Theory and Explicit Design of a Path Planner for an $SE(3)$ Robot*

Zhaoqi Zhang[1], Yi-Jen Chiang[2], and Chee Yap[1]

[1] Department of Computer Science, Courant Institute, New York University, New York, NY, USA. zz1918@nyu.edu; yap@cs.nyu.edu

[2] Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA. chiang@nyu.edu

Abstract. We consider path planning for a rigid spatial robot with 6 degrees of freedom (6 DOFs), moving amidst polyhedral obstacles. A correct, complete and practical path planner for such a robot has never been achieved, although this is widely recognized as a key challenge in robotics. This paper provides a complete "explicit" design, down to explicit geometric primitives that are easily implementable.

Our design is within an algorithmic framework for path planners, called **Soft Subdivision Search** (SSS). The framework is based on the twin foundations of ε-exactness and soft predicates, two concepts that are critical for rigorous numerical implementations. These concepts allow us to escape from "Zero Problems" that prevent the correct or practical implementations of most exact algorithms of Computational Geometry. The practicality of SSS has been previously demonstrated for various robots including 5-DOF spatial robots.

In this paper, we solve several significant technical challenges for $SE(3)$ robots: (1) We first ensure the correct theory by proving a general form of the Fundamental Theorem of the SSS theory. We prove this within an axiomatic framework, thus making it easy for future applications of this theory. (2) One component of $SE(3) = \mathbb{R}^3 \times SO(3)$ is the non-Euclidean space $SO(3)$. We design a novel topologically correct data structure for $SO(3)$. Using the concept of **subdivision charts and atlases** for $SO(3)$, we can now carry out subdivision of $SO(3)$. (3) The geometric problem of collision detection takes place in $\mathbb{R}^3$, via the footprint map. Unlike sampling-based approaches, we must reason with the notion of **footprints of configuration boxes**, which is much harder to characterize. Exploiting the theory of **soft predicates**, we design suitable approximate footprints which, when combined with the highly effective feature-set technique, lead to soft predicates. (4) Finally, we make the underlying geometric computation "explicit", i.e., avoiding a general solver of polynomial systems, in order to allow a direct implementation.

Keywords: Algorithmic Motion Planning; Subdivision Methods; Resolution-Exact Algorithms; Soft Predicates; Spatial 6DOF Robots; Soft Subdivision Search.

* This work is supported in part by NSF Grant #CCF-2008768.

1 Introduction

Motion planning [10,27] is a fundamental topic in robotics because a robot, almost by definition, is capable of movement. There is growing interest in motion planners because of the wide availability of inexpensive commercial robots, from domestic robots for vacuuming the floor, to drones that deliver packages. We focus on **path planning** which, in its elemental form, asks for a collision-free path from a start to a goal robot position, assuming a known map of the environment. Path planning is based on robot kinematics and collision-detection only, and the variety of such problems are surveyed in [21]. Although we ignore the issues of dynamics (timing, velocity, acceleration), a path is often used as the basis for solving restricted dynamics problems.

Exact path planning have been studied from the 1980s [38], and is reducible to the existential theory of connectivity of semi-algebraic sets (e.g., [14]). The output of an exact path planner is either a robot path, or a `NO-PATH` indicator if no path exists. Unfortunately, the exact path planning is largely impractical. Even in simpler cases, correct implementation are rare for two reasons: it requires exact algebraic number computation and has numerous degenerate conditions (even in the plane) that are hard to enumerate or detect (e.g., [16, p.32]). Correct implementations are possible using libraries such as LEDA or CGAL or our own Core Library that support exact algebraic number types (see [43,20]).

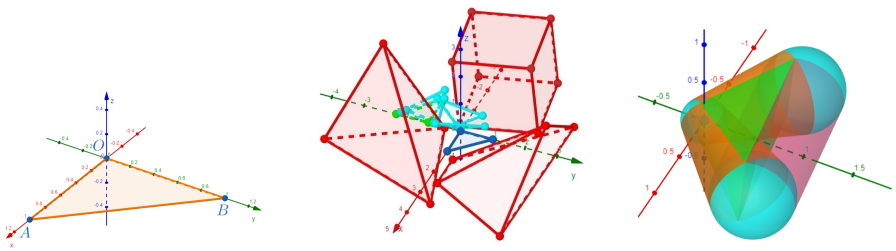
The last 30 years saw a flowering of practical path planning algorithms based on either the **Sampling Approach** (e.g., PRM, EST, RRT, SRT [10]) or the **Subdivision Approach** [26]. The dominance of Sampling Approach is described in a standard textbook in this area: "*PRM, EST, RRT, SRT, and their 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.* " [10, p.201]. Remarkably, the single bit of information, as encoded by `NO-PATH` output, is missing in the correctness criteria of these approaches as noted in [46]. The standard notions of **resolution completeness** (for Subdivision Approach) or **probabilistic completeness** (for Sampling Approach) ([10, Chapter 7.4]) do not talk about detecting no paths. Instead, they speak of *eventually finding a path* when "the resolution is small enough" (Subdivision Approach) or "when the sampling is large enough" (Sampling Approach). Both are recipes for non-terminating algorithms[3] but these are couched as "narrow passage issues" (e.g., [34,13]). See Appendix A in the full version of this paper [53] for the literature on this issue.

The Subdivision Approach goes back to the beginning of algorithmic robotics – see [6,58]. The present paper falls under this approach, but clearly a new theoretical foundation is needed. This foundation is ultimately based on interval methods [33] which is needed to provide guarantees in the presence of numerical approximation. The interval idea is encoded in the concept of **soft predicates**

[3] These are overcome by user-supplied "hyperparameters" that are not part of the original problem specification. Typically, it is some quitting criteria based on time-out or maximum sampling size.

[46]. The other foundation is the concept[4] of ε**-exactness** [46,47]. The idea here is rooted in an issue that afflicts all *exact* geometric algorithms: such algorithms must ultimately decide the sign of various computed numerical quantities, say x. For path planning, x might represent the clearance of the path, and we need x to be positive. Deciding the sign of x is easily reduced [43] to deciding if $x = 0$ ("the Zero Problem") . The Zero Problem might well be undecidable [9,43]. The concept of ε-exactness allows us to escape the Zero Problem. Clearly, both of the above concepts have wide spread ramification for computational geometry since all exact algorithms have implicit Zero Problems.

Based on this dual foundation, a general framework for path planning called **Soft Subdivision Search** (SSS) was formulated [46,47]. A series of papers [47,46,32,49,56,22], has shown that SSS planners are implementable and practical. They included planar fat robots [49] and complex robots [56], as well as spatial 5-DOF robots (rod and ring [22]). The latter represents the first rigorous and complete planner for a 5-DOF spatial robot. In each case, it was experimentally shown that SSS planners match or surpass the performance of state-of-art sampling algorithms. This is surprising, considering the much stronger theoretical guarantees of SSS, including its ability to decide `NO-PATH`.



(a) (b) (c)

Fig. 1: Delta Robot amidst obstacles Ω:
(a) Delta Robot defined by points $\mathcal{A} = (1, 0, 0), \mathcal{O} = (0, 0, 0), \mathcal{B} = (0, 1, 0)$.
(b) Sampled path ($\mathcal{AOB}$) from start ($\mathcal{AOB}$) to goal ($\mathcal{AOB}$) configurations.
(c) Approximate Footprint $\widetilde{Fp}(B)$ of box B.

In this paper, we address a well-known challenge of path planning: *to design a complete, rigorous and practical planner for a "spatial 6-DOF robot"*. It is not the 6 degrees of freedom per se (this is routinely achieved for robot arms), but the configuration space $SE(3) = \mathbb{R}^3 \times SO(3)$ that is challenging. Like similar challenges in the past (rod for $SE(2)$ and in $\mathbb{R}^3 \times S^2$), we choose a simple $SE(3)$ robot to demonstrate the principles. The robot is a planar triangle $\mathcal{AOB}$ in $\mathbb{R}^3$, a.k.a. **Delta robot**.[5] This is illustrated in Figure 1(a). Its "approximate

[4] For the reader's convenience, we reproduce the basic definitions such as soft predicates and ε-exactness in [53, Appendix B].

[5] Not to be confused with a class of parallel manipulator robots called **delta robots** E.g., https://en.wikipedia.org/wiki/Delta_robot

footprint" at some configuration box $B \subseteq SE(3)$ is shown in Figure 1(c). The path planning problem is specified as follows:

Given a polyhedral set $\Omega \subseteq \mathbb{R}^3$ of obstacles, we want to find an Ω-avoiding path from a start α to a goal β configuration:

Path Planning for $\mathcal{AOB}$-robot:
  Input: $(\alpha, \beta, \Omega, B_0, \varepsilon)$
    where $\alpha, \beta \in SE(3)$, B_0 is a box in $SE(3)$,
    $\Omega \in \mathbb{R}^3$ is a polyhedral obstacle set, and $\varepsilon > 0$ is the resolution.
  Output: an Ω-avoiding path of $\mathcal{AOB}$ restricted to B_0,
    from α to β or `NO-PATH`.

The ε parameter is used as follows:

Definition 1. *A path planner is said to be* ***resolution-exact*** *if it always terminates with an output satisfying these conditions: there is a constant $K > 1$ independent of the input (but depending on the planner) such that:*
*(**Path**) If the optimal clearance of a solution path is $> K\varepsilon$, then the planner outputs a path.*
*(**NoPath**) If there is no path of essential clearance $< \varepsilon/K$, then the planner outputs* `NO-PATH`.

The definition of clearance and other concepts are found in [53, Appendix B] and Section 2.1. The output is indeterminate because when the optimal clearance lies in $[\varepsilon/K, K\varepsilon]$, it can output (Path) or (NoPath). It can be argued that ε-exactness is an appropriate notion of "exactness" for real world applications because the physical world is inherently[6] inexact and uncertain. We believe this is the first completely rigorous alternative to exact path planning; see the Literature Review below for other attempts to resolve this issue.

1.1 What is an Explicit Algorithm in Computational Geometry?

As suggested by the title of this paper, our 6-DOF path planner is "explicit". This is an informal idea, attempting to characterize algorithms that are recognizably in computational geometry (CG). Classic CG algorithms (see [12,19]) are explicit in the sense that they construct well-defined combinatorial objects using explicit predicates. Moreover, these objects are embedded in the continuum such as $\mathbb{R}^n$ via approximate numerical constructions, called semi-algebraic models in [27, Sect.3.1.2, p.87]. E.g., Voronoi vertices are not just abstract vertices of a graph defined by their closest sites, but we typically need their approximate coordinates in $\mathbb{R}^n$. But when we address geometric problems which are non-linear or in non-Euclidean spaces many algorithms start to introduce highly non-trivial primitives such as the following:

[6] All common constants of physics and chemistry have less than 8 digits of accuracy. Among the few exceptions is the speed of light, which is exact by definition.

(P1) (Numerical Iteration) In their path planner for a spatial rod, Lee and Choset [28] used a retraction approach. To construct edges of the generalized Voronoi diagram in $\mathbb{R}^3 \times S^2$, they invoke a numerical gradient ascent method [28, p.355, column 2] to connect Voronoi vertices. Such constructions are not certified or guaranteed.

(P2) (Optimization) We will need to compute the distance between a line and a cone in $\mathbb{R}^3$ (see [53, Appendix C]). There is no known closed form expression, but one can reduce this to an optimization problem (using the Lagrangian formulation) or invoke an iterative procedure (e.g., [55]).

(P3) (Purely combinatorial description) Nowakiewicz [34] described a sampling-and-subdivision algorithm for a 6-DOF robot. The combinatorial steps and data structures are explicit, but the geometric/numerical primitives are unspecified (presumably out sourced to various numerical routines).

(P4) (Algebraic operations and solving systems) Is the intersection of two surfaces in $\mathbb{R}^3$ a geometric construction? Depending on the surface representation, this may be seen as a purely algebraic construction. As noted above, CG needs to extract numerical data from algebraic representations, and this amounts to solving of systems of algebraic equations (e.g., to compute Voronoi diagram of ellipses [17, Theorem 4.2]).

We regard algorithms such as (P1)-(P3) as "non-explicit". But (P4) is a harder call because nonlinear CG is inextricably connected to algebra. Some algebraic operations and analysis are inevitable. Moreover, solving polynomials systems can be seen as necessary geometric constructions for extracting numerical data from algebra. But invoking a generic polynomial solver inevitably gives rise to many irrelevant solutions (complex ones or geometrically wrong ones [17]) that must be culled. To the extent possible, we seek explicit expressions for such constructions. Non-explicit CG algorithms are useful and sometimes unavoidable, but their overall correctness and complexity is hard to characterize. We could largely identify "explicit" algorithms with those in semi-algebraic geometry [4].

To illustrate the "explicitness" achieved in this paper, we prove that our SSS planner for the Delta robot is ε-exact with resolution constant $K = 4\sqrt{6}+6\sqrt{2} < 18.3$. This constant is a small, manageable constant. It would be hard to derive such a constant if our primitives were not explicit.

1.2 Challenges in $SE(3)$ Path Planning

Despite the successful SSS planners from previous papers [47,46,32,49,56,22], there remain significant challenges in the theory and details. We expand on the four issues noted in the abstract:

(C1) By a "fundamental theorem" of SSS, we mean one that says that the SSS planner is resolution exact. Such a theorem was proved in the original paper [46], albeit for a disc robot. Subsequent papers implicitly assumed that the fundamental theorem extends to other robots. This became less clear in subsequent development as the underlying techniques were generalized

and configuration spaces became more complex. Partly to remedy this, [48] gave an axiomatic account of the Fundamental Theorem. The power of the axiomatic approach is that, to verify the correctness of any future instantiations of SSS, one only has to check the axioms. Part of axiomatization involves identifying the underlying mathematical spaces (called X, Y, Z, W below). There were 5 axioms, **(A0)**-**(A4)** in [48]. These axioms introduced constants C_0, D_0, L_0, σ and reveal their role in the implicit constant $K > 1$ of the definition of ε-exactness. The last axiom **(A4)** was problematic, and is remedied in this paper. In [48], the general Fundamental Theorem was stated but its proof was deferred.[7] We now complete this program.

(C2) The configuration space[8] $SE(3) = \mathbb{R}^3 \times SO(3)$ is the most general space for a rigid spatial robot, often simply called "6-DOF robot". A rigorous path planner for a $SE(3)$ robot would be a recognized milestone in robotics. The space $SO(3)$ is a non-Euclidean 3-dimensional space that lives naturally in 4-dimensions [24]. We will develop the algorithms and data structures to exploit a **Cubic Model** $\widehat{SO}(3)$ for $SO(3)$. This model is illustrated in Figure 2, and was known to Canny [8, p. 36]. The design of good

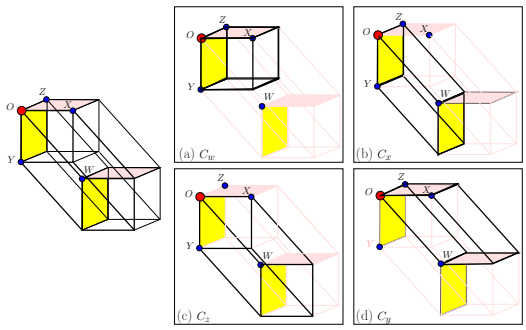


Fig. 2: The Cubic Model $\widehat{SO}(3)$ of $SO(3)$ from [48]

data structures in higher dimensions is generally challenging. For our application, our subdivisions must support the operation of splitting and adjacency query. The latter is a nontrivial issue and raises the question of maintaining smooth subdivisions [5]. In contrast, the sampling use of subdivision as in [34] has no need for adjacency queries.

(C3) The main primitive of sampling approaches is the classic **collision detection problem** (see [31]): *is a given configuration γ free?* There are off-the-shelf solutions from well-known libraries [31]. Our interval-based approach represents a nontrivial generalization: *is a box B of configurations free or stuck or neither?* Exact algorithms for this generalization is

[7] In retrospect, this deferment was appropriate in view of the problematic axiom (A4).

[8] Some authors write "$SE(3) = SO(3) \ltimes \mathbb{R}^3$" where $\ltimes$ is the semi-direct product [40] on the groups $SO(3)$ and $\mathbb{R}^3$. We forgo this algebraic detail as we are not interested in the group properties of $SE(3)$. We are only interested in $SE(3)$ as a metric space.

in general not possible (i.e., the footprint of B may not be semi-algebraic). But we can use the theory of **soft predicates** to design practical solutions.

(C4) The last challenge is to make the numerical/geometric computations "explicit" as explained above. This amounts to designing predicates and explicit algebraic expressions which allow a direct implementation. In short, we must avoid iterative procedures or general polynomial system solvers. Instead, we refine the general technique of Σ_2-decomposition from [22].

1.3 Literature Review

Lavalle [27] is a comprehensive overview of path planning; Halperin et al [21] gave a general survey of path planning. An early survey is [50] where two universal approaches to exact path planning were described: cell-decomposition [37] and retraction [36,35,7]. Since exact path planning is a semi-algebraic problem [38], it is reducible to general (double-exponential) cylindrical algebraic decomposition techniques [4]. But exploiting path planning as a connectivity problem yields singly-exponential time (e.g, [15]). The case of a planar rod (called "ladder") was first studied in [37] using cell-decomposition. More efficient (quadratic time) methods based on the retraction method were introduced in [41,42].

Spatial rods were first treated in [39]. The combinatorial complexity of its free space is $\Omega(n^4)$ in the worst case and this can be closely matched by an $O(n^{4+\epsilon})$ time algorithm [25]. Lee and Choset [28] gives a planner for a 3D rod using a retraction approach. Outside of the SSS planners, perhaps the closest to this paper is Nowakiewicz [34, p. 5383], who uses subdivision of the Cubic Model. But like many subdivision methods, this approach ultimately takes sample configurations (at the corners or centers) in subdivision boxes, and is actually a sampling method. The results were very favorable compared to pure sampling methods (PRM). For sampling-based planners, the main predicate is checking if a configuration is free; this is well-known **collision-detection problem** [31].

The theory of soft subdivision search is the first complete theory of path planning that overcomes the halting issue in non-exact planners. The following series of papers demonstrate that this theory leads to implementable algorithms whose efficiency beats the state-of-the-art sampling methods, up to 5 DOFs: [47,46,32,49,56,22].

There is a persistent misunderstanding of the fundamental "Zero Problem" of path planning. Since the problem has various names ("disconnection proof" [3], "non-existence of path" [52], "infeasibility proof" [30], etc), we will simply call it the NOPATH problem, and separately review this literature in [53, Appendix A].

1.4 Overview of Paper

Notation: We use bold font for vectors. E.g., $\boldsymbol{p} \in Z$ *where* $\boldsymbol{p} = (p_x, p_y, p_z)$. *Elements in* $SO(3)$ *are viewed either as* 3×3 *rotation matrices or as unit quaternions. In the latter case, we write* $\boldsymbol{q} = (q_0, \ldots, q_3) = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3 \in SO(3)$.

In Sect. 2, we present the axiomatic framework for SSS theory, and prove the Fundamental Theorem of SSS. In Sect. 3, we introduce the main geometric

primitive in the design of a soft predicate for the Delta Robot. Various techniques for its explicit evaluation are presented. Sect. 4 describes the data structures for representing Cubic Model $\widehat{SE}(3)$ of $SE(3)$. We conclude in Sect. 5.

Because of space limitation, additional details are deferred to five Appendices in the full version of this paper [53]: App. A reviews the NOPATH literature. App. B reviews basic concepts of SSS. App. C gives explicit "parameterized collision detection predicates" for special Σ_2-sets. App. D gives details about the adjacency structures for $\widehat{SE}(3)$. App. E proves the Fundamental Theorem.

2 The Fundamental Theorem of SSS

The Fundamental Theorem is about the SSS framework, which we review in [53, Appendix B]. This framework uses two standard data structures: a priority queue Q and a union-find structure U. The queue Q holds boxes in $\mathbb{R}^d$, and U maintains connectivity of boxes through their adjacency relations (B, B' are adjacent if $\dim(B \cap B') = d-1$). SSS has 3 subroutines.

- Subroutine $B \leftarrow Q.\texttt{GetNext}()$ that removes a box B of highest priority from Q. The search strategy of SSS amounts to defining this priority.
- Subroutine $\texttt{Expand}(B)$ that splits a box B into its set of children (subcells).
- A classifier $\widetilde{C}$ that assigns to each box B one of three values $\widetilde{C}(B) \in \{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$.

The search strategy has no effect on correctness, but our axioms will impose requirements on the other two subroutines.

2.1 The spaces of SSS Theory: W, X, Y and Z

Before stating the axioms, we review some spaces that are central to SSS theory.

Call $W := \mathbb{R}^d$ the **computational space** because the SSS algorithm operates on boxes in W. Here, $d \geq 1$ is at least the degree of freedom (DOF) of our robot. For $SE(3)$, we choose $d = 7$ (not $d = 6$) because we embed $SO(3)$ in $\mathbb{R}^4$ to achieve the correct topology of $SO(3)$. Let $\square W = \square\mathbb{R}^d$ denote[9] the set of tiles where a **tile** is defined to be a d-dimensional, compact and convex polytope of $\mathbb{R}^d$. Subdivision can be carried out using tiles (see [48]). By a **subdivision** of a tile B, we mean a finite set of tiles $\{B_1, \ldots, B_m\}$ such that $B = \bigcup_{i=1}^m B_i$ and $\dim(B_i \cap B_j) < d$ for all $i \neq j$. Suppose $\texttt{Expand}$ is a non-deterministic (i.e., multi-valued) function on $B \in \square W$ such that $\texttt{Expand}(B)$ is a subdivision of B. Using $\texttt{Expand}$, we can grow a subdivision tree $\mathcal{T}(B)$ rooted in $B \in \square W$, by repeated application of $\texttt{Expand}$ to leaves of $\mathcal{T}(B)$. The set of leaves of $\mathcal{T}(B)$ forms a subdivision of B. *General tiles are beyond the present scope; so we restrict them to axes-parallel boxes in this paper.*

[9] In [48], tiles were called **test cells**. The present tiling terminology comes from the literature on tiling or tessellation.

Next, $X := C_{space}(R_0)$ is the **configuration space** of our robot R_0. The robot lives in some **physical space** $Z := \mathbb{R}^k$ (typically $k = 2, 3$), formalized via the robot's **footprint map** $Fp = Fp^{R_0} : X \to 2^Z$ (power set of Z). In path planning, the input includes an **obstacle set** $\Omega \subseteq Z$. This induces the **clearance function** $C\ell : X \to \mathbb{R}_{\geq 0}$ where $C\ell(\gamma) := \text{Sep}(Fp(\gamma), \Omega))$ and $\text{Sep}(A, B) := \inf_{a \in A, b \in B} \|a - b\|$ denotes the **separation** between sets $A, B \subseteq Z$. We say γ is **free** iff $C\ell(\gamma) > 0$. Finally, $Y := C_{free}(R_0, \Omega)$ is the **free space**, comprised of all the free configurations.

What kind[10] of mathematical spaces are W, X, Y, Z? Minimally, we view them as metric spaces, each with its own metric: d_W, d_X, d_Y, d_Z. Since Z, W are normed linear spaces, we can take $d_Z(\boldsymbol{a}, \boldsymbol{b}) := \|\boldsymbol{a} - \boldsymbol{b}\|$ $(\boldsymbol{a}, \boldsymbol{b} \in Z)$, and similarly for d_W. Here, $\|\cdot\|$ is the Euclidean norm, i.e., 2-norm. Since $Y \subseteq X$, we can take d_Y to be d_X. But what is d_X? The space X can be[11] very diverse in robotics. For this paper, we assume $X = X^t \times X^r$ is the product of two metric spaces, a translational (X^t, d_T) and rotational (X^r, d_R) one. There are standard choices for d_T and d_R in practice. We can derive the metric d_X from d_T and d_R in several ways. If $a = (a^t, a^r), b = (b^t, b^r) \in X$, we have three possibilities:

$$d_X(a, b) := \sqrt{(d_T(a^t, b^t))^2 + \lambda \cdot (d_R(a^r, b^r))^2} \tag{1}$$

$$d_X(a, b) := \max\left\{d_T(a^t, b^t), \lambda \cdot d_R(a^r, b^r)\right\} \tag{2}$$

$$d_X(a, b) := d_T(a^t, b^t) + \lambda \cdot d_R(a^r, b^r) \tag{3}$$

where $\lambda > 0$ is a fixed constant. For definiteness, this paper uses the definition of (3) with $\lambda = 1$. To understand the use of λ, recall that X^r is a compact (angle) space and so d_R is bounded by a constant. We can take λ to be radius of the ball containing[12] R_0 and centered at the relative center of R_0. In this way, the pseudo-metric $d_H(a, b)$ (see next) bounds the maximum physical displacement.

We also need a pseudo-metric on X induced by the footprint map: given sets $A, B \subseteq Z$, let $d_H(A, B)$ denote the standard Hausdorff distance between them [26, p.86]. Given $\gamma, \gamma' \in X$, we define

$$d_H(\gamma, \gamma') := d_H(Fp(\gamma), Fp(\gamma')),$$

called the **Hausdorff pseudo-metric** on X. Although the original Hausdorff distance d_H is a metric on closed sets, the induced d_H is only a pseudo-metric in general: $d_H(\gamma, \gamma') = 0$ may not imply $\gamma = \gamma'$. E.g., if R_0 is a rod, two configurations can have the same footprint.

The case of $X = SE(3)$: Here $X^t = \mathbb{R}^3$ and $X^r = SO(3)$. As $X^t = Z$, we can choose $d_T = d_Z$ as above. Mathematically there is a natural choice for d_R as well: if $M, N \in SO(3)$ are viewed as 3×3 rotation matrices, we choose

[10] These spaces have many properties: this question asks for the minimal set of properties needed for SSS theory.

[11] For instance, if X is the configuration space of $m \geq 2$ independent, non-intersection discs in $\mathbb{R}^2$, then X is a subset of $\mathbb{R}^{2m}$ whose characterization is highly combinatorial.

[12] For a rigid robot R_0, we identify it with its footprint at $\mathbf{0} = (\mathbf{0}_t, \mathbf{0}_r) \in X^t \times X^r$. The **relative center** of R_0 is the point $\boldsymbol{c} \in Z$ which is invariant under any pure rotation $\gamma = (\mathbf{0}_t, \boldsymbol{q})$. Typically, we choose $\boldsymbol{c}$ to belong to $R_0 \subseteq Z$.

$d_R(M,N) = \|\log(MN^\mathsf{T})\|$ where $\log(MN^\mathsf{T})$ is an angular measure; see Huynh [24] who investigated 6 metrics Φ_i $(i = 1,\ldots,6)$ for $SO(3)$. Our d_R is the **natural metric** denoted Φ_6 in [24].

Normed linear spaces. It is not enough for Z and W to be metric spaces. For example, we need to decompose sets in Z using Minkowski sum $A \oplus B$. For W, we need to scale a tile B by some $\sigma > 0$ about a center $\boldsymbol{m}_B \in B$, denoted σB. This is used in defining **σ-effectivity**. These construction exploit the fact that Z, W are normed linear spaces.

2.2 Subdivision Charts and Atlases

We must now connect W and X. Subdivision in Euclidean space is standard, but the configuration space X is rarely Euclidean so that we cannot subdivide X directly. To solve this, we use the language of charts and atlases from differential geometry. By a **(subdivision) chart** of X, we mean a function $h : B \to X$ where $B \in \square W$ and h is a homeomorphism between B and its image $h(B) \subseteq X$. An **(subdivision) atlas** of X is a set $\mu = \{\mu_t : t \in I\}$ for some finite index set I such that each μ_t $(t \in I)$ is a chart, and if $X_t \subseteq X$ is the image of μ_t, then $\dim(\mu_t^{-1}(X_t \cap X_s)) < d$ $(t \neq s)$. From μ, we can construct a **tile model** of X, denoted X_μ, that is homeomorphic to X via a map $\overline{\mu} : X \to X_\mu$ (see [53, Appendix B.2]). Note that $\overline{\mu}$ is basically the inverse of the μ_t's: if $x \in B_t$, then $\overline{\mu}(\mu_t(x)) = x$.

A chart $\mu : B_t \to X$ is **good** if there exists a **chart constant** $C_0 > 0$ such that for all $q, q' \in B_t$, $1/C_0 \leq \frac{d_X(\mu(q),\mu(q'))}{\|q-q'\|} \leq C_0$. The subdivision atlas is **good** if there is an **atlas constant** C_0 that is common to its charts.

The case of $X = SE(3)$: First consider $X^r = SO(3)$, viewed as unit quaternions: the 4-cube $[-1,1]^4$ has eight 3-dimensional cubes as faces. After identifying the opposite faces, we have four faces denoted C_w, C_x, C_y, C_z (as illustrated in Figure 2). Let $I := \{w,x,y,z\} = \{0,1,2,3\}$ and $t \in I$. We view C_t as a subset of $\mathbb{R}^4$ where $\boldsymbol{q} = (q_0,\ldots,q_3) \in C_t$ implies $q_t = -1$. Define the chart: $\mu_t : C_t \to SO(3)$ by $\mu_t(\boldsymbol{q}) = \boldsymbol{q}/\|\boldsymbol{q}\|$ $(t \in I, \boldsymbol{q} \in C_t)$. The **cubic atlas** for $SO(3)$ is $\mu = \{\mu_t : t \in I\}$. The construction in [53, Appendix B.2] of the quotient space X_μ^r is called the **cubic model** of $SO(3)$, also denoted $\widehat{SO}(3)$. Moreover, our special construction ensures that X_μ^r is embedded in $\mathbb{R}^4$. Therefore $\widehat{SE}(3) := \mathbb{R}^3 \times \widehat{SO}(3)$ can be embedded in $\mathbb{R}^7$. We define $W := \mathbb{R}^7$.

2.3 The Axioms

We now state the 5 axioms in terms of the spaces X, Y, Z, W. Please refer to [53, Appendix B] for the definitions of the terms used here.

(A0) *(Softness)* $\widetilde{C}$ is a soft classifier for $Y \subseteq X$.

(A1) *(Bounded dyadic expansion)* The expansion $\texttt{Expand}(B)$ is dyadic and there is a constant $D_0 > 2$ such that $|\texttt{Expand}(B)| \leq D_0$, and each $B' \in \texttt{Expand}(B)$ has at most D_0 vertices and has aspect ratio at most D_0.

(A2) *(Pseudo-metric d_H is Lipschitz)* There is a constant $L_0 > 0$ such that for all $\gamma, \gamma' \in Y$, $d_H(\gamma, \gamma') < L_0 \cdot d_X(\gamma, \gamma')$.

(A3) *(Good Atlas)* The subdivision atlas μ has an atlas constant $C_0 \geq 1$:

$$\tfrac{1}{C_0} d_W(\overline{\mu}(\gamma), \overline{\mu}(\gamma')) < d_X(\gamma, \gamma') < C_0 \cdot d_W(\overline{\mu}(\gamma), \overline{\mu}(\gamma'))$$

(A4) *(Translational Cells)* Each box $B \subseteq \square W$ has the form $B = B^t \times B^r$ where $B^t \in \square Z$ and $Fp(B) = B^t \oplus Fp(B^r)$. Such boxes[13] are called **translational**.

Theorem 1 (Fundamental Theorem of SSS). *Assuming Axioms* **(A0)**-**(A4)**. *If the soft classifier is σ-effective, then SSS Planner is resolution exact with resolution constant*

$$K = L_0 C_0 D_0 \sigma$$

Application to our $SE(3)$ path planner: For our $SE(3)$ robot design, `Expand`(B) has at most 2^d congruent subboxes. Thus, we can choose $D_0 = 2^d$ of Axiom **(A1)**. We can easily show that the *the cubic atlases for $SO(3)$ is good*. However, to prove the exact bound for the distortion constant C_0 for $SO(n)$, we need the tools of differential geometry as in [54]. The remaining issue is Axiom **(A0)**, that classifier $\widetilde{C}$ must be σ-effective for some $\sigma > 1$. We will develop $\widetilde{C}$ in the next section and prove that it is $(2 + \sqrt{3})$-effective. Hence the Fundamental Theorem implies our SE(3) planner is resolution exact.

Next we briefly comment on these axioms. Axiom **(A1)** refers to "dyadic expansion": a tile is **dyadic** if its vertices are represented exactly by dyadic numbers (binary floats). Dyadic subdivision means that each tile is the expansion remains dyadic – this implies that we can carry out subdivision without any numerical error. Axiom **(A2)** shows that the Hausdorff pseudo metric d_H is Lipschitz in the metric d_X. This is actually a strengthening of the original axiom. It is strictly not necessary for the Fundamental Theorem.

We said that the advantage of the axiomatic approach is that it tells us precisely which axioms are needed for any property of our SSS planner. In particular, [48, Theorem 2] shows that Axioms **(A0)** and **(A1)** ensure the SSS planner halts. Very often, roboticists argue the correctness of their algorithms under the assumption of **exact** predicates and operations. What can we prove if the soft predicate of Axiom **(A0)** were exact? Then it can be shown [48, Theorem 3] that when the clearance is $> 2C_0 D_0 L_0 \varepsilon$, the planner produces a path under Axioms **(A0)**-**(A3)**. But what if we want an ε-exact algorithm? That means that an output of `NO-PATH` comes with a guarantee the clearance is $\leq K\varepsilon$ for some K. For such a result, [48, Theorem 5] invokes the problematic Axiom **(A4)**. We fix this issue in [53, Appendix E].

[13] The original definition of translational cells in [48] reads as follows: *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 $C\ell(c_0) \geq K_0 \cdot r_0(B)$.*

3 Approximate Footprint for Delta Robot: Computational Techniques

In this section, we describe the design of the approximate footprint of a box, and the techniques to compute the necessary predicates explicitly.

Axiom **(A0)** requires an effective soft predicate for boxes $B \in \square W$. To compute the exact classifier function, $C(B) \in \{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$, the method of features [46] says that it can be reduced to asking "is $Fp(B) \cap f$ empty?" for features $f \in \Phi(\Omega)$. Since the geometry of $Fp(B)$ is too involved, the paper [22] introduced the idea of **approximate footprint** $\widetilde{Fp}(B)$ as substitute for $Fp(B)$. To achieve soft predicates with effectivity $\sigma > 1$, we need:

$$Fp(B) \subseteq \widetilde{Fp}(B) \subseteq Fp(\sigma B). \tag{4}$$

We say $\widetilde{Fp}$ is σ-effective if it satisfies (4) for all B.

3.1 Design of $\widetilde{Fp}(B)$ for Delta Robot

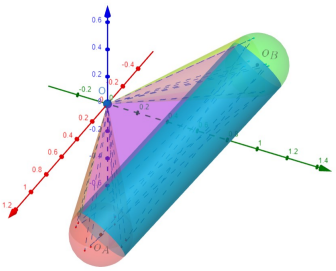


Fig. 3: Approximate rotation footprint $\widetilde{Fp}(B^r)$. Cf. Fig. 1(c).

Given $B = B^t \times B^r$, we have $Fp(B) = B^t \oplus Fp(B^r)$ (by translational axiom (A4)). Its **approximate footprint** of B is

$$\widetilde{Fp}(B) := Ball(B^t) \oplus \widetilde{Fp}(B^r) \tag{5}$$

where $\widetilde{Fp}(B^r) := \bigcup_{i=1}^{6} P_i = S_A \cup S_B \cup Cyl \cup Cone_A \cup Cone_B \cup Pyr$.

The sets $P_1, \ldots, P_6$ are comprised of two balls (S_A, S_B), a cylinder Cyl, two finite cones $(Cone_A, Cone_B)$ and a convex polytope Pyr (a pyramid with a rectangular base). The approximate footprint of B^r is illustrated in Figure 3. See [53, Appendix C]. In [53, Appendix E] we prove the following:

Theorem 2. *The approximate footprint of the Delta Robot is σ-effective where $\sigma = (2 + \sqrt{3}) < 3.8$.*

Theorem 3 (Correctness of Delta Robot Planner). *Our SSS planner for the Delta Robot is resolution exact with constant* $K = 4\sqrt{6} + 6\sqrt{2} < 18.3$.

Note that such constant is not excessive as it just means that we need at most five additional subdivision steps ($2^5 > 18.3$) to reach any desired resolution.

3.2 Parametric Separation Query and Boundary Reduction

Detecting collision [31] between two Euclidean sets $A, C \subseteq Z$ amounts to querying if their separation is positive: $\text{Sep}(A, C) > 0$. We generalize it to the query "Is $\text{Sep}(A, C) > s$?" which we call a **parametric separation query** (with parameter s). Note that we need not compute the $\text{Sep}(A, C)$ to answer this Yes/No query. The parametric query is useful because we are often interested in **fat objects**, i.e., sets of the form $A \oplus Ball(s)$. Detecting their collision with C reduces to a parametric query on A as in this simple lemma:

Lemma 1. *Let* $A, C \subseteq \mathbb{R}^n$ *be closed sets. Then* $(A \oplus Ball(s)) \cap C$ *is empty iff* $\text{Sep}(A, C) > s$.

In this and the next two subsections, we discuss techniques that are used to reduce the parametric separation query into ultimately explicit and implementable subroutines. Initially, the sets A, C in Lemma 1 are the approximate footprint $A = \widetilde{Fp}(B)$ (see (5)), and $C = \Omega$. Since $\widetilde{Fp}(B)$ is a fat version of $\widetilde{Fp}(B^r)$, we can replace $\widetilde{Fp}(B)$ by $\widetilde{Fp}(B^r)$. Using our method of features ([53, Appendix B]), we can replace C by a feature f of $\partial\Omega$. **Remark:** this technique could be used to simplify similar computations in the rod robot in [22].

Next, we address the problem of computing the separation $\text{Sep}(A, B) = \inf\{\|a - b\| : a \in A, b \in B\}$ between two closed semi-algebraic sets $A, B \subseteq \mathbb{R}^3$. Note that A is **semi-algebraic** means that it is the set of points that satisfy a set of equations and/or inequalities. If only equations are used, then A is **algebraic**. We say A is **simple** if there is a unique algebraic set $\overline{A}$ such that $A \subseteq \overline{A}$ and $\dim(A) = \dim(\overline{A})$. Call $\overline{A}$ the **algebraic span** of A. For instance, every feature $f \in \Phi(\Omega)$ is simple since, when A is a point/line-segment/triangle, then $\overline{A}$ is a point/line/plane (Ω is rational). But if $\dim(A) = 3$ then $\overline{A} = \mathbb{R}^3$.

For any two closed sets A, B, let $cp(A, B)$ be the **closest pair** set of $(\boldsymbol{a}, \boldsymbol{b}) \in A^\circ \times B^\circ$ such that $(\boldsymbol{a}, \boldsymbol{b})$ is a locally closest pair. Here A° is the relative interior of A in $\overline{A}$ (e.g., if A is a closed line segment, A° is a relatively open line segment). Using the algebraic spans $\overline{A}$ and $\overline{B}$, the set $cp(A, B)$ is (generically) contained in a finite zero-dimensional algebraic set S. Then $cp(A, B) = S \cap (A^\circ \times B^\circ)$.

Let us illustrate this idea. Assume the algebraic span $\overline{A}$ is the curve defined by the polynomial system $f_1 = f_2 = 0$; similarly $\overline{B}$ is the curve $g_1 = g_2 = 0$. Then the closest pair $(\boldsymbol{p}, \boldsymbol{q}) \in A^\circ \times B^\circ$ is among the solutions to the system

$$\begin{aligned} 0 &= f_1(\boldsymbol{p}) = f_2(\boldsymbol{p}) \\ 0 &= g_1(\boldsymbol{q}) = g_2(\boldsymbol{q}) \\ 0 &= \langle(\boldsymbol{p} - \boldsymbol{q}), \nabla f_1(\boldsymbol{p}) \times \nabla f_2(\boldsymbol{p})\rangle \\ 0 &= \langle(\boldsymbol{p} - \boldsymbol{q}), \nabla g_1(\boldsymbol{q}) \times \nabla g_2(\boldsymbol{q})\rangle \end{aligned} \tag{6}$$

where ∇f_i is the gradient of f_i, $\boldsymbol{u} \times \boldsymbol{v}$ and $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ are the cross-product and dot product of $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^3$. Note that (6) is a square system in 6 unknown variables $(\boldsymbol{p}, \boldsymbol{q})$ and generically has finitely many solutions. We say (A, B) is **degenerate** if the system has infinitely many solutions. The degenerate case is easily disposed of. Other examples of such computation are given in [53, Appendix C]. Using $cp(A, B)$, we now have a simple "reduction formula" for $\mathrm{Sep}(A, B)$:

Lemma 2 (Boundary Reduction Method). *Let $A \subseteq \mathbb{R}^3$ be a simple closed semi-algebraic set, and f be a feature. Assume $A \cap clos(f) = \emptyset$ where $clos(f)$ is the closure of f. Then* $\mathrm{Sep}(A, clos(f)) > s$ *iff*

$$(Q_0 > s) \wedge (Q_f > s) \wedge (Q_A > s)$$

where
$$Q_0 := \min\{\|\boldsymbol{a} - \boldsymbol{b}\| : (\boldsymbol{a}, \boldsymbol{b}) \in cp(A, B)\},$$
$$Q_f := \mathrm{Sep}(\partial A, f),$$
$$Q_A := \mathrm{Sep}(A, \partial f).$$

By definition, $Q_0 = \infty$ if $cp(A, B)$ is empty, and $Q_A = \infty$ if f is a corner.

This lemma reduces the parametric query to checking $Q_i > s$ for all $i = 0, A, f$. Note that $Q_0 > s$ can be reduced to solving a system like (6). By the method of features, checking if $\mathrm{Sep}(A, \Omega) > s$ can be reduced to checking if $\mathrm{Sep}(A, f) > s$ for all features $f \in \Phi(\Omega)$. Inevitably, we check all the $(i-1)$-dimensional features before checking the i-dimensional features $(i = 1, 2)$. Therefore, in application of this lemma, we would already know that $Q_A > s$ is true. Ultimately, the query reduces to an easy computation of $\mathrm{Sep}(\boldsymbol{a}, f) > s$ or $\mathrm{Sep}(A, \boldsymbol{a}) > s$ where $\boldsymbol{a}$ is a point. This technique had been exploited in our work, but becomes more important as the primitives becomes more complex. See its application in the next subsection and in [53, Appendix C].

3.3 On the Σ_2 Decomposition Technique

The reduction technique of Lemma 2 does not work when A is a complex 3-dimensional object like our approximate footprint. More precisely, the reduction requires us to characterize the various semi-algebraic patches that form the boundary of A. Instead, we use a different approach based on expressing A as a Σ_2-set as first introduced in [22].

First, we say that a set $B \subseteq \mathbb{R}^3$ is **elementary** if $B = \{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) \leq 0\}$ for some polynomial $f(X, Y, Z)$ of total degree at most 2, and the coefficients of f are algebraic numbers. Thus elementary sets include half-spaces, infinite cylinders, doubly-infinite cones, ellipsoids, etc. In our Delta robot, we will show that the algebraic coefficients of f are degree ≤ 2; by allowing a small increase in the effectivity constant, we can even assume degree 1 (i.e., $f(X, Y, Z)$ has integer coefficients). Next, a **Π_1-set** is defined as a finite intersection of elementary sets, and a **Σ_2-set** is a finite union of Π_1-sets. So A is a Σ_2-set if it can be written as

$$A = \bigcup_{i=1}^{m} \bigcap_{j=1}^{n} A_{ij}$$

where each A_{ij} is an elementary set, and each $A_i = \bigcap_{j=1}^{n} A_{ij}$ is a Π_1-set. We allow $A_{ij} = \emptyset$ to simplify notations. The simple double loop below can answer the question: "Is $f \cap A$ empty?" In [53, Appendix C] we show that our $\widetilde{Fp}(B)$ is a Σ_2-set.

Σ_2-Collision Detection(f, A):
  Input: f and $A = \bigcup_{i=1}^{m} \bigcap_{j=1}^{n} A_{ij}$.
  Output: `success` if $A \cap f = \emptyset$, `failure` else.
    For $i = 1$ to m
      $R \leftarrow f$
      For $j = 1$ to n
        $R \leftarrow R \cap A_{ij}$ (*)
        If $R = \emptyset$ break ◁ *exit current loop*
      If $R \neq \emptyset$, return `failure`
    Return `success`

The step (*) maintains R as the intersection of f with successive primitives. If f is a point or a line segment, this is trivial. When f is a triangle, this could still be solved in our previous paper for rod and ring robots [22]. But the present $\mathcal{AOB}$ robot requires us to maintain a planar set bounded by degree 2 curves; this requires a non-trivial algebraic algorithm. We do not consider this "explicit". Our solution is to explicitly write $A = \bigcup_{i=1}^{m} A_i$ where each $A_i = \bigcap_{j=1}^{n} A_{ij}$ has a very special form, namely, a convex and bounded Π_1-set of the following types:

$$\text{right cylinder, right cone, right frustum, convex polyhedron.} \tag{7}$$

By **right** cylinder, we mean that it is obtained by intersecting an infinite cylinder with two half-spaces whose bounding planes are perpendicular to the cylinder axis. The notion of right frustum is similar, but using a doubly-infinite cone instead of a cylinder. Thus the two "ends" of a right cylinder and a right frustum are bounded by two discs, rather than general ellipses. A right cone is a special case of a right frustum when one disc is just a single point.

We call the sets in (7) **special Π_1-sets**. A finite union of special Π_1-sets is called a **special Σ_2-set**. While the above Σ_2-collision detection does not extend to parametric queries, this becomes possible with special Σ_2-sets:

Theorem 4 (Parametric Special Π_1-set queries).
There are explicit methods for parametric separation queries of the form "Is $\mathrm{Sep}(P, f) > s$*?" where P is a special Π_1-set and f is a feature.*

REMARK: In [53, Appendix C], we introduce a further simplification to show that $\widetilde{Fp}(B)$ is the union of "very special" Σ_2-sets which are defined by polynomials of degree 2 whose coefficients have algebraic degree 2.

4 Subdivision in $\widehat{SE}(3)$: Adjacency and Splitting

We now address subdivision in the space $\widehat{SE}(3) = \mathbb{R}^3 \times \widehat{SO}(3)$. Let a box B in $\widehat{SE}(3)$ be decomposed as $B^t \times B^r$ where $B^t \in \square\mathbb{R}^3$ and $B^r \in \square\widehat{SO}(3)$. B^t

is standard, but B^r is slightly involved as shown next. Given an initial box $B_0 = B_0^t \times \widehat{SO}(3)$, the SSS algorithm will construct a subdivision tree $\mathcal{T} = \mathcal{T}(B_0)$ that is rooted at B_0. The leaves of $\mathcal{T}$ represent the (current) subdivision of B_0. We need an efficient method to access the adjacent boxes in the (current) subdivision. The number of adjacent boxes is unbounded; instead, we maintain only a bounded number of "principal" neighbors from which we can access all the other neighbors. For $\mathbb{R}^n$, this has been solved in [1] using $2n$ principal neighbors. We will show that for $\widehat{SO}(3)$ boxes, 8 principal neighbors suffice. So 14=6+8 principal neighbors suffice for boxes in $\mathcal{T}(B_0)$.

It remains to discuss principal neighbors in $\widehat{SO}(3)$. Following Section 2.2 and Figure 2, $\widehat{SO}(3)$ can be regarded as the union of four cubes, $\widehat{SO}(3) = \cup_{i=0}^3 C_i$ where $C_i := \big\{(a_0, \ldots, a_3) \in [-1, 1]^4 : a_i = -1\big\}$. The indices in $(0, 1, 2, 3)$ will also be identified with (w, x, y, z): thus $C_0 = C_w$, $C_1 = C_x$, etc. Let $d \in \{\pm e_0, \ldots, \pm e_3\}$ identify one of the 8 semi-axis directions (here e_i denotes the i-th standard basis vector). If two boxes B and B' are neighbors, there is a unique d such that B' is adjacent to B **in direction** d, denoted by $B \xrightarrow{d} B'$. In general, B' is not unique for a given B and d. See [53, Appendix D.1] for details.

We now describe the subdivision tree rooted at $\widehat{SO}(3)$: the first subdivision is special, and splits $\widehat{SO}(3)$ into 4 boxes C_i for $i = 0, 1, 2, 3$. Subsequently, each box is split into 8 children in an "octree-type" split. Each non-root box B maintains 8 **principal neighbor pointers**, denoted $B.d$ ($d \in \{\pm e_0, \ldots, \pm e_3\}$). However, only 6 of these pointers are non-null: if $B \subseteq C_i$ then $B.d$ is null iff $d = \pm e_i$. The non-null pointer $B.d$ points to the **principal d-neighbor** of B, which is defined as the box B' that is a d-neighbor of B whose depth is *maximal* subject to the restriction that $depth(B') \leq depth(B)$. Note that B' is unique and has size at least that of B. The non-null pointers for B are set up according to two cases. **Case** $B = C_i$: Each $B.d = C_j$ if $d = \pm e_j$ and $j \neq i$ (see Fig. 2). **Case** $B \neq C_i$: Three of the non-null pointers of B point to siblings and the other three point to non-siblings, and are determined as in [53, Appendix D].

5 Conclusion

Limitations. We are currently in the midst of implementing this work. We are not aware of any theoretical limitations, or implementability issues. One concern is how practically efficient is the current design (cf. [22, Sect. 1, Desiderata]). It is possible that additional techniques (mostly about searching and/or splitting strategies) may be needed to achieve real-time performance.

Extensions and Open Problems. Our SSS path planner for the Delta Robot is easily generalized to any "fat Delta robot" defined as the Minkowski sum $\mathcal{AOB} \oplus B(r)$ of $\mathcal{AOB}$ with a ball $B(r)$. Here are two useful extensions that appear to be reachable: (1) Extensions would be to "complex $SE(3)$ robots", where the robot geometry is non-trivial. In principle, the SSS framework allows such extensions [48]. (2) Spatial 7-DOF Robot Arm. In real world applications, robot arms normally need more than 6 degrees of freedom. In this case, the configuration space is a product of 2 or more rotational spaces.

References

1. B. Aronov, H. Bronnimann, A. Chang, and Y.-J. Chiang. Cost prediction for ray shooting in octrees. *Computational Geometry: Theory and Applications*, 34(3):159–181, 2006.
2. 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):198–214, 1995.
3. J. Basch, L. Guibas, D. Hsu, and A. Nguyen. Disconnection proofs for motion planning. In *IEEE Int'l Conf. on Robotics Animation*, pages 1765–1772, 2001.
4. S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry.* Algorithms and Computation in Mathematics. Springer, 2nd edition, 2006.
5. H. Bennett and C. Yap. Amortized analysis of smooth box subdivisions in all dimensions. In *14th Scandinavian Symp. and Workshops on Algorithm Theory (SWAT)*, volume 8503 of *LNCS*, pages 38–49. Springer-Verlag, 2014. July 2-4 2014. Copenhagen, Denmark. Appeared in CGTA.
6. 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.
7. J. Canny. Computing roadmaps of general semi-algebraic sets. *The Computer Journal*, 36(5):504–514, 1993.
8. J. F. Canny. *The complexity of robot motion planning.* ACM Doctoral Dissertion Award Series. The MIT Press, Cambridge, MA, 1988. PhD thesis, M.I.T.
9. 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. (SoCG'05)*, pages 116–125, 2005. June 5-8, Pisa, Italy.
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. D. Dayan, K. Solovey, M. Pavone, and D. Halperin. Near-optimal multi-robot motion planning with finite sampling. *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 39(5):9190–9196, 2021.
12. M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, revised 3rd edition edition, 2008.
13. J. Denny, K. Shi, and N. M. Amato. Lazy Toggle PRM: a Single Query approach to motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2407–2414, 2013. Karlsrube, Germany. May 2013.
14. M. S. E. Din and E. Schost. A nearly optimal algorithm for deciding connectivity queries in smooth and bounded real algebraic sets. *J. ACM*, 63(6):48:1–48:37, Jan. 2017.
15. M. S. el Din and E. Schost. A baby steps/giant steps probabilistic algorithm for computing roadmaps in smooth bounded real hypersurface. *Discrete and Comp. Geom.*, 45(1):181–220, 2011.
16. 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.

17. I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. The predicates for the Voronoi diagram of ellipses. *22nd ACM Symp. on Comp. Geom.*, pages 227–236, 2006.
18. O. Goldreich. On Promise Problems (a survey). In *Theoretical Computer Science: Essays in memory of Shimon Even*, pages 254 – 290. Springer, 2006. LNCS. Vol. 3895.
19. J. E. Goodman, J. O'Rourke, and C. Tóth, editors. *Handbook of Discrete and Computational Geometry*. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017.
20. D. Halperin, E. Fogel, and R. Wein. *CGAL Arrangements and Their Applications*. Springer-Verlag, Berlin and Heidelberg, 2012.
21. D. Halperin, O. Salzman, and M. Sharir. Algorithmic motion planning. In J. E. Goodman, J. O'Rourke, and C. Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017. Expanded from second edition.
22. C.-H. Hsu, Y.-J. Chiang, and C. Yap. Rods and rings: Soft subdivision planner for **R**ˆ3 x **S**ˆ2. In *Proc. 35th Symp. on Comp. Geometry (SoCG 2019)*, pages 43:1–43:17, 2019.
23. C.-H. Hsu, Y.-J. Chiang, and C. Yap. Rods and rings: Soft subdivision planner for **R**ˆ3 x **S**ˆ2. *arXiv e-prints*, arXiv:1903.09416, Mar 2019. This version includes appendices A–F, as well as calculations for disc-line separation.
24. D. Q. Huynh. Metrics for 3D rotations: Comparison and analysis. *J. Math. Imaging Vis.*, 35:155–164, 2009.
25. V. Koltun. Pianos are not flat: rigid motion planning in three dimensions. In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, pages 505–514, 2005.
26. J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
27. S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006.
28. J. Y. Lee and H. Choset. Sensor-based planning for a rod-shaped robot in 3 dimensions: Piecewise retracts of $R^3 \times S^2$. *Int'l. J. Robotics Research*, 24(5):343–383, 2005.
29. S. Li and N. Dantam. Learning proofs of motion planning infeasibility. *Robotics: Science and Systems*, 2021. Virtual Conference.
30. S. Li and N. Dantam. Scaling infeasibility proofs via concurrent, codimension-one, locally-updated coxeter triangulation. *IEEE Robotics and Automation Letters*, pages PP(99):1–8, Dec. 2023.
31. M. C. Lin and D. Manocha. Collision detection. In *Handbook of Data Structures and Applications*, pages 889–902. Chapman and Hall/CRC, 2018.
32. Z. Luo, Y.-J. Chiang, J.-M. Lien, and C. Yap. Resolution exact algorithms for link robots. In *Proc. 11th Intl. Workshop on Algorithmic Foundations of Robotics (WAFR '14)*, volume 107 of *Springer Tracts in Advanced Robotics (STAR)*, pages 353–370, 2015. Aug. 3-5, 2014, Boğazici University, Istanbul, Turkey.
33. R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, PA, 2009.
34. M. Nowakiewicz. MST-Based method for 6DOF rigid body motion planning in narrow passages. In *Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems*, pages 5380–5385, 2010. Oct 18–22, 2010. Taipei, Taiwan.
35. C. Ó'Dúnlaing, M. Sharir, and C. K. Yap. Retraction: a new approach to motion-planning. *ACM Symp. Theory of Comput.*, 15:207–220, 1983.
36. 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.
37. 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.
38. J. T. Schwartz and M. Sharir. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Appl. Math.*, 4:298–351, 1983.
39. J. T. Schwartz and M. Sharir. On the piano movers' problem: V. the case of a rod moving in three-dimensional space amidst polyhedral obstacles. *Comm. Pure and Applied Math.*, 37(6):815–848, 1984.
40. J. Selig. *Geometric Fundamentals of Robotics*. Springer, second edition, 2005.
41. 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.
42. 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.
43. V. Sharma and C. K. Yap. Robust geometric computation. In Goodman et al. [44], chapter 45, pages 1189–1224. Revised and expanded from 2004 version.
44. V. Sharma and C. K. Yap. Robust geometric computation. In J. E. Goodman, J. O'Rourke, and C. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 45, pages 1189–1224. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017.
45. Y. Sung and P. Stone. Motion planning (In)feasibility detection using a prior roadmap via path and cut search. In *Proc. Robotics: Science and Systems 2023*, July 2023.
46. C. Wang, Y.-J. Chiang, and C. Yap. On soft predicates in subdivision motion planning. *Comput. Geometry: Theory and Appl. (Special Issue for SoCG'13)*, 48(8):589–605, Sept. 2015.
47. C. Yap. Soft subdivision search in motion planning. In A. Aladren et al., editor, *Proc. 1st Workshop on Robotics Challenge and Vision (RCV 2013)*, 2013. A Computing Community Consortium (CCC) **Best Paper Award**, Robotics Science and Systems Conf. (RSS 2013), Berlin. In arXiv:1402.3213.
48. C. Yap. Soft subdivision search and motion planning, II: Axiomatics. In *Frontiers in Algorithmics*, volume 9130 of *Lecture Notes in Comp. Sci.*, pages 7–22. Springer, 2015. Plenary talk at 9th FAW. Guilin, China. Aug. 3-5, 2015.
49. C. Yap, Z. Luo, and C.-H. Hsu. Resolution-exact planner for thick non-crossing 2-link robots. In K. Goldberg, P. Abbeel, K. Bekris, and L. Miller, editors, *Algorithmic Foundations of Robotics XII: Proc. 12th WAFR 2016*, Springer Proceedings in Advanced Robotics, pages 576–591. Springer, 2020. (WAFR 2016: Dec. 13-16, 2016, San Francisco.) Book link: `https://www.springer.com/gp/book/9783030430887`. For proofs and more experimental data, see `arXiv:1704.05123 [cs.CG]`.
50. 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.
51. C. K. Yap. How to move a chair through a door. *IEEE J. of Robotics and Automation*, RA-3:172–181, 1987. Also: NYU-Courant Institute, Robotics Lab., No. 76, Aug 1986.

52. L. Zhang, Y. J. Kim, and D. Manocha. Efficient cell labeling and path non-existence computation using C-obstacle query. *Int'l. J. Robotics Research*, 27(11–12):1246–1257, 2008.
53. Z. Zhang, Y.-J. Chiang, and C. Yap. Theory and explicit design of a path planner for an SE(3) robot, arXiv:2407.05135, December 2024. Includes five appendices A–E.
54. Z. Zhang and C. Yap. Subdivision atlas and distortion bounds for SO(3), 2023. In Preparation.
55. Y. Zheng and C.-M. Chew. Distance between a point and a convex cone in n-dimensional space: Computation and applications. *IEEE Trans. on Robotics*, 25(6):1397–1412, 2009.
56. B. Zhou, Y.-J. Chiang, and C. Yap. Soft subdivision motion planning for complex planar robots. *Computational Geometry*, 92, Jan. 2021. Article 101683. Originally, in 26th ESA, 2018.
57. D. Zhu and J.-C. Latombe. Constraint reformulation in a hierarchical path planner. *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 1918–1923, 1990.
58. D. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991.