Resolution-Exact Planner for Thick Non-Crossing 2-Link Robots*

Chee K. Yap, Zhongdi Luo, and Ching-Hsiang Hsu

Department of Computer Science
Courant Institute, NYU
New York, NY 10012, USA
{yap,zl562,chhsu}@cs.nyu.edu

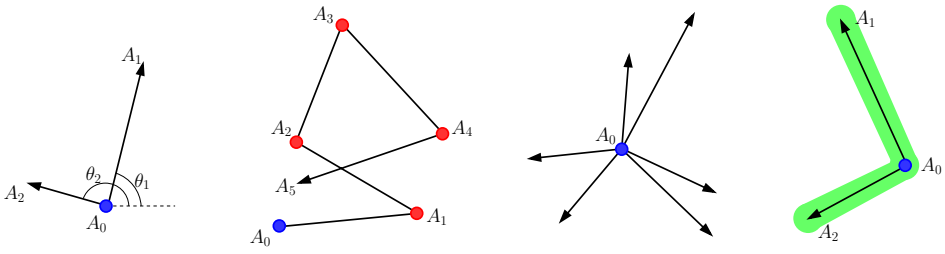
Abstract. We consider the path planning problem for a 2-link robot amidst polygonal obstacles. Our robot is parametrizable by the lengths $\ell_1, \ell_2 > 0$ of its two links, the thickness $\tau \geq 0$ of the links, and an angle κ that constrains the angle between the 2 links to be strictly greater than κ. The case $\tau > 0$ and $\kappa \geq 0$ corresponds to "thick non-crossing" robots. This results in a novel 4DOF configuration space $\mathbb{R}^2 \times (\mathbb{T}^2 \setminus \Delta(\kappa))$ where $\mathbb{T}^2$ is the torus and $\Delta(\kappa)$ the diagonal band of width κ.
We design a resolution-exact planner for this robot using the framework of Soft Subdivision Search (SSS). First, we provide an analysis of the space of forbidden angles, leading to a soft predicate for classifying configuration boxes. We further exploit the T/R splitting technique which was previously introduced for self-crossing thin 2-link robots.
Our open-source implementation in Core Library achieves real-time performance for a suite of combinatorially non-trivial obstacle sets. We also show that our algorithm performs very favorably compared with several state-of-art sampling algorithms in the OMPL software.

1 Introduction

Motion planning is one of the key topics of robotics [7, 3]. The dominant approach to motion planning for the last two decades has been based on sampling, as represented by PRM [5] or RRT [6] and their many variants. An alternative (older) approach is based on subdivision [2, 16, 1]. Recently, we introduced the notion of **resolution-exactness** which might be regarded[1] as the well-known idea of "resolution completeness" with a suitable converse [12, 13]. This provides the theoretical basis for exploiting the concept of **soft predicates**, which is roughly speaking the numerical approximation of exact predicates. Such predicates avoids the hard problem of deciding zero, leading to much more practical algorithms than exact algorithms. To support this new class of algorithms, and inspired by the success of the PRM framework, we introduce an algorithmic framework [13, 14] based on subdivision called **Soft Subdivision Search** (SSS). The present paper continues our exploration of algorithms in this framework.

* This work is supported by NSF Grants CCF-0917093 and CCF-1423228.

[1] In the theory of computation, a computability concept that has no such converse (e.g., recursive enumerability) is only "partially complete".



(a) Thin R_2 Robot (b) Chain Robot (c) Spider Robot (d) Thick R_2 Robot

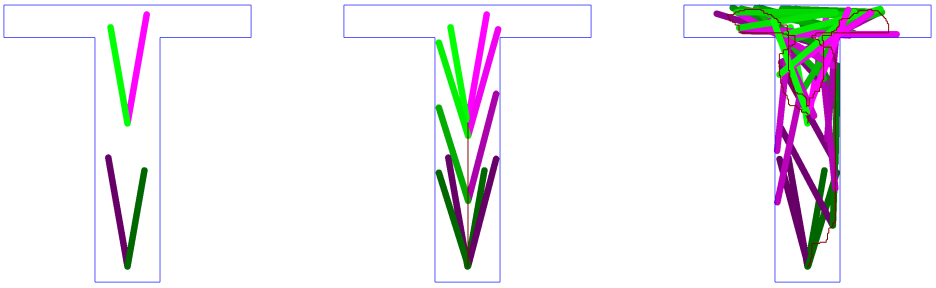
Fig. 1: Link Robots

Link robots offer a compelling class of non-trivial robots for exploring path planning (see [4, chap. 7]). In the plane, the simplest example of a non-rigid robot is the **2-link robot**, $R_2 = R_2(\ell_1, \ell_2)$, with links of lengths $\ell_1, \ell_2 > 0$. The two links are connected through a rotational joint A_0 called the **robot origin** as illustrated in Figure 1(a). The 2-link robot is in the intersection of two well-known families of link robots (see [9] for definition): **chain robots** and **spider robots** (Figure 1(b,c)).

One limitation of link robots is that the usual model of links as line segments is not physically realistic. On the other hand, a model of mechanical links involving complex details may require algorithms that currently do not exist (at least in the case of exact algorithms) or have high computational complexity. As a compromise, we introduce **thick links** by forming the Minkowski sum of each link with a ball of radius $\tau > 0$ (and recover "thin links" by setting $\tau = 0$). See Figure 1(d). To our knowledge, no exact algorithm for thick R_2 is known; for a single link R_1, an exact algorithm based on retraction follows from [10]. In this paper, we further parametrize the 2-link robot by a "bandwidth" κ which constrains the angle between the 2 links to be strictly larger than κ ("self-crossing" links is recovered by setting $\kappa < 0$). Thus, our full robot model is denoted

$$R_2(\ell_1, \ell_2, \tau, \kappa).$$

To illustrate the non-crossing constraint, we use a simple "T-room" environment as shown in Figure 2. Suppose the robot has to move between the two indicated configurations in Figure 2(a): from start configuration α (above) to goal configuration β (below). There is an obvious path from α to β as illustrated in Figure 2(b): the robot origin moves directly from its start to goal positions, while the link angles simultaneously adjust to their goal angles. However, such paths require the two links to cross each other. To achieve a "non-crossing" solution from α to β, we need a less obvious path as illustrated in Figure 2(c): the robot origin must first move away from the goal configuration towards the T-junction, in order to maneuver the two links into the appropriate relative order before it can move toward the goal configuration.



(a) α (above) and β (below) (b) Self-crossing path (c) Non-crossing path

Fig. 2: Path from configurations α to β in T-Room Environment

We had chosen $\varepsilon = 2$ in Figure 2(b,c); furthermore, $\kappa = 7$ for the non-crossing instance. But if we increase either ε to 3 or κ to 8, then the non-crossing instance would report NO-PATH. The self-crossing instance still finds a path with $\varepsilon = 3$. It is important to know that the NO-PATH output from resolution-exact algorithms is not never due to exhaustion ("time-out"). It is a principled answer, guaranteeing the non-existence of paths with clearance $> K \cdot \varepsilon$ (for some $K > 1$ depending on the algorithm). This is the key strength in our approach: *in contrast to sampling approaches, there is no "narrow passage" for resolution-exact algorithms.* Ultimately, we view the narrow passage as just the "halting problem" for path planning: how do you detect non-existence of paths? If, as our experimental study below indicates, our algorithms are just as fast (usually faster) than the sampling approaches, then it seems reasonable to say that the "narrow passage problem" has been solved, albeit through a non-sampling approach, and up to 4DOF. But we suspect this is not really about DOF's.

The T-Room Environment has trivial combinatorial complexity, designed to illustrate the non-crossing phenomenon. But our algorithm scales well with the combinatorial complexity of the environment; all our solutions are "realtime". An interesting environment is Figure 3 with 100 randomly generated triangles.

Overview of Paper. This paper explains the theory and construction of a resolution-exact planner for thick non-crossing 2-link robots. For the reader's convenience, we provide an appendix describing the theory of resolution-exactness, soft predicate and the SSS Framework. This appendix will be removed in the final paper. Besides implementing our SSS algorithm in our open-source Core Library [15], we also conduct experiments to compare with state-of-art sampling algorithms found in OMPL [11].

(a) Trace of the robot origin (b) Sub-sampled path (c) Subdivision boxes

Fig. 3: 100 Random Triangles Environment: non-crossing path found ($\kappa = 115°$)

2 Configuration Space of Non-Crossing 2-Link Robot

The configuration space of R_2 is $C_{space} := \mathbb{R}^2 \times \mathbb{T}^2$ where $\mathbb{T}^2 = S^1 \times S^1$ is the torus and $S^1 = SO(2)$ is the unit circle. We represent S^1 by the interval $[0, 2\pi]$ with the identification $0 = 2\pi$. Closed angular intervals of S^1 are denoted by $[s, t]$ where $s, t \in [0, 2\pi]$ using the convention

$$[s,t] := \begin{cases} \{\theta : s \le \theta \le t\} & \text{if } \ s \le t, \\ [s, 2\pi] \cup [0, t] & \text{if } \ s > t. \end{cases}$$

In particular, $[0, 2\pi] = S^1$ and $[2\pi, 0] = [0, 0]$. The standard Riemannian metric $d : S^1 \times S^1 \to \mathbb{R}_{\ge 0}$ on S^1 is given by $d(\theta, \theta') = \min\{|\theta - \theta'|, 2\pi - |\theta - \theta'|\}$. Thus $0 \le d(\theta, \theta') \le \pi$.

To represent the non-crossing configuration space, we must be more specific about interpreting the parameters in a configuration $(x, y, \theta_1, \theta_2) \in C_{space}$: there are two natural interpretations, depending on whether we view R_2 as a chain robot or a spider robot (see Figure 1(b,c)). We choose the latter view, in which case (x, y) is the **footprint** of the joint A_0 at the center of the spider. Then the angles θ_1, θ_2 of the two links are independent. This leads to somewhat simpler analysis. It is not entirely unclear to us what possible tradeoffs might accrue by viewing R_2 as a chain robot, but certainly the subdivision process would be organized differently; this is future research. If we consider chain robots R_k for $k \ge 3$, we would be forced to explore this alternative. In the terminology of [9], the robot R_2 has three named points A_0, A_1, A_2. Viewing R_2 as a spider robot, then point A_0 is the spider center (or origin) as shown in Figure 1(a). The **footprints** of these points at configuration $\gamma = (x, y, \theta_1, \theta_2)$ are given by

$$\begin{aligned} A_0[\gamma] &:= (x, y), \\ A_1[\gamma] &:= (x, y) + \ell_1(\cos\theta_1, \sin\theta_1), \\ A_2[\gamma] &:= (x, y) + \ell_2(\cos\theta_2, \sin\theta_2). \end{aligned}$$

The **thin footprint** of R_2 at γ, denoted $R_2[\gamma]$, is defined as the union of the line segments $[A_0[\gamma], A_1[\gamma]]$ and $[A_0[\gamma], A_2[\gamma]]$. Finally the **thick footprint** of R_2 is given by $Fprint_\tau(\gamma) := D(\mathbf{0}, \tau) \oplus R_2[\gamma]$, the Minkowski sum $\oplus$ of the thin footprint with the disc $D(\mathbf{0}, \tau)$ centered at the origin $\mathbf{0}$ of radius τ.

The **non-crossing configuration space** of bandwidth κ is defined to be

$$C_{space}(\kappa) := \mathbb{R}^2 \times (\mathbb{T}^2 \setminus \Delta(\kappa))$$

where $\Delta(\kappa)$ is the **diagonal band**

$$\Delta(\kappa) := \big\{(\theta, \theta') \in \mathbb{T}^2 : d(\theta, \theta') \leq \kappa\big\} \subseteq \mathbb{T}^2.$$

Note three special cases:

- If $\kappa < 0$ then $\Delta(\kappa)$ is the empty set.
- If $\kappa = 0$ then $\Delta(0)$ is a closed curve in $\mathbb{T}^2$.
- If $\kappa \geq \pi$ then $\Delta(\kappa) = S^1$.

Configurations in $\mathbb{R}^2 \times \Delta(0)$ are said to be **self-crossing**; all other configurations are **non-crossing**. Here we focus on the case $\kappa \geq 0$. For our subdivision below, we will split $\mathbb{T}^2 \setminus \Delta(0)$ into two connected sets: $\mathbb{T}^2_< := \big\{(\theta, \theta') \in \mathbb{T}^2 : 0 \leq \theta < \theta' < 2\pi\big\}$ and $\mathbb{T}^2_> := \big\{(\theta, \theta') \in \mathbb{T}^2 : 0 \leq \theta' < \theta < 2\pi\big\}$. For $\kappa \geq 0$, the diagonal band $\Delta(\kappa)$ retracts to the closed curve $\Delta(0)$. In $\mathbb{R}^2$, if we omit such a set, we will get two connected components. In contrast, that $\mathbb{T}^2 \setminus \Delta(\kappa)$ remains connected. CLAIM: $\mathbb{T}^2 \setminus \Delta(\kappa)$ *is topologically a cylinder with two boundary components.* The point is that the non-crossing constraint has changed the topology of the configuration space. To see claim, consider the standard model of $\mathbb{T}^2$ represented by a square with opposite sides identified as in Figure 4(a) (we show the case $\kappa = 0$). By rearranging the two triangles $\mathbb{T}^2_<$ and $\mathbb{T}^2_>$ as in Figure 4(b), our claim is now visually obvious.

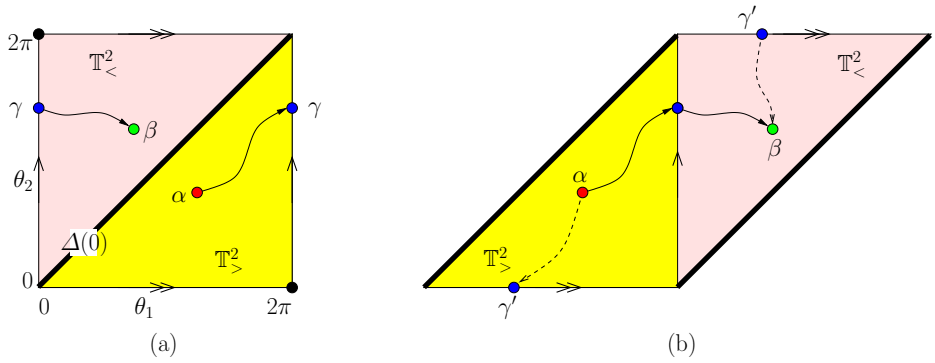


Fig. 4: Paths in $\mathbb{T}^2 \setminus \Delta(0)$ from $\alpha \in \mathbb{T}^2_>$ to $\beta \in \mathbb{T}^2_<$

3 Forbidden Angle Analysis of Thick Links

Towards the development of a soft-predicate for thick links, we must first extend our analysis in [9] which introduced the concept of forbidden angles for thin links. Let $L(\ell, \tau)$ be a single link robot of length $\ell > 0$ and thickness $\tau \geq 0$. Its configuration space is $SE(2) = \mathbb{R}^2 \times S^1$. Given a configuration $(b, \theta) \in SE(2)$, the **footprint** of $L(\ell, \tau)$ at (b, θ) is

$$Fprint_{\ell,\tau}(b, \theta) := L \oplus D(\mathbf{0}, \tau)$$

where $\oplus$ denotes Minkowski sum, L is the line segment $[b, b + \ell(\cos\theta, \sin\theta)]$ and $D(\mathbf{0}, \tau)$ is the disk as above. When ℓ, τ is understood, we simply write "$Fprint(b, \theta)$" instead of $Fprint_{\ell,\tau}(b, \theta)$.

Let $S, T \subseteq \mathbb{R}^2$ be closed sets. An angle θ is **forbidden** for (S, T) if there exists $s \in S$ such that $Fprint(s, \theta) \cap T$ is non-empty. If $t \in Fprint(s, \theta) \cap T$, then the pair $(s, t) \in S \times T$ is a **witness** for the forbidden-ness of θ for (S, T). The set of forbidden angles of (S, T) is called the **forbidden zone** of S, T and denoted $\text{Forb}_{\ell,\tau}(S, T)$. Clearly, $\theta \in \text{Forb}_{\ell,\tau}(S, T)$ iff there exists a witness pair $(s, t) \in S \times T$. Moreover, we call (s, t) a **minimum witness** of θ if the Euclidean norm $\|s - t\|$ is minimum among all witnesses of θ. If (s, t) is a minimum witness, then clearly $s \in \partial S$ and $t \in \partial T$.

Lemma 1. *For any sets $S, T \subseteq \mathbb{R}^2$, we have*

$$\text{Forb}_{\ell,\tau}(S, T) = \pi + \text{Forb}_{\ell,\tau}(T, S).$$

Proof. For any pair (s, t) and any angle α, we see that

$$t \in Fprint(s, \alpha) \text{ iff } \ s \in Fprint(t, \pi + \alpha).$$

Thus, there is a witness (s, t) for α in $\text{Forb}_{\ell,\tau}(S, T)$ iff there is a witness (t, s) for $\pi + \alpha$ in $\text{Forb}_{\ell,\tau}(T, S)$. The lemma follows. **Q.E.D.**

¶1. The Forbidden Zone of two points Consider the forbidden zone $\text{Forb}_{\ell,\tau}(V, C)$ defined by two points $V, C \in \mathbb{R}^2$ with $d = \|V - C\|$. (The notation V suggests a vertex of a translational box B^t and C suggests a corner of the obstacle set.) In our previous paper [9] on thin links (i.e., $\tau = 0$), this case is not discussed for reasons of triviality. When $\tau > 0$, the set $\text{Forb}_{\ell,\tau}(V, C)$ is more interesting. Clearly, $\text{Forb}_{\ell,\tau}(V, C)$ is empty iff $d > \ell + \tau$ (and a singleton if $d = \ell + \tau$). Also $\text{Forb}_{\ell,\tau}(V, C) = S^1$ iff $d \leq \tau$. Henceforth, we may assume

$$\tau < d < \ell + \tau. \tag{1}$$

The forbidden zone of V, C can be written in the form

$$\text{Forb}_{\ell,\tau}(V, C) := [\nu - \delta, \nu + \delta]$$

for some ν, δ. We call ν the **nominal angle** and δ the **correction angle**. From the symmetry of the footprint, we see that nominal angle ν is equal to $\theta(V, C)$.

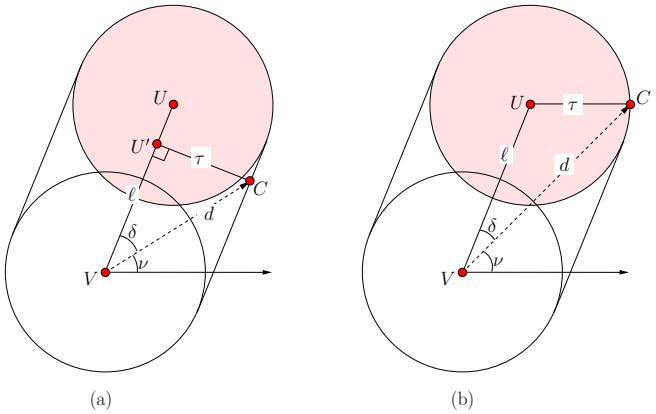


Fig. 5: $\text{Forb}_{\ell,\tau}(V, C)$

It remains to determine δ. Consider the configuration $(V, \theta) \in SE(2)$ of our link $L(\ell, \tau)$ where link origin is at V and the link makes an angle θ with the positive x-axis. The angle δ is determined when the point C lies on the boundary of $Fprint(V, \theta)$. The two cases are illustrated in Figure 5 where $\theta = \nu + \delta$ and other endpoint of the link is U; thus $\|VU\| = \ell$ and $\|VC\| = d$, and $\delta = \angle(CVU)$. Under the constraint (1), there are two ranges for d:

(a) d is short: $d^2 \leq \tau^2 + \ell^2$. In this case, the point C lies on the straight portion of the boundary of the footprint, as in Figure 5(a). From the right-angle triangle $CU'V$, we see that $\delta = \arcsin(\tau/d)$.

(b) d is long: $d^2 > \tau^2 + \ell^2$. In this case, the point C lies on the circular portion of the boundary of the footprint, as in Figure 5(b). Consider the triangle CUV with side lengths of d, ℓ, τ. By the cosine law, $\tau^2 = d^2 + \ell^2 - 2d\ell\cos\delta$ and thus

$$\delta = \arccos\left(\frac{\ell^2 + d^2 - \tau^2}{2d\ell}\right).$$

This proves:

Lemma 2. *Assume $\|VC\| = d$ satisfies (1). Then*

$$\text{Forb}_{\ell,\tau}(V, C) = [\nu - \delta, \nu + \delta]$$

where $\nu = \theta(V, C)$ and

$$\delta = \delta(V, C) = \begin{cases} \arcsin(\tau/d) & \text{if } \ d^2 \leq \tau^2 + \ell^2, \\ \arccos\left(\frac{\ell^2+d^2-\tau^2}{2d\ell}\right) & \text{if } \ d^2 > \tau^2 + \ell^2. \end{cases} \tag{2}$$

¶2. The Forbidden Zone of a Vertex and a Wall Recall that the boundary of a box B^t is divided into four **sides**, and two adjacent sides share

a common endpoint which we call a **vertex**. We now determine $\text{Forb}_{\ell,\tau}(V, W)$ where V is a vertex and W a wall feature. Choose the coordinate axes such that W lies on the x-axis, and $V = (0, -\sigma)$ lies on the negative y-axis, for some $\sigma > 0$. Let the two corners of W be C, C' with C' lying to the left of C. See Figure 6.

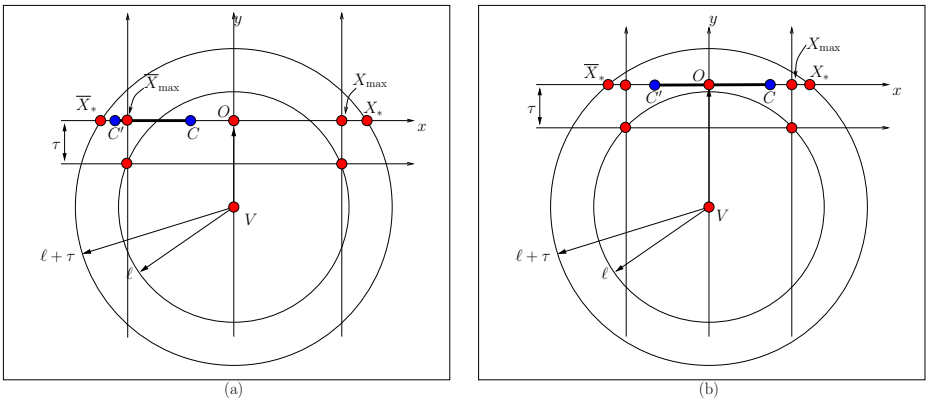


Fig. 6: Stop Analysis for $\text{Forb}_{\ell,\tau}(V, W)$ (assuming $\sigma > \tau$)

We first show that the interesting case is when

$$\tau < \sigma < \ell + \tau. \tag{3}$$

If $\sigma \geq \ell + \tau$ then $\text{Forb}_{\ell,\tau}(V, W)$ is either a singleton ($\sigma = \ell + \tau$) or else is empty ($\sigma > \ell + \tau$). Likewise, the following lemma shows that when $\sigma \leq \tau$, we are to point-point case of Lemma 2:

Lemma 3. *Assume $\sigma \leq \tau$. We have*

$$\text{Forb}_{\ell,\tau}(V, W) = \begin{cases} S^1 & \text{if } \ D(V, \tau) \cap W \neq \emptyset, \\ \text{Forb}_{\ell,\tau}(V, c) & \text{else} \end{cases}$$

where $c = C$ or C'.

Proof. Recall that we have chosen the coordinate system so that W lies on the x-axes and $V = (0, -\sigma)$. It is easy to see that $\text{Forb}_{\ell,\tau}(V, W) = S^1$ iff the disc $D(V, \tau)$ intersects W. So assume otherwise. In that case, the closest point in W to V is c, one of the two corners of W. The lemma is proved if we show that

$$\text{Forb}_{\ell,\tau}(V, W) = \text{Forb}_{\ell,\tau}(V, c).$$

It suffices to show $\text{Forb}_{\ell,\tau}(V, W) \subseteq \text{Forb}_{\ell,\tau}(V, c)$. Suppose $\theta \in \text{Forb}_{\ell,\tau}(V, W)$. So it has a witness (V, c') for some $c' \in W$. However, we see that the minimal witness for this case is (V, c). This proves that $\theta \in \text{Forb}_{\ell,\tau}(V, c)$. **Q.E.D.**

In addition to (3), we may also assume the wall lies within the annulus of radii $(\tau, \tau+\ell)$ centered at V:

$$\|VC\|, \|VC'\| \in (\tau, \ell+\tau) \tag{4}$$

Using the fact that $V = (0, -\sigma)$ and W lies in the x-axis, the following is immediate:

Lemma 4. *Assume (3) and (4).*
Then $\mathrm{Forb}_{\ell,\tau}(V, W)$ *is a non-empty connected interval of* S^1,

$$\mathrm{Forb}_{\ell,\tau}(V, W) = [\alpha, \beta] \subseteq (0, \pi).$$

Our next goal is to determine the angles α, β in this lemma. Consider the footprints of the link at the extreme configurations $(V, \alpha), (V, \beta) \in SE(2)$. Clearly, W intersects the boundary (but not interior) of these footprints, $Fprint(V, \alpha)$ and $Fprint(V, \beta)$. Except for some special configurations, these intersections are singleton sets. Regardless, pick any $A \in W \cap Fprint(V, \alpha)$ and $B \in W \cap Fprint(V, \beta)$. Since α is an endpoint of $\mathrm{Forb}_{\ell,\tau}(V, W)$, we see that $A \in (\partial W) \cap \partial(Fprint(V, \alpha))$. We call A a **left stop** for the pair (V, W) because[2] for any $\delta' > 0$ small enough, $A \in Fprint(V, \alpha + \delta')$ while $W \cap (V, \alpha - \delta') = \emptyset$. Similarly the point B is called a **right stop** for the pair (V, W). Clearly, we can write

$$\alpha = \theta(V, A) - \delta(V, A), \qquad \beta = \theta(V, B) + \delta(V, B)$$

where $\delta(V, \cdot)$ is given by Lemma 2. We have thus reduced the determination of angles α and β to the computation of the left A and right B stops.

We might initially guess that the left stop of (V, W) is C, and right stop of (V, W) is C'. But the truth is a bit more subtle. Define the following points $X_*, X_{\max}$ on the positive x-axis using the equation:

$$\|OX_*\| = \sqrt{(\ell+\tau)^2 - \sigma^2}$$
$$\|OX_{\max}\| = \sqrt{\ell^2 - (\sigma-\tau)^2}$$

These two points are illustrated in Figure 6. Also, let $\overline{X}_*$ and $\overline{X}_{\max}$ be mirror reflections of X_* and $X_{\max}$ across the y-axis. The points $X_*, \overline{X}_*$ are the two points at distance $\ell + \tau$ from V. The points $X_{\max}, \overline{X}_{\max}$ are the left and right stops in we replace W by the infinite line through W (i.e., the x-axis).

With the natural ordering of points on the x-axis, we can show that

$$\overline{X}_* < \overline{X}_{\max} < O < X_{\max} < X_*$$

where O is the origin. Since $\|VC\|$ and $\|VC'\|$ lie in $(\tau, \tau+\ell)$, it follows that

$$\overline{X}_* < C' < C < X_*.$$

Two situations are shown in Figure 6. The next lemma is essentially routine, once the points $X_{\max}, \overline{X}_{\max}$ have defined:

[2] Intuitively: At configuration (V, α), the single-link robot can rotate about V to the right, but if it tries to rotate to the left, it is "stopped" by A.

Lemma 5. *Assume (3) and (4).*
The left stop of (V, W) *is*

$$\begin{cases} C' & \text{if } X_{\max} \le C' & (L1) \\ X_{\max} & \text{if } C' < X_{\max} < C & (L2) \\ C & \text{if } C \le X_{\max} & (L3) \end{cases}$$

The right stop of (V, W) *is*

$$\begin{cases} C & \text{if } C \le \overline{X}_{\max} & (R1) \\ \overline{X}_{\max} & \text{if } C' < \overline{X}_{\max} < C & (R2) \\ C' & \text{if } \overline{X}_{\max} \le C' & (R3) \end{cases}$$

The cases (L1-3) and (R1-3) in this lemma suggests 9 combinations, but 3 are logically impossible: (L1-R1), (L1-R2), (L2-R1). The remaining 6 possibilities for left and right stops are summarized in the following table:

	(R1)	(R2)	(R3)
(L1)	*	*	(C', C')
(L2)	*	$(X_{\max}, \overline{X}_{\max})$	$(X_{\max}, C')$
(L3)	(C, C)	$(C, \overline{X}_{\max})$	(C, C')

Observe the extreme situations (L1-R3) or (L3-R1) where the the left and right stops are equal to the same corner, and we are reduced to the point-point analysis.

Once we know the left and right stops for (V, W), then we can use Lemma 2 to calculate the angles α and β.

¶3. The Forbidden Zone of a Side and a Corner We now consider the forbidden zone $\mathrm{Forb}_{\ell,\tau}(S, C)$ where S is a side and C a corner feature. Note that is complementary to the previous case of $\mathrm{Forb}_{\ell,\tau}(V, W)$ since C and V are points and S and W are line segments. We can exploit the principle of reflection symmetry of Lemma 1:

$$\mathrm{Forb}_{\ell,\tau}(S, C) = \pi + \mathrm{Forb}_{\ell,\tau}(C, S)$$

where $\mathrm{Forb}_{\ell,\tau}(C, S)$ is provided by previous Lemma (writing C, S in place of V, W).

¶4. Cone Decomposition We have now provided formulas for computing sets of the form $\mathrm{Forb}_{\ell,\tau}(V, W)$ or $\mathrm{Forb}_{\ell,\tau}(S, C)$; such sets are called **cones**. We now address the problem of computing $\mathrm{Forb}_{\ell,\tau}(B^t, W)$ where $B^t \subseteq \mathbb{R}^2$ is a (translational) box. We show that this set of forbidden angles can be written as the union of at most 3 cones, generalizes a similar result in [9]. Towards such a cone decomposition, we first classify the disposition of a wall W relative to a box B^t. But there is a preliminary case: if W intersects $B^t \oplus D(0, \tau)$, then it is we see that

$$\mathrm{Forb}_{\ell}(B^t, W) = S^1.$$

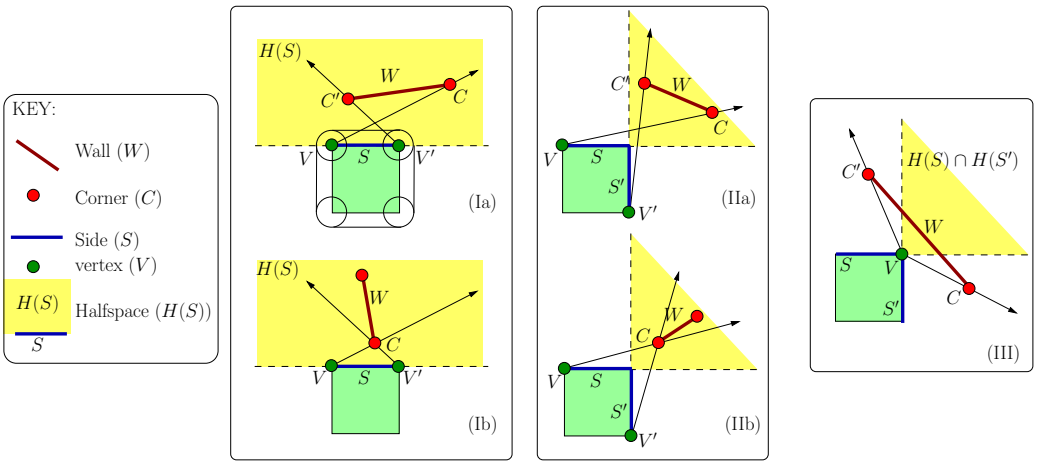


Fig. 7: Cases (I-III) of $\mathrm{Forb}_{\ell,\tau}(B^t, W)$

Call this **Case (0)**. Assuming W does not intersect $B^t \oplus D(0,\tau)$, there are three other possibilities, **Cases (I-III)** illustrated Figure 7.

We first need a notation: if $S \subseteq \partial(B^t)$ is a side of the box B^t, let $H(S)$ denote the open half-space which is disjoint from B^t and is bounded by the line through S. Then we have these three cases:

(I) $W \subseteq H(S)$ for some side s of box B^t.
(II) $W \subseteq H(S) \cap H(S')$ for two adjacent sides S, S' of box B^t.
(III) None of the above. This implies that $W \subseteq H(S) \cup H(S')$ for two adjacent sides S, S' of box B^t.

Theorem 1. $\mathrm{Forb}_{\ell,\tau}(B^t, W)$ *is the union of at most three thick cones.*

Sketch proof: we try to reduce the argument to the case $\tau = 0$ which is given in [9]. In that case, we could write

$$\mathrm{Forb}_{\ell}(B^t, W) = C_1 \cup C_2 \cup C_3$$

where each C_i is a thin cone or an empty set. In the non-empty case, the cone C_i has the form $\mathrm{Forb}_{\ell}(S_i, T_i)$ where $S_i \subseteq \partial B^t, T_i \subseteq W$. The basic idea is that we now "transpose" $\mathrm{Forb}_{\ell}(S_i, T_i)$ to the thick version $C_i' := \mathrm{Forb}_{\ell,\tau}(S_i, T_i)$. In case C_i is empty, C_i' remains empty. Thus we would like to claim that

$$\mathrm{Forb}_{\ell}(B^t, W) = C_1' \cup C_2' \cup C_3'.$$

This is almost correct, except for one issue. It is possible that some C_i is empty, and yet its transpose C_i' is non empty. In the full paper, we will fill in these detail.

Remark: in case of thin cones, the C_i's are non-overlapping (i.e., they may only share endpoints). But for thick cone decomposition, the cones will in general overlap.

4 Subdivision for Thick Non-Crossing 2-Link Robot

A resolution-exact planner for a thin self-crossing 2-link robot was described in [9]. We now extend that planner to the thick non-crossing case.

We will briefly review the ideas of the algorithm for the thin self-crossing 2-link robot. We begin with a box $B_0 \subseteq \mathbb{R}^2$ and it is in the subspace $B_0 \times \mathbb{T}^2 \subseteq C_{space}$ where our planning problem takes place. We are also given a polygonal obstacle set $\Omega \subseteq \mathbb{R}^2$; we may decompose its boundary $\partial\Omega$ into a disjoint union of corners (=points) and edges (=open line segments) which are called (boundary) **features**. Let $B \subseteq C_{space}$ be a box; there is an exact classification of B as $C(B) \in \{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$ relative to Ω. But we want a soft classification $\widetilde{C}(B)$ which is correct whenever $\widetilde{C}(B) \neq \texttt{MIXED}$, and which is equal to $C(B)$ when the width of B is small enough. Our method of computing $\widetilde{C}(B)$ is based on computing a set $\phi(B)$ of features that are relevant to B. A box $B \subseteq C_{space}$ may be written as a Cartesian product $B = B^t \times B^r$ of its translational subbox $B^t \subseteq \mathbb{R}^2$ and rotational subbox $B^r \subseteq \mathbb{T}^2$. In the T/R splitting method (simple version), we split B^t until the width of B^t is $\leq \varepsilon$. Then we do a single split of the rotational subbox B^r into all the subboxes obtained by removing all the forbidden angles determined by the walls and corners in $\widetilde{\phi}(B^t)$. This "rotational split" of B^r is determined by the obstacles, unlike the "translational splits" of B^t.

¶5. Boxes for Non-Crossing Robot. Our basic idea for representing boxes in the non-crossing configuration space $C_{space}(\kappa)$ is to write it as a pair $(B, \texttt{XT})$ where $\texttt{XT} \in \{\texttt{LT}, \texttt{GT}\}$, and $B \subseteq C_{space}$. The pair $(B, \texttt{XT})$ represents the set $B \cap (\mathbb{R}^2 \times \mathbb{T}^2_{\texttt{XT}})$ (with the identification $\mathbb{T}^2_{\texttt{LT}} = \mathbb{T}^2_<$ and $\mathbb{T}^2_{\texttt{GT}} = \mathbb{T}^2_>$). It is convenient to call $(B, \texttt{XT})$ an X**-box** since they are no longer "boxes" in the usual sense.

An angular interval $\Theta \subseteq S^1$ that[3] contains a open neighborhood of $0 = 2\pi$ is said to be **wrapping**. Also, call $B^r = \Theta_1 \times \Theta_2$ wrapping if either Θ_1 or Θ_2 is wrapping. Given any B^r, we can decompose the set $B^r \cap (\mathbb{T}^2 \setminus \Delta(\kappa))$ into the union of two subsets $B^r_{\texttt{LT}}$ and $B^r_{\texttt{GT}}$, where $B^r_{\texttt{XT}}$ denote the set $B^r \cap \mathbb{T}^2_{\texttt{XT}}$. In case B^r is non-wrapping, this decomposition has the nice property that each subset $B^r_{\texttt{XT}}$ is connected. For this reason, we prefer to work with non-wrapping boxes. Initially, the box $B^r = \mathbb{T}^2$ is wrapping. The initial split of $\mathbb{T}^2$ should be done in such a way that the children are all non-wrapping: the "natural" (quadtree-like) way to split $\mathbb{T}^2$ into four congruent children has[4] this property. Thereafter, subsequent splitting of these non-wrapping boxes will remain non-wrapping.

Of course, $B^r_{\texttt{XT}}$ might be empty, and this is easily checked: say $\Theta_i = [s_i, t_i]$ $(i = 1, 2)$. Then $B^r_<$ is empty iff $t_2 \leq s_1$. and $B^r_>$ is empty iff $s_2 \geq t_1$. Moreover, these two conditions are mutually exclusive.

[3] Wrapping intervals are either equal to S^1 or has the form $[s, t]$ where $2\pi > s > t > 0$.

[4] This is not a vacuous remark – the quadtree-like split is determined by the choice of a "center" for splitting. To ensure non-wrapping children, this center is necessarily $(0, 0)$ or equivalently $(2\pi, 2\pi)$. Furthermore, our T/R splitting method (to be introduced) does not follow the conventional quadtree-like subdivision at all.

We now modify the algorithm of [9] as follows: as long as we are just splitting boxes in the translational dimensions, there is no difference. When we decide to split the rotational dimensions, we use the T/R splitting method of [9], but each child is further split into two X-boxes annotated by `LT` or `GT` (they are filtered out if empty). We build the connectivity graph G (see Appendix A) with these X-boxes as nodes. This ensures that we only find non-crossing paths. Our algorithm inherits resolution-exactness from the original self-crossing algorithm.

The predicate $\texttt{isBoxEmpty}(B^r, \kappa, \texttt{XT})$ which returns true iff $(B^r_{\texttt{XT}}) \cap (\mathbb{T}^2 \setminus \Delta(\kappa))$ is empty is useful in implementation. It has a simple expression when restricted to non-wrapping translational box B^r:

Lemma 6.
Let $B^r = [a, b] \times [a', b']$ be a non-wrapping box.
(a) $\texttt{isBoxEmpty}(B^r, \kappa, \texttt{LT}) = \texttt{true}$ *iff* $\kappa \geq b' - a$ *or* $2\pi - \kappa \leq a' - b$.
(b) $\texttt{isBoxEmpty}(B^r, \kappa, \texttt{GT}) = \texttt{true}$ *iff* $\kappa \geq b - a'$ *or* $2\pi - \kappa \leq a - b'$.

5 Implementation and Experiments

We implemented our thick non-crossing planner in `C++` and OpenGL on the Qt platform (the Qt part is new). A preliminary heuristic version appeared [9, 8]. Our code, data and experiments are distributed[5] with our open source `Core Library`. To evaluate our planner, we compare it with several sampling algorithms in the open source OMPL [11]. The platform for our experiments is Mac OS X 10.10.5 (Yosemite) on MacBook Pro (Mid 2015). The processor is a 2.5 GHz Intel Core i7 with 16GB DDR3-1600 MHz RAM and 500GB Flash Storage. Details about each of these experiments are found in a folder in Core Library for this paper, and should be reproducible from the data there. This includes our configurations for OMPL.

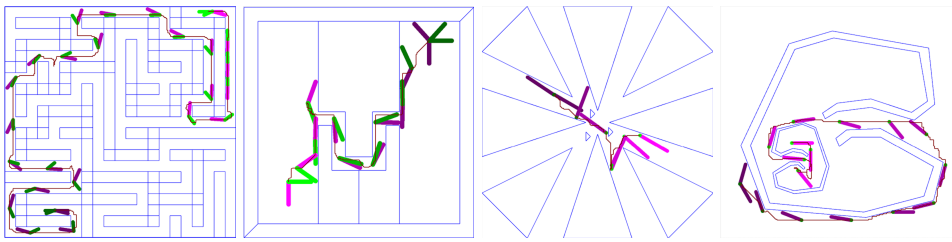
Two preliminary remarks are in order before discussing Table 1. Our planner shares some of the code base for the SSS framework begun in [12]. Similar to previous work, our planner can choose one of several search strategies; for simplicity, below we will consistently use the Greedy Best First (GBF) strategy, which is generally one of our best strategies. Next, OMPL does not natively support articulated robots such as R_2. So for the experiments in Table 1, we artificially set $\ell_2 = 0$ so that it is effectively a one-link thick robot; Table 2 will discuss the consequences of this artifice. Our planner, unlike those in OMPL, needs an ε parameter; we fix $\varepsilon = 2$ in Table 1. On the other hand, OMPL has various tuning parameters, but we choose the default.

Each row of Table 1 represents an experiment, which is specified by an environment, robot parameters and initial and goal configurations. For reference,

[5] http://cs.nyu.edu/exact/core/download/core/.

Environment (ℓ_1, τ)	Record Statistics (Avg./Best/STD/Success)	Ratios Relative to Record Statistics				
		Ours	RRT	PRM	BIT*	EST
T-Room (100, 10)	186.91/27.07/6.57/1	1/6.56/1/1	3.47/1/68.91/1	1.58/1.15/41.80/0.99	5.43/37.02/2.63/1	2.20/2.25/43.13/1
Maze (30, 4)	988.08/935.66/3.53/1	1/1/82.71/1	1.02/1.07/1/0.3	1.03/1.07/3.14/0	1.02/1.07/2.89/0	1.02/1.07/1.14/0
100 Triangles (100, 8)	867.98/.92/3.37/1	1/8.27/16.24/1	1.04/1/72.97/1	1.08/3.56/52.65/0.99	1.16/9.92/1/1	1.01/1.26/61.96/0.96
Narrow Passage (50, 10)	187.54/175.72/5.24/1	1/1/3.80/1	5.40/5.71/1.07/0	5.54/5.71/5.12/0	5.40/5.70/1.08/0	5.39/5.70/1/0
8-Ways (50, 10)	277.01/265.16/3.40/1	1/1/4.23/1	3.64/3.78/1.07/0	3.69/3.79/3.42/0.02	3.69/3.78/5.29/0.01	3.64/3.78/1/0
Double Bug Trap-A (45, 8)	6.72/739.50/3.04/1	1.64/2.15/9.32/1	1/1/1/0	1/1.35/1.33/0	1.01/1.35/2.34/0	1/1.35/1.08/0
Double Bug Trap-B (70, 8), No-Path	336.386/314.36/2.87/1	1/1/7.49/1	3/3.19/1.21/0	3.01/3.19/2.14/0	3.03/3.19/3.34/0	3/3.19/1/0

Table 1: Comparison with Sampling-based Planners in OMPL



(a) Maze (b) Narrow Passage (c) 8-Way Corridor (d) Double Bugtrap

Fig. 8: Some environments in our experiments

the experiment is named after the environment. Figure 8 depicts[6] all the environments which we have not met earlier. The robot parameters are the length ℓ_1 and thickness τ of the first link (see Column 1). E.g., $\ell_1 = 30, \tau = 4$ in the Maze experiment. For each experiment, we perform 100 runs of the following planners: RRT, PRM, BIT*, EST (all in OMPL) and ours. Each planner produces these 4 statistics:

Average Time / Best Time / Standard Deviation / Success Rate.

These 4 statistics are abbreviated as **Avg**/**Best**/**STD**/**Success**, respectively. Success Rate is the fraction of the 100 runs for which the planner finds a path (assuming there is one) out of 100 runs. But if there is no path, our planner will always discover this, so its **Success** is 1; simultaneously, the sampling methods will time out and hence their **Success** is 0. All our time is in milliseconds (ms). Column 2 contains the **Record Statistics**, i.e., the row optimum for these 4 statistics. For instance, the Record Statistics for the T-Room experiment is **186.91/27.07/6.57/1**. This tells us the row optimum for **Avg** is 186.91 ms,

[6] These environments are described in [9].

for **Best** is 27.06 ms, for **STD** is 6.57 ms, and for **Success** is 1. Note that "optimum" for the first 3 statistics (resp., last statistic) means minimum (resp., maximum) value. These 4 optimal values may be achieved by different planners. In the rest of the Table, we have one column for each Planner, showing the ratio of the planner's statistics relative to the Record Statistics of that row. The best performance is always indicated by the ratio of 1. The ratio for **Avg**/**Best**/**STD** (resp., **Success**) is ≥ 1 (resp., ≤ 1). E.g., for T-Room experiment, the row maximum for **Success** is 1, and it is achieved by every planner. The row minimum for **Avg** and **STD** are achieved by Our planner, but the row minimum for **Best** is achieved by RRT.

We regard the achievement of row optimum for **Success** and **Avg** (in that order) to be the main indicator of superiority. In this sense, Table 1 shows that our planner is consistently superior to the sampling based planners, with one exception: in the Double Bug Trap, RRT has minimum **Avg**, but we are only 1.64 times slower. In the final paper, we expect to remedy the lack of an articulated planner in OMPL in order to give a more accurate picture.

Environment (ℓ_1, τ)	Record Statistics (Avg./Best/STD/Success)	Ratios Relative to Record Statistics		
		Ours (0) (ℓ_2: 0)	Ours (I) (ℓ_2: 20, 20, 10, 10, 10, 5, 5)	Ours (II) (non-crossing)
T-Room (100, 10)	186.91/177.62/6.57/1	1/1/1/1	1.04/1.04/1/1	1.7/1.66/3.35/1
Maze (30, 4)	988.08/935.66/291.89/1	1/1/1/1	1.32/1.33/1.03/1	1.85/1.83/1.01/1
100 Triangles (100, 8)	867.98/834.43/54.74/1	1/1/1/1	1.1/1.03/1.31/1	1.3/1.3/1.21/1
Narrow Passage (50, 10)	187.54/175.72/17.63/1	1/1/1.13/1	1.04/1.05/1/1	1.39/1.4/1/1
8-Ways (50, 10)	277.01/265.16/12.97/1	1/1/1.11/1	1.04/1.04/1.16/1	1.28/1.28/1/1
Double Bug Trap-A (45, 8)	1636.96/1584.13/30.70/1	1.01/1.01/1.01/1	1/1/1.03/1	1.4/1.4/1/1
Double Bug Trap-B (70, 8), No-Path	334.21/322.29/6.27/1	1/1/1/1	1.03/1.02/1.75/1	1.1/1.1/1.17/1

Table 2: Comparing the effects of Crossing and Non-Crossing Planners

Table 2 shows the effects of increasing ℓ_2 starting from 0, and the effects of non-crossing constraint. We see that they are negligible in the experiments (the slow down in time is always less than 2), but the largest ratio in the entire table is the value of 3.35 for **STD** (this large value appears to be a fluke since our algorithm is deterministic, and the only randomness comes from background processes). In other words, the performance of our planner in Table 1 has not been much affected by the fact that we set $\ell_2 = 0$. There are three version of our

algorithm with fixed $\varepsilon = 2$. In Ours(0), the length of the second link is $\ell_2 = 0$, so this is effectively a single link. In Ours(I), we choose ℓ_2 to be 20 (resp., 20, 10, 10, 10, 5, 5) for the first (resp., second to seventh) experiment. In Ours(II), the length of ℓ_2 are as in Ours(I) except that the links are non-crossing ($\delta = 0$).

Environment (ℓ_1, τ)	Record Statistics (Avg./Best/STD/Success)	Ratios Relative to Record Statistics						
		Ours (0) (ℓ_2: 0)	Ours (I) (ℓ_2: 20, 20, 10, 10, 10, 5, 5)	Ours (II) (non-crossing)	RRT	PRM	BIT*	EST
T-Room (100, 73)	490.65/406.67/6.36/1	1/1.17/1.76/1	1.18/1.4/1/1	1.79/2.11/1.95/1	37.95/1/2204.95/0.4	49.94/4.28/1271.43/0.45	61.28/73.79/5.68/1	41.09/7.1/1691.84/0.5
Maze (30, 11)	2976.38/2940.97/20.79/1	1/1/1/1	1.35/1.35/1.11/1	1.95/1.91/7.49/1	9.97/9.34/40.9/0.2	9.29/4.04/236.96/0.3	10.11/10.21/2.1/1	10.11/10.2/2.1/0
100 Triangles (100, 22)	13389.28/1731.96/24.90/1	1/7.6/7.15/1	1.04/7.97/3.26/1	1.17/8.93/11.54/1	2.25/17.33/1/0	1.84/1/392.52/0.35	2.24/17.32/1.66/0.6	2.24/17.33/1.27/0
Narrow Passage (50, 12)	664.46/640.19/15.96/1	1/1/1.68/1	1.09/1.07/2.61/1	1.41/1.43/1/1	45.26/46.89/2.32/0	45.28/46.87/2.78/0	45.24/46.87/2.15/0	45.24/46.89/1.49/0
8-Ways (70, 11)	3566.80/3503.01/20.32/1	1/1/3.05/1	1.02/1.03/1.06/1	1.43/1.42/3.16/1	8.44/8.57/1.32/0	8.43/8.57/2.47/0	8.43/8.57/1/0.05	8.43/8.57/1.53/0
Double Bug Trap-A (45, 9)	6116.03/6049.1/29.94/1	1/1/1.93/1	1.02/1.02/1/1	1.48/1.47/4.57/1	4.92/4.96/1.25/0	4.91/4.96/1.5/0	4.92/4.96/2.11/0	4.92/4.96/1.27/0
Double Bug Trap-B (70, 9), No-Path	1314.12/1255.47/23.51/1	1/1/1.15/1	1/1.01/1/1	1.08/1.09/1.29/1	22.87/23.9/1.19/0	22.88/23.9/1.57/0	22.98/23.93/3.44/0	22.86/23.9/1.26/0

Table 3: Narrow Passages

Table 3 illustrates the "narrow passage" problems for the sampling approaches: in each experiment, we choose the largest (integer) thickness τ for which there is a path; in other words, if we increase the thickness to $\tau + 1$, our program will report no path. In fact, we include a case (the last row, Double Bug Trap-B) for which there is no path. Again, we have three version of our algorithm with fixed $\varepsilon = 1$. The fair comparison would be Ours(0) against the Sampling methods, because this runs the single link robot, but we perform better than Sampling methods even with Ours(I) and Ours(II).

Instead of using default 1 second time-out, we set time-out to be 30 seconds. We see that BIT* (resp., EST) has the best (worst) success rate among the Sampling methods. BIT* was able to find paths for all, with the exception of Narrow Passage and Double Bug Trap-A. EST could not find paths for any, with the exception of T-Room.

6 Conclusion and Limitations

The introduction of non-crossing link robots is theoretically novel, and points the way for many similar extensions. Our work is a contribution to the development of practical and theoretically sound subdivision planners [12, 13].

One might expects a tradeoff between the stronger guarantees of subdivision approaches versus the faster performance of sample approaches. But our experiments suggest no such tradeoffs since subdivision is consistently faster than sampling. Of course, our conclusions are preliminary because the experimental

settings for the two approaches are far from ideal. Also, we accepted all the default parameters of OMPL; perhaps tuning these parameters would greatly improve their performance. We expect to remedy some of this in the final paper.

Moreover, conventional wisdom maintains that subdivision will not scale to higher DOF's, and our current experiments are limited to at most 4DOF. But we interpret this wisdom as telling us that new subdivision techniques (such as the T/R splitting idea) are needed to make higher DOF's robots perform in real-time. It is a nontrivial but worthy challenge.

References

1. M. Barbehenn and S. Hutchinson. Toward an exact incremental geometric robot motion planner. In *Proc. Intelligent Robots and Systems 95.*, volume 3, pages 39–44, 1995. 1995 IEEE/RSJ Intl. Conf., 5–9, Aug 1995. Pittsburgh, PA, USA.
2. R. A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. 8th IJCAI – Vol. 2*, pp. 799–806, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.
3. 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.
4. S. L. Devadoss and J. O'Rourke. *Discrete and Computatational Geometry*. Princeton University Press, 2011.
5. L. Kavraki, P. Švestka, C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation*, 12(4):566–580, 1996.
6. J. J. Kuffner Jr and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.
7. S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
8. Z. Luo. Resolution-exact Planner for a 2-link planar robot using Soft Predicates. Master thesis, New York Univ., Jan. 2014. Department's Master Thesis Prize 2014.
9. 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)*, vol. 107 of *Springer Tracts in Advanced Robotics* , pp. 353–370, 2015.
10. 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.
11. I. Şucan, M. Moll, and L. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Auto. Magazine*, 19(4):72–82, 2012. `http://ompl.kavrakilab.org`.
12. C. Wang, Y.-J. Chiang, and C. Yap. On Soft Predicates in Subdivision Motion Planning. In *29th ACM Symp. on Comp. Geom.*, pages 349–358, 2013. SoCG'13, Rio de Janeiro, Brazil, June 17-20, 2013. Journal version in Special Issue of *CGTA*.
13. C. K. Yap. Soft Subdivision Search in Motion Planning. In A. Aladren et al., editor, *Proceedings, 1st Workshop on Robotics Challenge and Vision (RCV 2013)*, 2013. A Computing Community Consortium (CCC) **Best Paper Award**, Robotics Science and Systems Conference (RSS 2013), Berlin. In arXiv:1402.3213.
14. C. K. 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.

15. J. Yu, C. Yap, Z. Du, S. Pion, and H. Bronnimann. Core 2: A library for Exact Numeric Computation in Geometry and Algebra. In *3rd Proc. Int'l Congress on Mathematical Software (ICMS)*, pp. 121–141. Springer, 2010. LNCS No. 6327.
16. D. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991.

APPENDIX A: Elements of Soft Subdivision Search

We review the the notion of soft predicates and how it is used in the SSS Framework. See [12, 13, 9] for more details.

¶6. Soft Predicates. The concept of a "soft predicate" is relative to some exact predicate. Define the exact predicate $C : C_{space} \to \{0, +1, -1\}$ where $C(x) = 0/+1/-1$ (resp.) if configuration x is semi-free/free/stuck. The semi-free configurations are those on the boundary of C_{free}. Call $+1$ and -1 the **definite values**, and 0 the **indefinite value**. Extend the definition to any set $B \subseteq C_{space}$: for a definite value v, define $C(B) = v$ iff $C(x) = v$ for all x. Otherwise, $C(B) = 0$. Let $\square(C_{space})$ denote the set of d-dimensional boxes in C_{space}. A predicate $\widetilde{C} : \square(C_{space}) \to \{0, +1, -1\}$ 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 $B_i \to p \in C_{space}$ as $i \to \infty$, then $\widetilde{C}(B_i) = C(p)$ 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 $C(B)$ is definite, then $\widetilde{C}(B/\sigma)$ is definite.

¶7. The Soft Subdivision Search Framework. An SSS algorithm maintains a subdivision tree $\mathcal{T} = \mathcal{T}(B_0)$ rooted at a given box B_0. Each tree node is a subbox of B_0. We assume a procedure $\texttt{S}split(B)$ that subdivides a given leaf box B into a bounded number of subboxes which becomes the children of B in $\mathcal{T}$. Thus B is "expanded" and no longer a leaf. For example, $\texttt{S}split(B)$ might create 2^d congruent subboxes as children. Initially $\mathcal{T}$ has just the root B_0; we grow $\mathcal{T}$ by repeatedly expanding its leaves. The set of leaves of $\mathcal{T}$ at any moment constitute a subdivision of B_0. Each node $B \in \mathcal{T}$ is classified using a soft predicate $\widetilde{C}$ as $\widetilde{C}(B) \in \{\texttt{MIXED}, \texttt{FREE}, \texttt{STUCK}/\} = \{0, +1, -1\}$. Only `MIXED` leaves with radius $\geq \varepsilon$ are candidates for expansion. We need to maintain three auxiliary data structures:

- A priority queue Q which contains all candidate boxes. Let Q.`GetNext`() remove the box of highest priority from Q. The tree $\mathcal{T}$ grows by splitting Q.`GetNext`().
- A **connectivity graph** G whose nodes are the `FREE` leaves in $\mathcal{T}$, and whose edges connect pairs of boxes that are adjacent, i.e., that share a $(d-1)$-face.
- A Union-Find data structure for connected components of G. After each $\texttt{S}split(B)$, we update G and insert new `FREE` boxes into the Union-Find data structure and perform unions of new pairs of adjacent `FREE` boxes.

Let $Box_{\mathcal{T}}(\alpha)$ denote the leaf box containing α (similarly for $Box_{\mathcal{T}}(\alpha)$). The SSS Algorithm has three WHILE-loops. The first WHILE-loop will keep splitting $Box_{\mathcal{T}}(\alpha)$ until it becomes `FREE`, or declare NO-PATH when $Box_{\mathcal{T}}(\alpha)$ has radius less than ε. The second WHILE-loop does the same for $Box_{\mathcal{T}}(\beta)$. The third WHILE-loop is the main one: it will keep splitting Q.`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. Thus we get:

SSS Framework:
Input: Configurations α, β, tolerance $\varepsilon > 0$, box $B_0 \in C_{space}$.
Initialize a subdivision tree $\mathcal{T}$ with root B_0.
Initialize Q, G and union-find data structure.
1. While ($Box_{\mathcal{T}}(\alpha) \neq$ `FREE`)
   If radius of $Box_{\mathcal{T}}(\alpha)$) is $< \varepsilon$, Return(NO-PATH)
   Else `S`$split(Box_{\mathcal{T}}(\alpha))$
2. While ($Box_{\mathcal{T}}(\beta) \neq$ `FREE`)
   If radius of $Box_{\mathcal{T}}(\beta)$) is $< \varepsilon$, Return(NO-PATH)
   Else `S`$split(Box_{\mathcal{T}}(\beta))$
▷ *MAIN LOOP:*
3. While ($Find(Box_{\mathcal{T}}(\alpha)) \neq Find(Box_{\mathcal{T}}(\beta))$)
   If $Q_{\mathcal{T}}$ is empty, Return(NO-PATH)
   $B \leftarrow Q_{\mathcal{T}}$.`GetNext`()
   `S`$plit(B)$
4. Generate and return a path from α to β using G.

The correctness of our algorithm does not depend on how the priority of Q is designed. See [13] for the correctness of this framework under fairly general conditions.