcal{T}$ rooted at a box $B_0 \subseteq \mathbb{R}^d$ (see Figure 1 for $d = 2$). The nodes of $\mathcal{T}$ are subboxes of B_0, where boxes are closed subsets of full dimension d, and each internal node B is split into 2^i ($i = 1, \ldots, d$) congruent subboxes which form the children of B. We remark that boxes B are axes-parallel and not assumed to be square, with **width** $w(B)$ and **length** $\ell(B)$ defined to be the lengths of the shortest and longest side (resp.). For convergence, we must assume that the **aspect ratio** $\ell(B)/w(B) \geq 1$ is bounded. Any box that can be obtained as a descendent of B_0 in a subdivision tree is said to be **aligned**. Let $m(B)$ denote the **midpoint** and **radius** $r(B)$ be the distance from $m(B)$ to any corner of B. For any real number $s > 0$, let $s \cdot B$ denote the congruent box centered at $m(B)$ with radius $s \cdot r(B)$. Two boxes B, B' are **adjacent** if $B \cap B'$ is a **facet** F of B or of B', where facets refer to faces of co-dimension 1. Also, let $D_m(r)$ denote the **closed ball** centered at m with radius r.

To allow domains of arbitrarily complex geometry, the input to our algorithm is an initial subdivision tree $\mathcal{T}_0$ whose leaves are arbitrarily marked `ON` or `OFF`. The set of `ON`-leaves forms a **subdivision** of the **region-of-interest** $ROI(\mathcal{T})$ of the tree. Subsequently, $\mathcal{T}$ can be **expanded** at any `ON`-leaf B, by **splitting** B into 2^i ($1 \leq i \leq d$) congruent subboxes who become the children of B. These children remain `ON`-boxes, so $ROI(\mathcal{T})$ is preserved.

¶3. An Exact Subdivision Algorithm. Our algorithm is given $\varepsilon > 0$ and an initial $\mathcal{T}_0$ rooted at B_0. The algorithm is parametrized by two subroutines: a classification predicate $C(B)$ for boxes, and a subroutine

$\texttt{S}split(B, \varepsilon)$ which returns a subdivision of B into 2^i (for some $i = 0, \ldots, d$) congruent subboxes; the split subroutine is said to fail if $w(B) < \varepsilon$ (in this case $i = 0$). We use $\mathcal{T}$ to search for a path in $B_0 \cap C_{free}$ as follows. Let $V(\mathcal{T})$ denote the set of free leaves in $\mathcal{T}$. We define an undirected graph $G(\mathcal{T})$ with vertex set $V(\mathcal{T})$ and edges connecting pairs of adjacent free boxes. We maintain the connected components of $G(\mathcal{T})$ using the well-known **Union-Find** data structure on $V(\mathcal{T})$: given $B, B' \in V(\mathcal{T})$, $Find(B)$ returns the index of the component containing B, and $Union(B, B')$ merges the components of B and of B'.

We associate with $\mathcal{T}$ a priority queue $Q = Q_{\mathcal{T}}$ to store all the mixed leaves B with width $w(B) \geq \varepsilon$. Let $\mathcal{T}.getNext()$ remove a box in Q of highest "priority". This priority is discussed below. If B is the box returned by $\mathcal{T}.getNext()$, we will expand B as follows: first call $\texttt{S}split(B, \varepsilon)$. If $\texttt{S}split(B, \varepsilon)$ fails, we return fail. Otherwise, each of the subboxes B' returned by $\texttt{S}split(B, \varepsilon)$ is made a child of B. We label B' with the predicate $C(B')$. If $C(B') = \texttt{FREE}$, we insert B' into $V(\mathcal{T})$ and into the union-find structure, and for each $B'' \in V(\mathcal{T})$ adjacent to B', we call $Union(B', B'')$. Finally, if $C(B') = \texttt{MIXED}$ and $w(B') \geq \varepsilon$, we insert B' into Q. Thus, mixed box of width $< \varepsilon$ are discarded (effectively regarded as STUCK). Now we are ready to present a simple but useful exact subdivision algorithm:

EXACT FINDPATH:
Input: Configurations α, β, tolerance $\epsilon > 0$, box $B_0 \in \mathbb{R}^d$.
Output: Path from α to β in $Free(R_0, \Omega) \cap B_0$.
Initialize a subdivision tree $\mathcal{T}$ with only a root B_0.
1. While ($Box_{\mathcal{T}}(\alpha) \neq \texttt{FREE}$)
If (Expand $Box_{\mathcal{T}}(\alpha)$ fails) Return("No Path").
2. While ($Box_{\mathcal{T}}(\beta) \neq \texttt{FREE}$)
If (Expand $Box_{\mathcal{T}}(\beta)$ fails) Return("No Path").
3. While ($Find(Box_{\mathcal{T}}(\alpha)) \neq Find(Box_{\mathcal{T}}(\beta))$)
If $Q_{\mathcal{T}}$ is empty, Return("No Path")
(*) $B \leftarrow \mathcal{T}.getNext()$
Expand B
4. Compute a channel P from $Box_{\mathcal{T}}(\alpha)$ to $Box_{\mathcal{T}}(\beta)$.
General a motion $\overline{P}$ from P and Return($\overline{P}$)

In Step 4, the **channel** P is a sequence $(B_1, \ldots, B_m)$ of boxes where B_i, B_{i+1} are adjacent. We convert the channel into a motion (or trajectory) which is a parametrized path $\overline{P} : [0, 1] \to C_{free}$ from α to β. We can easily generate $\overline{P}$ to satisfy reasonable constraints such as smoothness. As noted, this ability to generate a motion is a big win of subdivision methods over pure algebraic methods. Note that our channels are free, in contrast to the M-channels (each a sequence of adjacent FREE or MIXED leaf boxes) of Zhu-Latombe [42], Barbehenn-Hutchinson [2] and Zhang-Manocha-Kim [41]. Freeness is essential in the Union-Find application.

The routine $\mathcal{T}.getNext()$ in Step (*) is not fully specified, but critical. To ensure "completeness" of this algorithm, a simple solution is to return any mixed leaf of minimum depth. Below, we will provide careful analysis of completeness. But many other interesting heuristics are possible: If $getNext()$ is random, we obtain a form of Sampling Method. By alternating between randomness and some complete strategy, we can get the best of both worlds. If $getNext()$ always return a mixed leaf that is adjacent to the connected component of $Box_{\mathcal{T}}(\alpha)$, we get a sort of Dijkstra's algorithm or A*-search (see Barbehenn and Hutchinson [2, 1]). Another idea is to use some entropy criteria. Recent work on shortest-path algorithms in GIS road systems offers many other heuristics.

4 Let us Design Soft Predicates!

The above exact subdivision is not our claim to novelty. Nevertheless, our framework has interesting features, and seems more adaptive than some suggestions in the literature. For instance, the (uniform) grid [23,

p. 185] is widely used. According to the Wikipedia[4] article on resolution completeness, most resolution complete planners are grid-based and uniform. Although grids are superficially similar to subdivisions, a critical difference is that grids use point-based operations while our theory is based on box (interval) operations with guarantees. Uniform grid translates into breadth-first search strategy for $\mathcal{T}.getNext()$, but we can do much better. Zhu and Latombe [42] suggests a more goal directed form of this expansion: pick some "shortest" M-channel (sequence of adjacent `FREE` or `MIXED` leaf boxes) and expand all the `MIXED` boxes in the channel. To support this kind of channel expansion, Barbehenn and Hutchinson introduced the highly efficient Dijkstra-search or its extension to A* search [2, 1]. While finding the shortest A*-path is efficient, the efficient update of the A*-structure after expansion is not well-understood. The justification of expanding along a shortest M-channel is also unclear.

Our true interest lies in replacing the exact predicate $C(B)$. Generally, exact predicates need algebraic computation and can be highly non-trivial to compute. In the present case, especially for $d = 2$, computing $C(B)$ is actually feasible. Even so, we prefer to avoid exact computation. We will replace $C(B)$ by some $\widetilde{C}(B)$ which is easy to compute and "correct in the limit".

¶4. Soft Predicates. We now formalize the needed properties. Let $\widetilde{C}(B)$ be a box predicate that returns a value in $\{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$. We call $\widetilde{C}$ a **soft version** of C if two conditions hold:

> **(A1)** It is **safe**, i.e., $\widetilde{C}(B) \neq \texttt{MIXED}$ implies $\widetilde{C}(B) = C(B)$.
> **(A2)** It is **convergent**, i.e., if $\{B_i : i = 1, 2, \ldots, \infty\}$ converges to a configuration γ and $C(\gamma) \neq \texttt{MIXED}$, then $\widetilde{C}(B_i) = C(\gamma)$ for large enough i.

We need a quantitative measure of the convergence rate. Relative to any class $\mathcal{B}$ of boxes, a soft version $\widetilde{C}$ of C is said to have a **effectivity factor** of σ if $C(B) = \texttt{FREE}$ implies $\widetilde{C}(\sigma B) = \texttt{FREE}$ for all $B \in \mathcal{B}$. One might imagine a stronger condition that $C(B) \neq \texttt{MIXED}$ implies $\widetilde{C}(\sigma B) \neq \texttt{MIXED}$ for all $B \in \mathcal{B}$. However, it seems difficult to effectively compute $\widetilde{C}$ with such strong properties. Our weaker definition suffices for our main Theorem A. Clearly, $0 \leq \sigma \leq 1$. Call $\widetilde{C}$ **effective** for $\mathcal{B}$ if it has some effectivity factor relative to $\mathcal{B}$. For example, we will prove that our soft predicates below are effective for the class $\mathcal{B}$ of square boxes.

We now design soft predicates $\widetilde{C}$ assuming Ω is a polyhedral set, and the boundary of Ω is partitioned into a simplicial complex comprising open cells of each dimension. These cells are called **features** of Ω. For $d = 3$, the features of dimensions $0, 1, 2$ (resp.) are called **corners**, **edges** and **walls**. Each box B is associated with three sets: its **outer domain** $W^+(B) \subseteq \mathbb{R}^2$, **inner domain** $W^-(B) \subseteq \mathbb{R}^2$, and **feature set** $\phi(B) \subseteq \Phi(\Omega)$. We define $W^+(B)$ and $W^-(B)$ as the discs $D_{m(B)}(r_0 + r(B))$ and $D_{m(B)}(r_0 - r(B))$, respectively. If $r_0 < r(B)$, the $W^-(B)$ is empty. Also, $\phi(B)$ comprises the features of Ω that intersects $W^+(B)$. We call B **simple** if one of the following conditions holds:

> **(S0)** Its feature set $\phi(B)$ is empty.
> **(S1)** Some feature of Ω intersects its inner domain.

The soft predicate $\widetilde{C}$ can now be defined: for our purposes, we only need to define $\widetilde{C}(B)$ for aligned boxes B. Thus we can use induction by depth. If B is non-simple, declare $\widetilde{C}(B) = \texttt{MIXED}$. Else if (S1) holds, declare $\widetilde{C}(B) = \texttt{STUCK}$. Otherwise, (S0) holds and clearly B is either free or stuck, and we define $\widetilde{C}(B) = C(B)$ accordingly.

We now come to computing $\widetilde{C}(B)$, but only in the context where B is a leaf of a subdivision tree. In this case, we can easily compute $\phi(B)$ by induction: when we expand B, we can easily distribute the features in $\phi(B)$ to each of its children (note that a feature can be given to more than one child, or to no child). Moreover, we can check the conditions (S1) and (S0) during this distribution. Finally, if (S0) holds, we determine $\widetilde{C}(B)$ as follows: $\widetilde{C}(B) = \texttt{FREE}$ (resp., `STUCK`) iff $m(B)$ is outside (resp., inside) the obstacle Ω.

[4] `http://en.wikipedia.org/wiki/Motion_planning#Completeness_and_Performance`, retrieved Oct 27, 2011.

To distinguish these two cases, we just check the feature set $\phi(B.\texttt{parent})$ of its parent box $B.\texttt{parent}$ of B. The set $\phi(B.\texttt{parent})$ is non-empty, and by a linear search, we find the feature f in this set that is closest to $m(B)$. This feature may be a wall, edge or corner but it is assumed that the features are locally oriented so that we can decide whether $m(B)$ is inside or outside Ω in the neighborhood of f.

LEMMA 1. *The predicate $\widetilde{C}$ is a soft version of C for the robot R_0. When boxes are squares, $\widetilde{C}$ has an effectivity factor of $1/\sqrt{2}$.*

Proof. See Appendix. □

All we do is to substitute $\widetilde{C}$ for C in the exact algorithm of the previous section to get a complete motion planning algorithm — this will be proved below, when we introduce resolution-exact algorithms.

What have we gained? This algorithm is extremely easy to implement for $d = 2$ and 3. For $d > 3$, it is also easily implemented if we assume all features are simplicial. In particular, the computation of $\widetilde{C}$ (which includes maintaining $\phi(B)$) uses only distance computation between points and features, and numerical approximations can be employed in these computations. Already, for $d = 3$, we have achieved a resolution-exact algorithm for a problem for which there is no known exact solution.

¶5. Improvements. We can speed up the computation as follows: we will define the set $\phi(B)$ in a slightly different way by recognizing two regimes for boxes. In the "small B regime", i.e., $r(B) < r_0$, we compute $\phi(B)$ as before. In the "large B regime", i.e., $r(B) \geq r_0$, we can define $\phi(B)$ to comprise those features that intersects the box αB where $\alpha = 1 + \sqrt{2}r_0/r(B)$. In general, for any box B, and real number $\alpha > 0$, αB denotes the box centered at $m(B)$ of radius α. Checking if a feature intersects αB is also extremely simple. This new definition should generally result in smaller sizes for $\phi(B)$.

We remark that the condition (S1) could have been omitted without affecting the correctness of the algorithm. It's role is to provide an early stuck decision. So for fast implementation, it could be omitted.

5 Rotational Degree of Freedom

Next consider the case where robot $R_1 \subseteq \mathbb{R}^2$ has a simple shape. Assume R_1 is a triangle, and R_1 is contained in a circumscribing disc R_0 of radius r_0. Now, $C_{space} = SE(2) = \mathbb{R}^2 \times S^1$. Each box $B \subseteq C_{space}$ is decomposed as $R \times \Theta$ where $R \subseteq \mathbb{R}^2$ is a rectangle and $\Theta \subseteq S^1$ is an angular range. We also write $m(B), r(B), w(B)$ to denote the previously defined $m(R), r(R), w(R)$. Two boxes $B = R \times \Theta$ and $B' = R' \times \Theta'$ are **adjacent** iff R and R' are adjacent, and Θ and Θ' are adjacent in the circular geometry of S^1.

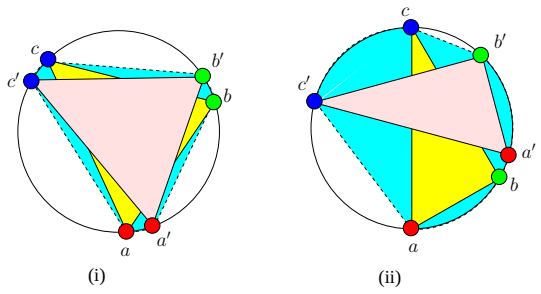

Figure 2: Shaded area represent round triangles: (i) $aa'bb'cc'$, (ii) $ab'cc'$.

¶6. ε-Smallness. We discuss the issue of **splitting** B: we can obviously simply split B into 8 congruent children. However there are two issues. First of all, we may want to avoid splitting the angular range when B is in the "large regime": as long as $w(B) \geq r_0$, we can approximate R_1 by R_0 and ignore the rotational degree of freedom. So B is split into 4 children (based on splitting R but not Θ). When B is in the "small regime", i.e., $w(R) < r_0$, we begin to split the angular range. But here, we want to treat Θ differently than R. To understand this, recall we previously do not split a box R when $w(R) < \varepsilon$. Let us say R is ε-**small** if $w(B) < \varepsilon$. We need a similar criteria for Θ, and here, we say Θ is ε-**small** if $|\Theta| < \varepsilon/r_0$. This assumes that angles are in radians, and Θ is represented as a subset $[\theta_1, \theta_2] \subseteq [0, 2\pi]$; also $|\Theta|$ is defined as $\theta_2 - \theta_1$. Finally, we say $B = R \times \Theta$ is ε-**small** if both R and Θ are ε-small. We now define our procedure $\textsc{Split}(B, \varepsilon)$ as follows: to split B, we split R and Θ separately. These will not be split if they are already ε-small. Thus,

splitting B will result in 2^i children for $i = 0, 1, 2, 3$. Our definition of ε-smallness is motivated by this lemma:

LEMMA 2. *Assume $0 < \varepsilon \leq \pi/2$. If B is ε-small, then the Hausdorff distance between the footprints of R_1 at any two configurations in B is at most $(1 + \sqrt{2})\varepsilon$.*

This result uses the fact that if we rotate R_1 by θ about the center of R_0, then the vertices of R_1 moves by at most $2r_0 \sin(\theta/2) \leq r_0\theta$ since $\sin\theta \leq \theta$ for θ in the said range. Also, the translational distance between any two configurations in B is at most $\sqrt{2}\varepsilon$.

¶7. Soft Predicate for Rotation. We now design a soft version $\widetilde{C}$ of C. The strategy follows the case of disc robot: we define the feature set $\phi(B)$ associated with a box $B = R \times \Theta$ as comprising those features of Ω that intersects the set $W^+(B)$ where $W^+(B)$ is a "round triangle" associated with B. We call R a **round triangle** if it is given as the intersection of a disc D with a triangular region T (see Figure 2). If $(D, T) \neq (D', T')$ but $D \cap T = D' \cap T'$, we regard these two round triangles as different. For any real number s, we denote the **s-expansion** of various shapes $S \subseteq \mathbb{R}^2$ by $(S)^s$. If $S = D(m, r)$ is a disc, $(D)^s := D(m, r + s)$. If S a convex polygon P, then $(P)^s$ is the triangle obtained by shifting each defining line of its edges in an outward normal direction by a distance of s. Typically, P is a triangle or a box. Finally, if S is a round triangle $R = D \cap T$, then $(R)^s = (D)^s \cap (T)^s$. Note that $(R)^s$ depends on the representation D and T. Usually, $s \geq 0$. If $s < 0$, this is actually a shrinking operation and $(S)^s$ may be the empty set.

Consider a configuration $(m, \theta) \in C_{space}$ the footprint $R_1[m, \theta]$ is a triangle in $D_m(r_0))$. Let $RT(m, \Theta)$ be a convex hull of the union of these footprints as θ range over Θ. Note that $RT(m, \Theta)$ is a round triangle. In Figure 2, we show $RT(m, \Theta)$ for two choices of R_1's. We define the **outer domain** $W^+(B)$ to be the $r(B)$-expansion of $RT(m(B), \Theta)$. As before, the **feature set** $\phi(B)$ is defined as those features in $\Phi(\Omega)$ that intersects $W^+(B)$. Finally, we define $\widetilde{C}(B)$ using $\phi(B)$ as before. Computing $\widetilde{C}(B)$ in the context of an expanding subdivision tree is also similar.

LEMMA 3. *$\widetilde{C}$ is a soft version of C for the robot R_1. Also $\widetilde{C}$ is effective for the class of squares.*

¶8. Improvements. We can improve by providing some heuristic for quick detection of stuck boxes, in analogy to Property (S1) for a disc robot. For any box B, we can define an **inner domain** $W^-(B)$ such that if any feature intersects $W^-(B)$, then B is stuck. Indeed $W^-(B)$ can be defined to be a suitable triangle: in Figure 2(i), $W^-(B)$ is the triangle bounded by the lines $\overline{a'c}$, $\overline{ab'}$ and $\overline{bc'}$.

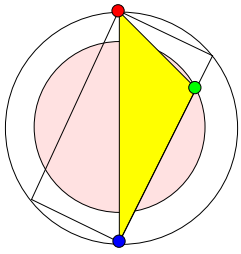

Figure 3: Enclosing circle of enclosing rectangle for obtuse triangle: their rotation

Notice that we defined R_0 as the circumscribing circle for R_1. This can give us very large r_0 when R_1 is very thin (obtuse). Why not define R_0 as the smallest disc that contains R_1? If R_1 is an obtuse triangle, then its longest side will be a diameter of R_0, and one vertex of R_1 will be in the interior of R_0. In general, obtuse or not, we may need to deal with shapes more general than round triangles. Consider the triangle in Figure 3 in its smallest enclosing circle C. Two vertices of the triangle (red and blue) are on C, but the green vertex is in the interior of the circle. If we slightly rotate the triangle slightly counter-clockwise about

the center of C, the green vertex will lie outside the swept area. Therefore, the convex hull of the swept area of the triangle is bounded in part by an arc of the inner circle (bounding the pink disk). The resulting convex hull of the swept may comprise of 3 arcs, where one of them comes from the inner circle. But we expect that the circumscribing circle approach is more effective as an approximation.

6 Resolution Exactness

We have designed some non-trivial algorithms in the previous sections. We now clarify what sort of algorithms these are. They are "resolution complete" in the informal sense of the literature. But what exactly does this mean? Here is one common[5] definition: *"Resolution completeness" is the property that the planner is guaranteed to find a path if the resolution of an underlying grid is fine enough.* Presumably, "fine enough" means "as the resolution parameter h go to 0". But if h is not bounded away from 0, this entails an infinite search and such algorithms will suffer from the halting problem that plaque Sampling Methods.

Notice that our algorithms in Sections 4 and 5 have an explicit input $\varepsilon > 0$, called the **resolution parameter**. It is essential that ε be different from 0. To use this parameter, we recall the concept of "clearance". Let $R_0 \subseteq \mathbb{R}^d$ be any robot moving amidst the obstacles $\Omega \subseteq \mathbb{R}^d$. The **clearance** of a configuration $\gamma \in C_{space}$ is the closest distance between a point in $R_0[\gamma]$ and a point in Ω. Moreover, the **clearance** of a motion $\mu : [0,1] \to C_{space}$ is the minimum clearance of $\mu(t)$ for $t \in [0,1]$. Here is another attempt to define resolution completeness, where we now state a converse condition for "no path': (i) *if there is a path with clearance ε, then the algorithm will find a free path,* and (ii) *if there is no path with clearance ε, it will report "no-path"*. Taken together, this pair of statements cannot be correct, as it implies that we can detect the case where the clearance is exactly ε, a feat that only Exact Methods can perform (in which case we might as well design algorithms with $\varepsilon = 0$). What is missing in current discussions of resolution completeness is an **accuracy parameter** $K \geq 1$. We say that a planner has an **accuracy** $K \geq 1$ if the following holds:

- If there is a path with clearance $K\varepsilon$, it outputs a path with clearance ε/K.
- If there is no path with clearance ε/K, it reports "no-path".

Now we can define a concept noted in the introduction: a planner is said to be **resolution-exact** if it has an accuracy $K \geq 1$. What if the maximum clearance of free paths lies strictly in the range $(\varepsilon/K, K\varepsilon]$? According to this definition, the planner is free to report a path or "no path". In our Theorem A below, we prove that this cannot be avoided! *This indeterminacy is the necessary price to pay for resolution-exactness*. In our view, this price is not a serious one because the user has the option to decrease the ε parameter as desired. Of course, if you decrease ε to ε/K, the indeterminacy will reappear for input instances that only has paths with clearance in the range $(\varepsilon/K^2, \varepsilon]$.

The result of Theorem A below concerns our algorithm EXACT FINDPATH in ¶3 in the 2D case, assuming that all boxes are squares and we use the exact classifier predicate $C(B)$. Recall that in our EXACT FINDPATH algorithm, we subdivide a box only if its width $w(B)$ is *larger than* the input resolution parameter $\epsilon > 0$. So the smallest boxes in the subdivision tree $\mathcal{T}$ have width t with $\epsilon/2 < t \leq \epsilon$. Now consider the "full expansion" of the subdivision tree $\mathcal{T}$ whose leaves are of the smallest size possible. Recall from ¶3 that a channel is a sequence $(B_1, \ldots, B_m)$ where B_i, B_{i+1} are adjacent. We are interested in a free channel where $\alpha \in B_1$ and $\beta \in B_m$.

LEMMA 4. *If there exists a motion μ with clearance $\delta = \sqrt{2}\epsilon$, then our* EXACT FINDPATH *algorithm outputs a path with clearance $\epsilon/4$.*

Proof. See Appendix. □

We define an **essential path** to be a path from the center a of a free box $B(\alpha)$ containing α to the center b of a free box $B(\beta)$ containing β (e.g., path P in Figure 7). A **canonical path** consists of line segments

[5] E.g., http://en.wikipedia.org/wiki/Motion_planning#Grid-Based_Search.

$\overline{\alpha a}, \overline{b\beta}$, and an essential path P from a to b. Note that the major task in motion planning is to find an essential path, while constructing $\overline{\alpha a}$ and $\overline{b\beta}$ is straightforward. We define the *essential clearance* of a canonical path to be the clearance of its essential path.

LEMMA 5. *If there is no free canonical path with essential clearance $\epsilon/4$, then our* EXACT FINDPATH *algorithm reports "no path".*

Proof. See Appendix. □

Putting together Lemmas 4 and 5, we have the following results for 2D, assuming that all boxes are squares and we use the exact classifier predicate $C(B)$.

THEOREM A: [Hard Predicate] *Let $C, c \geq 1$ and consider our planner* EXACT FINDPATH.
(i) For $C = \sqrt{2}$, if there is a path with clearance $C\varepsilon$, then our planner outputs a path with clearance $\varepsilon/4$.
(ii) For $c = 4$, if there is no free canonical path of essential clearance ε/c, then our planner reports "no path".
The results in (i) and (ii) are tight in the following sense:
(i') If $C < \sqrt{2}$, there are obstacle inputs Ω admitting paths with clearance $C\varepsilon$, but our planner reports "no path".
(ii') If $c < 4$, there are obstacle inputs Ω admitting no free canonical paths of essential clearance ε/c but our planner outputs a path.

Proof. See Appendix. □

Theorem A implies an accuracy factor $K = 4$, but it is clear that K can be reduced by adjusting our algorithm to use the resolution parameter ε in a more equitable way.

The general form of this result is perhaps no surprise, but the accuracy constants might not be what we initially expect, since we are talking about an "exact algorithm". There are two sources for the loss of accuracy: first, subdivision boxes are "aligned" with the integer grid in the sense that their coordinates are dyadic numbers. Second, the width of our smallest boxes, the ε-`MIXED` boxes, lies between $\varepsilon/2$ and ε.

An additional reason for accuracy loss is, of course, the use of soft predicates. In particular, what is the accuracy of our prototype algorithm in ¶3 when using the soft predicates of ¶4? Recall from Lemma 1 that when boxes are squares, our soft predicate $\widetilde{C}$ has an effectivity factor $\sigma = 1/\sqrt{2}$. In our algorithm, we can replace our input resolution parameter with $\bar{\varepsilon} = \sigma\epsilon$, i.e., we split boxes until the smallest box width is between $\bar{\varepsilon}/2$ and $\bar{\varepsilon}$, i.e., between $\sigma\epsilon/2$ and $\sigma\epsilon$.

LEMMA 6. *If there exists a motion μ with clearance $\delta = \sqrt{2}\epsilon$, then our algorithm using soft predicate $\widetilde{C}$ outputs a path with clearance $\sigma\epsilon/4$.*

Proof. See Appendix. □

LEMMA 7. *If there is no free canonical path with essential clearance $\sigma\epsilon/4$, then our algorithm using soft predicate $\widetilde{C}$ reports "no path".*

Proof. See Appendix. □

Combining Lemmas 6 and 7, we have the following.

THEOREM B: [Soft Predicate] *With the same assumptions as Theorem A, but with the exact predicate $C(B)$ replaced by a soft predicate $\widetilde{C}(B)$ with effectivity factor σ, we have:*
(i) For $C = \sqrt{2}$, if there is a path with clearance $C\varepsilon$, then our planner outputs a path of clearance $\sigma\varepsilon/4$.
(ii) For $c = 4$, if there is no free canonical path with essential clearance $\sigma\varepsilon/c$, then our planner reports "no path".

This implies that the accuracy factor K now becomes $4/\sigma$. In general, we have:

Corollary: If the Exact version of our planner has an accuracy factor of K, then the Soft version of our planner using a soft predicate with effectivity factor σ has an accuracy factor of K/σ.

7 Robots with Complex Geometry

We shall now show how to extend our soft predicates techniques to robots with complex geometry or multiple degrees-of-freedom (DOF). The state-of-the-art for what could be practically implemented is discussed in Zhang, Kim and Manocha [41]. They considered a series of challenging robot configurations: a "five-gear" robot moving amidst a collection of static gear obstacles, a "2-D puzzle" robot in a maze-like environment, a certain "star" robot with four DOF, and "serial link" robot with four DOF. Another famous robot in this literature is from Kavraki, with 10 DOF. Except for the "star", the rest are planar robots. See Figure 4 for (a) the "five-gear" of Zhang et.al, and (b) Kavraki's robot.

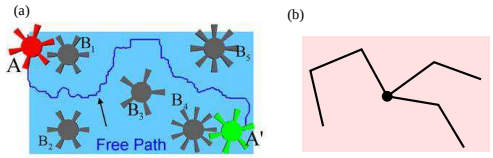

Figure 4: Complex Robots

¶9. Decomposition Principle for Complex Rigid Robots First consider the case of a rigid polygonal robot $R_2 \subseteq \mathbb{R}^2$, not necessarily convex. The "5-gear" robot in [41] is an instance. We first "cover" R_2 with a set S of triangles. More precisely, a set S of subsets of $\mathbb{R}^2$ is called a **cover** of R_2 if $\int(R_2) \subseteq \cup_{T\in S} \int(T)$ where $\int(T)$ denotes the interior of a set $T \subseteq \mathbb{R}^2$. The cover S is **exact** if the inclusion is an equality: $\int(R_2) = \cup_{T\in S} \int(T)$. In our application, each $T \in S$ is a triangle. Let $C_{R_2}(B)$ denote the box classification predicate for R_2; we reduce C_{R_2} to the classification predicates C_T for each triangle $T \in S$ as follows:

$$(\forall T \in S)[C_T(B) = \texttt{FREE}] \quad \Leftrightarrow \quad C_{R_2}(B) = \texttt{FREE}, \tag{1}$$

and

$$(\exists T \in S)[C_T(B) = \texttt{STUCK}] \quad \Rightarrow \quad C_{R_2}(B) = \texttt{STUCK}. \tag{2}$$

Thus the condition for stuckness is only one-sided. Despite the weakness of this criterion, our next result shows that the weakness vanishes when we consider soft predicates:

THEOREM 8. *Let S be an exact cover for a robot R_2. Consider the predicate:*

$$\widetilde{C}_{R_2}(B) = \begin{cases} \texttt{FREE} & \text{if } \widetilde{C}_T(B) = \texttt{FREE} \text{ for all } T \in S \\ \texttt{STUCK} & \text{if } \widetilde{C}_{T_1}(B) = \texttt{STUCK} \text{ for some } T_1 \in S \\ \texttt{MIXED} & \text{else.} \end{cases}$$

If each $\widetilde{C}_T$ is a soft version of $C_T(B)$, then $\widetilde{C}_{R_2}$ is a soft version of C_{R_2}.

Proof. We must prove that $\widetilde{C}_{R_2}$ is conservative and convergent. The safety of the conclusion $\widetilde{C}_{R_2}(B) = \texttt{FREE}$ follows (1) and the safety of $\widetilde{C}_T$ for each $T \in S$. Similarly, the safety of the conclusion $\widetilde{C}_{R_2}(B) = \texttt{STUCK}$ follows (2) and the safety of $\widetilde{C}_{T_1}$. Suppose $(B_i : i = 0, 1, \ldots)$ is a sequence of strictly decreasing boxes that converges to a configuration γ that is not semi-free:

$$B_i \to \gamma \in C_{space}(R_2) \quad (i \to \infty).$$

If γ is free, then we see that for i large enough, $\widetilde{C}_T(B_i) = \texttt{FREE}$ for all $T \in S$. Thus, $\widetilde{C}_{R_2}(B_i) = \texttt{FREE}$. If γ is stuck, then we see that there is some $T \in S$ such that for i large enough, $\widetilde{C}_T(B_i) = \texttt{STUCK}$. Thus $\widetilde{C}_{R_2}(B_i) = \texttt{STUCK}$. **Q.E.D.**

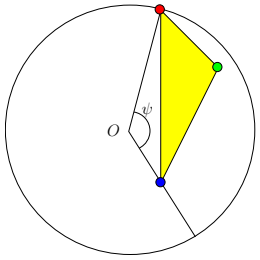

Figure 5: Sector of $R_0(T)$ defined by a robot T not containing the origin O.

It remains to see how to classify boxes relative to a triangle robot $T \in S$. Unlike the triangular robot R_1 above, we must now choose a common origin O for all the triangles in S. It is not hard to ensure that S contains at least one acute triangle T_0 whose circumcenter O is not covered by any other triangle in S. We choose this O as our origin. The soft version of $C_{T_0}(B)$ can be computed as for simple robots above. We now address computing the soft version of $C_T(B)$ for $T \neq T_0$. Enclose T in the smallest disc $R_0(T)$ centered at O. By assumption, $O \notin T$ and T lies in sector of $R_0(T)$ with angle ψ where $\psi < \pi$. The soft version of $C_T(B)$ when $B \subseteq C_{space}$ is in the large regime can be based on just the disc $R_0(T)$ (i.e., we ignore the angular range in B). When B is in the small regime, we develop a shape $W^+(B,T)$ that is analogous to the round triangle. We define the set $\phi(B,T)$ comprising those features that intersect $W^+(B,T)$. As usual, we use $\phi(B,T)$ to define the soft predicate $\widetilde{C}_T(B)$. Finally, we reduce the soft predicate $\widetilde{C}_{R_2}(B)$ to $\widetilde{C}_T(B)$ ($T \in S$) using Theorem 8. This completes our resolution-complete algorithm for a complex robot R_2.

In the full paper, we will also work out the details for the case of non-rigid robots such an link or Kavraki's 10 DOF robot.

8 Conclusion

In this paper, we introduced the notion of soft predicates and demonstrated their use in subdivision planners. We defined the concept of resolution-exact planners, and proved that the algorithms we designed are resolution-exact. The notion of resolution-exactness turned out to offer previously unnoticed subtleties.

Several open problems are raised by this research: first of all, we can in principle extend our work to subdivision of $SE(3) = \mathbb{R}^3 \times S^3$. But the proper method for subdividing S^3 is an interesting research. Second, notice that we have not tried to compute the connected components of STUCK boxes. We could do this, and it has some use for fast termination in the case of no-path. However, the best way to maintain this information seems to run into interesting issues of computational topology. Edelsbrunner and Delfinado's work on computing the Betti number of a 3-complex yields some clue about this issue [11].

According to Zhang et al [41], implementation of exact motion planning algorithms are only known for simple planer robots (like ladders or discs) and up to 3 degrees of freedom. Thus it is important to pay more attention to implementability of algorithms in this area. In robotics, we propose to give up exactness for the weaker notion of resolution-exactness. Little is lost by this step, since exact algorithms are ill-matched to the inherent limits on accuracy in physical systems. But we have much to gain: Subdivision algorithms are more holistic, integrating the concerns of topological correctness with geometric accuracy into one algorithm.

We believe Subdivision Methods can match the performance of Sampling Methods in many problems, but with superior properties. We plan to implement our algorithms and make comparisons.

In the full paper, we explore other variants of these algorithms, with an eye to simplicity and implementability, and as always, correctness. We plan to implement and compare our method with other approaches, including those with exact predicates and probabilistic approaches. Our general philosophy can clearly be extended to more complex motion planning problems such as kinodynamic problems or those with differential constraints. Combined with suitable $\mathcal{T}.getNext()$ heuristics, the complexity of our algorithms can be highly adaptive.

References

[1] M. Barbehenn and S. Hutchinson. Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees. *IEEE Trans. Robotics and Automation*, 11(2), 1995.

[2] M. Barbehenn and S. Hutchinson. Toward an exact incremental geometric robot motion planner. In *Proc. Intelligent Robots and Systems 95*, volume 3, pages 39–44, 1995. 1995 IEEE/RSJ International Conference on 'Human Robot Interaction and Cooperative Robots', 5–9, Aug 1995. Pittsburgh, PA , USA.

[3] R. Bohlin and L. Kavraki. A randomized algorithm for robot path planning based on lazy evaluation. In P. Pardalos, S. Rajasekaran, and J. Rolim, editors, *Handbook on Randomized Computing*, pages 221–249. Kluwer Academic Publishers, 2001.

[4] M. Brady, J. Hollerbach, T. Johnson, T. Lozano-Perez, and M. Mason. *Robot Motion: Planning and Control*. MIT Press, 1982.

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

[6] M. Burr, F. Krahmer, and C. Yap. Continuous amortization: A non-probabilistic adaptive analysis technique. *Electronic Colloquium on Computational Complexity (ECCC)*, TR09(136), December 2009.

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

[8] E.-C. Chang, S. W. Choi, D. Kwon, H. Park, and C. Yap. Shortest paths for disc obstacles is computable. In *21st ACM Symp. on Comp. Geom.*, pages 116–125, 2005. June 5-8, Pisa, Italy.

[9] P. Cheng and S. M. Lavalle. Resolution complete rapidly-exploring random trees. In *In Proc. IEEE Intl Conf. on Robotics and Automation*, pages 267–272, 2002.

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

[11] C. Delfinado and H.Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geom. Design*, 12:771–784, 1995.

[12] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14:443–479, 1995.

[13] A. Eigenwillig, V. Sharma, and C. Yap. Almost tight complexity bounds for the Descartes method. In *31st Int'l Symp. Symbolic and Alge. Comp. (ISSAC'06)*, pages 71–78, 2006. Genova, Italy. Jul 9-12, 2006. Eigenwillig and Sharma won the Best Student Author Award for this paper, shared with G.Moroz.

[14] I. Z. Emiris and M. I. Karavelas. The predicates of the Apollonius diagram: Algorithmic analysis and implementation. *Comput. Geometry: Theory and Appl.*, 33(1–2):18–57, 2006. Special Issue on Robust Geometric Algorithms and their Implementations.

[15] H. Everett, C. Gillot, D. Lazard, S. Lazard, and M. Pouget. The Voronoi diagram of three arbitrary lines in r^3. In *25th European Workshop on Computational Geometry (EuroCG'09)*, 2009. Mar 2009, Bruxelles, Belgium.

[16] H. Everett, D. Lazard, S. Lazard, and M. S. E. Din. The Voronoi diagram of three lines. *Discrete and Comp. Geom.*, 42(1):94–130, 2009. See also 23rd SoCG, 2007.

[17] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[18] D. Halperin, L. Kavraki, and J.-C. Latombe. Robotics. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 755–778. CRC Press LLC, 1997.

[19] K. Hauser. *Motion planning for legged and humanoid robots*. PhD thesis, Stanford University, Dec 2008. Department of Computer Science.

[20] L. Kavraki, P. Švestka, C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation*, 12(4):566–580, 1996.

[21] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C. Yap. Classroom examples of robustness problems in geometric computation. *Comput. Geometry: Theory and Appl.*, 40(1):61–78, 2007.

[22] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

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

[24] R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

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

[26] J. H. Reif and H. Wang. Nonuniform discretization for kinodynamic motion planning and its applications. *SIAM J. Computing*, 30:161–190, 2000.

[27] M. Sagraloff and C. K. Yap. A simple but exact and efficient algorithm for complex root isolation. In I. Z. Emiris, editor, *36th Int'l Symp. Symbolic and Alge. Comp. (ISSAC'11)*, pages 353–360, 2011. June 8-11, San Jose, California.

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

[29] J. T. Schwartz, M. Sharir, and J. Hopcroft, editors. *Planning, Geometry and Complexity of Robot Motion*. Ablex Series in Artificial Intelligence. Ablex Publishing Corp., Norwood, New Jersey, 1987.

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

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

[32] V. Sharma and C. Yap. Near optimal tree size bounds on a simple real root isolation algorithm, 2011. In Preparation.

[33] G. Varadhan, S. Krishnan, T. Sriram, and D. Manocha. Topology preserving surface extraction using adaptive subdivision. In *Proc. Symp. on Geometry Processing (SGP'04)*, pages 235–244, 2004.

[34] G. Varadhan and D. Manocha. Accurate Minkowski sum approximation of polyhedral models. *Graph. Models*, 68(4):343–355, 2006.

[35] C. K. Yap. Coordinating the motion of several discs. Robotics Report 16, Dept. of Computer Science, New York University, Feb. 1984.

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

[37] C. K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram for a set of simple curve segments. *Discrete and Comp. Geom.*, 2:365–394, 1987. Also: NYU-Courant Institute, Robotics Lab., No. 43, May 1985.

[38] C. K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 927–952. Chapman & Hall/CRC, Boca Raton, FL, 2nd edition, 2004.

[39] C. K. Yap. Theory of real computation according to EGC. In P. Hertling, C. Hoffmann, W. Luther, and N.Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, number 5045 in Lect. Notes in C.S., pages 193–237. Springer, 2008.

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

[41] L. Zhang, Y. J. Kim, and D. Manocha. Efficient cell labelling and path non-existence computation using C-obstacle query. *The International Journal of Robotics Research*, 27(11–12), 2008.

[42] D. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991. 1995 IEEE/RSJ International Conference on 'Human Robot Interaction and Cooperative Robots', 5–9, Aug 1995. Pittsburgh, PA , USA.

APPENDIX

In this appendix we provide all the missing proofs.

Lemma 1. *The predicate $\widetilde{C}$ is a soft version of C for the robot R_0. When boxes are squares, $\widetilde{C}$ has an effectivity factor of $1/\sqrt{2}$.*

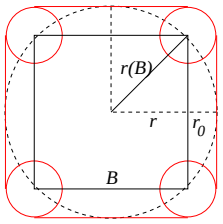

Figure 6: Effectivity factor $1/\sqrt{2}$.

Proof. To see the effectivity factor, suppose that $C(B) = \texttt{FREE}$. Referring to Figure 6, we see that the region bounded by the outer four red segments and the outer four red circular arcs has no obstacle. Clearly, the dotted circle also contains no obstacle. Note that this dotted circle is centered at $m(B)$ with radius $r + r_0$, and is the outer domain $W^+(\sigma B)$ of box σB whose radius is r, where $r = r(B)/\sqrt{2}$. This means that $\sigma = 1/\sqrt{2}$ and we have $\widetilde{C}(\sigma B) = \texttt{FREE}$. Therefore $\widetilde{C}$ has an effectivity factor of $1/\sqrt{2}$. **Q.E.D.**

Lemma 4. *If there exists a motion μ with clearance $\delta = \sqrt{2}\varepsilon$, then our* EXACT FINDPATH *algorithm outputs a path with clearance $\varepsilon/4$.*

Proof. Consider the "full expansion" of $\mathcal{T}$ as mentioned above, where the leaves have a width t with $\varepsilon/2 < t \leq \varepsilon$. Consider the subset $\mathcal{A}$ of such leaves that cover μ. We claim that each leaf box in $\mathcal{A}$ is *free*: let p be a point in μ and B_ℓ be the leaf box where p lies; since the diagonal of B_ℓ is $\sqrt{2}t \leq \sqrt{2}\varepsilon = \delta$, B_ℓ lies entirely within the "clearance region" of p and thus B_ℓ is free. Therefore $\mathcal{A}$ consists of free leaf boxes of width t that covers μ; in other words, $\mathcal{A}$ is a **free channel** Π that covers μ.

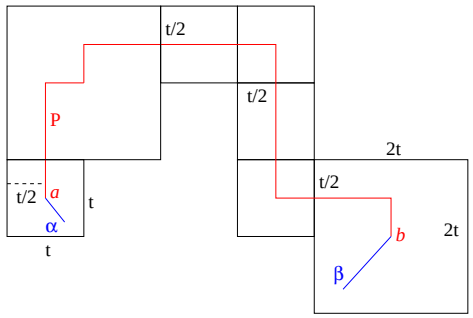

Figure 7: Path P with clearance $t/2 > \varepsilon/4$.

Since there exists a free channel Π connecting α and β, our EXACT FINDPATH algorithm will find *some* free channel Π' connecting α and β (Π' is *not necessarily* Π, but at least Π exists as a candidate to be found by our algorithm). This can be justified as follows: consider the subdivision tree $\mathcal{T}$ *produced by our algorithm*. It produces a subdivision of $ROI(\mathcal{T})$, and for each free box B in $\mathcal{A}$, there is a corresponding free leaf B^* in $\mathcal{T}$ that contains B. These free leaves B^*, after pruning redundancies, yield a free channel Π^* that covers Π. By definition of the correctness of any path finding algorithms, a free channel Π' connecting α and β will be found iff there exists a free channel Π^* connecting α and β.

Note that Π' consists of free aligned boxes connecting from $B(\alpha)$, the free (aligned) box containing α, to $B(\beta)$, the free (aligned) box containing β. Since each free box in Π' has width at least t, we can construct a rectilinear path P, from the box center a of $B(\alpha)$ to the box center b of $B(\beta)$, through the free boxes in Π' where each point of P is away from the box boundary by a distance at least $t/2$ (see Figure 7 for an example), and thus P has clearance $t/2 > \varepsilon/4$.

Our final reported path P_f is given by $P_f = \overline{\alpha a} \cup P \cup \overline{b\beta}$. It remains to show that $\overline{\alpha a}$ has clearance $\varepsilon/4$

(and similarly for $\overline{b\beta}$ by the same argument). The key point is to use the fact that α belongs to μ and thus has a clearance $\delta = \sqrt{2}\varepsilon$. We consider the following two cases.

Case (1): The width of $B(\alpha)$ is t. Then for any point $q \in \overline{\alpha a}$, $d(\alpha, q)$ is at most half of the diagonal of $B(\alpha)$, i.e., $d(\alpha, q) \leq \sqrt{2}t/2 \leq \sqrt{2}\varepsilon/2 = \delta/2$. However, α has clearance δ, and thus $q \in \overline{\alpha a}$ has clearance $\delta - d(\alpha, q) \geq \delta/2 > \varepsilon/4$.

Case (2): The width of $B(\alpha)$ is at least $2t$. We refer to Figure 8, where the boundaries of the inner box and of $B(\alpha)$ are apart by a distance $t/2$. Clearly, any point of $\overline{\alpha a}$ lying inside the inner box has clearance at least $t/2 > \varepsilon/4$. Now consider the portion of $\overline{\alpha a}$ outside the inner box. Without loss of generality, suppose such portion lies in the green shaded rectangle and the slope of $\overline{\alpha a}$ is in the range $[0, 1]$ (for other cases the slopes are in the ranges $(1, \infty)$, $[-1, 0)$, and $(-\infty, -1)$ and symmetric arguments apply). Note that $w = t/2$ and $h \leq w$ (since the slope of $\overline{\alpha a}$ is in $[0, 1]$), the diagonal of the green shaded rectangle is at most $\sqrt{2}t/2 \leq \sqrt{2}\varepsilon/2 = \delta/2$, i.e., any point $q \in \overline{\alpha a}$ lying in the green shaded rectangle has $d(\alpha, q) \leq \delta/2$. Since α has clearance δ, such q has clearance $\delta - d(\alpha, q) \geq \delta/2 > \varepsilon/4$. Therefore every point of $\overline{\alpha a}$ has clearance $\varepsilon/4$.

□

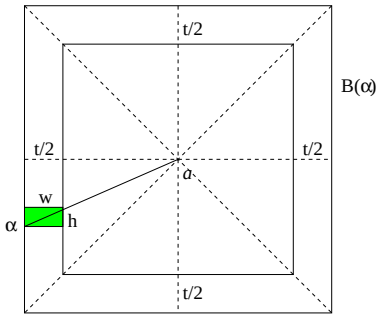

Figure 8: Segment $\overline{\alpha a}$ has clearance $\varepsilon/4$.

Lemma 5. *If there is no free canonical path with essential clearance $\varepsilon/4$, then our* EXACT FINDPATH *algorithm reports "no path".*

Proof. We prove the contrapositive: When our EXACT FINDPATH algorithm finds a path, there exists a free canonical path with essential clearance $\varepsilon/4$. Indeed, when our algorithm finds a free path, it finds a set of free aligned boxes connecting from $B(\alpha)$ to $B(\beta)$. Since each such free box has width at least t, we can construct an essential path, which is a rectilinear path P where each point of P is away from the box boundary by a distance at least $t/2$ (see Figure 7). Clearly $\overline{\alpha a} \cup P \cup \overline{b\beta}$ is a free canonical path with essential clearance at least $t/2 = \varepsilon/4$. □

THEOREM A: [Hard Predicate] *Let $C, c \geq 1$ and consider our planner* EXACT FINDPATH.
(i) For $C = \sqrt{2}$, if there exists a path with clearance $C\varepsilon$, then our planner outputs a path with clearance $\varepsilon/4$.
(ii) For $c = 4$, if there is no free canonical path with essential clearance ε/c, then our planner reports "no path".
The results in (i) and (ii) are tight in the following sense:
(i') If $C < \sqrt{2}$, there are obstacle inputs Ω admitting paths with clearance $C\varepsilon$, but our planner reports "no path".
(ii') If $c < 4$, there are obstacle inputs Ω admitting no free canonical paths of essential clearance ε/c but our planner outputs a path.

Proof.

(i) and (ii) are Lemmas 4 and 5 respectively.

(i'). Consider any $C < \sqrt{2}$. We can have an obstacle input Ω such that it admits a path with clearance $C\varepsilon$, where α lies in the aligned box $B = B(\alpha)$ of our subdivision tree, with width $w(B) = \varepsilon$, but the robot center cannot be placed in the red shaded triangle region (see Fig. Figure 9). Note that the diagonal of B is $\sqrt{2}\varepsilon$ and α can still have clearance $C\varepsilon$. However, B is a mixed box with $w(B) = \varepsilon$ and thus the expansion of B fails. Therefore our planner reports "no path".

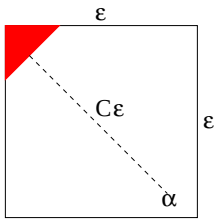

Figure 9: Proof of Theorem A (i').

(ii'). Suppose $c = 4 - \delta$ for some $1 > \delta > 0$. Choose a dyadic number $t > \delta$ and define $\varepsilon := 2t - \delta$. Since we subdivide our boxes until they are just smaller than ε, this implies our smallest boxes will have width exactly t. Suppose we have a channel comprising a "linear" sequence $(B_1, B_2, \ldots, B_k)$ ($k \geq 3$) of free boxes of width t. Linear sequence means there is a line ℓ passing through the centers of these boxes. We may place obstacles such that the clearance of any point on ℓ is $t/2$, and all other points in this channel has clearance $< t/2$. Assume $\alpha = m(B_1)$ and $\beta = m(B_k)$. Note that ℓ is both a free canonical path and an essential path, and the essential clearance of ℓ is the same as its clearance and vice versa. Clearly, our planner will report the channel path $(B_1, \ldots, B_k)$. It remains to show that there are no (free canonical) paths of (essential) clearance $\varepsilon/c = \frac{2t-\delta}{4-\delta}$. Observe that

$$\frac{2t - \delta}{4 - \delta} < t/2.$$

Hence there are no (free canonical) paths of (essential) clearance $\varepsilon/c = \frac{2t-\delta}{4-\delta}$. **Q.E.D.**

Lemma 6. *If there exists a motion μ with clearance $\delta = \sqrt{2}\varepsilon$, then our algorithm using soft predicate $\widetilde{C}$ outputs a path with clearance $\sigma\varepsilon/4$.*

Proof. This is a "soft version" of Lemma 4. Consider the "full expansion" of our subdivision tree $\mathcal{T}$; now the smallest boxes have width σt (instead of t). Look at the subset $\mathcal{A}$ of such leaf boxes that cover μ. For each such leaf box B_ℓ, let B_ℓ/σ be the box centered at $m(B_\ell)$ with width t. We claim that B_ℓ/σ is free: let p be a point on μ that lies in B_ℓ; clearly p also lies in B_ℓ/σ. Since the diagonal of B_ℓ/σ is $\sqrt{2}t \leq \sqrt{2}\varepsilon = \delta$, B_ℓ/σ lies entirely within the "clearance region" of p and thus B_ℓ/σ is free. Therefore we have $C(B_\ell/\sigma) = \texttt{FREE}$. By the effectivity factor σ for $\widetilde{C}$, $C(B_\ell/\sigma) = \texttt{FREE}$ implies $\widetilde{C}(B_\ell) = \widetilde{C}(\sigma(B_\ell/\sigma)) = \texttt{FREE}$. Therefore we can use $\widetilde{C}$ to classify each B_ℓ to be free, and thus to classify $\mathcal{A}$ as a **free channel** covering μ. This is the same as the free channel $\mathcal{A}$ covering μ in the proof of Lemma 4, but now each channel box has width σt rather than t. The rest of the proof of Lemma 4 carries over, with the reported path having a clearance $\sigma\varepsilon/4$ rather than $\varepsilon/4$. □

Lemma 7. *If there is no free canonical path with essential clearance $\sigma\varepsilon/4$, then our algorithm using soft predicate $\widetilde{C}$ reports "no path".*

Proof. This is a "soft version" of Lemma 5. Again we prove the contrapositive: When our algorithm finds a path, there exists a free canonical path with essential clearance $\sigma\varepsilon/4$. The proof of Lemma 5 carries over, but now each free aligned box has width σt rather than t, and thus the essential clearance is at least $\sigma t/2 = \sigma\varepsilon/4$. □