Soft Subdivision Motion Planning for Complex Planar Robots[☆]

Bo Zhou[a], Yi-Jen Chiang[a,*], Chee Yap[b]

[a]*Department of Computer Science and Engineering, New York University, Brooklyn, NY, USA.*
[b]*Department of Computer Science, New York University, New York, NY, USA.*

Abstract

The design and implementation of theoretically-sound robot motion planning algorithms is challenging. Within the framework of *resolution-exact algorithms*, it is possible to exploit *soft predicates* for collision detection. The design of soft predicates is a balancing act between their implementability and their accuracy/effectivity.

In this paper, we focus on the class of planar polygonal rigid robots with arbitrarily complex geometry. We exploit the remarkable *decomposability* property of soft collision-detection predicates of such robots. We introduce a general technique to produce such a decomposition. If the robot is an m-gon, the complexity of this approach scales linearly in m. This contrasts with the $O(m^3)$ complexity known for exact planners. It follows that we can now routinely produce soft predicates for any rigid polygonal robot. This results in resolution-exact planners for such robots within the general *Soft Subdivision Search* (SSS) framework. This is a significant advancement in the theory of sound and complete planners for planar robots.

We implemented such decomposed predicates in our open-source `Core Library`. The experiments show that our algorithms are effective, perform in real time on non-trivial environments, and can outperform many sampling-based methods.

Keywords: Computational Geometry; Algorithmic Motion Planning; Resolution-Exact Algorithms; Soft Predicates; Planar Robots with Complex Geometry.

1. Introduction

Motion planning is widely studied in robotics [10, 11, 5]. Many planners are heuristic, i.e., without a priori guarantees of their performance (see below for what we mean by guarantees). In this paper, we are interested in non-heuristic algorithms for the **basic planning problem**: this basic problem considers only kinematics and the existence of paths. The robot R_0 is fixed, and the input is a triple (α, β, Ω) where α, β are the start and goal configurations of R_0, and $\Omega \subseteq \mathbb{R}^d$ is a polyhedral environment in $d = 2$ or 3. The algorithm outputs an Ω-avoiding path from α to β if one

[☆]The conference version of this paper [21] appeared in *Proc. 26th European Symposium on Algorithms (ESA 2018)*, pages 73:1-73:14, 2018. Helsinki, Finland, Aug. 20-24, 2018. This work is supported in part by National Science Foundation (NSF) Grants CCF-1564132 and CCF-2008768.

[*]Corresponding author.
Email addresses: `bz387@nyu.edu` (Bo Zhou), `chiang@nyu.edu` (Yi-Jen Chiang), `yap@cs.nyu.edu` (Chee Yap)

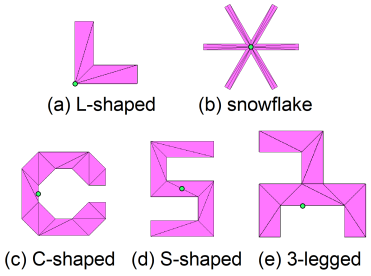


Figure 1: Some rigid planar robots ((a)-(b): star-shaped; (c)-(e): general shaped).

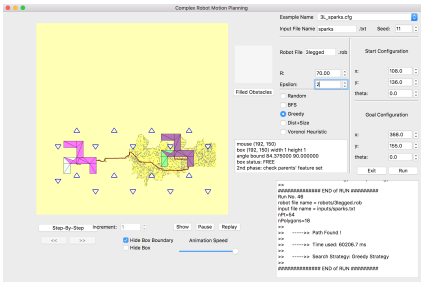


Figure 2: GUI interface for planner for a 3-legged robot.

exists, and `NO-PATH` otherwise. See Figure 1 for some rigid robots, and also Figure 2 for our GUI interface for path planning.

The basic planning problem ignores issues such as the optimality of paths, robot dynamics, planning in the time dimension, non-holonomic constraints, and other considerations of a real scenario. Despite such an idealization, the solution to this basic planning problem is often useful as the basis for finding solutions that do take into account the omitted considerations. E.g., given a kinematic path, we can plan a smooth trajectory with a trace that is homotopic to the given path and the ε-closeness.

The algorithms for this basic problem are called "planners." In theory, it is possible to design *exact* planners because the basic path planning is a semi-algebraic (non-transcendental) problem. Even when such algorithms are available, exact planners have relatively high complexity and are non-adaptive, even in the plane (see [13]). So we tend to see inexact implementations of exact algorithms, with unclear guarantees. When fully explicit algorithms are known, exact implementation of exact planners is possible using suitable software tools such as the `CGAL` library [7]. But such algorithms are rare.

In current robotics [11, 5], those algorithms that are considered practical and have some guarantees may be classified as either resolution-based or sampling-based. The guarantees for the former is the notion of **resolution completeness** and for the latter, **sampling completeness**. Roughly speaking, *if there exists a path* then:
– resolution completeness says that a path will be found if the resolution is fine enough;
– sampling completeness says that a path will be found with high probability if "enough" random samples are taken.
But notice that if there is no path, these criteria are silent; indeed, such algorithms would not halt except by artificial cut-offs. Thus a major effort in the last 20 years of sampling research has been devoted to the so-called "Narrow Passage" problem. It is possible to view this problem as a manifestation of the **Halting Problem** for the sampling approaches: how can the algorithm halt when there is no path? (A possible approach to address this problem might be to combine sampling with exact computation, as in [14].)

Motivated by such issues, as well as trying to avoid the need for exact computation, we in [16, 17] introduced the following replacement for resolution complete planners: a **resolution-exact planner** takes an extra input parameter $\epsilon > 0$ in addition to (α, β, Ω), and it always halts and outputs either

an Ω-avoiding path from α to β or `NO-PATH`. The output satisfies this condition: there is a constant $K > 1$ depending on the planner, but independent of the inputs, such that:

(PATH) If there is a path of clearance $K\epsilon$, it will output a path;

(NOPATH) if there is no path of clearance ϵ/K, it will output `NO-PATH`.

Notice that if the optimal clearance[1] lies between $K\epsilon$ and ϵ/K, then the algorithm may output either a path or `NO-PATH`. So there is output indeterminacy. Note that the traditional way of using ϵ is to fix $K = 1$, killing off indeterminacy. Unfortunately, this also leads us right back to exact computation which we had wanted to avoid. We believe that indeterminacy is a small price to pay in exchange for avoiding exact computation [16]. The practical efficiency of resolution-exact algorithms is demonstrated by implementations of planar robots with 2, 3 and 4 degrees of freedom (DOF) [16, 12, 19], and also 5-DOF spatial robots [9]. All these robots perform in real-time in non-trivial environments. In view of the much stronger guarantees of performance, resolution-exact algorithms might reasonably be expected to have a lower efficiency compared to sampling algorithms. Surprisingly, no such trade-offs were observed: resolution-exact algorithms consistently outperform sampling algorithms. Our 2-link robot [12, 19] was further generalized to have thickness (a feat that exact methods cannot easily duplicate), and can satisfy a non-self-crossing constraint, all without any appreciable slowdown. Finally, these planners are more general than the basic problem: they all work for parameterized families $R_0(t_1, t_2 \ldots)$ of robots, where t_i's are robot parameters. All these suggest the great promise of our approach.

What is New in This Paper. In theoretical path planning, the algorithms often considered simple robots like discs or line segments. In this paper, we consider robots with complex shapes, which are more realistic models for real-world robots. We call them "complex robots" (where the complexity comes from the robot geometry rather than from the degrees of freedom). We focus on planar robots that are rigid and connected. Such a robot can be represented by a compact connected polygonal set $R_0 \subseteq \mathbb{R}^2$ whose boundary is an m-sided polygon, i.e., an m-gon. Informally, we call R_0 a "complex robot" if it is a non-convex m-gon for "moderately large" values of m, say $m \geq 5$. By this criterion, all the robots in Figure 1 are "complex." According to [20], no exact algorithms for $m > 3$ have been implemented; in this paper, we have robots with $m = 18$. To see why complex robots may be challenging, recall that the free space of such robots may have complexity $O((mn)^3)$ (see [1]) when the robot and environment have complexity m and n, respectively. Even with m fixed, this can render the algorithm impractical. For instance, if $m = 10$, the algorithm may slow down by 3 orders of magnitude. But our subdivision approach does not have to compute the entire free space before planning a path; hence the worst-case cubic complexity of the free space is not necessarily an issue.

More importantly, we show that the complexity of our new method grows only linearly with m. To achieve this, we exploit a remarkable property of soft predicates called "decomposability." We show how an arbitrary complex robot can be decomposed (via triangulation that may introduce new vertices) into an ensemble of "nice triangles" for which soft predicates are easy to implement. As we see below, there is a significant difference between a single triangle and an ensemble of triangles. *In consequence of our new techniques, we can now routinely construct resolution-exact planners for any reasonably complex robot provided by a user.* This could lead to a flowering of experimentation algorithmics in this subfield.

[1] See [16, p. 591, just before the last paragraph] for definition of clearance and similar terminology.

Technically, it is important to note that the previous soft predicate construction for a triangle robot in [16, 18] requires that the rotation center, i.e., the origin of the (rotational) coordinate system, be chosen to be the circumcenter of the triangle. But for our new soft predicates the triangles in the triangulation of the complex robot cannot be treated in the same way. This is because all the triangles of the triangulation must share a **common origin**, to serve as the rotation center of the robot. To ensure easy-to-compute predicates, we introduce the notion of a "nice triangulation" relative to a chosen origin: all triangles must be "nice" relative to this origin. These ideas apply for arbitrary complex robots, but we also exploit the special case of star-shaped robots to achieve stronger results.

Figure 2 shows our experimental setup for complex robots. A demo showing the real-time performance of our algorithms is found in the video clip available through this web link: `https://cs.nyu.edu/exact/gallery/complex/complex-robot-demo.mp4`.

Remark. Although it is not our immediate concern to address noisy environments and uncertainties, it is clear that our work can be leveraged to address these issues. E.g., users can choose $\epsilon > 0$ to be correlated with the uncertainty in the environment and the precision of the robot sensors. By using weighted Voronoi diagrams [4], we can achieve practical planners that have obstacle-dependent clearances (larger clearance for "dangerous" obstacles).

Previous Related Work. An early work is Zhu-Latombe [22] who also classify boxes into `FREE` or `MIXED` or `STUCK` (using our terminology below). They introduced the concept of **M-channels** (comprised of `FREE` or `MIXED` leaf boxes), as a heuristic basis to find an F-channel comprising only of `FREE` boxes. Subsequent researchers (Barbehenn-Hutchinson [2] and Zhang-Manocha-Kim [20]) continued this approach. Researchers in resolution-based approaches were interested in detecting the non-existence of paths, but their solutions remain partial because they do not guarantee to always detect non-existence of paths (of sufficient clearances) [3, 20]. The challenge of complex robots was taken up by Manocha's group who implemented a series of such examples [20]: a "five-gear" robot, a "2-D puzzle" robot a certain "star" robot with 4 DOFs, and a "serial link" robot with 4 DOFs. Except for the "star," the rest are planar robots.

Overview of the Paper. Section 2 reviews the fundamentals of our soft subdivision approach. Sections 3 and 4 describe our new techniques for star-shaped robots and for general complex robots, respectively. We present the experimental results in Section 5, and conclude in Section 6. The conference version of this paper appeared in [21].

2. Review: Fundamentals of Soft Subdivision Approach

Our soft subdivision approach includes the following three fundamental concepts (see [16] and the Appendix of [12] for the details):

- Resolution-exactness. This is an alternative replacement for the standard concept of "resolution completeness" in the subdivision literature. Briefly, a planner is **resolution-exact** if there is a constant $K > 1$ such that if there is a path of clearance $K\epsilon$, it will return a path, and if there is no path of clearance ϵ/K, it will return `NO-PATH`. Here, $\epsilon > 0$ is an additional input to the planner, in addition to the normal parameters.

- Soft Predicates. Let $\square\mathbb{R}^d$ be the set of closed axes-aligned boxes in $\mathbb{R}^d$. We are interested in predicates that classify boxes. Let $C : \mathbb{R}^d \to \{+1, 0, -1\}$ be an (exact) predicate where $+1, -1$ are called definite values, and 0 the indefinite value. For motion planning, we may

also identify $+1/-1/0$ with FREE/STUCK/MIXED, respectively. In our application, if p is a free configuration, then $C(p) = \texttt{FREE}$; if p is on the boundary of the free space, $C(p) = \texttt{MIXED}$; otherwise $C(p) = \texttt{STUCK}$. We extend C to boxes $B \in \square\mathbb{R}^d$ as follows: for a definite value $v \in \{+1, -1\}$, $C(B) := v$ if $C(x) = v$ for every $x \in B$. Otherwise, $C(B) := 0$. Call $\widetilde{C} : \square\mathbb{R}^d \to \{+1, 0, -1\}$ a "soft version" of C if whenever $\widetilde{C}(B)$ is a definite value, $\widetilde{C}(B) = C(B)$, and moreover, if for any sequence of boxes B_i $(i \geq 1)$ that converges monotonically to a point p, $\widetilde{C}(B_i) = C(p)$ for i large enough.

- Soft Subdivision Search (SSS) Framework. This is a general framework for a broad class of motion planning algorithms. One must supply a small number of subroutines with fairly general properties in order to derive a specific algorithm. For SSS, we need a predicate to classify boxes in the configuration space as FREE/STUCK/MIXED, a method to split boxes, a method to test if two FREE boxes are connected by a path of FREE boxes, and a method to pick MIXED boxes for splitting. The power of such frameworks is that we can explore a great variety of techniques and strategies. Indeed we introduced the SSS framework to emulate such properties found in the sampling framework. It is easy to put these elements together into an "SSS FindPath" algorithm as in [16, p. 592] or in the ArXiv version [8, Appendix A.2] of [9].

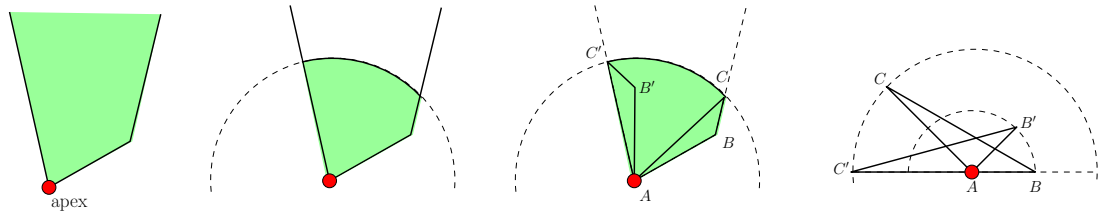
Feature-Based Approach. Following our previous work [16, 12], our computation and predicates are "feature based" whereby the evaluations of box primitives are based on a set $\widetilde{\phi}(B)$ of features associated with the box B. Given a polygonal set $\Omega \subseteq \mathbb{R}^2$ of obstacles, the boundary $\partial\Omega$ may be subdivided into a unique set of **corners** (points) and **edges** (open line segments), called the **features** of Ω. Let $\Phi(\Omega)$ denote this feature set. Our representation of $f \in \Phi(\Omega)$ ensures this **local property of** f: *for any point q, if f is the closest feature to q, then we can decide if q is inside Ω or not.* To see this, first note that if f is a corner, then q is outside Ω (since the closest feature f is a corner, f must be a convex corner of Ω). But if f is an edge, our representation assigns an orientation to f such that q is inside Ω iff q lies to the left of the oriented line through f.

3. Star-Shaped Robots

We first consider star-shaped robots. A star-shaped region R is one for which there exists a point $A \in R$ such that any line through A intersects R in a single line segment. We call A a **center** of R. Note that A is not unique. When a robot R_0 is a star-shaped polygon, we decompose R_0 into a set of triangles that share a **common vertex** at a center A. The rotations of the robot R_0 about the point A can then be reduced to the rotations of "nice" triangles about A. The soft predicates of nice triangles will be easy to implement because their footprints have special representations.

3.1. Nice Shapes for Rotation

From now on, by a **triangular set** we mean a subset $T \subseteq \mathbb{R}^2$ which is written as the non-redundant intersection of three closed half-spaces: $T = H_1 \cap H_2 \cap H_3$. Non-redundant means that we cannot express T as the intersection of only two half-spaces. Note that if T is bounded, this is our familiar notion of a triangle with 3 vertices. But T might be unbounded and have only 2 vertices as in Figure 3(a). If T is a triangular set, we may arbitrarily call one of its vertices the **apex** and call the resulting T a **pointed triangular set**. By a **truncated triangular set (TTS)**, we



(a) triangular set (unbounded case) (b) truncated triangular set (c) swept area by a nice triangle (d) sweeping $[A, B, C]$ to $[A, B', C']$

Figure 3: Truncated triangular set and swept areas.

mean the intersection of a pointed triangular set T with any disc centered at its apex A, as shown in Figure 3(b).

Notation for Angular Range: It is usual to identify S^1 (unit circle) with the interval $[0, 2\pi]$ where 0 and 2π are identified. Let $\alpha \neq \beta \in S^1$. Then $[\alpha, \beta]$ denote the range of angles from α counter-clockwise to β. Thus $[\alpha, \beta]$ and $[\beta, \alpha]$ are complementary ranges in S^1. If $\Theta = [\alpha, \beta]$, then its **width**, $|\Theta|$ is defined as $\beta - \alpha$ if $\beta > \alpha$, and $2\pi + \beta - \alpha$ otherwise. Moreover, we will write "$\alpha < \theta < \beta$" to mean that $\theta \in [\alpha, \beta]$.

Fix an arbitrary bounded triangular set T_0, represented by its three vertices A, B, C where A is the apex. For $\theta \in S^1$, let $T_0[\theta]$ denote the footprint of T_0 after rotating T_0 counter-clockwise (CCW) by θ about the apex. If $\Theta \subseteq S^1$, we write $T_0[\Theta] = \bigcup \{T_0[\theta] : \theta \in \Theta\}$. The sets $T_0[\theta]$ and $T_0[\Theta]$ are called **footprints** of T_0 at θ and Θ, respectively. If $\Theta = [\alpha, \beta]$, write $T_0[\alpha, \beta]$ for $T_0[\Theta]$, and call $T_0[\alpha, \beta]$ the **swept area** as T_0 rotates from α to β.

One of our concerns is to ensure that the swept area $T_0[\Theta]$ is "nice." Consider an example where $[A, B, C]$ is a triangular set with apex A (see Figure 3(c)). Consider the area swept by rotating $[A, B, C]$ in a CCW direction about its apex to position $[A, B', C']$. This sweeps out the truncated triangular set shown in Figure 3(b). This truncated triangular set (TTS) is desirable since it can be easily specified by the intersection of three half-spaces and a disc. On the other hand, if $[A, B, C]$ is the triangular set in Figure 3(d), then no rotation of $[A, B, C]$ would sweep out a truncated triangular set. So the triangular set in Figure 3(d) is "not nice," unlike the triangular set in Figure 3(c).

In general, let $T = [A, B, C]$ be a bounded triangular set. Let a, b, c denote the corresponding angles at A, B, C. We say T is **nice** if either b or c is at least $\pi/2$ ($= 90°$). We call the corresponding vertex (B or C) a **nice vertex**. Assuming T is non-degenerate and nice, there is a unique nice vertex. In the following, we assume (w.l.o.g.) that B is the nice vertex. The reason for defining niceness is the following.

Lemma 1. Let T be a pointed triangular set. Then T is nice iff for all $\alpha \in S^1$ ($0 < \alpha < \pi - a$), the footprints $T[0, \alpha]$ and $T[-\alpha, 0]$ are truncated triangular sets (TTS).

Proof. If T is nice, $T[0, \alpha]$ and $T[-\alpha, 0]$ are truncated triangular sets (TTS); this is easily seen in Figure 3(c).

Conversely, if T is not nice, let us assume that $\|A - B\| \leq \|A - C\|$ (e.g., Figure 3(d)). We claim that for sufficiently small $\alpha > 0$, either $T[0, \alpha]$ or $T[-\alpha, 0]$ is not a TTS. Assume (w.l.o.g.) that A, B, C are in CCW order; we show that $T[0, \alpha]$ is not a TTS.

If T is not nice, then $b < 90°$. Let $B - C$ intersect the $CircleB$ (the circle centered at A that passes through B) at D. Let $\alpha_{max} = \angle BAD = 180° - 2b = 2(90° - b)$, since $b = \angle ABD = \angle ADB$. Note that a TTS is a **convex** set as it is the intersection of three half-spaces and one disc; all of them are convex and thus the intersection is also convex. However, for any $\alpha < \alpha_{max}$, $T[0, \alpha]$ is not a TTS since $B - C$ will intersect $B' - C'$ inside $CircleB$ (see Figure 4) — this makes $T[0, \alpha]$ **non-convex** and thus it is not a TTS. □

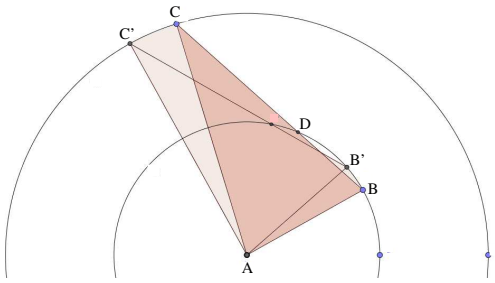


Figure 4: Proof of Lemma 1: $T[0, \alpha]$ is not a truncated triangular set (TTS).

Lemma 2. Let R_0 be a star-shaped polygonal region with A as center. If the boundary of R_0 is an n-gon, then we can decompose R_0 into an essentially disjoint[2] union of at most $2n$ bounded triangular sets (i.e., at most $2n$ triangles) that are **nice** and have A as the apex.

Proof. First, for each vertex v of R_0 we add a segment connecting A and v. This decomposes R_0 into a disjoint union of n triangles (since R_0 is star-shaped). Now consider each of the resulting triangle $T = [A, B, C]$ and let A be the apex of T. If T is not nice, then both angles b and c (corresponding to vertices B and C) are less than $90°$, and we can add a segment $[A, D]$ that is perpendicular to edge $[B, C]$ and intersects $[B, C]$ at D where D is in the interior of $[B, C]$. This effectively decomposes T into two *nice* triangles $[A, D, B]$ and $[A, D, C]$ with A being the common apex. In this way, we can decompose R_0 into at most $2n$ nice triangles that have A as the apex. □

3.2. Complex Predicates and T/R Subdivision Scheme

For complex robots in general (not necessarily star-shaped), we can exploit the remarkable **decomposability** property of soft predicates. More specifically, suppose $R_0 = \cup_{j=1}^{m} T_j$ where each T_j is a triangle or other shapes and not necessarily pairwise disjoint. If we have soft predicates $\widetilde{C}_j(B)$ for each T_j (where B is a box), then we immediately obtain a soft predicate for R_0 defined as follows:

$$\widetilde{C}(B) = \begin{cases} \texttt{FREE} & \text{if each } \widetilde{C}_j(B) \text{ is } \texttt{FREE} \\ \texttt{STUCK} & \text{if some } \widetilde{C}_j(B) \text{ is } \texttt{STUCK} \\ \texttt{MIXED} & \text{otherwise.} \end{cases} \tag{1}$$

[2] A set $\{A_1, \ldots, A_k\}$ where each $A_i \subseteq \mathbb{R}^2$ is said to be **essentially disjoint** if the interiors of the A_i's are pairwise disjoint.

Let $\sigma > 1$ and $\widetilde{C}$ be the soft version of an exact predicate C. Recall [16, 18] that $\widetilde{C}$ is **σ-effective** if for all boxes B, if $C(B) = \texttt{FREE}$ then $\widetilde{C}(B/\sigma) = \texttt{FREE}$.

Proposition A.
(1) $\widetilde{C}$ is a soft version of the exact classification predicate for R_0.
(2) Moreover, if each $\widetilde{C}_j$ is σ-effective, then $\widetilde{C}$ is σ-effective.

We need σ-effectivity in soft predicates in order to ensure resolution-exactness; see [16, 18] where this proposition was proved. Intuitively, σ-effectivity ensures that we can satisfy the **(PATH)** requirement of resolution-exactness as defined in the introduction. Indeed, σ is proportional to the constant K implicit in the definition of resolution-exactness (see [18]). There are two important remarks. First, this proposition is **false** if the $\widetilde{C}_j$ and $\widetilde{C}$ were exact predicates. More precisely, suppose C is the exact predicate for R_0 and C_j is the exact predicate for each T_j. It is true that if $C(B) = \texttt{FREE}$ then $C_j(B) = \texttt{FREE}$ for all j. But if $C(B) = \texttt{STUCK}$, it does not follow that $C_j(B) = \texttt{STUCK}$ for some j. Second, the predicates $\widetilde{C}_j(B)$ for all the T_j's must be based on a **common coordinate system**. As mentioned in Sec. 1, the soft predicate construction for a triangle robot in [16] does not work here. A technical contribution of this paper is the design of soft predicates $\widetilde{C}_j(B)$ for all the T_j's that are based on a common coordinate system. In the case of star-shaped robots, we apply Lemma 2 and use the apex A as the origin of this common coordinate system. Let r_j be the length of the longer edge out of A in T_j. We define r_0 as $r_0 = \max_j r_j$ (i.e., r_0 is the radius of the circumcircle of R_0 centered at A).

T/R Splitting. The simplest splitting strategy is to split a box $B \subseteq \mathbb{R}^d$ into 2^d congruent subboxes. In the worst case, to reduce all boxes to size $< \epsilon$ requires time $\Omega(\log(1/\epsilon)^d)$; this complexity would not be practical for $d > 3$. In [12, 19] we introduced an effective solution called *T/R splitting* which can be adapted to configuration space[3] $SE(2)$ in the current paper. Write a box $B \subseteq SE(2)$ as a pair (B^t, B^r) where $B^t \subseteq \mathbb{R}^2$ is the translational box and $B^r \subseteq S^1$ an angular range Θ. We say box $B = (B^t, B^r)$ is **ε-small** if B^t and B^r are both ε-small; the former means the width of B^t is $\leq \varepsilon$; the latter means the angle (in radians) satisfies $|B^r| \leq \varepsilon/r_0$. Our splitting strategy is to only split B^t (leaving $B^r = S^1$) as long as B^t is not ε-small. This is called a **T-split**, and produces 4 children. Once B^t is ε-small, we do binary splits of B^r (called **R-split**) until B^r is ε-small. We discard B when it is ε-small. The following lemma (and proof) in [16] can be carried over here:

Lemma 3. *([16])* Assume $0 < \varepsilon \leq \pi/2$. If $B = (B^t, B^r)$ is ε-small and B^t is a square, then the Hausdorff distance between the footprints of R_0 at any two configurations in B is at most $(1+\sqrt{2})\varepsilon$.

Soft Predicates. Suppose we want to compute a soft predicate $\widetilde{C}(B)$ to classify boxes B. Following the previous work [16, 12], we reduce this to computing a feature set $\widetilde{\phi}(B) \subseteq \Phi(\Omega)$. The **feature set** $\widetilde{\phi}(B)$ of B is defined as comprising those features f such that

$$\mathrm{Sep}(m_B, f) \leq r_B + r_0 \tag{2}$$

where m_B and r_B are respectively the **midpoint** and **radius** of the translational box B^t of $B = (B^t, B^r)$ (also call them the **midpoint** and **radius** of B), and $\mathrm{Sep}(X, Y) := \inf\{\|x - y\| :$

[3] The configuration space of planar rigid robots is $SE(2) = \mathbb{R}^2 \times S^1$ where S^1 is the unit circle representing angles $[0, 2\pi)$.

$x \in X, y \in Y\}$ denotes the **separation** of two Euclidean sets $X, Y \subseteq \mathbb{R}^2$. We say that B is **empty** if $\widetilde{\phi}(B)$ is empty but $\widetilde{\phi}(B_1)$ is not, where B_1 is the parent of B. We may assume the root is never empty. If B is empty, it is easy to decide whether B is `FREE` or `STUCK`: since the feature set $\widetilde{\phi}(B_1)$ is non-empty, we can find the $f_1 \in \widetilde{\phi}(B_1)$ such that $\mathrm{Sep}(m_B, f_1)$ is minimized. Then $\mathrm{Sep}(m_B, f_1) > r_B$, and by the local property of features (see Feature-Based Approach in Sec. 2), we can decide if m_B is inside (B is `STUCK`) or outside Ω (B is `FREE`).

For a box B where $B^r = S^1$, we maintain its feature set $\widetilde{\phi}(B)$ as above. But when $B^r \neq S^1$, we compute its feature set $\widetilde{\phi}(B)$ as follows. Recall that we decompose R_0 into a set of nice triangles T_j with a common apex A. For each T_j, consider the footprint of T_j with A at m_B and rotating T_j about A from θ_1 to θ_2, where $B^r = [\theta_1, \theta_2]$. By Lemma 1 the resulting swept area is a truncated triangular set (TTS); call it TTS_j. We define (cf. [16]) for a 2D shape S the **s-expansion** of S, denoted by $(S)^s$, to be the Minkowski sum of S with the $Disc(s)$ of radius s centered at the origin. For a TTS, recall that $TTS = T \cap D$ where $T = H_1 \cap H_2 \cap H_3$ is an unbounded triangular set (with each H_i a half space) and D is a disk (Figure 3). Note that $(TTS)^s$ is a proper subset of $(H_1)^s \cap (H_2)^s \cap (H_3)^s \cap (D)^s$; a theorem in the next section gives an exact representation of $(TTS)^s$. We now specify the feature set $\widetilde{\phi}(B)$: for each T_j, let $\widetilde{\phi}_j(B)$ comprise those features f satisfying $\mathrm{Sep}(m_B, f) \leq r_B + r_j$ (replacing r_0 with r_j in Eq. (2)), such that f also **intersects the r_B-expansion of TTS_j**. We can think of $\widetilde{\phi}(B)$ as a collection of these $\widetilde{\phi}_j(B)$'s, each of which is used by the soft predicate $\widetilde{C}_j(B)$ so that we can apply Proposition A.

4. General Complex Robots

When R_0 is a general polygon, not necessarily star-shaped, we can still decompose R_0 into a set of triangles T_j ($j = 1, \ldots, m$), and consider the rotation of these triangles relative to a fixed point O (we may identify O with the origin). In this section, we define what it means for T_j to be "nice" relative to a point O. If O lies in the interior of T_j, we could decompose T_j into at most 6 nice pointed triangles at O, as in the previous section. Henceforth, assume that O does not lie in the interior of T_j.

4.1. Basic Representation of Nicely Swept Sets

Let $T = [A, B, C]$ be any non-degenerate triangular region defined by the vertices A, B, C. Let the origin O be outside the interior of T. We define what it means for T to be "nice relative to O." W.l.o.g., let $0 \leq \|A\| \leq \|B\| \leq \|C\|$ where $\|A\|$ is the Euclidean norm.

We say that T is **nice** if the following three conditions hold:

$$\langle A, B - A\rangle \geq 0, \quad \langle A, C - A\rangle \geq 0, \quad \langle B, C - B\rangle \geq 0. \tag{3}$$

Here $\langle u, v\rangle$ denotes the dot product of vectors u, v.

A more geometric view of niceness is as follows (see Figure 5). Draw three concentric circles centered at O with radii $\|A\|, \|B\|, \|C\|$, respectively. Two circles would coincide if their radii are equal, but we will see that the distinctness of the vertices and niceness prevent such coincidences. Let L_A be the line tangent to the circle of radius $\|A\|$ and passing through the point A. Let H_A denote the closed half-space bounded by L_A and not containing O. The first condition in (3) $\langle A, B - A\rangle \geq 0$ says that $B \in H_A$. Similarly, the second condition says that $C \in H_A$. Finally, the last condition says that $C \in H_B$ (where H_B is analogous to H_A).

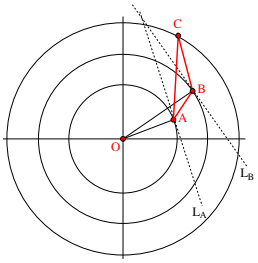


Figure 5: Nice triangle $[A, B, C]$.

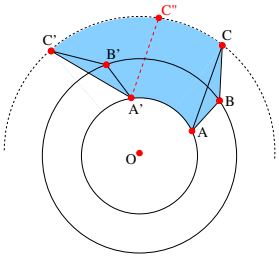


Figure 6: Nicely swept set (NSS, in blue) with A, B, C in CCW order.

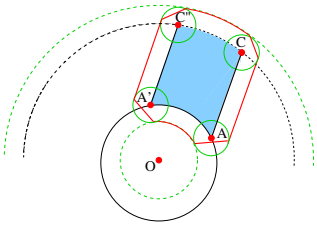


Figure 7: Expansion of $TruncStrip(A, C; A', C'')$ of Fig. 6 (in red).

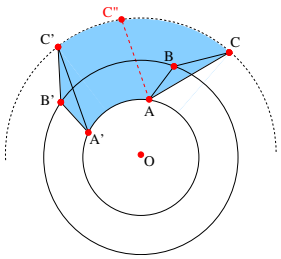


Figure 8: Nicely swept set (NSS, in blue) with A, B, C in CW order.

If T is a nice triangle, then $T[\alpha, \beta]$ is called a **nicely swept set (NSS)**. See Figure 6, where $T[\alpha, \beta]$ is shaded in blue. Let $T[\alpha]$ be the triangle $[A, B, C]$ and $T[\beta]$ be $[A', B', C']$. W.l.o.g., assume[4] that A, B, C appear in counter-clockwise (CCW) order as indicated in Figure 6. Then we can subdivide $T[\alpha, \beta]$ into two parts: the triangle $[A, B, C]$ and another part which we call a **swept segment**.

Notation for Swept Segment: if S is the line segment $[A, C]$, then write $S[\alpha, \beta]$ for this swept segment. The boundary of $S[\alpha, \beta]$ is decomposed into the following sequence of four curves given in clockwise (CW) order: (i) the arc (A, A') centered at O of radius $\|A\|$ from A to A', (ii) the segment $[A', C']$, (iii) the arc (C', C) centered at O of radius $\|C\|$ from C' to C, (iv) the segment $[C, A]$.

Our next goal is to consider s-expansion of the swept segment, i.e.,

$$X = S[\alpha, \beta] \oplus Disc(s). \tag{4}$$

Specifically, we want an easy way to detect the intersection between this expansion with any given feature (corner or edge). To do so, we want to express X as the union of "basic shapes." A subset of $\mathbb{R}^2$ is a **0-basic shape** if it is a half-space, a disc or complement of a disc. We write $Disc(r)$ for the disc of radius r centered at O, and $Ann(r, r')$ for the annulus with inner radius r and outer radius r' centered at O. A shape X is said to be **1-basic** if it can be written as the finite

[4] In case A, B, C appear in clockwise (CW) order, the boundary of $T[\alpha, \beta]$ can be similarly decomposed into two parts, comprising the swept segment $S[\alpha, \beta]$ and the triangle $[A', B', C']$. See Figure 8.

intersection $X = \bigcap_{j=1}^{k} X_j$ where X_j's are 0-basic shapes. The **1-size** of X is the minimum k in such an intersection. So polygons with n sides have 1-size of n. Truncated triangular sets have 1-size of 4. We need some other 1-basic shapes:

- **Strips**: $Strip(a, b; a', b')$ is the region between the two parallel lines $\overline{a, b}$ and $\overline{a', b'}$. Here a, b, a', b' are distinct points.
- **Truncated strips**: $TruncStrip(a, b; a', b')$ is the intersection of $Strip(a, b; a', b')$ with an annulus; the boundary of this shape is comprised of two line segments $[a, b]$ and $[a', b']$ and two arcs (a, a') and (b, b') from the boundary of the annulus.
- **Sectors**: $Sector(a, b, b')$ denotes any region bounded by a circular arc (b, b') and two segments $[a, b]$ and $[a, b']$.

Finally, a shape X is said to be **2-basic** if it can be written as a finite union of 1-basic shapes, $X = \bigcup_{j=1}^{m} X_j$ where X_j's are 1-basic. We call $\{X_1, \ldots, X_m\}$ a **basic representation** of X. The **2-size** of the representation is the sum of the 1-sizes of X_j's. Thus, for any box $B^t \subseteq \mathbb{R}^2$, the s-expansion of B^t is a 2-basic shape since it is the union of four discs and an octagon. We now consider the case where X is the s-expansion of a swept segment $S[\alpha, \beta]$. We first decompose $S[\alpha, \beta]$ into two shapes as follows: suppose C'' lies on the circle of radius $\|C\| = \|C'\|$. Considering both cases of A, B, C being in CCW and CW orders, there are two possible representations:
(i) If $[A', C'']$ is parallel to $[A, C]$ and $[A', C''] \subseteq Ann(\|A\|, \|C\|)$, then we have

$$S[\alpha, \beta] = Sector(A', C', C'') \cup TruncStrip(A, C; A', C''). \tag{5}$$

(ii) If $[A, C'']$ is parallel to $[A', C']$ and $[A, C''] \subseteq Ann(\|A\|, \|C\|)$, then we have

$$S[\alpha, \beta] = Sector(A, C, C'') \cup TruncStrip(A, C''; A', C'). \tag{6}$$

The swept segment in Figure 6 supports the representation (5) but not (6), while the swept segment in Figure 8 supports the representation (6) but not (5). Note that they are symmetric cases, with A, B, C in CCW order in Figure 6 and in CW order in Figure 8. Also, if the angular range of $[\alpha, \beta]$ is greater than 90 degrees and the points O, A, C are collinear, then both representations fail! We next show when at least one of the representations succeeds:

Lemma 4. Assume the width of the angular range $[\alpha, \beta]$ is at most $\pi/2$. Then swept segment $S[\alpha, \beta]$ can be decomposed into a sector and a truncated strip as in (5) or (6).

Proof. Our goal is to choose the point C'' so that either (5) or (6) holds. Let the swept segment be $S[\alpha, \beta]$, with $S[\alpha] = [A, C]$ and $S[\beta] = [A', C']$. Let D (resp., D') be the point such that $\|D\| = \|C\|$ (resp., $\|D'\| = \|C\|$) and O, A, D (resp., O, A', D') are collinear. Then $Sector(O, D, D')$, bounded by the arc (D, D') centered at O, contains either $[A, C]$ or $[A', C']$. If it contains $[A, C]$ (see Figure 6), then we choose C'' such that $[A', C''] \subseteq Sector(O, D, D')$ and $[A, C]$ is parallel to $[A', C'']$, and thus (5) holds. By symmetry, if the sector contains $[A', C']$ (see Figure 8), we can choose C'' so that (6) holds. □

Clearly, the s-expansion of a sector is 2-basic. This is also true for truncated strips (w.l.o.g., considering that in the representation (5)):

Lemma 5. Let $X = TruncStrip(A, C; A', C'')$. There is a basic representation of $X \oplus D(s)$ of the form $\{D_1, D_2, D_3, D_4, X'\}$ where D_i's are discs and X' is the intersection of a convex hexagon with an annulus.

Proof. See Figure 6 for a figure of X. Let $D_1 = Disc_A, D_2 = Disc_C, D_3 = Disc_{A'}, D_4 = Disc_{C''}$ where $Disc_P$ denote the disc with center P of radius s. These discs are outlined in green in Figure 7. The boundary of each D_i $(i = 1, \ldots, 4)$ intersects the boundary of $X \oplus D(s)$ in a circular arc (a_i, b_i) where a_i is closer to O than b_i. Let H_i be the half space containing X and bounded by the line through $[a_i, b_i]$. We need to check that these half spaces do indeed contain X. Also, let H_5 (resp., H_6) be the half space containing X and bounded by the line through b_1 and a_2 (resp., b_3 and a_4). Note that $[b_1, a_2]$ and $[b_3, a_4]$ are parallel. Then we see that $H = \bigcap_{i=1}^{6} H_i$ is a convex hexagon containing X, and the intersection $H \cap Ann(\|A\| - s, \|C\| + s)$ is outlined in red in Figure 7. Observe that this intersection covers all of $\Big(X \oplus Disc(s)\Big) \setminus \bigcup_{i=1}^{4} D_i$.

This construction is valid as long as $\|A\| \geq s$, i.e., the annulus $Ann(\|A\| - s, \|C\| + s)$ is a true annulus. When $\|A\| < s$, the boundary of $X \oplus Disc(s)$ no longer has an inner arc of radius $\|A\| - s$, but degenerates into a concave vertex where the two circles of radius s centered at A and A' (resp.) meet. □

Combining all these lemmas, we conclude:

Theorem 6. Let $T[\alpha, \beta]$ be a nicely swept set where $[\alpha, \beta]$ has width $\leq \pi/2$. Then $T[\alpha, \beta]$ can be decomposed into a triangle, a sector and a truncated strip. The s-expansion of $T[\alpha, \beta]$ has a basic representation which is the union of the s-expansions of the triangle, sector and truncated strip.

Proof. We know that $T[\alpha, \beta]$ can be decomposed into a triangle and a swept segment. The swept segment, since $[\alpha, \beta]$ has width $\leq \pi/2$, can be further decomposed into a sector and a truncated strip. The expansions of the triangle and sector are clear; the expansion of the truncated strip was the subject of the previous lemma. □

The complexity of testing intersection of 2-basic shapes with any feature is proportional to its 2-size, which is $O(1)$. This theorem assures us that the constants in "$O(1)$" are small. Note that it is *not* correct to test if a line segment L intersects a 1-basic shape $X = \bigcap_{j=1}^{k} X_j$ by just testing if L intersects every X_j, since L could intersect every X_j but not all in the same place(s) so that $L \cap X = \emptyset$. Therefore, we need to maintain the *common intersections* between L and all X_j's tested so far as we loop over all X_j's; at the end, L intersects X if and only if there is at least one non-empty set of common intersections. Since the complement of a disk is non-convex, in general this process could result in many sets/segments of common intersections to maintain. Fortunately, there is at most one complement of a disk in our decomposition of an NSS. Thus it is enough to maintain just a *single* set/segment of the common intersection of L with all other 0-basic shapes, and check with the complement of a disk only at the end.

4.2. Partitioning an n-gon into Nice Triangles

Suppose P is an n-gon. We can partition it into $n - 2$ triangles. W.l.o.g., there is at most one triangle that contains the origin O. We can split that triangle into at most 6 nice triangles, using our technique for star-shaped polygons (Lemma 2).

Lemma 7. If T is an arbitrary triangle and O is exterior to T, then we can partition T into at most 4 nice triangles.

Proof. Let $T = [A, B, C]$. In the worst case, all three niceness conditions for T (i.e., $B \in H_A, C \in H_A$, and $C \in H_B$, where $0 \le \|A\| \le \|B\| \le \|C\|$; recall the geometric view of niceness described right after Eq. (3)) are violated. W.l.o.g., suppose that among the three edges of T, $[A, B]$ is the closest to O. Let D be the point on $[A, B]$ such that $[O, D] \perp [A, B]$, and similarly for $E \in [A, C]$ and $F \in [B, C]$; see Figure 9. Then we add segments $[C, D], [D, E], [D, F]$ to decompose T into 4 triangles $[D, E, A], [D, E, C], [D, F, B]$ and $[D, F, C]$. Note that the line L_D tangent to the circle of radius $\|D\|$ (centered at O) and passing through the point D coincides with $[A, B]$; similarly, the line L_E coincides with $[A, C]$ and L_F coincides with $[B, C]$. As before, H_D is the half space bounded by L_D and not containing O; similarly for H_E and H_F. For the triangle $[D, E, A]$, note that $0 \le \|D\| \le \|E\| \le \|A\|$ since $[A, B]$ is closer to O than $[A, C]$ (so $\|D\| \le \|E\|$), $[O, D] \perp [D, A]$ (so $\|D\| \le \|A\|$) and $[O, E] \perp [E, A]$ (so $\|E\| \le \|A\|$). Thus the three niceness conditions for the triangle $[D, E, A]$ are: $E \in H_D, A \in H_D$, and $A \in H_E$. Again, these three conditions are satisfied due to the facts that $[A, B]$ is closer to O than $[A, C]$, $[O, D] \perp [D, A]$ and $[O, E] \perp [E, A]$, i.e., these conditions are automatically satisfied due to the construction of D and E. Similarly, the three niceness conditions for the triangle $[D, E, C]$ are: $E \in H_D, C \in H_D$, and $C \in H_E$, which are again satisfied due to the construction of D and E. Symmetrically, the triangles $[D, F, B]$ and $[D, F, C]$ are both nice due to the construction of D and F. Therefore T can be decomposed into at most 4 nice triangles.

□

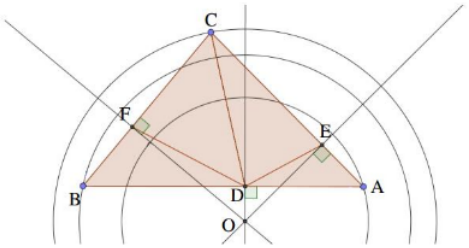


Figure 9: Proof of Lemma 7: A triangle $T = [A, B, C]$ with the origin O in the exterior can be decomposed into at most 4 nice triangles.

The number 4 in this lemma is the best possible: if T is a triangle with circumcenter O, then any partition of T into nice triangles would have at least 4 triangles because we need to introduce vertices in the middle of each side of T.

Theorem 8. Let P be an n-gon.
(i) Given any triangulation of P into $n-2$ triangles, we can refine the triangulation into a triangulation with $\le 4n-6$ nice triangles.
(ii) This bound is tight in this sense: for every $n \ge 3$, there is a triangulation of P whose refinement has size $4n-6$.

Proof. (i) In the given triangulation of P, we might have a triangle containing O. This triangle can be triangulated into at most 6 nice triangles (Lemma 2). By Lemma 7, the remaining $n-3$ triangles

can be refined into $4(n-3)$ nice triangles. The final count is $6+4(n-3)=4n-6$.

(ii) We construct an n-gon whose vertices are all on the unit circle. Note that all such vertices are of the form $e^{\mathbf{i}\theta}$. For the first triangle T_0, pick the vertices $1, e^{\mathbf{i}2\pi/3}, e^{\mathbf{i}4\pi/3}$. Call these vertices u_0, u_1, u_2. Choose the origin O inside T_0 so that for each triangle $[O, u_i, u_j], i \neq j \in \{0,1,2\}$ we have both $\angle O u_i u_j < 90°$ and $\angle O u_j u_i < 90°$. Therefore each triangle $[O, u_i, u_j]$ must be split into 2 nice triangles and overall T_0 must be split into 6 nice triangles. If $n=3$, our result is verified. If $n>3$, we must add $n-3$ additional vertices. Define the vertex $v_k := e^{\mathbf{i}\frac{2k\pi}{3(n-2)}}$, $k=0,1,\ldots,n-2$ (note that $v_0=1$ and $v_{n-2}=e^{\mathbf{i}2\pi/3}$ have previously been chosen). Thus we have added new vertices $v_1,\ldots,v_{n-3}$. Any triangle $[v_k, v_\ell, v_m]$ must be split into 4 nice triangles. This proves our claim. □

4.3. Soft Predicates and T/R Subdivision Scheme

We can now follow the same paradigm as for star-shaped robots in Sec. 3.2. We first apply Theorem 8(i) to partition the robot R_0 into a set of nice triangles, $R_0 = \cup_j T_j$, where all T_j's share a common origin O, and we will use the soft predicates developed for T_j and apply Proposition A. The origin O plays a similar role as the apex in Sec. 3.2. The T/R splitting scheme is exactly the same: we first perform T-splits, splitting only the translational boxes until they are ε-small, and then we perform R-splits, splitting only the rotational boxes until they are ε-small. Essentially the top part of the subdivision tree is a quad-tree, and the bottom parts are binary subtrees (see Sec. 3.2).

The feature set for a subdivision box B where we perform T-splits is the same as before; the only difference is that now for a box B where we perform R-splits, we use a new feature set $\widetilde{\phi}_j(B)$ for each nice triangle T_j where O is not at its vertex (there are at most 6 nice triangles with O at a vertex/apex; see Theorem 8(i)). Suppose $T_j = [a,b,c]$ with $0 \le \|a\| \le \|b\| \le \|c\|$. Let $r_j = \|c\|$. Also, suppose the angle range of box $B = (B^t, B^r)$ is $B^r = [\theta_1, \theta_2]$. Recall the footprint of $T_j[\theta_1,\theta_2]$ is a nicely swept set (NSS); denote it NSS_j. Then the new feature set $\widetilde{\phi}_j(B)$ for T_j comprises those f where $\mathrm{Sep}(m_B, f) \le r_B + r_j$ and f also **intersects the r_B-expansion of** NSS_j (where m_B and r_B are the midpoint and radius of B).

4.4. Worst-Case Time Complexity

We now give a worst-case time complexity bound for our SSS algorithm on an input environment with n polygonal features, assuming a fixed complex robot R_0 represented by an m-gon. We establish this bound in the following theorem, which justifies the claim in the abstract that our method scales linearly in m. Note that the worst-case bounds in (i) and (ii) of this theorem are highly pessimistic since our SSS algorithm is highly adaptive.

Theorem 9. Suppose our polygonal environment has n features whose corners are given by L-bit rational numbers. Assume a fixed complex robot R_0 represented by an m-gon, and $\varepsilon > 0$ is the input resolution parameter. Then
(i) the size of our subdivision tree is $O(S)$, where S is defined as

$$S := n 2^L / \varepsilon^3,$$

and the constant in $O(\cdot)$ depends on r_0^2 with r_0 being the radius of the circumcircle of R_0;
(ii) the time complexity of our SSS algorithm is $O(mS)$.

Proof. (i) Recall from Secs. 3.2 and 4.3 that in our subdivision tree, the top part is a quad tree subdividing the *translational* domain and the bottom parts are binary trees subdividing the *rotational* domain. The leaves of the quad tree can be `FREE` (green), `STUCK` (red), or ε-small (i.e., with width ε) `MIXED` boxes (yellow). Each bottom-part binary tree is attached to a yellow leaf in the quad tree. Note that the maximum size of each binary tree is reached when the leaves are all ε-small (i.e., when each leaf has an angle range (in radians) of ε/r_0) — and thus this maximum binary-tree size is $O(2\pi/(\varepsilon/r_0)) = O(2\pi r_0/\varepsilon) = O(r_0/\varepsilon)$.

Our remaining task is then to bound the number of leaves in the quad tree. To do so, observe that each green/red/yellow leaf has a parent, which must be a `MIXED` (yellow) node and we call it a *penultimate leaf*. Let M be the set of such penultimate leaves. Note that each penultimate leaf has at most 4 green/red/yellow leaves and thus the total number of leaves in the quad tree is $O(|M|)$. With each bottom-part binary tree attached to a yellow leaf of the quad tree, our overall subdivision tree has size $O(|M| \times r_0/\varepsilon)$.

To bound $|M|$, recall that our polygonal environment has n features whose corners are given by L-bit rational numbers. Then the length of each polygonal edge e is $< 2^{L+1}$. Also, each penultimate leaf in M is a `MIXED` (yellow) box B with a *non-empty* feature set, where a feature f is included into the feature set of B if $\text{Sep}(m_B, f) \le r_B + r_0$ (recall from the paragraph of Soft Predicates and Eq. (2) in Sec. 3.2; m_B and r_B are the midpoint and radius of B). In the worst case, each yellow box in M has the smallest width 2ε (parent of an ε-small green/red/yellow leaf with width ε). Since such boxes in M are *essentially disjoint*, each edge e gives rise to at most $X = A/(2\varepsilon)^2$ such boxes, where A is the area of the union of two rectangles of $2^{L+1} \times (r_B + r_0)$ each, one on each side of e, together with the two discs centered at the two endpoints of e with radius $(r_B + r_0)$. As each such disc has half of it intersected with the rectangles, only the other half of it contributes to a new area beyond the rectangles. The area A is thus $2 \times 2^{L+1} \times (r_B + r_0) + \pi(r_B + r_0)^2$, and X is the number of boxes, each of area $(2\varepsilon)^2$, to tile up the area of A. Since each box B has width 2ε, r_B is $\sqrt{2}\varepsilon$, and we have $X = O(2^L/\varepsilon \times (1 + (r_0/\varepsilon))) = O(r_0 \cdot 2^L/\varepsilon^2)$. Finally, there are n polygonal edges, each giving rise to X boxes in M, and thus we have $|M| = nX = O(r_0 \cdot n2^L/\varepsilon^2)$. Therefore, our overall subdivision tree has size $O(|M| \times r_0/\varepsilon) = O(r_0^2 \cdot n2^L/\varepsilon^3) = O(r_0^2 S) = O(S)$, where the constant in $O(\cdot)$ depends on r_0^2. This completes the proof of (i).

(ii) Our SSS algorithm constructs the subdivision tree by splitting boxes and calling our predicates to classify each box. Each classification takes $O(m)$ time since our predicate is a composition of $O(m)$ simpler predicates. This proves the overall time complexity of $O(mS)$. □

5. Experimental Results

Table 1: Robot Statistics.

Robot	m (# sides)	t (# triangles)
L-shaped	6	4
snowflake	18	24
S-shaped	12	26
3-legged	14	20
C-shaped	18	22

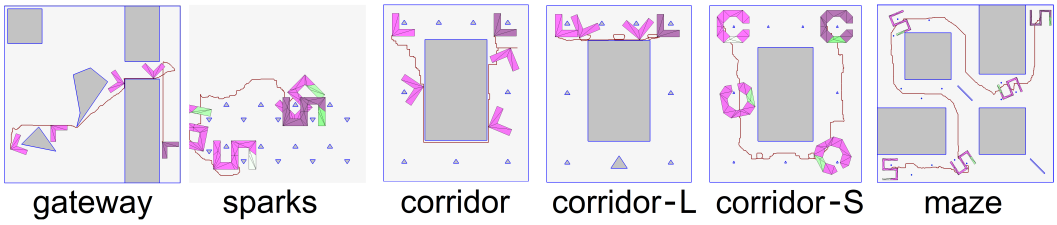


Figure 10: Six environments in our experiments.

Table 2: Running Our Planner (R: radius of the robot's circumcircle around its rotation center; P?: path found? (Yes/No); Time is in s; S-shaped*: thin version).

Exp#	Robot	Envir.	R	ϵ	α	β	P?	Time
0	L-shaped	gateway	50	2	(18, 98, 340°)	(458,119,270°)	Yes	10.106
1	L-shaped	gateway	50	4	(18, 98, 340°)	(458,119,270°)	No	8.431
2	snowflake	sparks	56	2	(108, 136, 0°)	(358, 155, 0°)	Yes	17.846
3	snowflake	sparks	56	2	(108, 136, 0°)	(358, 155, 180°)	Yes	3.370
4	S-shaped	sparks	74	4	(132, 80, 90°)	(333, 205, 90°)	Yes	34.284
5	S-shaped	sparks	74	4	(132, 80, 90°)	(333, 205, 60°)	No	57.371
6	3-legged	sparks	70	2	(108, 136, 0°)	(368, 155, 0°)	Yes	41.745
7	L-shaped	corridor	68	2	(75, 420, 0°)	(370, 420, 0°)	Yes	4.012
8	L-shaped	corridor	68	3	(75, 420, 0°)	(370, 420, 0°)	Yes	1.926
9	L-shaped	corridor	68	5	(75, 420, 0°)	(370, 420, 0°)	Yes	2.684
10	L-shaped	corridor-L	68	5	(75, 420, 0°)	(370, 420, 0°)	No	2.908
11	L-shaped	corridor-L	68	3	(75, 420, 0°)	(370, 420, 0°)	Yes	2.255
12	C-shaped	corridor-S	80	4	(80, 450, 0°)	(380, 450, 0°)	Yes	26.200
13	S-shaped	maze	38	2	(38, 38, 0°)	(474, 474, 90°)	No	90.097
14	S-shaped*	maze	38	2	(38, 38, 0°)	(474, 474, 90°)	Yes	79.518

We have implemented our approaches in `C/C++` with `Qt` GUI platform. The software and data sets are freely available from the web site for our open-source `Core Library` [6]. All experiments are reproducible as targets of Makefiles in `Core Library`. Our experiments are on a PC with one 3.4GHz Intel Quad Core i7-2600 CPU, 16GB RAM, nVidia GeForce GTX 570 graphics and Linux Ubuntu 16.04 OS. The results are summarized in Table 2 and Table 3. Table 2 is concerned only with the behavior of our complex robots; Table 3 gives comparisons with the open-source OMPL library [15]. The robots are as shown in Figure 1; their statistics are given in Table 1. Figure 10 shows the six environments in our experiments.

We select some interesting experiments to analyze characteristic behavior of our planner. Please see Table 2 and the video (`https://cs.nyu.edu/exact/gallery/complex/complex-robot-demo.mp4`). In Exp0-1, we show how the parameter ϵ affects the result. With a narrow gateway, when we change ϵ from 2 to 4, the output changes from a path to `NO-PATH` for the same configuration. In Exp2-3, we observe how the snowflake robot rotates and maneuvers to get from the start to two different goals. For Exp4-5, the difference is in the angles of the goal configuration; in Exp5 this is designed to be an isolated configuration and the planner outputs `NO-PATH` as desired. Exp6 shows how the robot squeezes among the obstacles to move its complex shape through the environment.

Table 3: Comparing with OMPL ("#": Exp#; "Time/P?": our run time (in s)/path found? (Y/N). Each OMPL method: Average Time (in s)/Standard Deviation/Success Rate, over 10 runs).

#	Time/P?	PRM	RRT	EST	KPIECE
0	10.106/Y	**4.18**/2.53/1	42.13/38.49/1	76.22/110.44/0.9	300/0/0
2	17.846/Y	**9.22**/6.82/1	210.41/144.25/0.3	271.75/89.31/0.1	240.00/126.47/0.2
3	**3.370**/Y	300/0/0	300/0/0	300/0/0	300/0/0
4	34.284/Y	**5.93**/7.20/1	217.33/134.53/0.3	300/0/0	300/0/0
5	**57.371**/N	300/0/0	300/0/0	300/0/0	300/0/0
6	41.745/Y	**2.72**/4.89/1	154.22/141.77/0.5	104.32/78.10/0.7	3.16/4.28/1
8	1.926/Y	0.63/0.55/1	300/0/0	3.02/4.71/1	**0.41**/0.28/1
11	2.255/Y	**1.49**/0.84/1	300/0/0	241.24/124.88/0.2	1.58/1.47/1
12	26.200/Y	**3.16**/4.21/1	300/0/0	172.506/120.38/0.7	93.88/88.03/0.8
13	**90.097**/N	300/0/0	300/0/0	300/0/0	300/0/0
14	79.518/Y	300/0/0	236.72/106.44/0.3	300/0/0	**39.81**/91.57/0.9

Exp7-9 use the same L-shaped robot, α, β configurations and the environment; only ϵ varies. The planner can find three totally different paths. When ϵ is small (Exp7), the path is very carefully adjusted to move the robot around the obstacles. When ϵ is larger (Exp8), the planner finds an upper path with a higher clearance. When ϵ is even larger (Exp9), the planner chooses a very safe but much longer path at the bottom. Note that using a larger ϵ usually makes the search faster, since we stop splitting boxes smaller than ϵ, but a longer path can make the search slower. In Exp10-11, we modify the environment of Exp7-9 by putting a large obstacle at the bottom, which forces the robot to find a path at the top. Exp12 uses an environment similar to those in Exp7-11 but with much smaller scattered obstacles. It is designed for the C-shaped robot, which can rotate while having an obstacle in its pocket. Exp13-14 use a challenging environment where the small scattered obstacles force the S-shaped robot to rotate around and only the "thin" version (Exp14, also in Fig. 10 "maze") can squeeze through.

In Table 3 we compare our planner with several sampling algorithms in OMPL: PRM, RRT, EST, and KPIECE. These experiments are correlated to those in Table 2 (see the Exp #). Each OMPL planner is run 10 times with a time limit 300 seconds (default), where all planner-specific parameters use the OMPL default values. We see that for OMPL planners there are often unsuccessful runs and they have to time out even when there is a path. On the other hand, our algorithm consistently solves the problems in a reasonable amount of time, often much faster than the OMPL planners, in addition to being able to report `NO-PATH`.

6. Conclusions

Although the study of rigorous algorithms for motion planning has been around for over 40 years, there has always been a gap between such theoretical algorithms and the practical methods. Our introduction of resolution-exactness and soft predicates on the theoretical front, together with matching implementations, closes this gap. Moreover, it eliminated the "narrow passage" problem that plagued the sampling approaches. The present paper extends our approach to challenging planning problems for which no exact algorithms exist.

What are the current limitations of our work? We implement everything in machine precision (the practice in this field). But it can be easily modified to achieve the theoretical guarantees of

resolution-exactness if we use arbitrary precision BigFloats number types. Another limitation is that we have not computed the implicit constant $K > 1$ of resolution-exactness; in [18], this constant is shown to be a product of a few constants such as Lipshitz constant and the effectivity constant (σ). However it can be visually seen from our experiments that the actual K is not unreasonably large.

We pose two open problems: One is to find an optimal decomposition of m-gons into nice triangles (currently, we simply give an upper bound). Such decomposition will have impact for practical complex robots. Second, we would like to develop similar decomposability of soft predicates for complex rigid robots in $\mathbb{R}^3$.

References

[1] F. Avnaim, J.-D. Boissonnat, and B. Faverjon. A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. In J.-D. Boissonnat and J. Laumond, editors, *Geometry and Robotics*, LNCS vol 391, pages 67–86. Springer, Berlin-Heidelberg, 1989.

[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] H. Bennett, E. Papadopoulou, and C. Yap. Planar minimization diagrams via subdivision with applications to anisotropic Voronoi diagrams. *Computer Graphics Forum (Special Issue for Eurographics Symp. on Geometric Processing (SGP) 2016)*, 35(5):229–247, 2016.

[5] 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.

[6] Core Library. https://cs.nyu.edu/exact/core_pages/downloads.html.

[7] D. Halperin, E. Fogel, and R. Wein. *CGAL Arrangements and Their Applications*. Springer-Verlag, Berlin and Heidelberg, 2012.

[8] C.-H. Hsu, Y.-J. Chiang, and C. Yap. Rods and rings: Soft subdivision planner for $\mathbf{R}\hat{}3$ x $\mathbf{S}\hat{}2$. *arXiv e-prints*, arXiv:1903.09416, Mar 2019. This version includes appendices A–F.

[9] C.-H. Hsu, Y.-J. Chiang, and C. Yap. Rods and rings: Soft subdivision planner for $\mathbf{R}\hat{}3$ x $\mathbf{S}\hat{}2$. In *Proc. 35th Symp. on Comp. Geometry (SoCG 2019)*, pages 43:1–43:17, June 18-21, 2019.

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

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

[12] 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.

[13] V. Milenkovic, E. Sacks, and S. Trac. Robust complete path planning in the plane. In *Proc. 10th Workshop on Algorithmic Foundations of Robotics (WAFR 2012)*, Springer Tracts in Advanced Robotics, vol.86, pages 37–52. Springer, 2012.

[14] O. Salzman, M. Hemmer, B. Raveh, and D. Halperin. Motion planning via manifold samples. In *Proc. European Symp. Algorithms (ESA)*, pages 493–505, 2011.

[15] I. Şucan, M. Moll, and L. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012. http://ompl.kavrakilab.org.

[16] 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.

[17] 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. Robotics Science and Systems Conf. (RSS 2013), Berlin. In arXiv:1402.3213. Full paper: http://cs.nyu.edu/exact/papers/.

[18] 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.

[19] C. Yap, Z. Luo, and C.-H. Hsu. Resolution-exact planner for thick non-crossing 2-link robots. In *Proc. 12th Intl. Workshop on Algorithmic Foundations of Robotics (WAFR '16)*, 2016. Dec. 13-16, 2016, San Francisco. The appendix in the full paper (and arXiv from `http://cs.nyu.edu/exact/` (and `arXiv:1704.05123 [cs.CG]`) contains proofs and additional experimental data.

[20] 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.

[21] B. Zhou, Y.-J. Chiang, and C. Yap. Soft subdivision motion planning for complex planar robots. In *Proc. 26th European Symp. on Algorithms (ESA 2018)*, pages 73:1–73:14, 2018. Helsinki, Finland, Aug. 20-24, 2018.

[22] D. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991.