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]).

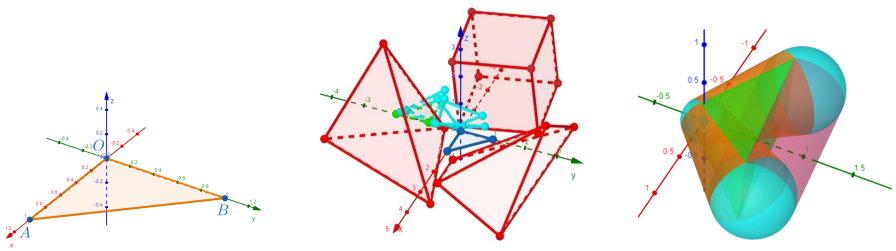
The last 30 years saw a flowering of practical path planning algorithms based on either the **Sampling Approach** (e.g., PRM, EST, RRT, SRT [10]) or the **Subdivision Approach** [26]. The dominance of Sampling Approach is described in a standard textbook in this area: "*PRM, EST, RRT, SRT, and their 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 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 Appendix B (see also [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 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 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

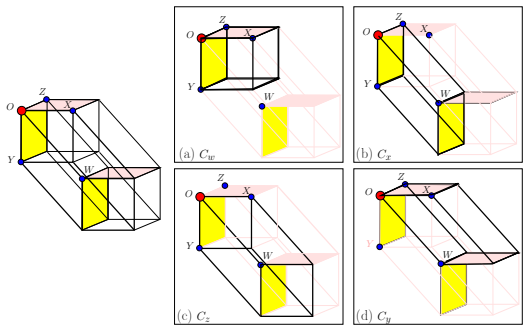


Fig. 2: The Cubic Model $\widehat{SO}(3)$ of $SO(3)$ from [48]

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 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 $SE(3)$.

Because of space limitation, additional details are deferred to five Appendices in the supplementary material of this paper (see also [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 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 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 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 Appendix B for definition of 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 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

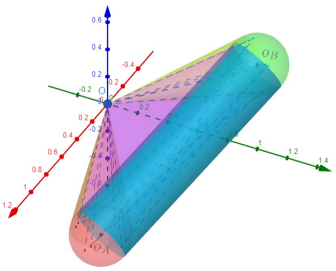


Fig. 3: Approximate rotation footprint $\widetilde{Fp}(B^r)$. Cf. Fig. 1(c).

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 Appendix C. In 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 (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 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 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?" Appendix C shows 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 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 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 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, July 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.

A Appendix: Review of the NOPATH Literature

We feel that the repeated misunderstanding about the fundamental "Zero Problem" [44] of path planning calls for a closer overview of the literature. As the problem goes by different names ("disconnection proof" [3], "non-existence of path" [52], "infeasibility proof" [30], etc), we will simply call it the NOPATH problem. In its starkest form, NOPATH is just a decision problem, with a YES/NO answer. Classical path planning extends this problem by asking for a path (if YES), and a simple "NO" otherwise. Call this the FINDPATH problem. In recent years (e.g., [45,30]), we see FINDPATH extended by requiring an "infeasibility proof" in case of a "NO" answer.[14] Since the NOPATH problem is embedded in FINDPATH or its extension, we can evaluate all path planners with respect to their completeness for NOPATH.

Before reviewing the literature, there are two preliminary remarks. First of all, the NOPATH problem is clearly solved by exact path planning, so we may only focus on algorithms based on numerical approximations and sampling. Second, we note that some algorithms only solve "promise problems" (e.g., [18]). In such problems, the inputs in addition to being well-formed, must satisfy some semantic conditions ("promise"). A **promise algorithm** is one whose correctness depends on such promises. Below, we will see such promise algorithms. If the promise implies the existence of paths, then clearly they do not solve the NOPATH problem.

The earliest paper that appears to offer a solution to NOPATH is Zhu and Latombe [57,58], whose "hierarchical framework" shares many features of our SSS framework. They made a nice observation that if there is no path in the adjacency graph of cells that are either EMPTY or MIXED[15], then it constitutes a proof of NOPATH. But they do not offer a complete method to detect NOPATH; see also Barbehenn and Hutchinson [2]. Unfortunately, the non-termination issue persists.

Non-Halting Example: Consider a planar disc robot of radius 1 and the obstacle set $\Omega = \{(x, 0) : |x| \geq 1\}$ where the robot must move from $\alpha = (0, -1)$ to $\beta = (0, 1)$. Then every box that contains $(0, 0)$ will be MIXED. This causes the halting problem for any subdivision approaches (and, a fortiori, for any sampling approaches).

An early reference for infeasibility proofs is Basch et al. [3] who aimed to find proofs when a robot cannot move through a "gate" in a 3D wall (they do not claim completeness). The exact solution for a 2D gate was first solved in [51]. Next, Zhang et al. [52] gave infeasibility proofs by giving a sufficient

[14] Note that the concept of providing a proof is always relative to some "proof checker" and this could vary dramatically depending on how powerful a proof checker one has in mind. E.g., for a proof of NOPATH, one often thinks of a cut that separates the start and goal configurations in the adjacency graph (so we need a "cut checker"). But one could also output the entire adjacency graph, and the checker can verify that there is no path.

[15] Our classification of boxes as `FREE`/`STUCK`/`MIXED` in SSS corresponds to their **labeling** of cells as EMPTY/FULL/MIXED.

criterion for classifying a cell as FULL (i.e., `STUCK`). Their heuristic is clearly incomplete. More recently, Li and Dantam [30,29] aimed to find infeasibility proofs by learning and other techniques. But they only have a promise algorithm, conditioned on the input having the ε-blocked property [30, p.2, Section III.A]. Here $\varepsilon > 0$ is a hyperparameter [30, p.7, Section VII.B] that the user must provide. Note that the concept of ε-blocked property is very close to our idea of ε-exactness. The difference is that they make this property a "promise" while our SSS algorithms take ε as an input.

Li and Dantam [30] proposed the idea to dovetail a PRM planner with an "infeasibility prover" (see [30, Figure 1]). This allows sampling based methods to sometimes produce NOPATH outputs. More generally, the concept of providing "infeasibility proofs" is only needed for incomplete algorithms (such as sampling methods). A complete algorithm does not need to provide proofs. In particular, exact algorithms or our SSS algorithms do not need to provide such proofs.

Another attempt at infeasibility proofs is Sung and Stone [45], but in the limited setting of a prior roadmap.

Path planners that accept $\varepsilon > 0, \delta > 0$ as part of the input are clearly trying to do approximation. Do they escape the NOPATH issue? E.g., Dayan et al. [11] defined a graph $G = G(\mathcal{X}, r, x^s, x^g)$ where $\mathcal{X} \subseteq C_{space}$, $r > 0$ and $x^s, x^g \in C_{space}$. An edge (u, v) of G represents a feasible path from u to v. Given ε, δ, they construct $\mathcal{X}$ and $r > 0$ such that $(\mathcal{X}, r)$ is (ε, δ)-complete in this sense: the inequality $d_G(x^s, x^g) \leq (1 + \varepsilon)OPT_\delta$ holds. Here $d_G(u, v)$ is the length of the shortest path in G, and OPT_δ is the length of the shortest path among the δ-clear paths. Unfortunately, if there is no path from x^s to x^g in C_{free} then $d_G(x^s, x^g) = OPT_\delta = \infty$. But if we assume that $OPT_\delta < \infty$, then this becomes a promise algorithm that assumes the existence of a path.

B Appendix: Soft Subdivision Search

We review the elements of SSS Framework [46,48]. Recall that in Section 2.1, we introduced the four spaces for path planning: the configuration space X, free space Y, physical space Z, and computation space W given by

$$W = \mathbb{R}^d, \quad X = C_{space}(R_0), \quad Y = C_{free}(R_0, \Omega), \quad Z = \mathbb{R}^k.$$

B.1 Soft Predicates

Given $A, B \subseteq Z$, their **separation** is defined as $\mathrm{Sep}(A, B) := \inf_{\boldsymbol{a} \in A, \boldsymbol{b} \in B} \|\boldsymbol{a} - \boldsymbol{b}\|$. A robot R_0 is defined by a continuous **footprint map** $Fp : X \to 2^Z$. This map may be regarded as a generalization[16] of the more well-known **forward kinematic map**. Continuity of Fp comes from the fact that X and 2^Z are

[16] Typically, a fixed point $\mathcal{A}$ on the robot is chosen and the footprint map $Fp_{\mathcal{A}}(\gamma)$ is the location of $\mathcal{A}$ in physical space. We generalize this to any set $\mathcal{S}$ of points on the robot: if $\mathcal{S}$ is the entire robot, we just write $Fp(\gamma)$ instead of $Fp_{\mathcal{S}}(\gamma)$.

topological spaces where 2^Z has the topology induced from the Hausdorff pseudo-metric on subsets of Z. Relative to an obstacle set $\Omega \subseteq Z$, its **clearance function** is $C\ell : X \to \mathbb{R}_{\geq 0}$ where $C\ell(x) := \text{Sep}(Fp(x), \Omega)$. We say $x \in X$ is **free** iff $C\ell(x) > 0$. We define Y as the set of free configurations in X. A **motion** is a continuous function $\mu : [0,1] \to X$. We call μ a **path** if the range of μ lies in Y (so $\mu(t)$ is free for all $t \in [0,1]$). We also call $\mu(0)$ and $\mu(1)$ the start and goal configurations of the path, and μ is a path from its start to its goal configuration. The **clearance** of a path μ is $\min\{C\ell(\mu(t)) : t \in [0,1]\}$. The **optimal clearance** between $x, y \in Y$ is the largest clearance of any path from x to y.

We now consider a somewhat non-intuitive concept of "essential clearance" first introduced in [46]: let μ be a path. We say μ has **essential clearance** $C > 0$ if there exists t_0, t_1 $(0 \leq t_0 < t_1 \leq 1)$ such that for all $t \in [0,1]$:

$$C\ell(\mu(t)) \begin{cases} \geq C & \text{if } \ t \in [t_0, t_1] \\ < C & \text{else.} \end{cases}$$

Note that having essential clearance C means that, with the exception of an initial segment $[0, t_0)$ and a final segment $(t_1, 1]$, the path has clearance at least C. If $t_0 = 0$ or $t_1 = 1$, this initial or final segment is null. The motivation for this definition is to make our resolution exact algorithms simpler, by not having to check such initial and final segments. E.g, our SSS planner finds a path by discovering a **channel**, i.e., a sequence of adjacent free boxes that connect the start and goal configurations α, β. All the free boxes in the channel have widths $\geq \varepsilon$. Then it is clear that we can find an path μ from α to β inside this channel whose essential clearance is $\geq \varepsilon/K$ where $K > 0$ depends on the algorithm. We cannot discount the possibility that the initial or final segment of μ may have clearance arbitrarily close to 0.

The concept of a "soft predicate" is relative to some exact predicate. Based on the clearance function, the **exact predicate** is $C : X \to \{0, +1, -1\}$ where $C(x) = 0/+1$ (resp.) if configuration x is semi-free/free; else $C(x) = -1$. The semi-free configurations are those on the boundary of Y. Call $+1$ and -1 the **definite values**, and 0 the **indefinite value**.

$$C(x) = \begin{cases} +1 & \text{if } \ x \in Y^\circ, \\ 0 & \text{else if } x \in \partial Y \\ -1 & \text{else.} \end{cases}$$

We can extend the definition to any set $B \subseteq X$: for a definite value v, define $C(B) = v$ iff $C(x) = v$ for all x. Otherwise, $C(B) = 0$.

Let $\square W$ denote the set of d-dimensional boxes in $W = \mathbb{R}^d$. In general, a box $B \in \square\mathbb{R}^k$ is called a **hypercube** if $B = \prod_{i=1}^k I_i$ is the product of intervals I_i of the same width. Let $w(B) = \min_{i=1}^k w(I_i)$ where $w([a,b]) = b - a$. The **diameter** of any set $S \in \mathbb{R}^n$ is the $\text{diam}(S) := \max_{\boldsymbol{a},\boldsymbol{b} \in S} \|\boldsymbol{a} - \boldsymbol{b}\|_2$. The **aspect ratio** of B is defined as $\rho(B) := \text{diam}(B)/w(B)$.

In our application of $SE(3)$, $W = \mathbb{R}^7$ and each box $B \in \square W$ has the decomposition $B = B^t \times B^r$ where $B^t \in \square\mathbb{R}^3$, and $B^r \in \square\mathbb{R}^4$. We assume that B^t is a hypercube, and define the "width" of B to be $w(B) := w(B^t)$.

For any box $B \in \square W$, let $C(B)$ as a short-hand for $C(\overline{\mu}(B))$ where $\mu : X_\mu \to X$ is the homeomorphism from the "square model" X_μ to X. A predicate $\widetilde{C} : \square W \to \{0, +1, -1\}$ is a **soft version of** C if it is conservative and convergent. **Conservative** means that if $\widetilde{C}(B)$ is a definite value, then $\widetilde{C}(B) = C(B)$. **Convergent** means that if for any sequence $(B_1, B_2, \ldots)$ of boxes, if $B_i \to p \in W$ as $i \to \infty$, then $\widetilde{C}(B_i) = C(p)$ $(= C(\overline{\mu}(p)))$ for i large enough. To achieve resolution-exact algorithms, we must ensure $\widetilde{C}$ converges quickly in this sense: say $\widetilde{C}$ is **effective** if there is a constant $\sigma > 1$ such if $C(\sigma B) = \pm 1$ (i.e., definite) then $\widetilde{C}(B) = C(B)$.

B.2 The Cubic Model X_μ

For any set $S \subseteq \mathbb{R}^m$, a **subdivision** of S is a finite set $\{S_1, \ldots, S_k\}$ such that $S = \cup_{i=1}^k S_i$ and $\dim(S_i \cap S_j) < \dim(S_i)$ for all $i \neq j$. We assume that S_i's are nice sets for which the notion of dimension, $\dim(S_i)$ and $\dim(S_i \cap S_j)$, are well-defined. In our applications, S_i's are boxes.

Let X be a topological space and I a finite index set. Recall the definitions of charts and atlases in Section 2.2. Let $\mu = \{\mu_t : t \in I\}$ be a **subdivision atlas** of X. For each $t \in I$, let $\mu_t : B_t \to X$ where $B_t \in \square\mathbb{R}^m$ for some fixed m. We construct the space X_μ as the following quotient space: Let $X_\mu^+ := \uplus_{t \in I} B_t$ the disjoint union of the B_t's. Then X_μ is the quotient space $X_\mu^+ / \sim$ where $a \sim b$ for $a, b \in X_\mu^+$ iff $a \in B_s$ and $b \in B_t$ implies $\mu_s(a) = \mu_t(b)$. Let $[b]$ denote the equivalence class of b. Finally, we can define the map $\overline{\mu} : X \to X_\mu$ where $\overline{\mu}(x) = [b]$ iff $b \in B_t$ and $\mu_t(b) = x$.

For $SO(3)$, we have the atlas $\mu = \{\mu_w, \mu_x, \mu_y, \mu_z\}$ in which $\mu_t : B_t \to SO(3)$ $(t \in \{w, x, y, z\})$. The boxes B_w, B_x, B_y, B_z are specially chosen so that X_μ is embedded in $\mathbb{R}^4$. Obviously, X_μ has a non-Euclidean topology.

B.3 The SSS Framework

An SSS algorithm maintains a subdivision tree $\mathcal{T} = \mathcal{T}(B_0)$ rooted at a box $B_0 \subseteq \square W$. Each tree node is a subbox of B_0. In Axiom **(A1)**, we view $\texttt{Expand}(B)$ as a set of boxes that represent a subdivision of B. If $B \in \square\mathbb{R}^m$ has dimension $p \leq m$, the **canonical expansion** $\texttt{Expand}_0(B)$ of B is the set of 2^p congruent subboxes of B that form a subdivision of B; for simplicity, we may assume this canonical expansion; but see [48] for other expansions. In the SSS framework, we also have a procedure (still called) $\texttt{Expand}(B)$, which acts as follows: given a leaf B of $\mathcal{T}(B_0)$, it converts B into an internal node whose children form the set $\texttt{Expand}(B)$. The expand procedure will immediately classify each B' in the set $\texttt{Expand}(B)$ using a soft predicate $\widetilde{C}$, and perform some additional actions as outlined below. Thus, we see that the tree $\mathcal{T}$ is initially just the root B_0 and it grows by repeated expansion of its leaves. The set of leaves of $\mathcal{T}$ at any moment constitute a subdivision of B_0. For our SE(3) subdivision tree, the root B_0 is initially $B_0^t \times \widehat{SO}(3)$. Our expansion of any box $B = B^t \times B^r$ takes a specific form: it is either $\texttt{Expand}(B^t) \times B^r$ or $B^t \times \texttt{Expand}(B^r)$. This is the T/R Split

idea in [32, Section 3]. However, when $B^r = \widehat{SO}(3)$, this expansion is special: we always have $\texttt{Expand}(B^r) = \{C_w, C_x, C_y, C_z\}$ as illustrated in Figure 2.

The SSS Algorithm maintains the subdivision tree $\mathcal{T}$ using three WHILE-loops – see box below. The goal is to find a path from the start α to the goal β where $\alpha, \beta \in X$; or report `NO-PATH`, satisfying the requirements of ε-exactness (see Definition 1, Section 1). Let $Box_{\mathcal{T}}(\alpha)$ denote the leaf box of $\mathcal{T}$ that contains α. The first WHILE-loop keeps expanding $Box_{\mathcal{T}}(\alpha)$ until it becomes `FREE`, or returns `NO-PATH` when $Box_{\mathcal{T}}(\alpha)$ has width less than ε. The second WHILE-loop does the same for $Box_{\mathcal{T}}(\beta)$.

The last WHILE-loop (Main Loop) depends on three data structures, Q, G, U:

(a) A priority queue Q contains[17] only `MIXED` boxes.
(b) An **adjacency graph** G whose nodes are the `FREE` boxes in $\mathcal{T}$, and whose edges connect pairs of adjacent boxes, i.e., pairs that share a $(d-1)$-face.
(c) A Union-Find data structure U to represent the connected components of G.

The above $\texttt{Expand}(B)$ procedure takes these additional actions: for each B' in the set $\texttt{Expand}(B)$, if $w(B') < \varepsilon$ or $\widetilde{C}(B') = \texttt{STUCK}$, we discard B'. If $\widetilde{C}(B') = \texttt{FREE}$, we insert B' into the adjacency graph G; we also insert B' into U and perform $U.Union(B', B'')$ with boxes B'' in G which are adjacent. If $\widetilde{C}(B') = \texttt{MIXED}$, it is pushed into Q.

The main WHILE loop will keep expanding $Q.\texttt{GetNext}()$ until a path is detected or Q is empty. If Q is empty, it returns `NO-PATH`. Paths are detected when the Union-Find data structure tells us that $Box_{\mathcal{T}}(\alpha)$ and $Box_{\mathcal{T}}(\beta)$ are in the same connected component. It is then easy to construct a path in G to connect $Box_{\mathcal{T}}(\alpha)$ and $Box_{\mathcal{T}}(\beta)$. Thus we get:

[17] From the procedure $\texttt{Expand}(B)$, each box B' in the tree $\mathcal{T}$ has a classification $\widetilde{C}(B') \in \{\texttt{MIXED}, \texttt{FREE}, \texttt{STUCK}\}$.)

SSS Framework:

Input: start and goal configurations $\alpha, \beta \in X$,
obstacle set $\Omega \subseteq Z$, resolution $\varepsilon > 0$, initial box $B_0 \in \square W$.
Output: Path from α to β in $Y \cap \mu(B_0)$ or `NO-PATH`.
Initialize a subdivision tree $\mathcal{T}$ with root B_0.
Initialize data structures Q, G and U.
While $(Box_{\mathcal{T}}(\alpha) \neq$ `FREE`$)$
If width of $Box_{\mathcal{T}}(\alpha)$ is $< \varepsilon$, Return(`NO-PATH`)
Else `Expand`$(Box_{\mathcal{T}}(\alpha))$
While $(Box_{\mathcal{T}}(\beta) \neq$ `FREE`$)$
If width of $Box_{\mathcal{T}}(\beta)$ is $< \varepsilon$, Return(`NO-PATH`)
Else `Expand`$(Box_{\mathcal{T}}(\beta))$
▷ *MAIN LOOP:*
While $(U.Find(Box_{\mathcal{T}}(\alpha)) \neq U.Find(Box_{\mathcal{T}}(\beta)))$
If $Q_{\mathcal{T}}$ is empty, Return(`NO-PATH`)
$B \leftarrow Q_{\mathcal{T}}$.`GetNext`()
`Expand`(B)
Using G, construct a channel $P = (B_1, B_2, \ldots, B_k)$ comprising adjacent free boxes. Construct a **canonical path** $\overline{P} = (\alpha, b_1, c_1, b_2, \ldots, c_{k-1}, b_k, \beta)$ where $b_i = m(B_i)$ and $c_i = m(B_i \cap B_{i+1})$.

The correctness of SSS does not depend on search strategy (i.e., priority) of Q. However, choosing a good search strategy can have a great impact on performance. Two strategies with the best results are Greedy Best First and some kind of Voronoi strategy.

B.4 Method of Features

In SSS framework, the obstacle set $\Omega \subseteq Z = \mathbb{R}^3$ is a closed polyhedral set. We may assume its boundary $\partial\Omega$ is bounded. Then $\partial\Omega$ is partitioned into a set of boundary **features**: **corners** (points), **edges** (relatively open line segments), or **walls** (relatively open triangles). Let $\Phi(\Omega)$ denote the set of features of Ω. The (minimal) set of corners and edges are uniquely defined by Ω, but walls depend on a triangulation of $\partial\Omega$.

Our approach to soft predicates is based on the "method of features" [47,46]. The **exact feature set** of B is

$$\phi(B) := \{f \in \Phi(\Omega) : f \cap Fp(B) \neq \emptyset\}. \tag{8}$$

This is too hard to compute, and we want an approximation $\widetilde{\phi}(B)$ with the property $\phi(B) \subseteq \widetilde{\phi}(B)$. This property holds if we define the **approximate feature set** of box B as

$$\widetilde{\phi}(B) := \left\{f \in \Phi(\Omega) : f \cap \widetilde{Fp}(B) \neq \emptyset\right\} \tag{9}$$

where $\widetilde{Fp}(B)$ is the approximate footprint defined in Section 3.1. Moreover the property (4) on approximate footprint $\widetilde{Fp}(B)$, i.e.,

$$Fp(B) \subseteq \widetilde{Fp}(B) \subseteq Fp(\sigma B)$$

ensures the resolution exactness of our algorithm. Hence,

$$\phi(B) \subseteq \widetilde{\phi}(B) \subseteq \phi(\sigma B). \tag{10}$$

The idea is to maintain $\widetilde{\phi}(B)$ for each box B in the subdivision. We softly classify B as $\widetilde{C}(B) = \texttt{MIXED}$ as long as $\widetilde{\phi}(B)$ is non-empty; otherwise, we can decide whether $\widetilde{C}(B) = C(B)$ is `FREE` or `STUCK`. For computational efficiency, we want the approximate feature sets to have **inheritance property**, i.e.,

$$\widetilde{\phi}(B) \subseteq \widetilde{\phi}(parent(B)). \tag{11}$$

This can be ensured by a trick in [22].

C Appendix: Explicit Parameterized Collision Detection Computation

Our previous paper [22] already provided explicit algorithms for the collision detection between obstacle features and some simple Π_1-sets, as summarized in the lemma below.

Lemma 3. *([22]) Let A be a point, an edge or a ball, and let f be a feature. There are explicit procedures to answer the parametric separation query "Is* $\mathrm{Sep}(A, f) > s$*?".*

In this paper, we provide new algorithms so that A can be a special Σ_2-set. (cf. Sec. 3.2 on parameterized collision detection.)

We remark that when A is a cone and f is a line, there appears to be no known explicit solution. We only found iterative procedures [55] or Lagrangian minimization formulations, which are not explicit.

C.1 General Geometric Notations

We give some basic definitions and notations used in Euclidean geometry $\mathbb{R}^m$. Recall from Sec.3.2 that a set $A \subseteq \mathbb{R}^m$ is **simple** if there is a unique algebraic set $\overline{A}$ such that $A \subseteq \overline{A}$ and $\dim(A) = \dim(\overline{A})$. In this case, we call $\overline{A}$ an **algebraic span** of A.

Let ∂A denote the boundary of a set A. Given two points u, v, let $\mathrm{seg}(u, v)$ denote the segment connecting them, and $|\mathrm{seg}(u, v)|$ denote its length. Let $\mathrm{Ball}(v, r)$ denote the ball centered at v with radius r. If v is the origin $\mathbf{0}$, we just write $\mathrm{Ball}(r)$. Let $cone(v, o, r)$ denote the right cone with apex v whose base is a disc centered at o of radius $r > 0$. So the boundary of $cone(v, o, r)$ can be decomposed into a circular disc $disc(o, r)$ and a conic surface which we call a **traffic**

cone, denoted by $\mathrm{tc}(v, o, r)$. Note that the algebraic span $\overline{\mathrm{tc}(v, o, r)}$ is an algebraic double cone surface whose equation is $x^2 + y^2 = r^2 z^2$ up to a coordinate transformation.

We also introduce an important shape for our analysis, the **ice-cream cone** $\mathrm{icc}(v, o, r)$ which is defined as the union of $cone(v, o, r)$ and a ball B such that B is tangential to the traffic cone $\mathrm{tc}(v, o, r)$. Note that the $disc(o, r)$ is a section of B (see Figure 4). The relation between ice-cream cone and its corresponding traffic cone is given by this lemma:

Lemma 4. *The ice-cream cone $icc(v, o, r)$ is the union of $cone(v, o, r)$ with the ball $Ball(c, R)$ whose center c on the axis of the cone satisfies $|seg(v, c)| = \frac{h^2+r^2}{h}$ where $h = |seg(v, o)|$. The radius R is given by $\frac{h^2+r^2}{r}$.*

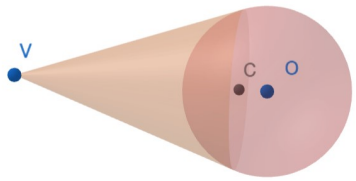


Fig. 4: An ice-cream cone $\mathrm{icc}(v, c, r)$ for some $r > 0$.

Given any box $B \in \square W$, we can express it as $B = B^t \times B^r$, where $B^t \in \square\mathbb{R}^3$ and $B^r \subseteq \widehat{SO}(3)$. We now define $m(B), w(B)$ and $r(B)$ as follows: The **center** of B, denoted $m_B = m(B)$, is the center of B^t. Note that $m(B)$ is just the center of B^t, and independent of B^r. The **width** of B, denoted by $w_B = w(B)$, is similarly just the width of B^t. The **radius** of B, denoted by $r_B = r(B)$, is given by $r(B) = \sqrt{3}w_B/2$. Thus, the circumball of B^t is simply $\mathrm{Ball}(B^t) = \mathrm{Ball}(m_B, r_B)$.

C.2 Rearranging Approximate Footprint into Special Σ_2 Sets

In equation (5), we informally define our approximate footprint $\widetilde{Fp}(B)$ as $\mathrm{Ball}(B^t) \oplus \widetilde{Fp}(B^r)$ where

$$\widetilde{Fp}(B^r) = \cup_{i=1}^{6} P_i.$$

Note that this means our $\widetilde{Fp}(B)$ is a fat version of $\widetilde{Fp}(B^r)$. In general, $A \oplus \mathrm{Ball}(r)$ is a **fat version** of A.

Previously the sets P_i were not fully specified there. We will see that each P_i is, in fact, a special Π_1-set. Therefore, $\widetilde{Fp}(B^r)$ a special Σ_2-set.

As preliminary, consider the 8 corners $c_1, \ldots, c_8$ of the box B^r. Each c_i represents a rotation of the robot $\Delta\mathcal{AOB}$. Let $\mathcal{A}_i \in \mathbb{R}^3$ denote the position of $\mathcal{A}$ under this rotation. Similarly for $\mathcal{B}_i$. Then $o_{\mathcal{A}}$ is defined as the center of gravity of these points, $o_{\mathcal{A}} = \frac{1}{8}\sum_{i=1}^{8} \mathcal{A}_i$. Similarly for $o_{\mathcal{B}}$. Let $d(B)$ be the maximum distance from $o_{\mathcal{A}}$ to any $\mathcal{A}_i$ and from $o_{\mathcal{B}}$ to any $\mathcal{B}_i$ $(i = 1, \ldots, 8)$. See Figure 5.

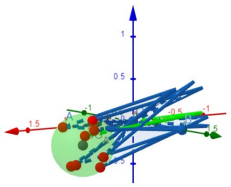


Fig. 5: Getting $S_{\mathcal{A}}$ from the $\mathcal{A}_1 \ldots \mathcal{A}_8$.

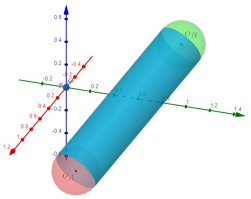


Fig. 6: The Cyl makes the convex hull of $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$

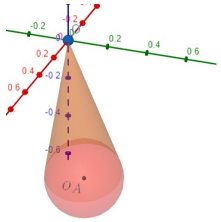


Fig. 7: The $Cone_{\mathcal{A}}$ makes the convex hull of $\mathcal{O}$ and $S_{\mathcal{A}}$

- $P_1 = S_{\mathcal{A}}$ is the Ball$(o_{\mathcal{A}}, d(B))$.
- $P_2 = S_{\mathcal{B}}$ is the Ball$(o_{\mathcal{B}}, d(B))$.
- $P_3 = Cyl$ is the right-cylinder with $o_{\mathcal{A}}$ and $o_{\mathcal{B}}$ as the centers of its two base discs and $d(B)$ as its radius. Thus, $Cyl \cup S_{\mathcal{A}} \cup S_{\mathcal{B}}$ is just the convex hull of $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$. See Figure 6
- $P_4 = Cone_{\mathcal{A}}$ is right-cone with origin $\mathcal{O}$ as its apex and a circular base that is tangent to $S_{\mathcal{A}}$. Then $Cone_{\mathcal{A}} \cup S_{\mathcal{A}}$ is the convex hull of $\mathcal{O}$ and $S_{\mathcal{B}}$. See Figure 7.
- $P_5 = Cone_{\mathcal{B}}$ is analogous to $Cone_{\mathcal{A}}$.
- $P_6 = Pry$ is a pyramid with apex at $\mathcal{O}$ and a rectangular base that is tangential to Cyl. Thus the union $\cup_{i=1}^{6} P_i$ is the convex hull of $\mathcal{O}$, $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$. See Figure 3. Note that each P_i is a special Π_1-set.

Now we apply $\widetilde{Fp}(B) = \text{Ball}(B^t) \oplus \widetilde{Fp}(B^r)$ to get the Σ_2 representation of $\widetilde{Fp}(B)$. The $\widetilde{Fp}(B)$ can be decomposed into the union of 7 special Π_1 sets. They are the following:

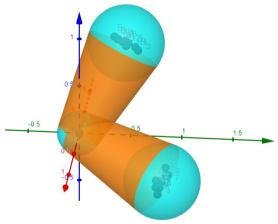


Fig. 8: The frustums make the convex hull of $S_{\mathcal{A}}$ and $S_{\mathcal{O}}$ as well as the convex hull of $S_{\mathcal{B}}$ and $S_{\mathcal{O}}$.

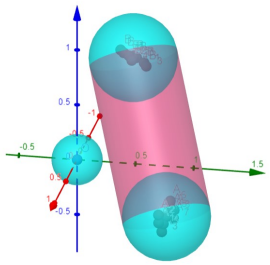


Fig. 9: The Cyl makes the convex hull of $S^+_{\mathcal{A}}$ and $S^+_{\mathcal{B}}$.

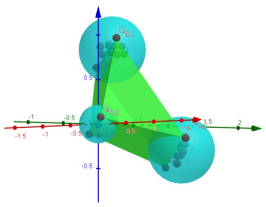


Fig. 10: The Pyr^+ makes the convex hull of $S_{\mathcal{A}}$, $S_{\mathcal{B}}$ and $S_{\mathcal{O}}$

- $S_{\mathcal{O}} = \text{Ball}(B^t)$ (small cyan ball center at $\mathcal{O}$);
- $S^+_{\mathcal{A}} = S_{\mathcal{A}} \oplus \text{Ball}(B^t) = \text{Ball}(o_{\mathcal{A}} + m_B, r(B) + d(B))$ (large cyan ball center at $\mathcal{A}$);
- $S^+_{\mathcal{B}} = S_{\mathcal{B}} \oplus \text{Ball}(B^t) = \text{Ball}(o_{\mathcal{B}} + m_B, r(B) + d(B))$ (large cyan ball center at $\mathcal{B}$);
- $Frust_A$ is the right-frustum whose union with $S_{\mathcal{O}}$ and $S^+_{\mathcal{A}}$ will result in the convex hull of $S_{\mathcal{O}}$ and $S^+_{\mathcal{A}}$ (orange frustum);
- $Frust_B$ is the right-frustum whose union with $S_{\mathcal{O}}$ and $S^+_{\mathcal{B}}$ will result in the convex hull of $S_{\mathcal{O}}$ and $S^+_{\mathcal{B}}$ (orange frustum), see Figure 8;
- Cyl^+ is the right-cylinder whose union with $S^+_{\mathcal{A}}$ and $S^+_{\mathcal{B}}$ will result in the convex hull of $S^+_{\mathcal{A}}$ and $S^+_{\mathcal{B}}$ (pink cylinder), see Figure 9;
- Pyr^+ is the polyhedron whose union with $S^+_{\mathcal{A}}$, $S^+_{\mathcal{B}}$, $S_{\mathcal{O}}$, $Frust_A$, $Frust_B$ and Cyl^+ will result in the convex hull of $S^+_{\mathcal{A}}$, $S^+_{\mathcal{B}}$ and $S_{\mathcal{O}}$ (green polyhedron), see Figure 10.

C.3 Exploiting a very special Σ_2-decomposition of $\widetilde{Fp}(B)$

To implement collision detection of the approximate footprint $\widetilde{Fp}(B)$ with a feature using the above special Σ_2-decomposition of $\widetilde{Fp}(B)$, our boundary reduction technique requires us to compute the separation of a disc and a feature. This requires computing roots of a degree 4 polynomial (see Appendix C.4. and [23, Appendix D.1]. The following lemma shows that we can reduce this computation to parametric separation query that amounts to checking a polynomial inequality of degree 2 and whose coefficients are algebraic of degree 2 (i.e., square-roots).

Lemma 5. *If $B \in \square W$ has corners whose coordinates are rational numbers, then $\widetilde{Fp}(B)$ is a Σ_2-set whose defining polynomials have coefficients of degree at most 2.*

Proof. If $q = (1, x, y, z) \in \widehat{SO}(3)$ has rational coordinates, $q/\|q\| \in SO(3)$, then the corresponding 3×3 orthogonal matrix M_q has rational entries. As a result, the ball $S_{\mathcal{A}}$ has rational center and radius is a square root of a rational number. So the equation defining $S_{\mathcal{A}}$ is a polynomial of degree 2 with rational coefficients. Moreover, $r(B) = \frac{\sqrt{3}}{2} w(B)$ is a square root of rational number. It implies that $S^+_{\mathcal{A}} = S_{\mathcal{A}} \oplus \text{Ball}(B^t)$ is a polynomial whose coefficients have algebraic degree 2. The other cases are similar. **Q.E.D.**

To exploit the above lemma, we need a new "very special" Σ_2-decomposition of our approximate footprint, namely as the union of special Σ_2-sets (such as ice-cream cones and fat line segments). Note that ice-cream cones and fat line segments avoid computations involving discs.

Lemma 6. *Given box $B \in \square W$ and feature f, $\widetilde{Fp}(B) \cap f = \emptyset$ if and only if*

$$\begin{aligned}(\text{Sep}(icc_{\mathcal{A}}, f) > r(B)) \wedge (\text{Sep}(icc_{\mathcal{B}}, f) > r(B)) \\ \wedge (\text{Sep}(seg(o_{\mathcal{A}}, o_{\mathcal{B}}), f) > d(B) + r(B)) \\ \wedge (\text{Sep}(Pyr^+, f) > 0),\end{aligned}$$

where $icc_{\mathcal{A}} = icc(m_B, m_B + o_{\mathcal{A}}, d(B))$ *and* $icc_{\mathcal{B}} = icc(m_B, m_B + o_{\mathcal{B}}, d(B))$ *are the ice-cream cones given by* $Cone_{\mathcal{A}} \cup S_{\mathcal{A}}$ *and* $Cone_{\mathcal{B}} \cup S_{\mathcal{B}}$ *with a translation of* m_B.

Proof. We rearrange our approximate footprint in this way:

$$\widetilde{Fp}(B) = \big(S^+_{\mathcal{A}} \cup Frust_{\mathcal{A}} \cup S_{\mathcal{O}}\big) \cup \big(S^+_{\mathcal{B}} \cup Frust_{\mathcal{B}} \cup S_{\mathcal{O}}\big) \cup \big(S^+_{\mathcal{A}} \cup Cyl^+ \cup S^+_{\mathcal{B}}\big) \cup Pyr^+.$$

We notice the following relations:

$$\begin{aligned} S^+_{\mathcal{A}} \cup Frust_{\mathcal{A}} \cup S_{\mathcal{O}} &= (Cone_{\mathcal{A}} \cup S_{\mathcal{A}}) \oplus \mathrm{Ball}(B^t) = \mathrm{icc}_{\mathcal{A}} \oplus \mathrm{Ball}(r(B)) \\ S^+_{\mathcal{B}} \cup Frust_{\mathcal{B}} \cup S_{\mathcal{O}} &= (Cone_{\mathcal{B}} \cup S_{\mathcal{B}}) \oplus \mathrm{Ball}(B^t) = \mathrm{icc}_{\mathcal{B}} \oplus \mathrm{Ball}(r(B)) \\ S^+_{\mathcal{A}} \cup Cyl^+ \cup S^+_{\mathcal{B}} &= \mathrm{seg}(o_{\mathcal{A}}, o_{\mathcal{B}}) \oplus \mathrm{Ball}\,(d(B) + r(B)) \end{aligned}$$

Then by Lemma 1, $\mathrm{Sep}(\widetilde{Fp}(B), f) > 0$ if and only if the four terms in the conjunction form described in the lemma are all true. **Q.E.D.**

C.4 Boundary Reduction Method for Disc-Edge

We can apply the boundary reduction method to produce a parametric separation query for discs. Note that discs are Π_1-sets.

In this section, we give an example of boundary reduction method. We consider a disc Π_1 set defined by $D = \{(x, y, z) \in \mathbb{R}^3 | x^2 + y^2 \leq 1, z = 0\}$, and an edge feature defined by $f = \{(a_0, b_0, c_0) + t(a, b, c) | t \in [0, 1]\}$. We compute the parametric collision detection "if $\mathrm{Sep}(D, f) > s$" for $s > 0$.

The collision detection is based on the following process:

```
Input: a disc D and a segment f, real number s > 0
Output: boolean (Sep(D, f) > s)
    If (Sep(D, ∂f) ≤ s), return false
    For each (p, q) ∈ cp(∂D, f̄),
        if (d(p, q) ≤ s and q ∈ f),
            return false
    For each w ∈ D̄ ∩ f̄ (this is unique)
        if (w ∈ D and w ∈ f)
            return false
    return true
```

Among the process above, we recursively call the query "$\mathrm{Sep}(D, \partial f) > s$?", where ∂f is a set of two points. This subquery is standard and thoroughly studied by many people. We focus on the computation of $cp(\partial D, \overline{f})$, which is a good example of solving the polynomial equations listed in (6).

To begin with, suppose $(\boldsymbol{p}, \boldsymbol{q}) \in cp(\partial D, \overline{f})$, where $\partial D = \{x^2 + y^2 = 1\} \cap \{z = 0\}$ and $\overline{f} = \{(a_0, b_0, c_0) + t(a, b, c) | t \in \mathbb{R}\}$. We denote the normal direction of

∂D to be $\boldsymbol{d} = (0, 0, 1)$ and the direction of $\overline{f}$ to be $\boldsymbol{f} = (a, b, c)$. Given any $\boldsymbol{q} = (x_q, y_q, z_q) \in \overline{f}$, let's compute $(\boldsymbol{p}, \boldsymbol{q}) \in cp(\partial D, \boldsymbol{q})$. Suppose that the solution is $\boldsymbol{p} = (x, y, z)$. Corresponding to symbols in (6), $f_1(x, y, z) = x^2 + y^2 - 1$ and $f_2(x, y, z) = z$. Let $\boldsymbol{v} = \nabla f_1(\boldsymbol{p}) \times \nabla f_2(\boldsymbol{p})$. Noticing that $\boldsymbol{p} \perp \boldsymbol{v}$ (since $\boldsymbol{p}$ is on a circle, which is equivalent to $f_1(\boldsymbol{p}) = 0$), and $(\boldsymbol{q} - \boldsymbol{p}) \perp \boldsymbol{v}$, $\boldsymbol{p}$ is the intersection between the plane constructed by $\boldsymbol{q}$ and $\boldsymbol{d}$ with ∂D. This gives us

$$\boldsymbol{p} = (x, y, z) = \left(\frac{x_q}{\sqrt{x_q^2 + y_q^2}}, \frac{y_q}{\sqrt{x_q^2 + y_q^2}}, 0 \right).$$

By applying $\langle \boldsymbol{q} - \boldsymbol{p}, \boldsymbol{f} \rangle = 0$, we have the equation:

$$ax_q \left(1 - \sqrt{x_q^2 + y_q^2}\right) + by_q \left(1 - \sqrt{x_q^2 + y_q^2}\right) + cz_q \sqrt{x_q^2 + y_q^2} = 0.$$

This is equivalent to

$$ax_q + by_q = (ax_q + by_q - cz_q)\sqrt{x_q^2 + y_q^2},$$

or

$$(ax_q + by_q)^2 = (ax_q + by_q - cz_q)^2 \left(x_q^2 + y_q^2\right).$$

Here, x_q, y_q, z_q are linear to the variable t. Then the equation above is a polynomial of t with degree 4. *Hence, solving the equation is the same with solving a quartic equation.* The root of the polynomial of t gives possible $\boldsymbol{q} \in \overline{f}$. Then if there is $\boldsymbol{q} \in f$ such that $d(\boldsymbol{p}, \boldsymbol{q}) \leq s$, the query returns false.

C.5 Parametric Query for Special Σ_2-sets

Theorem 5 (Theorem 4 in main paper).
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.*

Proof. Parametric separation queries for convex polyhedra is standard. Hence, it remains to consider parametric separation queries for other special Π_1-sets, viz., right cylinder, right cone and right frustum.

The main idea is based on the boundary reduction technique. The technique reduces the query to the following 3 subqueries $Q_0 > s$, $Q_A > s$ and $Q_f > s$. We note that $Q_0 > s$ is trivial: Given a special Π_1 set Π, its algebraic span is $\mathbb{R}^3$. Hence the closest pair $cp(\overline{\Pi}, f)$ for any feature f takes place at the whole feature. Then $cp(\Pi, f) \neq \emptyset$ if and only if $\Pi^\circ \cap f \neq \emptyset$. Since f is a closed set, the relation between Π and f can only be classified into three cases: $\partial\Pi \cap f \neq \emptyset$, $f \subset \Pi^\circ$ or $f \cap \Pi = \emptyset$. Checking the first case is solving the algebraic equations which is standard. To check the second case, we can pick any $\boldsymbol{q} \in f$ and check if $\boldsymbol{q} \in \Pi$ or not by checking each of algebraic inequalities that form the Π_1 set. So we suppose that we are in the third case, i.e. $f \cap \Pi = \emptyset$ where $cp(\Pi, f) = \emptyset$.

It remains to consider the subqueries $Q_A > s$ and $Q_f > s$. Checking if $Q_A > s$, i.e., $\text{Sep}(\Pi, \partial f) > s$ is solved recursively. Hence we only focus on deciding $Q_f > s$, i.e., $\text{Sep}(\partial\Pi, f) > s$. We observe that for our special Π_1-sets, $\partial\Pi$ can be decomposed into two or three surfaces of the form: (i) the intersection of a quadric surface and a slab (which is bounded by two parallel planes). (ii) a disc. The case of a disc has been addressed in Appendix C.4. The quadric surface has two possibilities: one is the non-planar surface of a right frustum, which is treated in Appendix C.7. This includes the traffic cone as a special case. The other possibility is the non-planar surface of a right cylinder, denoted by Cyl. Noticing that if $(\boldsymbol{p}, \boldsymbol{q}) \in cp(\text{Cyl}, f)$, then $\boldsymbol{q}$ is a projection of the axis of Cyl onto f. Finding the projection will give us a potential $\boldsymbol{q}$ and we can find a corresponding $\boldsymbol{p}$. This finishes our proof. **Q.E.D.**

C.6 Ice-cream Cone: Reducing the maximum degree from 4 to 2

We solve the parametric separation query for the ice-cream cone, namely $\text{Sep}(\text{icc}(v, o, r), f) > 0$ where f is a feature.

Input: an ice-cream cone $\text{icc}(v, o, r)$, a feature f and a real number $s > 0$.
Output: boolean $(\text{Sep}(\text{icc}(v, o, r), f) > s)$.
Method:
  If $\text{Sep}(v, f) \leq s$, return `false`
  If $\text{Sep}(\text{Ball}(o, r), f) \leq s$, return `false`
  Let $h := |\text{seg}(v, o)|$, $k := \sqrt{h^2 - r^2}$, a point $c := o + \frac{r^2}{h^2}(v - o)$
  If $\text{Sep}(\text{tc}(v, c, \frac{rk}{h}), f) \leq s$, return `false`
  return `true`

The check for $\text{Sep}(v, f) \leq s$ and $\text{Sep}(\text{Ball}(o, r), f) \leq s$ are in Lemma 3. For the problem $\text{Sep}(\text{tc}(v, c, \frac{rk}{h}), f) \leq s$, we solve it by a reduction from each feature to its boundary features, according to the Lemma 2. The case f is a corner is easy, so we focus on the other two cases.

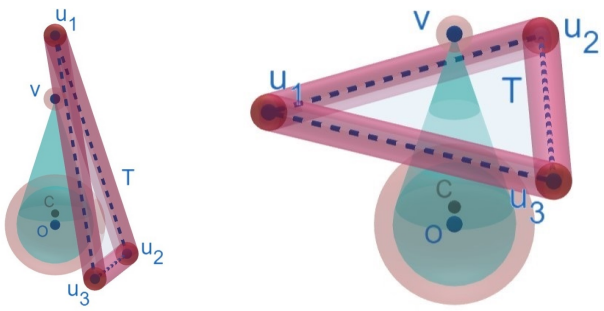
Case f is an edge (line segment):

Input: a traffic cone $\mathrm{tc}(v, c, r)$ and a line segment g, a real number $s > 0$.
Output: boolean $(\mathrm{Sep}(\mathrm{tc}(v, c, r), g) > s)$.
Method:
  If $\mathrm{Sep}(\mathrm{tc}(v, c, r), \partial g) \leq s$, return `false`
  If $cp(\overline{\mathrm{tc}(v, c, r)}, \overline{\ell}) = \emptyset$ (closest pair in Sec. 3.2), return `true`
  For each pair of $(u, w) \in cp(\overline{\mathrm{tc}(v, c, r)}, \overline{\ell})$,
    if $d(u, w) \leq s$
      if $u \in \mathrm{tc}(v, c, r)$ and $w \in g$,
        return `false`
  return `true`

Case f is a triangle:

Input: a traffic cone $\mathrm{tc}(v, c, r)$ and a triangle T, a real number $s > 0$.
Output: boolean $(\mathrm{Sep}(\mathrm{tc}(v, c, r), T) > s)$.
Method:
  If $\mathrm{Sep}(\mathrm{tc}(v, c, r), \partial T) \leq s$, return `true`
  return $(\mathrm{seg}(v, c) \cap T \neq \emptyset)$

The return $\mathrm{seg}(v, c) \cap T \neq \emptyset$ in the last case may not be so obvious, but we can see it from the Figure 11.



(a) (b)

Fig. 11: Separation between triangle and ice-cream cone compared to the separation between their boundaries:
(a) When the triangle does not intersect across the ice-cream cone, the boundary will always reach the minimum separation since the traffic cone is a developable surface.
(b) If the triangle intersects across the ice-cream cone, it will intersect with the axis of the cone.

C.7 Closest Pairs between Cone and Line

In this section, let $\mathrm{TC} = \mathrm{tc}((0,0,0),(0,0,-1),r)$, i.e., the algebraic representation for $\overline{\mathrm{TC}}$ is $x^2+y^2-r^2z^2=0$. We illustrate how to find $cp(\overline{\mathrm{TC}},\overline{\ell})$ for traffic cone TC and line segment ℓ. We assume that $\overline{\ell}$ is represented parametrically by $\boldsymbol{q}(t)=\boldsymbol{v}_0+\boldsymbol{v}t$, for $t\in\mathbb{R}$, where $\boldsymbol{v}=(a,b,c)$. Given a point $\boldsymbol{p}\in\overline{\mathrm{TC}}$, the normal vector at that point is $\boldsymbol{n_p}$.

Before applying (6) to find local minima, we first check if $\overline{\mathrm{TC}}\cap\overline{\ell}$ is non-empty. This is solving the two equations. If there are solutions, we regard them as additional closest pairs in $cp(\overline{\mathrm{TC}},\overline{\ell})$ in addition to the solutions of (6).

Now we apply (6) to our problem. For our problem, we only have $f_1(x,y,z)=x^2+y^2-r^2z^2$. Our formulation of the system (6) does not involve the polynomials g_1 or g_2, instead we have $\nabla g_1(\boldsymbol{q})\times\nabla g_2(\boldsymbol{q})$ given by the vector $\boldsymbol{v}$. We do not have $\nabla f_1(\boldsymbol{p})\times\nabla f_2(\boldsymbol{p})$, but we have the $\boldsymbol{n_p}$ which is equal to $\boldsymbol{p}-\boldsymbol{q}$ up to a constant multiple. The perpendicular conditions can be reduced to $\langle\boldsymbol{n_p},\boldsymbol{v}\rangle=0$. This yields two equations to be solved for $cp(\overline{\mathrm{TC}},\overline{\ell})$, where $(x,y,z)=\boldsymbol{p}$:

$$x^2+y^2=r^2z^2 \tag{12}$$

$$ax+by=r^2cz \tag{13}$$

Let $\kappa := rc$. Multiplying (12) by κ^2 and subtracting square of (13), we get a quadratic equation for x and y:

$$(\kappa^2-a^2)x^2+(\kappa^2-b^2)y^2-2abxy=0,$$

which is equivalently:

$$\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} \kappa^2-a^2 & -ab \\ -ab & \kappa^2-b^2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=0.$$

Let

$$A=\begin{pmatrix} \kappa^2-a^2 & -ab \\ -ab & \kappa^2-b^2 \end{pmatrix}.$$

Then

$$\det(A)=(\kappa^2-a^2)(\kappa^2-b^2)-a^2b^2=\kappa^2(\kappa^2-a^2-b^2).$$

When $\det(A)>0$, the equation has no real non-trivial ($x=y=0$ is trivial) solution. Hence $\mathrm{cp}(\overline{\mathrm{TC}},\overline{\ell})=\emptyset$. In this case, $\overline{\mathrm{TC}}\cap\overline{\ell}\neq\emptyset$, which we have excluded in previous discussions.

When $\det(A)\leq 0$, the equation has real non-trivial solutions $(\kappa^2-a^2)x+(ab\pm\sqrt{-\det(A)})y=0$, which are one or two planes. We denote the planes by P_1 and P_2.

To find the exact $\boldsymbol{p}$ and $\boldsymbol{q}$, one only need to notice that $\boldsymbol{q}_1=P_1\cap\overline{\ell}$ and $\boldsymbol{q}_2=P_2\cap\overline{\ell}$ ($P_i\cap\overline{\ell}\neq\emptyset$ since $\boldsymbol{p}\perp\overline{\ell}$ and P_i contains the direction of $\boldsymbol{p}$). Moreover, let $\mathrm{we}_i=P_i\cap\overline{\mathrm{TC}}$ which is a cross of two lines by the apex of $\overline{\mathrm{TC}}$, there are $\boldsymbol{p}_{i1},\boldsymbol{p}_{i2}\in\mathrm{we}_i$ such that $(\boldsymbol{p}_{ij}-\boldsymbol{q}_i)\perp\mathrm{we}_i$ for $i=1,2$ and $j=1,2$. This gives four pairs of $(\boldsymbol{p}_{ij},\boldsymbol{q}_i)$. Noticing that some of the pairs gives a local maximum for $d(\boldsymbol{p}_{ij},\boldsymbol{q}_i)$, we should get rid of those pairs, which may result in at most two pairs. The remaining possible pairs give $cp(\overline{\mathrm{TC}},\overline{\ell})$.

D Appendix: Subdivision for $\widehat{SE}(3)$

D.1 Box Adjacency Calculus

We first introduce a notation for discussing the j-th component of a box $B = \prod_{i=1}^n I_i$ where each I_i is an interval. For $j = 1, \ldots, n$, let $\text{Proj}_j(B) := \prod_{i=1, i\neq j}^n I_i$ denote the $n-1$ dimensional box obtained by omitting the j-th component. Then define the operator $\otimes_j$ using this identity: if $B = \prod_{i=1}^n I_i$, then

$$B = \underset{j}{\text{Proj}}(B) \otimes_j I_j$$

For $j \neq k$, we extend the notation to $\otimes_{j,k}$ using the identity

$$B = \underset{j,k}{\text{Proj}}(B) \otimes_{j,k} (I_j \times I_k)$$

If $I = [a, b]$ and $I' = [a', b']$ are intervals, we write $I \xrightarrow{+1} I'$ if $b = a'$, and $I \xrightarrow{-1} I'$ if $a = b'$. Let e_j be the elementary j-th vector in $\mathbb{R}^n$ (so e_j is a n-vector of all 0's except for a 1 in the j-th position). We call $d \in \{\pm e_1, \ldots, \pm e_n\}$ to be a **semi-direction**. For $n \geq 2$, we say that B is **adjacent to B' in the direction** $\pm e_j$, denoted $B \xrightarrow{\pm e_j} B'$, if $B = \text{Proj}_j(B) \otimes_j I_j$ and $B' = \text{Proj}_j(B') \otimes_j I'_j$ and $I_j \xrightarrow{\pm 1} I'_j$ and ($\text{Proj}_j(B) \subseteq \text{Proj}_j(B')$ or $\text{Proj}_j(B) \supseteq \text{Proj}_j(B')$).

The case of $\widehat{SO}(3)$: We extend the notion of adjacencies to the 3-dimensional boxes embedded in $\mathbb{R}^4$. The boundary of $[-1, 1]^4 \subseteq \mathbb{R}^4$ is subdivided into eight 3-dimensions boxes denoted $\pm C_i$ for $i = 0, \ldots, 3$, where[18]

$$C_i = [-1, 1]^3 \otimes_i \overline{1}$$

where $\overline{1}$ is an alternative symbol for -1. Thus $-\overline{1} = 1$. For instance, $C_2 = [-1, 1] \times [-1, 1] \times \overline{1} \times [-1, 1]$, and $-C_2 = [-1, 1] \times [-1, 1] \times 1 \times [-1, 1]$, However, $\widehat{SO}(3)$ is viewed as $\cup_{i=0}^3 C_i$ because of the equivalence of points $\boldsymbol{p}, \boldsymbol{q} \in \partial[-1, 1]^4$ where $\boldsymbol{p} \equiv \boldsymbol{q}$ iff $\boldsymbol{p} = -\boldsymbol{q}$. Therefore, $C_i \equiv -C_i$.

Suppose $B \subseteq C_i$ and $B' \subseteq C_j$ are boxes in $\widehat{SO}(3)$. We want to define the relation

$$B \xrightarrow{\pm e_k} B'.$$

This relation is not defined if $k = i$. Otherwise:

(Case 1) $i = j$: Then $B = \text{Proj}_i(B) \otimes_i \overline{1}$ and $B' = \text{Proj}_i(B') \otimes_i \overline{1}$. We define $B \xrightarrow{\pm e_k} B'$ if and only if "$\text{Proj}_i(B) \xrightarrow{\pm e_k} \text{Proj}_i(B')$." The precise definition requires us to shift the index k after projection in case $k > i$: let

$$k' = \begin{cases} k & \text{if } k < i, \\ k - 1 & \text{if } k > i. \end{cases}$$

So, $B \xrightarrow{\pm e_k} B'$ if and only if $\text{Proj}_i(B) \xrightarrow{\pm e_{k'}} \text{Proj}_i(B')$.

[18] We now index our components from $0, \ldots, 3$ instead of $1, \ldots, 4$. Moreover, $(0, 1, 2, 3)$ is also written (w, x, y, z), as in $C_0 = C_w$, etc.

(Case 2) $i \neq j$: Then $B = \text{Proj}_{i,j}(B) \otimes_{i,j} (\overline{1} \times I_j)$ and $B' = \text{Proj}_{i,j}(B') \otimes_{i,j} (I'_i \times \overline{1})$. We say the relation $B \xrightarrow{\pm e_k} B'$ is undefined if $k \neq j$. Otherwise, we have 2 possibilities:
(i) $B \xrightarrow{-e_j} B'$ if $I_j = [a_j, b_j] = [\overline{1}, b_j]$ and $\text{Proj}_{i,j}(B) \xrightarrow{-e_j} \text{Proj}_{i,j}(B')$.
(ii) $B \xrightarrow{+e_j} B'$ if $I_j = [a_j, b_j] = [a_j, 1]$ and $\text{Proj}_{i,j}(B) \xrightarrow{+e_j} -(\text{Proj}_{i,j}(B'))$.

D.2 Maintaining Principal Neighbor Pointers

Recall that in Section 4 we introduced the principal neighbor pointers for boxes in $\mathbb{R}^3$ and in $\widehat{SO}(3)$. If $B \in \square\mathbb{R}^3$, it has 6 principal neighbor pointers as in [1]. But if $B \subseteq C_i \subseteq \widehat{SO}(3)$, then it has 8 principal neighbor pointers, denoted $B.d$ where $d \in \{\pm e_0, \ldots, \pm e_3\}$. If $B = C_i$, all 8 pointers are non-null; otherwise two of them are null, namely $B.e_i = B.(-e_i) = \text{null}$.

We assume the **T/R Splitting scheme** in which we split a box $B = B^t \times B^r$ by either splitting B^t or splitting B^r. We need to update the principal neighbor pointers after such a split. Since the split of B^t is standard, we focus on B^r. Initially, $B^r = \widehat{SO}(3)$ and all its pointers are null. After the first split, we have four boxes, $C_0, \ldots, C_3$. Their pointers are initialized as follows: $C_i.(\pm e_j) = C_j$ for $j \neq i$, and $C_i.(\pm e_i) = \text{null}$. When $B \subseteq C_i$ is split, each of its eight children B_i $(i = 1, \ldots, 8)$ has its principal neighbor pointers set up as follows: Two of them are null, inherited from its parent. Three of them point to siblings (as in the standard octree split). Three of them point to non-siblings as follows: if $B_i.d$ is pointing to a non-sibling, then $B_i.d \leftarrow B.d$ in case $B.d$ is a leaf. Otherwise, $B_i.d$ will point to the child B' of $B.d$ such that $B_i \xrightarrow{d} B'$.

Now that we have set up the principal neighbor pointers of the children of B, we need to update the principal neighbor pointers of the other boxes: any box B' that used to point to B may need to be updated to point to a child of B (call this the "reverse pointer update"). First, we have to show how to get to such B's. Suppose $B.d = B'$ and $\mathit{depth}(B') < \mathit{depth}(B)$ then there is nothing to do in this direction. Otherwise, if $\mathit{depth}(B') = \mathit{depth}(B)$ and B' has children, we must go to all descendants of B' that pointed to B, and redirect their $-d$ pointers to point to the appropriate child of B. Note that there are always 6 non-null pointers, which can cross from C_i to some other C_j $(j \neq i)$. In contrast, for the translational boxes, there are boundary boxes that can have fewer than 6 non-null pointers.

We remark that Nowakiewicz [34] also discusses subdividing the translational and rotational boxes separately, and cubic model. However, the method does not classify boxes, and does not compute or use the adjacency information of boxes.

E Proof of Fundamental Theorem

E.1 Two Lemmas on Separation

Separation does not satisfy the triangular inequality because $\mathrm{Sep}(A,C) \leq \mathrm{Sep}(A,B) + \mathrm{Sep}(B,C)$ may be false. We have a kind of "triangular inequality" if one of the summands is replaced by the Hausdorff distance:

Lemma 7 (Pseudo Triangle Inequality). *Let (X,d) be a metric space. Then for any $A,B,C \subseteq X$,*

$$\mathrm{Sep}(A,C) \leq d_H(A,B) + \mathrm{Sep}(B,C).$$

Proof. Let $a \in A, b \in B, c \in C$:

$$\begin{aligned}
\mathrm{Sep}(A,C) &= \inf_{a,c} d_X(a,c) \\
&\leq \inf_b \left(\inf_{a,c} d_X(a,b) + d_X(b,c) \right) && \text{(triangular inequality)} \\
&\leq \inf_b \left(\mathrm{Sep}(A,b) + \mathrm{Sep}(b,C) \right) \\
&\leq \mathrm{Sep}(A,b^*) + \mathrm{Sep}(b^*,C) && \text{(choose } b^* \text{ so that } \mathrm{Sep}(b^*,C) = \mathrm{Sep}(B,C)) \\
&= \mathrm{Sep}(A,b^*) + \mathrm{Sep}(B,C) \\
&\leq d_H(A,B) + \mathrm{Sep}(B,C)
\end{aligned}$$

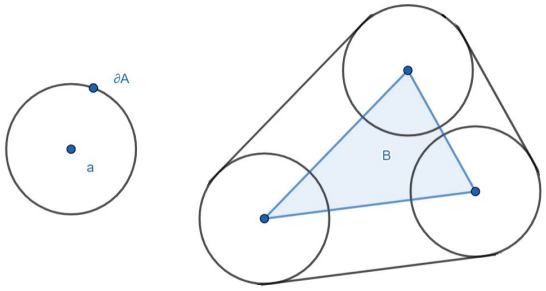
Q.E.D.

In particular, for $\gamma, \gamma' \in X$, the pseudo triangular inequality implies

$$|\, C\ell(\gamma) - C\ell(\gamma')| \leq d_H(\gamma, \gamma'). \tag{14}$$

We have another lemma on separation that relates to the Minkowski-sum.

Lemma 8 (Separation of Minkowski-Sum). *Suppose that closed sets $A,B,C \subseteq \mathbb{R}^m$. If $(A \oplus B) \cap C = \emptyset$, then for any $a \in A$, $\mathrm{Sep}(a \oplus B, C) \geq d_{\mathbb{R}^m}(a, \partial A)$.*



Proof. We denote the complement of a set S in $\mathbb{R}^m$ by $\complement S$. Then since $(A \oplus B) \cap C = \emptyset$, $C \subseteq \complement(A \oplus B)$. Since $a \oplus B \subseteq A \oplus B$, we have

$$\begin{aligned}
\mathrm{Sep}(a \oplus B, C) &\geq \mathrm{Sep}(a \oplus B, \complement(A \oplus B)) \\
&= \mathrm{Sep}(a \oplus B, \partial\complement(A \oplus B)) \\
&= \mathrm{Sep}(a \oplus B, \partial(A \oplus B)).
\end{aligned}$$

Then since

$$\begin{aligned}\mathrm{Sep}(a \oplus B, \partial(A \oplus B)) &= \inf_{(a'+b') \in \partial(A \oplus B), b \in B} d_{\mathbb{R}^m}(a+b, a'+b') \\ &\geq \inf_{a' \in \partial A, b \in B, b' \in \partial B} d_{\mathbb{R}^m}(a+b, a'+b') \\ &\geq \inf_{a' \in \partial A, b \in B} d_{\mathbb{R}^m}(a+b, a'+b) \\ &= \inf_{a' \in \partial A} d_{\mathbb{R}^m}(a, a') \\ &= d_{\mathbb{R}^m}(a, \partial A),\end{aligned}$$

we have

$$\mathrm{Sep}(a \oplus B, C) \geq d_{\mathbb{R}^m}(a, \partial A).$$

Q.E.D.

E.2 Proof of SSS framework

A subdivision is called a **uniform ε-subdivision**, if the widths of all the boxes in the subdivision are the same and it is less than ε.

Lemma 9. *Assume the Axioms* **(A0)**, **(A1)**, **(A2)** *and* **(A3)**. *If there exists a path π in W with clearance $K\varepsilon$ where $K = L_0 C_0 D_0 \sigma$, then the SSS framework can find a path $\overline{P}$.*

Proof. Suppose there exists a path π with clearance $K\varepsilon = L_0 C_0 D_0 \sigma\varepsilon$. We obtain a contradiction by assuming that the SSS framework could not find a path. In this case, the SSS framework will subdivide $\square W$ into minimum sizes. Hence, in this subdivision, all `MIXED` tiles have width less than ε. Without loss of generality, let us assume the subdivision is a uniform ε-subdivision. We consider all tiles in this subdivision that intersect with π. The union of these tiles forms a set $\mathrm{Cover}(\pi)$. It is easy to see that this is a path-connected and compact set. Hence, by Poincaré's duality, there is a sequence P of adjacent boxes in the $\mathrm{Cover}(\pi)$. We prove that these boxes are `FREE`, i.e., P is a channel. Each tile $B \in P$ is `FREE` under the soft predicate, that is, $\forall q \in B$, $C\ell(q) > 0$.

For each tile $B \in P$, we consider the tile σB. Since there is $t \in [0,1]$ such that $\pi(t) \in B \subseteq \sigma B$, for each $q \in \sigma B$, by axioms (**A2**) and (**A3**), we have

$$\begin{aligned}\| C\ell(q) - C\ell(\pi(t)) \| &\leq d_H(\mu(q), \mu(\pi(t))) && \text{(Pseudo Triangle Inequality)} \\ &< L_0 d_X(\mu(q), \mu(\pi(t))) && (\mathbf{A2}) \\ &< L_0 C_0 d_W(q, \pi(t)). && (\mathbf{A3})\end{aligned}$$

By axiom (**A1**) and since the diameter of tile σB is no greater than $l(\sigma B)$, we have

$$\begin{aligned}d_W(q, \pi(t)) &\leq l(\sigma B) && \text{(definition)} \\ &\leq D_0 w(\sigma B) && (\mathbf{A1}) \\ &< D_0 \sigma\varepsilon. && \text{(width is } \varepsilon)\end{aligned}$$

Hence $\| C\ell(q) - C\ell(\pi(t))\| < L_0 C_0 D_0 \sigma\varepsilon$ and then

$$C\ell(q) \geq C\ell(\pi(t)) - L_0 C_0 D_0 \sigma\varepsilon > 0.$$

This gives $\forall q \in \sigma B$, $C\ell(q) > 0$. So tile $B \in P$ is FREE under the soft predicate. **Q.E.D.**

Remark: It can be seen from the proof that the coefficient $L_0 C_0 D_0 \sigma$ is not related to the number of children each tile is split into. So the D_0 in this term can be taken as the aspect ratio of the tiles.

Lemma 10. *Assume the Axioms* **(A0)** *and* **(A4)**. *If the SSS framework returns a path $\overline{P}$, then there exists a path $\overline{P}'$ of essential clearance $\frac{1}{2}\varepsilon$.*

Proof. If $\overline{P}$ is returned, then it implies that there is a channel P' in the uniform ε-subdivision. Let $\overline{P}'$ be the concatenation of three polygonal paths,

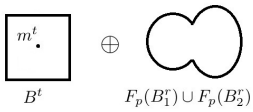
$$\overline{P}' = (S_\alpha; \overline{P}''; S_\beta)$$

where $\overline{P}''$ connects the midpoints of the adjacent boxes of P', and S_α is the segment connecting α to the midpoint of the initial box, and S_β is the segment connecting the midpoint of the last box to β. For the essential clearance of the path $\overline{P}'$, we only need to ensure that the clearance of $\overline{P}''$ is at least $\frac{1}{2}\varepsilon$.

Let $B_1 = B_1^t \times B_1^r$ and $B_2 = B_2^t \times B_2^r$ be the two adjacent tiles, their centers are $m_1 = m_1^t \times m_1^r$ and $m_2 = m_2^t \times m_2^r$ respectively. The line segment connecting m_1 and m_2 is $\overline{m_1 m_2}$. Since $\dim(B_1 \cap B_2) = k - 1$, we have either $B_1^t = B_2^t$ or $B_1^r = B_2^r$. We discuss these two cases respectively.

When $B_1^t = B_2^t = B^t$, and $m_1^t = m_2^t = m^t$, for any $q \in \overline{m_1 m_2}$, we have $q = m^t \times b^r$. Since, B_1 and B_2 are FREE,

$$\begin{aligned}
B_1 \text{ and } B_2 \text{ are FREE} &\Rightarrow Fp(B_1) \cap \Omega = \emptyset \text{ and } Fp(B_2) \cap \Omega = \emptyset \quad \text{(by } \mathbf{(A0)}\text{)} \\
&\Rightarrow (Fp(B_1) \cup Fp(B_2)) \cap \Omega = \emptyset \\
&\Rightarrow \big(\big(B^t \oplus Fp(B_1^r)\big) \cup \big(B^t \oplus Fp(B_2^r)\big)\big) \cap \Omega = \emptyset \quad \text{(by } \mathbf{(A4)}\text{)} \\
&\Rightarrow \big(B^t \oplus (Fp(B_1^r) \cup Fp(B_2^r))\big) \cap \Omega = \emptyset
\end{aligned}$$

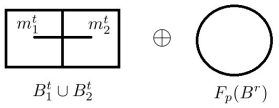


Then since $m^t \in B^t$, $Fp(\overline{m_1^r m_2^r}) \subseteq (Fp(B_1^r) \cup Fp(B_2^r))$, and so by separation of Minkowski-sum,

$$\begin{aligned}
C\ell(\overline{m_1 m_2}) &= \text{Sep}(m^t \oplus Fp(\overline{m_1^r m_2^r}), \Omega) \\
&\geq \text{Sep}(m^t \oplus (Fp(B_1^r) \cup Fp(B_2^r)), \Omega) \\
&\geq d_{\mathbb{R}^m}(m^t, \partial B^t) \\
&\geq \frac{1}{2}\varepsilon.
\end{aligned}$$

When $B_1^r = B_2^r = B^r$, and $m_1^r = m_2^r = m^r$, for any $q \in \overline{m_1 m_2}$, we have $q = q^t \times m^r$. Since, B_1 and B_2 are `FREE`,

$$\begin{aligned} B_1 \text{ and } B_2 \text{ are } \texttt{FREE} &\Rightarrow Fp(B_1) \cap \Omega = \emptyset \text{ and } Fp(B_2) \cap \Omega = \emptyset \qquad (\mathbf{A0}) \\ &\Rightarrow (Fp(B_1) \cup Fp(B_2)) \cap \Omega = \emptyset \\ &\Rightarrow \big(\big(B_1^t \oplus Fp(B^r)\big) \cup \big(B_2^t \oplus Fp(B^r)\big)\big) \cap \Omega = \emptyset \qquad (\mathbf{A4}) \\ &\Rightarrow \big(\big(B_1^t \cup B_2^t\big) \oplus Fp(B^r)\big) \cap \Omega = \emptyset. \end{aligned}$$



Then since for any $q \in \overline{m_1 m_2}$, we have $q \in B_1^t \cup B_2^t$, and so by Lemma on Separation of Minkowski-sum,

$$\begin{aligned} C\ell(\overline{m_1 m_2}) &= \inf_{q \in \overline{m_1 m_2}} \mathrm{Sep}(q^t \oplus Fp(B^r), \Omega) \\ &\geq \inf_{q \in \overline{m_1 m_2}} d_{\mathbb{R}^m}(q^t, \partial(B_1^t \cup B_2^t)) \\ &= \mathrm{Sep}(\overline{m_1^t m_2^t}, \partial(B_1^t \cup B_2^t)) \\ &\geq \frac{1}{2}\varepsilon. \end{aligned}$$

Hence, for any consecutive centers m_1 and m_2, the clearance of direct connection is always no less than $\frac{1}{2}\varepsilon$. Therefore, the path $\overline{P}''$ has clearance $\frac{1}{2}\varepsilon$. **Q.E.D.**

REMARK: our construction of $\overline{P}'$ uses a rectilinear path $\overline{P}''$, while our SSS algorithm uses a non-rectilinear path. This non-rectilinear path only has a clearance of $\frac{1}{2\sqrt{n}}\varepsilon$. Note that Axiom **(A4)** is extremely strong and ensures a clearance independent of constants such as L_0, D_0, σ.

The definition of resolution exactness has two requirements, (Path) and (NoPath). Lemma 9 satisfies the requirement of (Path), and Lemma 10 satisfies the requirement of (NoPath). This implies the resolution-exactness of our SSS planner:

Theorem 6. *Assuming the Axioms* **(A0)**, **(A1)**, **(A2)**, **(A3)** *and* **(A4)**, *the SSS algorithm is resolution exact with constant* $K = \max\{L_0 C_0 D_0 \sigma, 2\}$.

E.3 Analysis of the SSS Algorithm for the Delta robot

We now analyze the basic properties of our SSS path planner for the Delta robot.

Theorem 7. *The Lipschitz constant in Axiom* **(A2)** *is* $L_0 = 1$.

Proof. Given any two configurations, we could compare their Hausdorff distance by moving the relative center of a triangle to another without changing its direction in $r \in SO(3)$, i.e. for each point $T \in \triangle\mathcal{AOB}$, let $t = |T\mathcal{O}|$, and since our $\lambda = 1 = \sup t$, we have the inequalities:

$$\begin{aligned} d_H((\boldsymbol{x}, r), (\boldsymbol{x}', r')) &\leq d_H((\boldsymbol{x}, r), (\boldsymbol{x}, r')) + d_H((\boldsymbol{x}, r'), (\boldsymbol{x}', r')) \\ &< t\theta + d_{\mathbb{R}^3}(\boldsymbol{x}, \boldsymbol{x}') \\ &= t d_{SO(3)}(r, r') + d_{\mathbb{R}^3}(\boldsymbol{x}, \boldsymbol{x}') \\ &\leq \lambda d_{SO(3)}(r, r') + d_{\mathbb{R}^3}(\boldsymbol{x}, \boldsymbol{x}') \\ &= d_X((\boldsymbol{x}, r), (\boldsymbol{x}', r')). \end{aligned}$$

Hence, the minimum Lipschitz constant $L_0 \leq 1$. In fact, we could take $L_0 = 1$, since we have taken the physical length of the edge $\mathcal{AO}$ as 1 unit in $\mathbb{R}^3$. **Q.E.D.**

Theorem 8 (Theorem 2 in main paper). *The approximate footprint of the Delta robot is σ-effective with $\sigma = (2 + \sqrt{3}) < 3.8$.*

Proof. Notice that the radius of $\widetilde{Fp}_{\mathcal{A}}(B^r)$, which is the approximate footprint of $\mathcal{A}$ for a rotation box is less than θ (the rotation angle), which is restricted by $\sqrt{4}w(B)/2 = w(B)$ (box dimension is 4, atlas constant is 2), so $\widetilde{Fp}(B^r) \subseteq Ball(0, w(B)) \oplus Fp(B^r)$. Hence, we have

$$\begin{aligned} \widetilde{Fp}(B^t \times B^r) &= Ball(B^t) \oplus \widetilde{Fp}(B^r) \\ &\subseteq Ball(m(B^t), \sqrt{3}w(B^t)/2) \oplus Ball(0, w(B)) \oplus Fp(B^r) \\ &= Ball(m(B^t), (2 + \sqrt{3})w(B^t)/2) \oplus Fp(B^r) \\ &\subseteq Fp((2 + \sqrt{3})B^t) \oplus Fp(B^r) \\ &\subseteq Fp((2 + \sqrt{3})B). \end{aligned}$$

So this soft predicate is $(2 + \sqrt{3})$-effective. **Q.E.D.**

Theorem 9 (Theorem 3 in main paper). *Our SSS planner for the Delta Robot is resolution exact with constant $K = 4\sqrt{6} + 6\sqrt{2} < 18.3$.*

Proof. The SSS framework for the Delta robot is resolution exact with constant $K = 4\sqrt{6} + 6\sqrt{2}$ since $L_0 = 1, C_0 = 2, D_0 = \sqrt{6}, \sigma = 2 + \sqrt{3}$. **Q.E.D.**