Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Toward Certifiable Motion Planning for Medical Steerable Needles

Mengyu Fu*, Oren Salzman†, and Ron Alterovitz*
*Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA
Email: {mfu,ron}@cs.unc.edu
†Computer Science Department, Technion - Israel Institute of Technology, Israel
Email: osalzman@cs.technion.ac.il

***Abstract*—Medical steerable needles can move along 3D curvilinear trajectories to avoid anatomical obstacles and reach clinically significant targets inside the human body. Automating steerable needle procedures can enable physicians and patients to harness the full potential of steerable needles by maximally leveraging their steerability to safely and accurately reach targets for medical procedures such as biopsies and localized therapy delivery for cancer. For the automation of medical procedures to be clinically accepted, it is critical from a patient care, safety, and regulatory perspective to certify the correctness and effectiveness of the motion planning algorithms involved in procedure automation. In this paper, we take an important step toward creating a certifiable motion planner for steerable needles. We introduce the first motion planner for steerable needles that offers a guarantee, under clinically appropriate assumptions, that it will, in finite time, compute an exact, obstacle-avoiding motion plan to a specified target, or notify the user that no such plan exists. We present an efficient, resolution-complete motion planner for steerable needles based on a novel adaptation of multi-resolution planning. Compared to state-of-the-art steerable needle motion planners (none of which provide any completeness guarantees), we demonstrate that our new resolution-complete motion planner computes plans faster and with a higher success rate.**

I. Introduction

Steerable needles are highly flexible medical devices able to follow 3D curvilinear trajectories inside the human body, reaching clinically significant targets while safely avoiding critical anatomical structures [3, 12, 39, 52]. Compared with traditional rigid medical instruments, steerable needles can reduce a patient's trauma, increase safety, and provide minimally invasive access to previously inaccessible targets. Steerable needles have been considered for a wide range of diagnostic and treatment procedures including biopsy, drug therapy delivery, and radioactive seed implantation for cancer treatment [2].

Automating steerable needle procedures can enable physicians and patients to harness the full potential of steerable needles by maximally leveraging their steerability and ability to accurately and precisely reach targets. Automation is critical to harnessing the full potential of these needles since the nonholonomic constraints on the needle's 3D motion coupled with the cluttered nature of anatomical environments make direct manual control unintuitive and impractical for human operators. To automate steerable needle procedures, physicians first obtain a medical image (such as a computed tomography (CT) or magnetic resonance imaging (MRI) scan) of the relevant anatomy, from which we can segment (manually or automatically) the relevant anatomy, including the target to reach and obstacles to avoid. The next key ingredient to the automation of steerable needle procedures is motion planning, which requires computing feasible motions to steer the needle safely around the anatomical obstacles and to the target. An example scenario of a lung biopsy is shown in Fig. 1 (top).

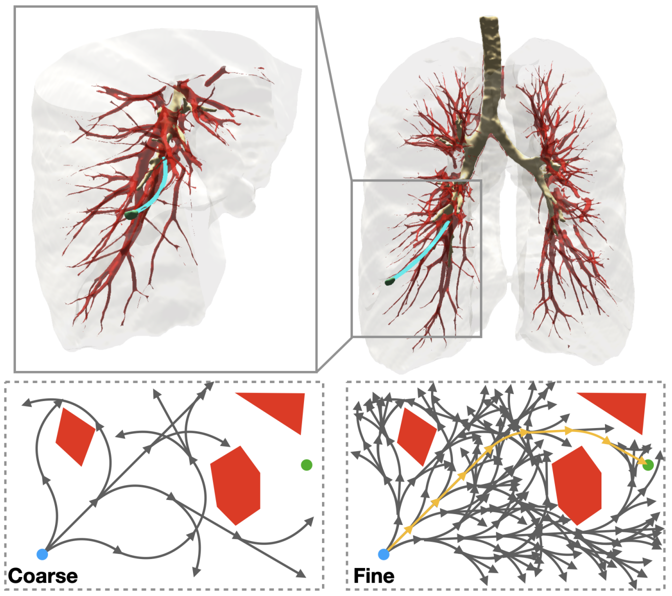


Fig. 1. **Top:** A medical steerable needle (cyan) used to reach a nodule (green) in the lung parenchyma for biopsy or cancer treatment while avoiding critical anatomical structures such as the bronchial tubes (brown) and major blood vessels (red). **Bottom:** Our resolution-complete motion planner uses search trees built using different resolutions, illustrated here in 2D. A valid motion plan goes from the start configuration (blue dot) to the goal point (green dot), while avoiding obstacles (red) and satisfying kinematic constraints. The left search tree uses a coarse resolution and failed to find a plan while the right one uses a finer resolution and successfully generated a motion plan (yellow).

For the automation of medical procedures to be clinically accepted, it is critical from a patient care, safety, and regulatory perspective to certify the correctness and effectiveness of the algorithms involved in procedure automation. Unfortunately, *no previously developed motion planner for steerable needles offers a formal guarantee that it will compute a solution, when one exists, in finite time, or notify the user that no solution exists*. Although many steerable needle motion planners have been proposed, for all prior methods, the method either is not guaranteed to return a solution (e.g., [15, 16, 20, 40, 47, 51, 53]) or is not guaranteed to find a solution within a clinically reasonable distance of the target [34] when a solution exists.

As an important step toward creating a certifiable motion planner for steerable needles, we introduce the first motion planner for steerable needles that enables us to offer a guarantee under clinically appropriate assumptions that it can, in finite time, compute an exact, obstacle-avoiding motion plan to a specified target, or notify the user that no such plan exists. In motion planning, such a guarantee is defined as *completeness* [32]. A motion planner that lacks a completeness guarantee may find solutions for only a subset of problem instances, and when no solution is found by the planner, a user has no way to distinguish whether the planner is incapable of finding an existing solution or if no solution exists.

Providing a completeness guarantee for a steerable needle motion planner is challenging in part because motion planning for steerable needles in 3D with curvature constraints is at least NP-hard [26, 48]. This challenge inspires us to consider variants of completeness relevant to medical applications. We note that some variants of completeness that only offer asymptotic guarantees, such as probabilistic completeness [32], are not useful for needle steering since they only are guaranteed to find a solution as computation time increases to infinity, but medical applications typically require guaranteeing the planner's behaviour within a finite time.

In this paper, we focus on a specific type of completeness relevant to real-world medical applications: resolution completeness [32]. Generally speaking, a resolution characterizes the discretization of some space (e.g., state space, configuration space, action space, and time). An algorithm is resolution complete if there exists a fine-enough resolution with which the algorithm finds a plan in finite time when a qualified solution exists, and otherwise correctly returns that no such plan exists. We illustrate at the bottom of Fig. 1 an example showing searches with different resolutions for needle steering.

In this work, we present an efficient, resolution-complete motion planner for steerable needles based on a novel adaptation of multi-resolution planning. The planner is resolution complete, which means under some mild assumptions on the system and the solution (detailed in Sec. V and Appendix A), the planner, in finite time, is guaranteed to find a motion plan as long as the problem admits a qualified solution. Our main contributions include: (i) carefully defining the motion primitives [17] used by our planner which are specifically tailored to our domain of 3D steerable needles (Sec. IV-B); (ii) introducing a set of domain-specific optimizations that improve the efficiency of the algorithm while maintaining resolution completeness (Sec. IV-G); and (iii) providing a proof sketch to show the resolution completeness of our method (Sec. V and Appendix A).

We demonstrate the performance of our planner in scenarios based on lung biopsy. In these scenarios, a steerable needle is deployed through a bronchoscope and must steer through the lung parenchyma (the substance of the lung outside the bronchial tubes) to a target while safely avoiding obstacles (e.g., blood vessels). We compare in simulation our planner with two existing steerable needle planners—one is sampling based while the other is search based. Not only does our motion planner provide a resolution-completeness guarantee, but compared to prior work it also computes plans of comparable quality, faster, and with a higher success rate.

II. Related Work

Steerable needles have many different designs, including bevel-tip flexible needles [52, 12], symmetric-tip needles [13], needles with curved stylet tips [38], needles with tendon-actuated tips [43], and programmable bevel-tip needles [28, 46]. In this paper, we focus on bevel-tip flexible needles but our approach can be easily used in any mechanical design as long as the major kinematic constraint to consider is the curvature of the needle trajectory.

A. Motion planning for steerable needles

Early work studied planning and control for steerable needles in the 2D plane [4, 6, 10, 44]. To fully utilize the capability of steerable needles, later work began to focus more on needle steering in 3D environments. Duindam et al. [15] used inverse kinematics for planning but the planner was tested only with simple geometrically shaped obstacles and provides no theoretical guarantees.

Other planners built upon the probabilistic completeness guarantees of sampling-based methods such as the Rapidly-exploring Random Tree (RRT) [31]. Xu et al. [53] used an RRT variant for needle steering but showed low efficiency in computing time. Patil et al. [40] developed an RRT-based needle planner that guides the tree expansion by sampling in the 3D workspace (instead of the configuration space). The efficiency obtained by sampling in the workspace and not accounting for the needle's orientation makes the planner extremely fast in practice. Unfortunately, this makes the completeness proofs of the original RRT inapplicable and probabilistic completeness is not guaranteed.

To avoid dealing with curvature constraints directly in the RRT algorithms, there are also hybrid methods that combine sampling and other techniques. Favaro et al. [16] proposed a method that uses RRT* [25] that builds a tree embedded in the 3D workspace to generate candidate plans of low cost, followed by a smoothing step to account for the curvature constraint. However, this decoupled approach does not provide any theoretical guarantees.

Liu et al. proposed the Adaptive Fractal Tree (AFT) [34] for needle steering and used a Graphics Processing Unit (GPU) to further speed up their algorithm. The method uses a greedy approach for path refinement—it iteratively uses the lowest-cost path in the previous iteration for plan refinement. However, expanding the best path of a coarse resolution does not necessarily lead to a best path of a finer resolution. Furthermore, the authors use a cost function consisting of three factors, only one of which is the distance to the goal, also known as the targeting error. Thus, when provided with a required targeting error, paths produced by the method are not guaranteed to adhere to this constraint since the targeting error may be sacrificed for a better cost for the other two terms. Pinzi et al. [41] later extended AFT to account for goal orientation constraints.

Other methods focus on accounting for uncertainty during needle insertion but do not account for completeness [20, 47, 51, 49]. To summarize, to the best of the authors' knowledge, none of the existing steerable needle planners provide provable guarantees on the planner's completeness.

B. Resolution-complete motion planners

Generally speaking, an algorithm is *resolution complete* if it generates a plan to the goal whenever a solution exists at the maximal resolution and returns failure otherwise [7]. This property guarantees that given a predefined maximal resolution, the algorithm terminates in finite time and provides a deterministic result.

Barraquand et al. [8] proposed a planner for single/multi-body mobile robots with nonholonomic constraints. They formally proved the planner is guaranteed to generate a solution path when the discretization of the search parameters is fine enough. This approach was later extended by Lindemann and LaValle [33] to suggest a multi-resolution approach for 2D car-like robots. Both these works [8, 33] serve as the algorithmic foundations to the planner we present in this paper.

Sampling-based planners (such as RRT) typically ensure probabilistic completeness (i.e., such a planner is guaranteed to find a solution, if one exists, with probability one when given infinite time). However, they can also be used to build resolution-complete planners given some mild assumptions on the minimal motion that the system can perform. Cheng et al. [11] proposed a resolution-complete version of RRT for systems that satisfy the Lipschitz condition. Yershov et al. [54] formally analyzed the system conditions for the existence of resolution-complete planners. Kleinbort et al. [27] later analyzed the assumptions for RRT's probabilistic completeness in kinodynamic planning. However their analysis can be adapted to resolution-completeness guarantees.

Ljungqvist et al. [35] proposed a planner for a general two-trailer system in 2D. They used a two-point boundary value problem (2pBVP) solver to generate a set of motion primitives connecting 2D grid points. Their planner is resolution-complete and resolution-optimal with respect to the resolution in the configuration space, which means the planner generates a plan with minimal cost among all solutions that can be represented as a sequence of motion primitives. Most of the above-mentioned planners can be used to plan for 2D nonholonomic robots. However, none account explicitly for the challenges of planning with curvature constraints in 3D, where the dimension of the search space is higher and there is no efficient 2pBVP solver. In this work, we provide a planner for 3D needle steering that is both efficient in practice and is guaranteed to be resolution complete.

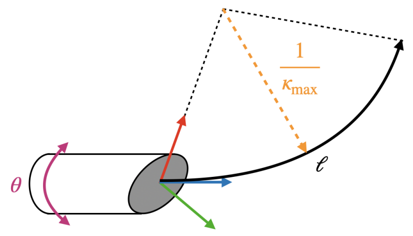


Fig. 2. The kinematics of a bevel-tip steerable needle. The needle can be inserted (characterized by ℓ) and axially rotated at its base (characterized by θ).

III. Problem Definition

In this work, we consider steerable needles that operate in a 3D workspace $\mathcal{W} \in \mathbb{R}^3$, which is cluttered with obstacles $\mathcal{W}_{\text{obs}} \subset \mathcal{W}$. We define the configuration space (or C-space) of the steerable needle as $\mathcal{X} \subset \mathcal{SE}(3)$. Each configuration $\mathbf{x} = (p, q) \in \mathcal{X}$ uniquely defines the pose (i.e., position $p \in \mathbb{R}^3$ and orientation $q \in \mathcal{SO}(3)$) of the needle tip. We define a projection function $\text{Proj}(\cdot) : \mathcal{X} \to \mathcal{W}$ that projects configurations to points in the workspace, i.e., $\text{Proj}(\mathbf{x}) = p$. A configuration $\mathbf{x}$ is *collision free* if $\text{Proj}(\mathbf{x}) \notin \mathcal{W}_{\text{obs}}$, and is *in collision* otherwise. The union of all collision-free configurations is denoted as $\mathcal{X}_{\text{free}}$. Since we assume the needle shaft perfectly follows its tip, a motion plan of the needle can be uniquely defined as a trajectory $\sigma : [0, 1] \to \mathcal{X}$. And such a motion plan σ is *collision free* if all configurations along the trajectory are collision free. Namely, $\forall s \in [0, 1], \sigma(s) \in \mathcal{X}_{\text{free}}$.

We also need to consider the kinematics of the steerable needle. We specifically consider steerable needles that are highly flexible and have an asymmetric tip (e.g., a bevel) [3, 12, 39, 52]; the asymmetric tip exerts asymmetric forces on the tissue in front of the needle tip, and the high flexibility enables the needle to curve substantially at maximum curvature $\kappa_{\max}$ as it moves through the tissue. Furthermore, rotating the needle axially at its base changes the direction of the needle's asymmetric tip, enabling the needle to change its direction of steering. See Fig. 2 for an illustration.

We say a motion plan is (kinematically) *feasible* if it never exceeds the maximum curvature $\kappa_{\max}$. A *valid* motion plan for the needle is both collision free and feasible. We also assume there exists a resolution describing the smallest interval or precision of the achievable motions, which may be limited by the physical system's hardware (e.g., motor, encoders, controller, etc.) and its interaction with the environment. In this paper, we determine this finest resolution by considering the hardware's ability to measurably change the steerable needle tip's position and orientation in tissue. Considering real-world effects such as torsional wind up of the needle shaft during actuation, the control resolution of the needle tip is coarser than the control resolution of the needle base where motors directly apply controls. Thus, we are not using minimal motions of the motors. Instead, we consider the minimal motions the tip of the needle can perform. We assume there exists a lower-level controller taking care of controlling the tip to the desired pose, as is common in needle steering systems.

We are now ready to state the steerable needle motion planning problem.

Problem 1. *A steerable needle motion planning problem is defined as the tuple $\Delta = (\mathcal{X}, \mathcal{W}_{\text{obs}}, \mathbf{x}_{\text{start}}, p_{\text{goal}}, \tau, \ell_{\max}, \kappa_{\max})$, where $\mathcal{W}_{\text{obs}}$ is the obstacle set, $\mathbf{x}_{\text{start}}$ is the start configuration, $p_{\text{goal}} \in \mathcal{W}$ is the goal point, $\tau > 0$ is the goal tolerance, $\ell_{\max}$ is the maximum insertion length, and $\kappa_{\max}$ is the maximum curvature. The problem calls for computing a valid motion plan σ that satisfies: (i) $\sigma(0) = \mathbf{x}_{\text{start}}$, (ii) the Euclidean norm $\|\text{Proj}(\sigma(1)) - p_{\text{goal}}\|_2 \leq \tau$, and (iii) trajectory length $\ell(\sigma) \leq \ell_{\max}$.*

As we show in our later discussion (in Sec. V and Appendix A), for any given instance of Problem 1, under some mild assumptions, there exists some fine-enough resolution $R_{\min} = \{\delta\ell_{\min}, \delta\theta_{\min}\}$ (corresponding to the needle's insertion and axial rotation, respectively) for which our planner is guaranteed to find a solution in finite time (when one exists) or to indicate that no solution exists.

IV. Method

A. Overview

Our needle planner builds a search tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ embedded in the C-space with $\mathbf{x}_{\text{start}}$ as its root. Each node $v \in \mathcal{V}$ is associated with a configuration $\mathbf{x}_v \in \mathcal{X}$, and each edge $e = (v, u) \in \mathcal{E}$ represents the transition from $\mathbf{x}_v$ to $\mathbf{x}_u$. To expand a node $v \in \mathcal{V}$, we construct new nodes (children of v) with *motion primitives* (to be explained shortly in Sec. IV-B), which are pre-defined feasible motions. A child node v_{child} is accepted and added to the search tree if the trajectory from v to v_{child} is collision-free and v_{child} is valid (will be detailed in Sec. IV-D). The search tree grows until there is some node v with configuration $\mathbf{x}_v$ whose projection is inside the τ-neighborhood of p_{goal} (condition (ii) in Problem 1).

A key aspect of our search method (which is similar in nature to other search-based planners [33]) is to use a set of motion primitives defined using *multiple* resolutions. Instead of expanding each node in our search tree using the entire set of motion primitives, we start with coarse motion primitives and use finer motion primitives as the search progresses. Thus, we start (Sec. IV-B) by describing the parameters required to define a motion primitive. After that, we continue (Sec. IV-C) to detail a hierarchy of motion primitives together with an ordering that will be used in our search algorithm. We then describe our search algorithm in detail (Sec. IV-D) and elaborate on the method we use to handle "similar" states, also known as *duplicate detection* [14] (Sec. IV-F). We conclude this section with some implementation details (Sec. IV-G).

B. Motion Primitives

Motion primitives, introduced by Frazzoli et al. [17], have been used in many motion planners [23, 24, 33, 42, 35]. In our setting, the motion primitives are a set of predefined kinematically feasible local motions. Roughly speaking, a motion primitive defines with what curvature the needle curves, how far the needle steers, and in what direction (see Fig. 3). Since

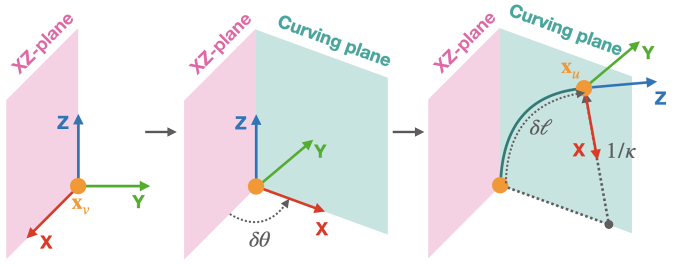


Fig. 3. A motion primitive is a circular arc defined as $\mathcal{M} = (\kappa, \delta\ell, \delta\theta)$. The circular arc (dark green) lies in the curving plane (light green) that contains the Z-axis (blue) at the start configuration $\mathbf{x}_v$. κ is the curvature of the arc, $\delta\theta$ is the angle between the curving plane and the XZ-plane, and $\delta\ell$ is the length of the arc. The figures show step-by-step how the child configuration $\mathbf{x}_u = \mathbf{x}_v \oplus \mathcal{M}$ is generated.

for each motion primitive, the curvature κ is explicitly defined, a motion primitive is guaranteed to be kinematically feasible as long as $\kappa \leq \kappa_{\max}$. As we will see in the proofs (Appendix A), our definition of motion primitives guarantees resolution completeness, and the experiments show that the definition also the enables computation efficiency of our algorithm.

More formally, to steer from configuration $\mathbf{x}_v$, a motion primitive is defined as a three-tuple $\mathcal{M} = (\kappa, \delta\ell, \delta\theta)$, where $\kappa \in [0, \kappa_{\max}]$ is the curvature, $\delta\ell > 0$ is the length of the circular arc, and $\delta\theta \in [0, 2\pi)$ is the angle between the curving plane and the XZ-plane of $\mathbf{x}_v$ (see Fig. 3). Thus the *action space* (or motion space) can be defined as $\mathcal{A} \subset \mathbb{R}^3$, which is the set of all motion primitives. We use $\mathbf{x}_u = \mathbf{x}_v \oplus \mathcal{M}$ to denote the operation of extending $\mathbf{x}_v$ with motion primitive $\mathcal{M}$ and obtaining the resultant configuration $\mathbf{x}_u$. See Fig. 3 for a step-by-step determination of $\mathbf{x}_u$. In the context of a search tree, by a slight abuse of notation, $u = v \oplus \mathcal{M}$ denotes the resultant node u, obtained by extending node v with the motion primitive $\mathcal{M}$. We call $\mathcal{M}$ the *extending primitive* of node u.

Using motion primitives allows pre-computing intermediate configurations and thus saving computation efforts during planning by transforming these configurations to the frame defined by $\mathbf{x}_v$. Since the trajectory produced with one motion primitive is a circular arc, it is possible to densely interpolate the trajectory for collision-checking purposes.

In the following sections, we show that $\delta\ell$ and $\delta\theta$ are gradually refined in the algorithm. In contrast, we keep a fixed set of curvatures, $\{0, \kappa_{\max}\}$, for all motion primitives. As we will see (Sec. V and Appendix A) this does not hinder the guarantees provided by our approach. Moreover, as we demonstrate in our experiments (Sec. VI), these primitives, coupled with our planner allow us to efficiently compute paths for non-trivial instances where other planners fail.

C. Motion Primitive Hierarchy

Our algorithm uses a sequence of motion primitives, whose resolution changes from coarse to fine. The coarsest motion primitives are defined by some parameters $\delta\ell_{\max}$ and $\delta\theta_{\max}$. In our implementation and examples (e.g., Fig. 4) we have that $\delta\theta_{\max} = \frac{\pi}{2}$ and $\delta\ell_{\max} > 0$ is a user-given parameter.

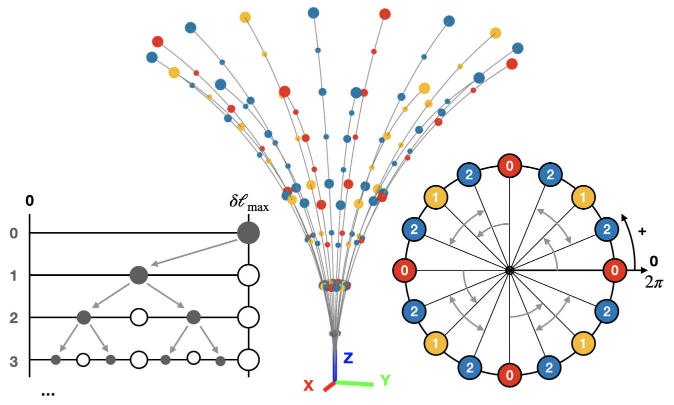


Fig. 4. **Visualization of length and angle levels. Left:** Visualization of length levels. Smaller node sizes correspond to higher length levels. The first length level ($l_\ell = 0$) corresponds to motion primitives of maximal length ($\delta\ell_{\max}$). As l_ℓ increases, the resolution of length becomes higher. The gray arrows illustrate how motion primitives with the first 4 length levels are generated during refinement. **Right:** Visualization of angle levels. Nodes with angle levels 0, 1, 2 are shown in red, yellow, and blue, respectively. The first angle level ($l_\theta = 0$) corresponds to motion primitives of $\delta\theta = \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. As l_θ increases, the resolution of orientation becomes higher. The circular arrows illustrate how nodes with the first three angle levels are generated during refinement. **Middle:** 3D visualization of length and angle levels.

Since $\delta\theta \in [0, 2\pi)$ and $\delta\theta_{\max} = \frac{\pi}{2}$, there exist four orientations ($\delta\theta \in \{0, 0.5\pi, \pi, 1.5\pi\}$) that have the coarsest orientation (see Fig. 4). There exists only one coarsest length, which is $\delta\ell_{\max}$, since path length is accumulated when we expand a node. To characterize how fine the resolution of a motion primitive $\mathcal{M} = (\kappa, \delta\ell, \delta\theta)$ is, we define the notions of length level l_ℓ and angle level l_θ. More formally,

$$l_\ell(\mathcal{M}) = \min\{l \in \mathbb{Z} \mid l \geq 0, \mathrm{MOD}(\delta\ell, 2^{-l} \cdot \delta\ell_{\max}) = 0\},$$
$$l_\theta(\mathcal{M}) = \min\{l \in \mathbb{Z} \mid l \geq 0, \mathrm{MOD}(\delta\theta, 2^{-l} \cdot \delta\theta_{\max}) = 0\},$$

where $\mathrm{MOD}(\cdot)$ is the modulo operation.

For a motion primitive $\mathcal{M} = (\kappa, \delta\ell, \delta\theta)$, we refine the resolution of both the insertion $\delta\ell$ and the orientation $\delta\theta$. The new motion primitives constructed by refining $\delta\ell$ are:

$$\mathcal{M}_{\ell\pm} = (\kappa, \delta\ell \pm 2^{-(l_\ell(\mathcal{M})+1)} \cdot \delta\ell_{\max}, \delta\theta). \quad (1)$$

Similarly, the motion primitives constructed by refining $\delta\theta$ are:

$$\mathcal{M}_{\theta\pm} = (\kappa, \delta\ell, \delta\theta \pm 2^{-(l_\theta(\mathcal{M})+1)} \cdot \delta\theta_{\max}). \quad (2)$$

It is straight-forward to see that the refined motion primitives $\mathcal{M}_{\ell-}$ and $\mathcal{M}_{\ell+}$ both have a length level of $l_\ell(\mathcal{M}) + 1$ and the refined motion primitives $\mathcal{M}_{\theta-}$ and $\mathcal{M}_{\theta+}$ both have an angle level of $l_\theta(\mathcal{M}) + 1$ (see Fig. 4).

Note that when refining a motion primitive with $l_\ell(\mathcal{M}) = 0$ (resp. $l_\theta(\mathcal{M}) = 0$), we ignore $\mathcal{M}_{\ell+}$ (resp. $\mathcal{M}_{\theta-}$) as they both exceed the range of exploration.

Similar to Lindemann and LaValle [33], our search algorithm expands nodes according to a node's *rank*. Rank captures both the depth of a node in the search tree and the fineness of resolution along the branch connecting the node from the root. We define the rank of the root node to be zero, the rank of any other node v is recursively defined as:

$$\mathrm{Rank}(v) = \mathrm{Rank}(v.\mathrm{parent}) + l_\ell(\mathcal{M}_v) + l_\theta(\mathcal{M}_v) + 1. \quad (3)$$

For a visualization, see Figs. 4 and 5.

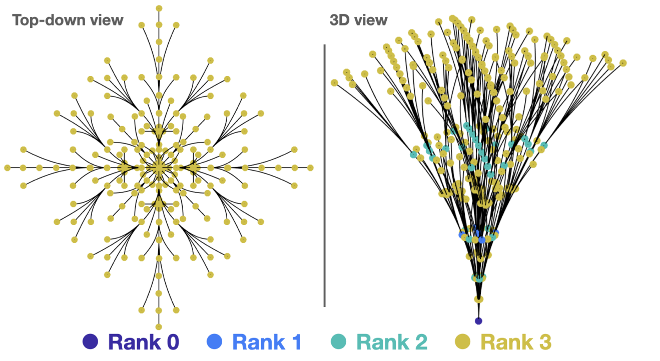


Fig. 5. Nodes of the first four ranks. We use motion primitives with $\kappa = 0$ (straight lines) and $\kappa = \kappa_{\max}$ (arcs with maximum curvature).

Algorithm 1 MultiResolutionSearch
Input: $\mathcal{W}_{\mathrm{obs}}, \mathbf{x}_{\mathrm{start}}, p_{\mathrm{goal}}, \tau, \kappa_{\max}, \ell_{\max}, \delta\ell_{\max}$

```
1:  $\Theta \leftarrow \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}, K \leftarrow \{0, \kappa_{\max}\}$
2:  root $\leftarrow (\mathbf{x}_{\mathrm{start}}, 0)$                    ▷ The root has rank 0
3:  OPEN $\leftarrow$ {root}, CLOSED $\leftarrow \emptyset$
4:  while not OPEN.empty() do
5:    $v \leftarrow$ OPEN.extract()
6:    if Valid($v, \mathcal{W}_{\mathrm{obs}}, p_{\mathrm{goal}}, \ell_{\max}$) then
7:      if not existSimilarConfig($v$, CLOSED) then
8:        if Terminate($v, p_{\mathrm{goal}}, \tau$) then
9:          return RetrievePlan($v$)
10:       for $\mathcal{M} \in$ Primitives($K, \delta\ell_{\max}, \Theta$) do
11:         OPEN.insert($v \oplus \mathcal{M}$)
12:       CLOSED.insert($v$)
13:   if $v$ != root then
14:     for $\mathcal{M} \in$ RefinedPrimitives($\mathcal{M}_v$) do
15:       OPEN.insert($v$.parent $\oplus \mathcal{M}$)
16: return NULL
```

D. Algorithm Description

We run an A*-like search where nodes are ordered according to their rank (Eq. 3). A distinctive feature from (vanilla) A* is that when we expand a node, we also increase the resolution of the motion primitives used to expand its parent and add nodes using these finer motion primitives to the search's priority queue. The rest of this section formalizes this idea.

Alg. 1 shows the pseudocode of our needle planner. We first initialize the coarsest orientations and the curvature set (line 1), then initialize the OPEN list and CLOSED set (line 3). The search algorithm then iteratively extracts nodes from the OPEN list (line 5), where nodes are ordered in a monotonically non-decreasing order according to their rank.

Only at this point (line 6) the extracted node is validated (also known as *lazy* validation [21, 36]). Validation of node

v involves ensuring that: (i) the accumulated trajectory length should not exceed the maximum insertion length $\ell_{\max}$; (ii) the goal point should be inside or close to the reachable region of $\mathbf{x}_v$ (Sec. IV-G); (iii) v should not be a duplicated node (Sec. IV-G); and that (iv) the circular arc connecting v.parent and v should be collision-free. An invalid node will be rejected and discarded. For a valid node v, we further check if there exists any similar configuration in the CLOSED set in order to avoid considering highly similar configurations (Sec. IV-F and Appendix A). A valid node without a similar configuration is accepted, expanded, and added to the CLOSED set (lines 10-12). The search terminates if the associated configuration of the accepted node satisfies the goal tolerance.

In our search algorithm, only the coarsest child nodes are added to the OPEN list during the initial expansion of a node (lines 10-11). But additional child nodes, created with finer motion primitives, are added when the coarse child nodes are extracted from the OPEN list (line 15). More specifically, when node v is extracted, we refine its extending motion primitive $\mathcal{M}_v$ following Eq. 1 and 2 (line 14), and use the refined motion primitives $\mathcal{M}_{\ell\pm}$ and $\mathcal{M}_{\theta\pm}$ to expand v.parent.

E. Cutoff Resolution

As specified in Sec. III, for a physical needle-steering robot there exists some smallest interval or precision of the achievable motions, which induces the minimal insertion and axial rotation $\delta\ell_{\min}$ and $\delta\theta_{\min}$, respectively. We term $\delta\ell_{\min}$ and $\delta\theta_{\min}$ as the *cutoff resolution* and stop adding refined nodes when the extending motion primitive $\mathcal{M}$ satisfies $2^{-l_\ell(\mathcal{M})} \cdot \delta\ell_{\max} < \delta\ell_{\min}$ or $2^{-l_\theta(\mathcal{M})} \cdot \delta\theta_{\max} < \delta\theta_{\min}$.

F. Duplicate Detection

To avoid re-expanding the same or highly similar nodes multiple times, search-based planners often employ *duplicate detection* [14] that prunes so-called "duplicate" nodes. To prune duplicate nodes and enable the planner to rapidly explore the entire C-space, we reject a node if there already exists a similar configuration in the search tree (line 7). More formally, we reject node v with configuration $\mathbf{x}_v$ if $\exists u \in \mathcal{V}, \rho(\mathbf{x}_u, \mathbf{x}_v) < d_{\text{sim}}$, where $d_{\text{sim}} > 0$ is a radius we use to identify similar configurations. Here, $\rho(\cdot)$ is a distance metric defined on $\mathcal{X}$ which in our work is defined as

$$\rho(\mathbf{x}_u, \mathbf{x}_v) = \|p_u - p_v\|_2 + \alpha \cdot \text{dist}_{\sphericalangle}(q_u, q_v), \tag{4}$$

where $\alpha > 0$ is a weighting parameter and $\text{dist}_{\sphericalangle}()$ is the angular distance between two orientations. Note that to guarantee resolution completeness, the value of d_{sim} depends on other system parameters detailed in Sec. V and Appendix A.

G. Implementation Details

We now describe several implementation details used to further speed up our approach. To distinguish between different implementations of our approach we refer to the basic version of our Resolution-Complete Search (i.e., without the implementation details described below) as RCS_BASIC and

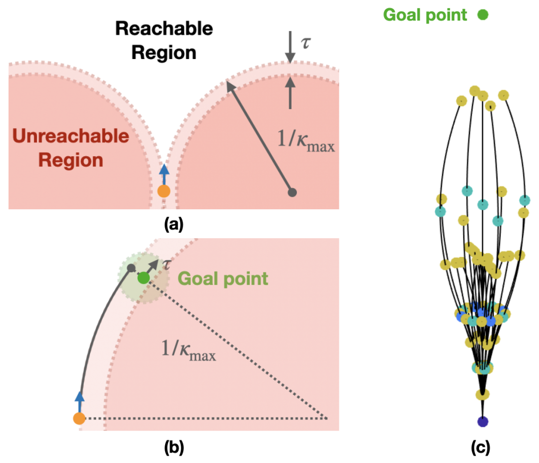


Fig. 6. **(a)** An illustration of reachable and unreachable regions in 2D. The case in 3D is similar. The unreachable region can be generated by rotating the circles around the Z-axis (blue vector), which creates a donut-like shape in 3D that is unreachable. It also visualizes how we check goal-reachability when considering tolerance τ. We reject a configuration if the relative position of p_{goal} falls in the inner region (darker orange). **(b)** The algorithm creates a direct connection to the goal when p_{goal} is outside but still close to the boundary of the reachable region. We use a circular arc with curvature $\kappa_{\max}$ to steer towards p_{goal} and the arc stops at the closest point to p_{goal}. **(c)** An example of valid nodes with rank 0-3 after checking goal reachability.

to the (basic) version that does not use similar-node rejection (i.e., when not performing the test in line 7) as RCS_NR. The versions that use all the following implementation details without and with parallelization (explained shortly) will be referred to as RCS and RCS_PARA, respectively.

1) Early pruning by testing for goal reachability: We can prune away nodes that, due to curvature constraints, cannot be part of a path that reaches the goal (see Fig. 6 for a 2D illustration). The curvature constraint defines so-called "unreachable regions" of a node and testing if the goal p_{goal} belongs to a node's unreachable region can be done efficiently (see Fig. 6). Such nodes are pruned away and not expanded.

However, recall that we allow some goal tolerance τ. Thus, instead of requiring the goal point to be inside a node's reachable region, we only require that the distance between p_{goal} and the boundary of the reachable region is smaller than τ.

Our model allows a needle to make "U-turns" and reach the region we currently mark as unreachable. But in our specific setting, the needle tip cannot (physically) turn more than 90° as the needle might buckle and shear through the tissue, so we discard such motions. Thus we don't need to account for a needle entering the unreachable region due to a "U-turn".

2) Direct goal connection: For each node v that is added to the search tree with corresponding configuration $\mathbf{x}_v$, we attempt to connect $\mathbf{x}_v$ to the goal point p_{goal} with a circular arc (a similar technique is used in the RRT-based needle planner [40]). This arc lies in the plane that is determined by the tangent vector of $\mathbf{x}_v$ and p_{goal}, and its curvature can be computed according to the relative position of $\mathbf{x}_v$ and p_{goal}.

If p_{goal} lies outside the reachable region of $\mathbf{x}_v$ but the distance between p_{goal} and the boundary of the reachable region is no larger than τ, we steer the needle in the plane following a circular arc of curvature $\kappa_{\max}$ to the point closest to p_{goal}. When the circular arc is collision-free, a solution has been found and we terminate the search. This approach can often dramatically speed up the search.

3) Equivalent node pruning: As we use a multi-resolution approach, there may exist multiple nodes representing the exact same configurations. Our approach for rejecting similar nodes (Sec. IV-F) can be used to reject equivalent ones. However, testing if two nodes are equivalent is more efficient and saves future computationally expensive collision checking.

As we are refining the arc length $\delta\ell$ and orientation $\delta\theta$ simultaneously, it is possible for a node to be expanded more than once with the same motion primitive: first as a node with finer arc length, then as a node with finer orientation. To avoid extending the same node with the same motion primitive, we give each motion primitive a unique index and the parent node keeps record of which motion primitives have been explored and allows only unexplored motion primitives when adding finer nodes (line 15).

4) Parallelism: One of the most time-consuming tasks in our search algorithm is processing a node after it is extracted from the OPEN list (namely, evaluating if the path to this node is collision free, computing the relevant motion primitives for its parent node and the corresponding new nodes). To this end, we implemented a multi-threaded version of the algorithm where each thread is tasked with processing a node extracted from the OPEN list. This enables processing nodes in parallel while maintaining the correctness of the algorithm by adding standard locking mechanisms to the shared data structures (i.e., OPEN list and CLOSED set).

V. Theoretical Guarantees

In this section we state and give a proof overview of the theoretical guarantees that our algorithm provides. We start with some general definitions pertaining to the notion of resolution completeness adapted from LaValle [32]. Unfortunately, their generality requires masking important problem-related details such as, "is planning defined in the C-space or in the control space?" or "what are the specific assumptions on the system?" This is also the reason that existing proofs (e.g., [7, Appendix A] and [11, Thm. 5.2]) cannot be used as is. Thus, we quickly move to the specific setting of motion planning for steerable needles which requires specifying the exact problem-related details and definitions. Here we start with an overview of our proof explaining where we rely on the aforementioned proofs and where we are required to account for our specific domain and planner.

A. General resolution-related definitions

Definition 1 (Resolution). *Resolution is a finite set of parameters $R = \{r_0, r_1, ..., r_n\}$, where each $r_i \in R$ characterizes the discretization of some space (e.g. state space, configuration space, action space, and time), and the smaller r_i is, the finer the corresponding resolution is.*

Definition 2 (Resolution completeness). *For a general motion planning problem Δ, a planner $\mathcal{P}$ is resolution complete if when a so-called qualified solution to Δ exists, there exists some resolution $R_{\min}$ such that running $\mathcal{P}$ with resolution $R_{\min}$ on Δ finds a solution in finite time.*

Clearly the above definition is more a general intuition than a precise definition. We need to define what a "qualified solution is" and what "running $\mathcal{P}$ with resolution $R_{\min}$ on Δ" means. These notions together with our main theoretic result (Thm. 2) are formalized in Appendix A.

B. Proof overview

As a first step we need to state how Def. 1 is instantiated in our setting. Here, the resolution is a pair $R = \{r_\ell, r_\theta\}$ that characterizes the action space (namely, the insertion $\delta\ell$ and rotation $\delta\theta$ of the needle). However, this geometric characterization of the needle motion is a simplification of the way we control a needle in practice—via insertion and rotation *velocity*. This difference is important as the relative insertion and rotation velocity creates paths that have curvatures ranging between zero and $\kappa_{\max}$. In contrast, our motion primitives either follow a straight line or a path of maximal curvature $\kappa_{\max}$. Thus, the first part of our proof shows that considering these two fixed curvatures allows us to approximate any path arbitrarily well. This is done using the notion of *duty cycling* [37] and is detailed in Sec. B of Appendix A. The original idea in [37] is designed specifically for bevel-tip needles. We look at the problem from a geometric perspective and decouple the guarantees and the needle mechanism, thus making it valid for needles with different designs.

The second part of our proof, detailed in Sec. C of Appendix A, states that any path that adheres to some mild assumptions can be approximated arbitrarily well by a very specific set of motion primitives—those with some *fixed* resolution. This is a somewhat technical but important step—it will allow us to argue that as long as the cutoff resolution (defined in Sec. IV-E) is fine enough, our algorithm RCS_NR is guaranteed to find a solution in finite time (when no node rejection is applied). Here we adapt the original proof by Barraquand et al. [7, Appendix A] that considers paths in a two-dimensional workspace. As our needle moves in a 3D workspace, we cannot use the proof as-is and detail some required adaptations.

These parts are summarized in the following theorem (stated informally to avoid using notations defined in Appendix A).

Theorem 1 (Resolution completeness of RCS_NR). *Let $\Delta = (\mathcal{X}, \mathcal{W}_{\text{obs}}, \mathbf{x}_{\text{start}}, p_{\text{goal}}, \tau, \ell_{\max}, \kappa_{\max})$ be a steerable needle motion planning problem. Under the assumption that the system is Lipschitz continuous and there exists a traceable solution with non-zero clearance*[1]*, there exists some cutoff resolution for which RCS_NR will find a solution in finite time.*

The third part of our proof, described in Sec. E of Appendix A, shows that even with similar node rejection, the basic version of our algorithm RCS_BASIC still finds a solution

[1]Refer to Appendix A for detailed definitions of Lipschitz continuous, traceable trajectory, and clearance.

when several conditions are satisfied. Here we adapt the proofs provided by Cheng and LaValle [11, Thm. 5.2]. In their proof, a fixed control period is assumed for every motion primitive. Thus, we need to incorporate the machinery developed in the second part of our proof (Sec. C of Appendix A) and obtain the following result (again, stated informally to avoid using notations defined only in Appendix A).

Theorem 2 (Resolution completeness with similar-node rejection)**.** *Let $\Delta = (\mathcal{X}, \mathcal{W}_{\text{obs}}, \mathbf{x}_{\text{start}}, p_{\text{goal}}, \tau, \ell_{\max}, \kappa_{\max})$ be a steerable needle motion planning problem. Under the assumption that the system is Lipschitz continuous and there exists a traceable solution that has sufficient clearance, there exists some cutoff resolution $\{\delta\ell_{\min}, \delta\theta_{\min}\}$ and some radius for similar node rejection d_{sim} (which is a function of $\tau, \ell_{\max}$ and $\delta\ell_{\min}$) for which RCS_BASIC will find a solution in finite time.*

In the final part of the proof (Sec. F of Appendix A), we show that none of the implementation details we use to improve the algorithm's efficiency hinder the theoretical guarantees of RCS_BASIC.

VI. Results

We evaluate our new resolution-complete motion planner for steerable needles using scenarios based on the medical task of lung biopsy. Lung cancer is the deadliest form of cancer in the United States, killing over 150,000 Americans each year [5]. Early diagnosis is critical for patient survival, and biopsy of suspicious nodules is required for diagnosis. Steerable needles deployed from bronchoscopes have the potential to safely and accurately reach nodules throughout the lung for biopsy and localized treatment [29, 50]. We illustrate in Fig. 1 a volumetric model of the relevant anatomy segmented from a CT scan [18]. In this procedure, the steerable needle is deployed from a bronchoscope inside the lung and must steer from the start pose just outside a bronchial tube (the furthest pose reachable by the bronchoscope) to the nodule while avoiding anatomical obstacles that include the large blood vessels, the bronchial tubes, and the lung boundary.

To create test cases, we randomly sampled 50 collision-free start configurations along the bronchial tube walls (i.e., points reachable by the bronchoscope from which the steerable needle can be deployed), each with 10 reachable goal points in the lung parenchyma (i.e., points in the tissue of the lung outside the bronchial tubes in which nodules requiring biopsy may occur), totaling 500 test cases (see Fig. 7 for 10 plans computed by RCS). To avoid skewing the data with trivial test cases, we discarded test cases where the start configuration can be connected directly to the goal point with a collision-free arc. Additionally, we also disallowed test cases where there are obstacles directly in front of the start configuration deeming the problem unsolvable. Finally, note that it is not guaranteed that a valid plan exists for a test case.

We consider a steerable needle with a maximum curvature of $\kappa_{\max} = 0.01(\text{mm}^{-1})$, device diameter of 2mm,

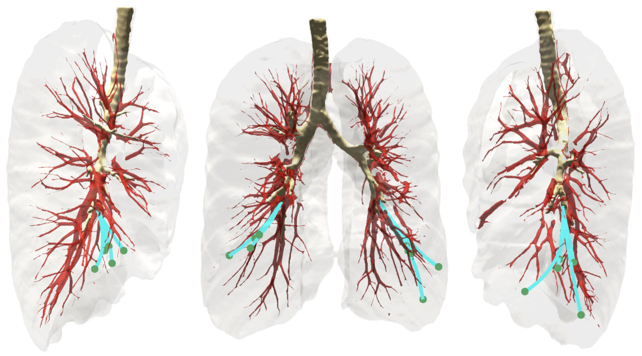


Fig. 7. Three views of the the lung environment. The needle steers to targets (green) while avoiding anatomical obstacles including large blood vessels (red), bronchial tubes (brown), and the lung boundary (gray). We also show 10 of the 500 test cases in which the steerable needle must deploy from the bronchoscope's tip in the bronchial tube to the nodule in the lung parenchyma. For these example test cases, we show plans computed by RCS (cyan).

and maximum insertion length of 100mm. The simulated workspace was reconstructed from a preoperative chest CT scan where $\mathcal{W}_{\text{obs}}$ is a point cloud representing the anatomical obstacles described above. We use a collision-checking resolution of 0.5mm and a goal tolerance of $\tau = 1.0\text{mm}$.

We compared in simulation the variants of RCS with two steerable needle planners: an RRT-based planner [30, 40] and AFT [34, 41]. While the original AFT algorithm is GPU accelerated, here we present results for our CPU-based implementation and only focus on the feasibility of the method and not on the computing times (we let AFT run until it terminates). Similar to [41], we define the cost function for AFT as

$$\text{Cost}(\sigma) = \ell(\sigma)/\ell_{\max} + \|\sigma(1) - p_{\text{goal}}\|_2/\tau, \tag{5}$$

which accounts both for insertion length and final tip error. We also ran a search-based planner denoted as SINGLE_RES that includes all optimizations of RCS mentioned in Sec. IV-G but that uses only the finest resolution (with no multiple resolutions). For additional details about the parameters used for each planner, see Appendix B. All experiments were run on a dual 2.1GHz 16-core Intel Xeon Silver 4216 CPU and 100GB of RAM. Code for our proposed method is available on GitHub [19].

We now present results pertaining to the success rate of the different algorithms. In our setting, the success rate is the ratio of solved cases among all 500 test cases. For RCS, RRT, and their variants, each planner was allowed 100 seconds. The results are shown in Fig. 8. First, among RCS variants, RCS performed much better than RCS_BASIC, indicating the first three optimizations introduced in Sec. IV-G dramatically improved the efficiency of the algorithm. Furthermore, except for the obvious overhead effect in the early stage ($< 10\text{ms}$), RCS_PARA achieved even better performance. The single-resolution planner SINGLE_RES only achieved a 24.2% success rate, suggesting that the multi-resolution approach in RCS variants is valuable. Second, the single-threaded RCS

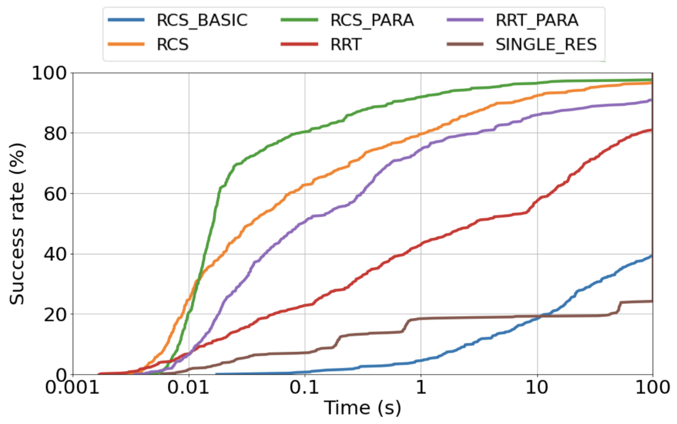


Fig. 8. Success rate as a function of time for RCS and RRT.

achieved better performance than the single-threaded RRT and multi-threaded RRT_PARA. From the perspective of running time, RCS_PARA's average running time for solved cases is 0.43 seconds, and it took 0.83 seconds to reach a success rate of 91.2%, which is roughly 120 times faster than RRT_PARA.

Since we only had a CPU-based version of the AFT algorithm, we do not compare success rate over time. Instead, we compare the success rate when RCS runs for 100 seconds and AFT finishes two tree refinements. Additionally, as AFT produces many paths while optimizing a cost function that does not necessarily favor paths with minimal goal tolerance (Eq. 5), we chose the one with the minimal goal tolerance (not with the minimal cost) for success rate analysis. The 5-level AFT achieved a success rate of 65.8%, with many of the failures due to the computed paths not satisfying the maximum allowed targeting error of $\tau = 1\text{mm}$.

For additional experiments evaluating the quality of the plans produced by each planner, see Appendix C.

VII. Conclusion & future work

In this paper, we took an important step toward creating a certifiable motion planner for steerable needles. Specifically, we introduced a resolution-complete planner that dramatically outperforms state-of-the-art needle planners in a clinically inspired simulation. This was achieved by carefully designing motion primitives and applying domain-specific optimizations. We formally showed that the planner is resolution complete, which means that under some mild assumptions on the system and the solution, the planner, in finite time, is guaranteed to find a plan as long as the problem admits a qualified solution.

We view this work as an *algorithmic foundation* required to obtain certifiable motion planning for steerable needles. Our planner is the first resolution-complete planner for steerable needles, but more work remains. Our analysis showed that, under some mild assumptions, when a qualified solution exists, *if* the cutoff resolution is fine enough and the path has *some* clearance (distance from the obstacles), the algorithm will find it. However, it would be valuable for medical applications to provide the precise relation between the system's controls and this cutoff resolution. Subsequently, we need to provide the precise relation between this cutoff resolution (i.e., what does it mean to be "fine enough") and the clearance of paths (i.e., what does it mean "some clearance"?). Future work will use this foundation to compute the relation between the aforementioned parameters in order to give physicians certifiable software for motion planning for steerable needles.

We believe that the algorithmic foundations laid out in this work will also allow us to provide guarantees on the *quality of the solution*—a critical requirement in our medical domain. Here, trajectory quality can correspond to minimizing damage to tissue, the time the patient is under anaesthesia, and more (see [9] and references within). Consequently, we plan to revisit the way nodes are ordered in our priority queue (recall that now they are ordered according to their rank) in order to provide optimality (or near-optimality) guarantees.

Acknowledgments

This research was supported in part by the U.S. National Institutes of Health (NIH) under award R01EB024864, the Israeli Ministry of Science & Technology (MOST) by grant No. 102583 and 2028142, and the United States-Israel Binational Science Foundation (BSF) by grant No. 1018193.

We thank Janine Hoelscher, Inbar Fried, Maxwell Emerson, Tayfun Efe Ertop, Margaret Rox, Josephine Granna, Alan Kuntz, Jason A. Akulian, and Robert J. Webster III for their discussions on steerable needles for lung applications.

References

[1] Aurora - NDI. https://www.ndigital.com/products/aurora/. Accessed: 2021-02-28.

[2] Niki Abolhassani, Rajni Patel, and Mehrdad Moallem. Needle insertion into soft tissue: A survey. *Medical Engineering & Physics*, 29(4):413–431, 2007.

[3] Ron Alterovitz, Ken Goldberg, and Allison Okamura. Planning for steerable bevel-tip needle insertion through 2D soft tissue with obstacles. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1640–1645. IEEE, 2005.

[4] Ron Alterovitz, Thierry Siméon, and Ken Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. In *Robotics: Science and Systems (RSS)*, 2007.

[5] American Cancer Society. Cancer Facts & Figures. Technical report, American Cancer Society, 2016.

[6] Ali Asadian, Mehrdad R Kermani, and Rajni V Patel. Robot-assisted needle steering using a control theoretic approach. *J. Intelligent and Robotic Systems*, 62(3):397–418, 2011.

[7] Jerome Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robotics Research (IJRR)*, 10(6):628–649, 1991.

[8] Jérôme Barraquand and Jean-Claude Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2):121–155, 1993.

[9] Michael Bentley, Caleb Rucker, Chakravarthy Reddy, Oren Salzman, and Alan Kuntz. A novel shaft-to-

tissue force model for safer motion planning of steerable needles. *Computing Research Repository (CoRR)*, abs/2101.02246, 2021.

[10] Mariana C Bernardes, Bruno V Adorno, Philippe Poignet, and Geovany A Borges. Semi-automatic needle steering system with robotic manipulator. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1595–1600. IEEE, 2012.

[11] Peng Cheng and Steven M LaValle. Resolution complete rapidly-exploring random trees. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, volume 1, pages 267–272. IEEE, 2002.

[12] Noah J Cowan, Ken Goldberg, Gregory S Chirikjian, Gabor Fichtinger, Ron Alterovitz, Kyle B Reed, Vinutha Kallem, Wooram Park, Sarthak Misra, and Allison M Okamura. Robotic needle steering: design, modeling, planning, and image guidance. In Jacob Rosen, Blake Hannaford, and Richard M Satava, editors, *Surgical Robotics: System Applications and Visions*, chapter 23, pages 557–582. Springer, 2011.

[13] Simon P DiMaio and Septimiu E Salcudean. Needle insertion modeling and simulation. *IEEE Trans. Robotics and Automation*, 19(5):864–875, 2003.

[14] Wei Du, Sung-Kyun Kim, Oren Salzman, and Maxim Likhachev. Escaping local minima in search-based planning using soft duplicate detection. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 2365–2371. IEEE, 2019.

[15] Vincent Duindam, Jijie Xu, Ron Alterovitz, Shankar Sastry, and Ken Goldberg. Three-dimensional motion planning algorithms for steerable needles using inverse kinematics. *Int. J. Robotics Research (IJRR)*, 29(7):789–800, 2010.

[16] Alberto Favaro, Leonardo Cerri, Stefano Galvan, Ferdinando Rodriguez Y Baena, and Elena De Momi. Automatic optimized 3D path planner for steerable catheters with heuristic search and uncertainty tolerance. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 9–16. IEEE, 2018.

[17] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25 (1):116–129, 2002.

[18] Mengyu Fu, Alan Kuntz, Robert J Webster III, and Ron Alterovitz. Safe motion planning for steerable needles using cost maps automatically extracted from pulmonary images. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 4942–4949. IEEE, 2018.

[19] Mengyu Fu, Oren Salzman, and Ron Alterovitz. steerable-needle-planner. https://github.com/UNC-Robotics/steerable-needle-planner, 2021. Accessed: 2021-6-9.

[20] K. Hauser, R. Alterovitz, N. Chentanez, A. Okamura, and K. Goldberg. Feedback control for steering needles through 3D deformable tissue using helical paths. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[21] Kris Hauser. Lazy collision checking in asymptotically-optimal motion planning. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2951–2957, 2015.

[22] Jeffrey Ichnowski and Ron Alterovitz. Motion planning templates: A motion planning framework for robots with low-power CPUs. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 612–618. IEEE, 2019.

[23] Fahad Islam, Oren Salzman, and Maxim Likhachev. Provable indefinite-horizon real-time planning for repetitive tasks. In *Int. Conf. Automated Planning and Scheduling (ICAPS)*, volume 29, pages 716–724, 2019.

[24] Fahad Islam, Anirudh Vemula, Sung-Kyun Kim, Andrew Dornbush, Oren Salzman, and Maxim Likhachev. Planning, learning and reasoning framework for robot truck unloading. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 5011–5017. IEEE, 2020.

[25] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Research (IJRR)*, 30(7):846–894, 2011.

[26] David G. Kirkpatrick, Irina Kostitsyna, and Valentin Polishchuk. Hardness results for two-dimensional curvature-constrained motion planning. In *Canadian Conference on Computational Geometry (CCCG)*, 2011.

[27] Michal Kleinbort, Kiril Solovey, Zakary Littlefield, Kostas E Bekris, and Dan Halperin. Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters*, 4(2):x–xvi, 2018.

[28] Seong Young Ko, Luca Frasson, and Ferdinando Rodriguez y Baena. Closed-loop planar motion control of a steerable probe with a "programmable bevel" inspired by nature. *IEEE Trans. Robotics*, 27(5):970–983, 2011.

[29] A Kuntz, P J Swaney, A Mahoney, R H Feins, Y Z Lee, Robert J Webster III, and Ron Alterovitz. Toward transoral peripheral lung access: Steering bronchoscope-deployed needles through porcine lung tissue. In *Hamlyn Symposium on Medical Robotics*, pages 9–10, 2016.

[30] Alan Kuntz, Luis G Torres, Richard H Feins, Robert J Webster III, and Ron Alterovitz. Motion planning for a three-stage multilumen transoral lung access system. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 3255–3261. IEEE, 2015.

[31] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[32] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[33] Stephen R Lindemann and Steven M LaValle. Multiresolution approach for motion planning under differential constraints. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 139–144. IEEE, 2006.

[34] Fangde Liu, Arnau Garriga-Casanovas, Riccardo Secoli, and Ferdinando Rodriguez y Baena. Fast and adaptive fractal tree-based path planning for programmable bevel tip steerable needles. *IEEE Robotics and Automation Letters*, 1(2):601–608, 2016.

[35] Oskar Ljungqvist, Niclas Evestedt, Marcello Cirillo, Daniel Axehill, and Olov Holmer. Lattice-based motion planning for a general 2-trailer system. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 819–824. IEEE, 2017.

[36] Aditya Mandalika, Sanjiban Choudhury, Oren Salzman, and Siddhartha S. Srinivasa. Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles. In *Int. Conf. Automated Planning and Scheduling (ICAPS)*, pages 745–753, 2019.

[37] Davneet S Minhas, Johnathan A Engh, Michele M Fenske, and Cameron N Riviere. Modeling of needle steering via duty-cycled spinning. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2756–2759. IEEE, 2007.

[38] Stephen Okazawa, Richelle Ebrahimi, Jason Chuang, Septimiu E Salcudean, and Robert Rohling. Hand-held steerable needle device. *IEEE/ASME Trans. Mechatronics*, 10(3):285–296, 2005.

[39] Wooram Park, Jin Seob Kim, Yu Zhou, Noah J Cowan, Allison M Okamura, and Gregory S Chirikjian. Diffusion-based motion planning for a nonholonomic flexible needle model. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4611–4616, April 2005.

[40] Sachin Patil, Jessica Burgner, Robert J Webster III, and Ron Alterovitz. Needle steering in 3D via rapid replanning. *IEEE Trans. Robotics*, 30(4):853–864, 2014.

[41] Marlene Pinzi, Stefano Galvan, and Ferdinando Rodriguez y Baena. The adaptive hermite fractal tree (AHFT): a novel surgical 3D path planning approach with curvature and heading constraints. *Int. J. Computer Assisted Radiology and Surgery*, 14(4):659–670, 2019.

[42] Mihail Pivtoraiko and Alonzo Kelly. Kinodynamic motion planning with state lattice motion primitives. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 2172–2179. IEEE, 2011.

[43] Peng Qi, Hongbin Liu, Lakmal Seneviratne, and Kaspar Althoefer. Towards kinematic modeling of a multi-DOF tendon driven robotic catheter. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3009–3012. IEEE, 2014.

[44] Kyle B Reed, Ann Majewicz, Vinutha Kallem, Ron Alterovitz, Ken Goldberg, Noah J Cowan, and Allison M Okamura. Robot-assisted needle steering. *IEEE Robotics and Automation Magazine*, 18(4):35–46, 2011.

[45] D Caleb Rucker, Jadav Das, Hunter B Gilbert, Philip J Swaney, Michael I Miga, Nilanjan Sarkar, and Robert J Webster III. Sliding mode control of steerable needles. *IEEE Trans. Robotics*, 29(5):1289–1299, 2013.

[46] Riccardo Secoli and Ferdinando Rodriguez y Baena. Adaptive path-following control for bio-inspired steerable needles. In *IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 87–93. IEEE, 2016.

[47] Konstantin M Seiler, Surya PN Singh, Salah Sukkarieh, and Hugh Durrant-Whyte. Using Lie group symmetries for fast corrective motion planning. *Int. J. Robotics Research (IJRR)*, 31(2):151–166, 2012.

[48] Kiril Solovey. Complexity of planning. *arXiv preprint arXiv:2003.03632v2 [cs.RO]*, 2020.

[49] Wen Sun, Sachin Patil, and Ron Alterovitz. High-frequency replanning under uncertainty using parallel sampling-based motion planning. *IEEE Trans. Robotics*, 31(1):104–116, 2015.

[50] Philip J Swaney, Arthur W Mahoney, Bryan I Hartley, Andria A Remirez, Erik Lamers, Richard H Feins, Ron Alterovitz, and Robert J Webster III. Toward transoral peripheral lung access: Combining continuum robots and steerable needles. *Journal of Medical Robotics Research*, 2(01):1750001, 2017.

[51] Jur Van Den Berg, Sachin Patil, Ron Alterovitz, Pieter Abbeel, and Ken Goldberg. LQG-based planning, sensing, and control of steerable needles. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 373–389. Springer, 2010.

[52] Robert J Webster III, Jin Seob Kim, Noah J Cowan, Gregory S Chirikjian, and Allison M Okamura. Nonholonomic modeling of needle steering. *Int. J. Robotics Research (IJRR)*, 25(5-6):509–525, 2006.

[53] Jijie Xu, Vincent Duindam, Ron Alterovitz, and Ken Goldberg. Motion planning for steerable needles in 3D environments with obstacles using rapidly-exploring random trees and backchaining. In *IEEE Int. Conf. Automation Science and Engineering*, pages 41–46. IEEE, 2008.

[54] Dmitry S Yershov and Steven M LaValle. Sufficient conditions for the existence of resolution complete planning algorithms. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 303–320. Springer, 2010.

Appendix A. Resolution Completeness

A. Preliminaries

Before we state the different parts of our proof, we introduce some definitions. Recall that a steerable needle motion planning problem is a tuple $\Delta = (\mathcal{X}, \mathcal{W}_{\mathrm{obs}}, \mathbf{x}_{\mathrm{start}}, p_{\mathrm{goal}}, \tau, \ell_{\max}, \kappa_{\max})$ and that $\rho(\cdot)$ is a distance metric defined on $\mathcal{X}$ (Eq. 4). Finally, recall that $\mathcal{A}$ is the action space, which is the set of all valid motion primitives. Throughout the proof, for some sequence of motion primitives M, we will use $\mathbf{x} \oplus M$ to denote the resultant trajectory obtained by sequentially applying elements in M to $\mathbf{x}$.

Definition 3 (Strong clearance)**.** *Let $\sigma : [0,1] \to \mathcal{X}$ be some trajectory. We say that σ has strong γ-clearance if*

$$\forall s \in [0,1], \min_{\mathbf{x} \in \mathcal{X}_{\mathrm{obs}}} \rho(\sigma(s), \mathbf{x}) > \gamma,$$

where $\mathcal{X}_{\mathrm{obs}} = \mathrm{cl}(\mathcal{X} \setminus \mathcal{X}_{\mathrm{free}})$ and $\mathrm{cl}(\cdot)$ is the closure of a set.

Definition 4 (Trajectory approximation)**.** *Let $\sigma : [0,1] \to \mathcal{X}$ be some trajectory. We say that another trajectory σ' is an ε-approximation of σ if the following conditions are satisfied:*

(i) boundary condition: $\forall s \in \{0,1\}, \rho(\sigma(s), \sigma'(s)) < \varepsilon$;

(ii) one-way Hausdorff distance:

$$\max_{t \in [0,1]} \{ \min_{s \in [0,1]} \rho(\sigma(s), \sigma'(t)) \} < \varepsilon.$$

Definition 5 (Decomposable trajectory)**.** *Let $\sigma : [0,1] \to \mathcal{X}$ be some trajectory. We say that σ is decomposable if it can be decomposed into a finite sequence of motion primitives. Namely, there exists $M_\sigma = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\} \subset \mathcal{A}$ such that $\sigma = \sigma(0) \oplus M_\sigma$.*

Definition 6 (Traceable trajectory)**.** *Let $\sigma : [0,1] \to \mathcal{X}$ be some trajectory. We say that σ is traceable if for any given $\varepsilon > 0$, there exists a decomposable trajectory that is an ε-approximation of σ.*

Note that in the definitions of decomposable and traceable trajectories we allow any arbitrary set of motion primitives. This allows us to decouple the planner's ability of finding a path using a given set of motion primitives with the expressiveness of the motion primitives.

Definition 7. *We define a distance metric on action space $\mathcal{A}$ as the two-way Hausdorff distance between two resultant trajectories $\mathbf{x} \oplus \mathcal{M}_1$ and $\mathbf{x} \oplus \mathcal{M}_2$. Formally, we have*

$$\rho_{\mathcal{A}}(\mathcal{M}_1, \mathcal{M}_2) = \max\Big\{ \max_{t \in [0,1]} \{ \min_{s \in [0,1]} \rho(\sigma_{\mathcal{M}_1}(s), \sigma_{\mathcal{M}_2}(t)) \}, \max_{s \in [0,1]} \{ \min_{t \in [0,1]} \rho(\sigma_{\mathcal{M}_1}(s), \sigma_{\mathcal{M}_2}(t)) \} \Big\},$$

where $\sigma_{\mathcal{M}_1} = \mathbf{x} \oplus \mathcal{M}_1$ and $\sigma_{\mathcal{M}_2} = \mathbf{x} \oplus \mathcal{M}_2$. It is worth to note that changing $\mathbf{x}$ does not change the relative position between the two trajectories. Thus, without losing generality, we have $\mathbf{x} = (p, q)$ where $p = (0,0,0)$ and $q = (1,0,0,0)$.

Definition 8 (Lipschitz continuous)**.** *The system is Lipschitz continuous if $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}, \forall \mathcal{M}_1, \mathcal{M}_2 \in \mathcal{A}$,*

$$\rho(\mathbf{x}_1 \oplus \mathcal{M}_1, \mathbf{x}_2 \oplus \mathcal{M}_2) \le L_s(\rho(\mathbf{x}_1, \mathbf{x}_2) + \rho_{\mathcal{A}}(\mathcal{M}_1, \mathcal{M}_2)),$$

where $L_s > 0$ is a constant.

Finally, as we will see, it will be convenient to introduce the notion of a *finest set* of motion primitives.

Definition 9 (Finest set of motion primitives)**.** *Given a resolution $R = \{r_\ell, r_\theta\}$, and a set of curvatures K, we define the* finest set of motion primitives *as*

$$M_{\mathrm{fs}}(R, K) = \left\{ (\kappa, r_\ell, n \cdot r_\theta) \;\middle|\; \kappa \in K, n \in \left[0, \left\lfloor \frac{2\pi}{r_\theta} \right\rfloor\right] \subset \mathbb{Z} \right\}.$$

B. Approximating curves with arbitrary curvatures

When a bevel-tip needle is inserted only, it follows a trajectory with curvature $\kappa_{\max}$. When the needle is inserted while applying axial rotational velocity that is relatively larger than the insertion velocity, it follows a straight line (i.e., of curvature zero). Minhans et al. [37] introduced the notion of duty-cycling to approximate any curvature for bevel-tip steerable needles. Roughly speaking, combining periods of needle spinning (i.e., zero-curvature trajectories) with periods of non-spinning (i.e., maximal-curvature trajectories) enables the needle to achieve any curvature up to the maximum needle curvature. This idea is formalized in the following lemma.

Lemma 1 (Arbitrary curvature approximation using duty-cycling)**.** *Let σ be a decomposable trajectory and let $\varepsilon_d > 0$ be some real value. There exists a finite sequence of motion primitives M_D in which every element has curvature $\kappa \in \{0, \kappa_{\max}\}$ such that the trajectory $\sigma(0) \oplus M_D$ is an ε_d-approximation of σ.*

Proof sketch. Here, to explicitly show how the approximation factor is used. And to provide a more general discussion, we provide a proof from a geometric perspective (and not control-based as in the original work by Minhas et al. [37]).

The trajectory σ is decomposable, thus there exists a sequence of motion primitives $M_\sigma = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ such that $\sigma = \sigma(0) \oplus M_\sigma$ and each motion primitive $\mathcal{M}_i$ has arbitrary curvature $\kappa_i \in [0, \kappa_{\max}]$. To approximate $\mathcal{M}_i$, we construct a sequence of motion primitives $M_i = \{\mathcal{M}_i^{(1)}, \ldots, \mathcal{M}_i^{(n_i)}\}$ that satisfies

$$\begin{aligned}
&\mathcal{M}_i^{(1)}.\delta\theta = \mathcal{M}_i.\delta\theta, \\
&\forall j \in [2, n_i], \mathcal{M}_i^{(j)}.\delta\theta = 0, \\
&\forall j \in [1, n_i], \mathcal{M}_i^{(j)}.\kappa \in \{0, \kappa_{\max}\}.
\end{aligned}$$

Namely, the first motion primitive $\mathcal{M}_i^{(1)}$ ensures that both trajectories use the same curving plane (see Fig. 3) and the the rest of the sequence stays within this curving plane and approximates the (arbitrary) curvature κ_i.

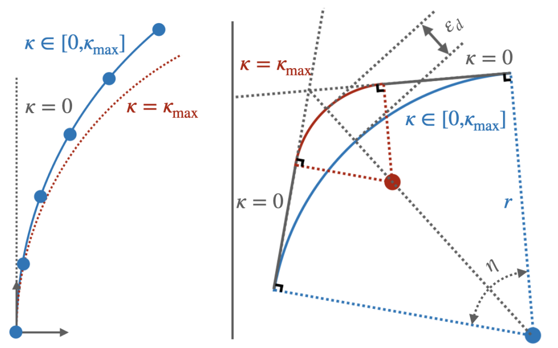


Fig. 9. **Illustration of approximation with duty-cycling. Left:** Decompose $\mathcal{M}_i$ into multiple segments with length ℓ_i. **Right:** Use three segments to approximate one segment of $\mathcal{M}_i$, where the segments have a curvature of $0, \kappa_{\max}, 0$, respectively. The one-way Hausdorff distance (marked as ε_d in the figure) depends on ℓ_i. For a given $\kappa_{\max}$, to approximate $\mathcal{M}_i$ (with curvature κ), the shorter ℓ_i is, the smaller ε_d is. This is because $\varepsilon_d < r \cdot (1/\cos(0.5\eta) - 1)$, where $r = 1/\kappa$ is the radius of curvature and $\eta = \ell_i / r$ is the central angle.

We then decompose $\mathcal{M}_i$ into small equal-length segments of length ℓ_i (except possibly the last segment) where the specific value of ℓ_i is chosen according to the value of ε_d. We then use three motion primitives to approximate each of these segments as illustrated in Fig. 9. It is not hard to see that (i) the start and end configurations of $\mathcal{M}_i$ and M_i are identical, and (ii) the one-way Hausdorff distance between $\mathcal{M}_i$ and each $\mathcal{M}_i^{(j)}$ is less than ε_d if ℓ_i is carefully chosen.

Let $M_\sigma^{\varepsilon_d} = M_1 \cdot M_2 \cdot \ldots \cdot M_n$ be this sequence of all the newly constructed motion primitives. Then it is straightforward that $\sigma(0) \oplus M_\sigma^{\varepsilon_d}$ is an ε_d-approximation of σ. ■

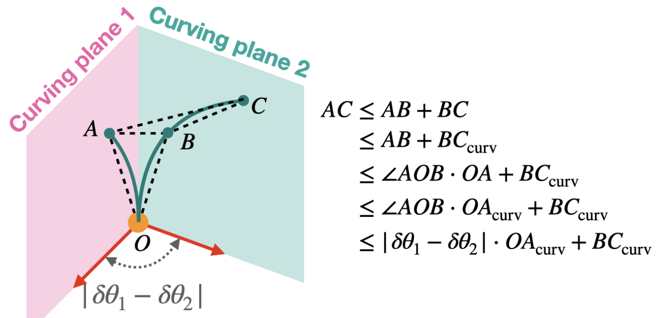
C. Approximating curves using fixed-length primitives

Lemma 2 (Fixed-resolution trajectory approximation)**.** *Let σ be a decomposable trajectory and let $\varepsilon_r > 0$ be some real value. If the system is Lipschitz continuous (Def. 8), there exists a fine resolution $R(\sigma, \varepsilon_r) = \{r_\ell, r_\theta\}$ and a finite sequence of motion primitives $M_{R(\sigma,\varepsilon_r)}$ such that $\sigma(0) \oplus M_{R(\sigma,\varepsilon_r)}$ is an ε_r-approximation of σ. Moreover $M_{R(\sigma,\varepsilon_r)} \subseteq \mathcal{M}_{\text{fs}}(R(\sigma,\varepsilon_r), K_\sigma)$, where K_σ is the set of curvatures that appear along σ.*

Proof sketch (adapted from [7, Appendix A]). The trajectory σ is decomposable, thus there exists a finite sequence of motion primitives $M_\sigma = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ such that $\sigma = \sigma(0) \oplus M_\sigma$. Set $K_\sigma = \bigcup_i \mathcal{M}_i.\kappa$ to be the set of all curvatures that appear in M_σ.

To approximate each motion primitive $\mathcal{M}_i$ using primitives from the finest set of motion primitives $\mathcal{M}_{\text{fs}}(R(\sigma,\varepsilon_r), K_\sigma)$ (Def. 9), we construct a sequence motion primitive $M_i = \{\mathcal{M}_i^{(1)}, \ldots \mathcal{M}_i^{(n_i)}\}$, where

$$
\begin{aligned}
&\mathcal{M}_i^{(1)}.\delta\theta = k_i \cdot r_\theta, \\
&\forall j \in [2, n_i], \mathcal{M}_i^{(j)}.\delta\theta = 0, \\
&\forall j \in [1, n_i], \mathcal{M}_i^{(j)}.\kappa = \mathcal{M}_i.\kappa, M_i^{(j)}.\delta\ell = r_\ell.
\end{aligned}
$$



Curving plane 1
Curving plane 2
A
B
C
O
$|\delta\theta_1 - \delta\theta_2|$
$AC \leq AB + BC$
$\leq AB + BC_{\text{curv}}$
$\leq \angle AOB \cdot OA + BC_{\text{curv}}$
$\leq \angle AOB \cdot OA_{\text{curv}} + BC_{\text{curv}}$
$\leq |\delta\theta_1 - \delta\theta_2| \cdot OA_{\text{curv}} + BC_{\text{curv}}$

Fig. 10. Illustration of the action distance between two motion primitives with the same curvature. Here the shorter motion primitive lies in curving plane 1, thus $\min\{\delta\ell_1, \delta\ell_2\} = OA_{\text{curv}}$ and $|\delta\ell_1 - \delta\ell_2| = OC_{\text{curv}} - OA_{\text{curv}} = BC_{\text{curv}}$.

Similar to the sequence constructed for Lemma 1, the first motion primitive $\mathcal{M}_i^{(1)}$ accounts for the curving plane (though here it can only be approximated) and the the rest of the sequence stays within this curving plane and accounts for the length of the circular arc the trajectory follows in this plane. Applying the sequence M_i is equivalent to applying one motion primitive $\tilde{\mathcal{M}}_i = (\mathcal{M}_i.\kappa, n_i \cdot r_\ell, k_i \cdot r_\theta)$. Thus, by carefully choosing r_ℓ and r_θ, distance between $\mathcal{M}_i$ and $\tilde{\mathcal{M}}_i$ (see Def. 7) can be arbitrarily small.

This is done for every motion primitive $\mathcal{M}_i$. As M is a finite sequence of size n, for any $\varepsilon > 0$ we can always find a fine-enough resolution $\{r_\ell, r_\theta\}$ that ensures that

$$\rho_{\mathcal{A}}(\mathcal{M}_i, \tilde{\mathcal{M}}_i) < \varepsilon, \forall i \in [1, n].$$

This is because, given that both motion primitives have equal curvature, $\rho_{\mathcal{A}}(\mathcal{M}_1, \mathcal{M}_2) < |\delta\theta_1 - \delta\theta_2| \cdot \min\{\delta\ell_1, \delta\ell_2\} + |\delta\ell_1 - \delta\ell_2|$, where $\delta\ell_i = \mathcal{M}_i.\delta\ell$ and $\delta\theta_i = \mathcal{M}_i.\delta\theta$. See Fig. 10 for illustration.

Since the system is Lipschitz continous,

$$
\begin{aligned}
&\rho(\sigma(0) \oplus \mathcal{M}_1 \cdots \oplus \mathcal{M}_n, \sigma(0) \oplus \tilde{\mathcal{M}}_1 \cdots \oplus \tilde{\mathcal{M}}_n) \\
\leq &L_s(\rho(\sigma(0) \oplus \mathcal{M}_1 \cdots \oplus \mathcal{M}_{n-1}, \sigma(0) \oplus \tilde{\mathcal{M}}_1 \cdots \oplus \tilde{\mathcal{M}}_{n-1}) \\
&+ \rho_{\mathcal{A}}(\mathcal{M}_n, \tilde{\mathcal{M}}_n) \\
\leq &L_s^n \cdot \rho(\sigma(0), \sigma(0)) + \sum_{i=1}^{n} L_s^{n-i+1} \cdot \rho_{\mathcal{A}}(\mathcal{M}_i, \tilde{\mathcal{M}}_i) \\
< &\varepsilon \cdot \frac{L_s(L_s^n - 1)}{L_s - 1}.
\end{aligned}
$$

Thus, to ensure that $\sigma(0) \oplus \{\tilde{\mathcal{M}}_1, \ldots, \tilde{\mathcal{M}}_n\}$ is an ε_r-approximation of σ, we only need to ensure that $\varepsilon \leq \frac{\varepsilon_r(L_s - 1)}{L_s(L_s^n - 1)}$. As both n and L_s are fixed, we can choose ε to be as small as needed thus the desired fine resolution exists which cocludes the proof. ■

Corollary 1. *Let σ be a traceable trajectory and let $\varepsilon > 0$ be some real value. If the system is Lipschitz continuous (Def. 8), there exists a fine resolution $R(\sigma, \varepsilon) = \{r_\ell, r_\theta\}$ and a finite sequence of motion primitives $M_{R(\sigma,\varepsilon_r)} \subseteq$*

$M_{\text{fs}}(R(\sigma,\varepsilon),\{0,\kappa_{\max}\})$ *such that* $\sigma(0) \oplus M_{R(\sigma,\varepsilon_r)}$ *is an* ε*-approximation of* σ*.*

Proof sketch. Set $\varepsilon_t = \varepsilon_d = \varepsilon_r = \varepsilon/3$. According to Def. 6, there exists a decomposable trajectory σ_t that is an ε_t-approximation of σ. Moreover, according to Lemma 1, there exists a finite sequence of motion primitives M_D in which every element has curvature $\kappa \in \{0,\kappa_{\max}\}$ such that the trajectory $\sigma_d = \sigma(0) \oplus M_D$ is an ε_d-approximation of σ_t.

Note that by construction σ_d is decomposable. Thus, according to Lemma 2, there exists a fine resolution $R(\sigma,\varepsilon_r) = \{r_\ell, r_\theta\}$ and a finite sequence of motion primitives $M_{R(\sigma,\varepsilon_r)}$ such that $\sigma_r = \sigma(0) \oplus M_{R(\sigma,\varepsilon_r)}$ is an ε_r-approximation of σ_d. Moreover, $M_{R(\sigma,\varepsilon_r)} \subseteq \mathcal{M}_{\text{fs}}(R(\sigma,\varepsilon_r),\{0,\kappa_{\max}\})$ as the construction in the proof of Lemma 2 does not add new curvatures.

Finally, as $\varepsilon_d = \varepsilon_d = \varepsilon_r = \varepsilon/3$. Then the trajectory σ_r is an ε-approximation of σ. ■

D. Resolution completeness (without similar node rejection)

Theorem 1 (Resolution completeness of RCS_NR). *Let* $\Delta = (\mathcal{X}, \mathcal{W}_{\text{obs}}, \mathbf{x}_{\text{start}}, p_{\text{goal}}, \tau, \ell_{\max}, \kappa_{\max})$ *be a steerable needle motion planning problem. If a solution to* Δ *is traceable, has strong* γ*-clearance for some* $\gamma > 0$*, and the system is Lipschitz continuous then there exists some cutoff resolution* $R_{\min}$ *for which RCS_NR will find a solution in finite time.*

Proof sketch. Let σ be a traceable solution with clearance γ. Following Cor. 1, there exists a fine resolution $R(\sigma,\varepsilon) = \{r_\ell, r_\theta\}$ and a finite sequence of motion primitives $M_{R(\sigma,\varepsilon)} \subseteq M_{\text{fs}}(R(\sigma,\varepsilon_r),\{0,\kappa_{\max}\})$ such that $\sigma(0) \oplus M_{R(\sigma,\varepsilon)}$ is an ε-approximation of σ. In our algorithm, the resolutions are divided by half as the length level l_ℓ and angle level l_θ increase. Thus, there exists a fine-enough resolution $\tilde{R} = \{2^{-l_\ell} \cdot \delta\ell_{\max}, 2^{-l_\theta} \cdot \delta\theta_{\max}\}$ that satisfies $2^{-k_\ell} \cdot \delta\ell_{\max} < r_\ell$, $2^{-k_\theta} \cdot \delta\theta_{\max} < r_\theta$. Setting the cutoff resolution $R_{\min}$ to be finer (both with respect to the insertion as well as rotation) than $\tilde{R}$ ensures that $M_{R(\sigma,\varepsilon)}$ can be approximated arbitrarily well.[2]

The search tree built with RCS_NR is a subtree of a dense tree in which each node is expanded with every element in $\mathcal{M}_{\text{fs}}(\tilde{R},\{0,\kappa_{\max}\})$. This is because every coarse motion primitive used in RCS_NR can be decomposed into a sequence of motion primitives in $\mathcal{M}_{\text{fs}}(\tilde{R},\{0,\kappa_{\max}\})$. Additionally, if we allow the algorithm run until the OPEN list is exhausted, every node in the dense tree (except for those that are in collision) will be explored by RCS_NR. Since the dense tree encodes all possible trajectories that can be decomposed with $\mathcal{M}_{\text{fs}}(\tilde{R},\{0,\kappa_{\max}\})$, when the solution σ is traceable, has γ-clearance, and the system is Lipschitz continuous, an ε-approximation (with $\varepsilon < \gamma$) of σ will be encoded in the dense tree and thus will be explored by RCS_NR. ■

[2]To be more precise, one needs to account for the cases where $R(\sigma,\varepsilon_r)$ is not in the sequence of resolutions considered by the algorithm and we may introduce additional error when approximating $R(\sigma,\varepsilon_r)$ with $\tilde{R}$. However, using the techniques we previously used this can be easily accounted for. We omit this in our proof sketch.

E. Resolution completeness (with similar node rejection)

We are now ready to show that even with similar node rejection, our algorithm is still resolution complete

Theorem 2 (Resolution completeness with similar-node rejection). *Let* $\Delta = (\mathcal{X}, \mathcal{W}_{\text{obs}}, \mathbf{x}_{\text{start}}, p_{\text{goal}}, \tau, \ell_{\max}, \kappa_{\max})$ *be a steerable needle motion planning problem. RCS_BASIC will find a solution in finite time, if the following conditions are satisfied:*

(C1) *The system is Lipschitz continuous.*

(C2) *The cutoff resolution* $R_{\min}$ *is fine enough and it satisfies* $\delta\ell_{\min} = 2^{-l_{\ell\max}} \cdot \delta\ell_{\max}, \delta\theta_{\min} = 2^{-l_{\theta\max}} \cdot \delta\theta_{\max}$.

(C3) *The radius* d_{sim} *used to reject similar nodes satisfies*

$$0 < d_{\text{sim}} < \min\left\{\delta\ell_{\min}, \frac{\tau(L_s-1)}{2(L_s^H-1)}\right\},$$

where $H = \lceil \ell_{\max}/\delta\ell_{\min} \rceil$.

(C4) *There exists a traceable solution plan* σ *with* $\frac{\tau}{2}$ *goal tolerance and strong* γ*-clearance (Def. 3) for* $\gamma > \frac{\tau+\delta\ell_{\min}}{2}$.

The proof of Thm. 2 uses Thm. 1 and then follows Cheng and LaValle [11, Thm. 5.2]. We include it here for completeness.

Proof. According to Thm. 1, RCS_NR terminates in finite time, thus RCS_BASIC also terminates in finite time since more nodes are rejected. We now prove that RCS_BASIC can find a solution plan if conditions (**C1**)-(**C4**) are satisfied.

Since σ is traceable, there exists some fine resolution $R(\sigma,\varepsilon)$ can be explored by RCS_BASIC (as discussed in Thm. 1), with which we can construct an ε-approximation of σ. Denote the decomposable approximation σ', and the sequence of motion primitives to compose it $M_{\sigma'} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$. When $M_{\sigma'}$ is sequentially applied to $\mathbf{x}_{\text{start}}$, we obtain a sequence of configurations $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_0 = \mathbf{x}_{\text{start}}, \mathbf{x}_i = \mathbf{x}_{i-1} \oplus \mathcal{M}_i, i \in [1,n]$. For the rest of the proof, we use $M_\sigma[i,j] = \{\mathcal{M}_i, \ldots, \mathcal{M}_j\}$ to denote a subsequence of $M_{\sigma'}$. We also use $\mathbf{x} + M_{\sigma'}[i,j]$ to denote the configuration after sequentially applying $\{\mathcal{M}_i, \ldots, \mathcal{M}_j\}$ to $\mathbf{x}$.

If we run RCS_NR, every $\mathbf{x}_i$ will be explored and σ will be constructed when the search terminates. However, if we run RCS_BASIC, we prune nodes using duplicate detection (Sec. IV-F). Thus, we need to show that even with pruning, RCS_BASIC will still find a plan. This will be done by showing that the same sequence of motion primitives can be applied to configurations that are "similar" to $\mathbf{x}_0 \ldots \mathbf{x}_n$ and the resultant plan $\tilde{\sigma}$ exists using the fact that $\tilde{\sigma}$ is "similar" to σ and that σ has γ-clearance. The rest of this proof formalizes this idea.

Recall that (**C3**) ensures that $d_{\text{sim}} < \delta\ell_{\min}$ which guarantees that any motion primitive will end up at a non-similar configuration. Now, let $\mathbf{x}_i$ be the first configuration that is pruned because of a similar configuration (see Alg. 1, line 7). We will say that $\mathbf{x}_i$ is *replaced* by the similar configuration $\mathbf{x}'_i$. As $i \geq 1$, in the worst case we have $i = 1$. We then apply $M_{\sigma'}[2,n]$ to $\mathbf{x}'_1$. According to (**C1**), the maximal error accumulated to $\mathbf{x}'_n = \mathbf{x}'_1 + M_{\sigma'}[2,n]$ is $\varepsilon_1 = \rho(\mathbf{x}'_n, \mathbf{x}_n) = L_s^{n-1} \cdot d_{\text{sim}}$.

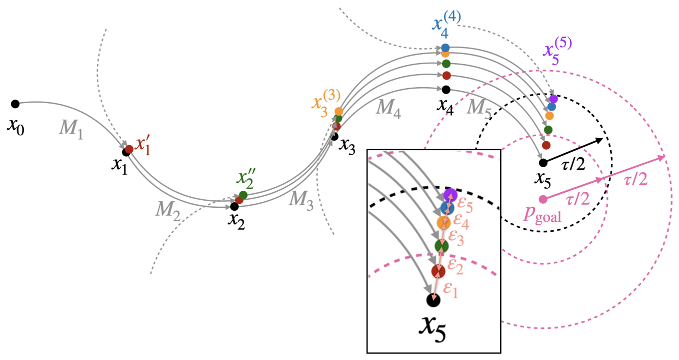


Fig. 11. A 2D illustration of configuration pruning. σ is shown as black nodes, the plan after $\mathbf{x}_1'$ prunes $\mathbf{x}_1$ is shown as red nodes, the plan after $\mathbf{x}_2''$ prunes $\mathbf{x}_2'$ is shown as green nodes, the plan after $\mathbf{x}_3^{(3)}$ prunes $\mathbf{x}_3''$ is shown as yellow nodes, the plan after $\mathbf{x}_4^{(4)}$ prunes $\mathbf{x}_4^{(3)}$ is shown as blue nodes, and the pruning configuration $\mathbf{x}_5^{(5)}$ is shown as a purple node. The solid circular arrows represent elements in M_σ, and the dashed circular arrows represent connections to predecessors of the pruning configurations. In this particular example, as long as we guarantee that $\|\text{Proj}(\mathbf{x}_5) - p_{\text{goal}}\|_2 \leq \frac{\tau}{2}$ and that $\varepsilon = \sum_{i=1}^{5} \varepsilon_i \leq \frac{\tau}{2}$, the resultant plan which ended at $\mathbf{x}_5^{(5)}$ still satisfies the required goal tolerance.

Similarly, when $\mathbf{x}_2'$ is replaced by $\mathbf{x}_2''$, we apply $M_{\sigma'}[3,n]$ to $\mathbf{x}_2''$ and for $\mathbf{x}_n'' = \mathbf{x}_2'' + M_{\sigma'}[3,n]$, the accumulated error is $\varepsilon_2 = \rho(\mathbf{x}_n'', \mathbf{x}_n') = L_s^{n-2} \cdot d_{\text{res}}$. The same analysis applies for $\{\mathbf{x}_3, \ldots, \mathbf{x}_n\}$. According to (**C3**), the total accumulated error then becomes:

$$\begin{aligned}\varepsilon = \rho(\mathbf{x}_n^{(n)}, \mathbf{x}_n) &\leq \rho(\mathbf{x}_n', \mathbf{x}_n) + \cdots + \rho(\mathbf{x}_n^{(n)}, \mathbf{x}_n^{(n-1)}) \\ = \varepsilon_1 + \cdots + \varepsilon_n &= \frac{L_s^n - 1}{L_s - 1} \cdot d_{\text{res}} < \frac{\tau}{2} \cdot \frac{L_s^n - 1}{L_s^H - 1} \leq \frac{\tau}{2}.\end{aligned}$$

According to (**C4**), we have that $\|\text{Prog}(\mathbf{x}_n) - g_{\text{goal}}\|_2 \leq \frac{\tau}{2}$. Thus,

$$\begin{aligned}&\|\text{Proj}(\mathbf{x}_n^{(n)}) - p_{\text{goal}}\|_2 \\ &\leq \|\text{Proj}(\mathbf{x}_n^{(n)}) - \text{Proj}(\mathbf{x}_n)\|_2 + \|\text{Proj}(\mathbf{x}_n) - p_{\text{goal}}\|_2 \\ &\leq \tau/2 + \tau/2 = \tau.\end{aligned}$$

This implies that even in the worst case where all possible replacements happen, the final configuration $\mathbf{x}_n^{(n)}$ still satisfies the required goal tolerance (see Fig. 11).

Additionally, we prove that when pruning happens for $\mathbf{x}_i^{(j)}$, the motion plan constructed with $M_{\sigma'}[i,n]$ is still collision-free. We have shown above that $\rho(\mathbf{x}_n^{(n)}, \mathbf{x}_n) < \frac{\tau}{2}$. Moreover, we have that $\forall i \in [0,n], \rho(\mathbf{x}_i^{(n)}, \mathbf{x}_i) < \frac{\tau}{2}$ since less error is accumulated for $i < n$. Thus we have that

$$\forall k \in [i,n], \|\text{Proj}((\mathbf{x}_k^{(j)}) - \text{Proj}((\mathbf{x}_k)\|_2 \leq \rho(\mathbf{x}_k^{(j)}, \mathbf{x}_k) < \frac{\tau}{2}.$$

And for any configuration $\tilde{\mathbf{x}}$ along edge $(\mathbf{x}_k^{(j)}, \mathbf{x}_{k+1}^{(j)})$, we have that

$$\begin{aligned}\min\{&\|\text{Proj}(\tilde{\mathbf{x}}) - \text{Proj}(\mathbf{x}_k)\|_2 \\ &\|\text{Proj}(\tilde{\mathbf{x}}) - \text{Proj}(\mathbf{x}_{k+1})\|_2\} < \frac{\tau + \delta\ell_{\min}}{2}.\end{aligned}$$

According to (**C4**), $\gamma > \frac{\tau+\delta\ell_{\min}}{2}$, there always exist a small value $\varepsilon = \gamma - \frac{\tau+\delta\ell_{\min}}{2}$. Thus, as long as σ' is an ε-approximation of σ, $\tilde{\sigma}$ is then guaranteed to be a γ-approximation of σ. σ's strong γ-clearance guarantees that the motion plan constructed with $M_{\sigma'}[i,n]$ is collision-free.

To summarize, as long as the required conditions are satisfied, RCS_BASIC still finds a motion plan. ■

F. Resolution completeness while incorporating implementation details

Corollary 2. *RCS and RCS_PARA will also find a solution in finite time, if the conditions in Thm. 2 are satisfied. In other words, none of the implementation details hinder the resolution completeness guarantees.*

Proof sketch. For RCS, goal reachability checks only reject invalid nodes, direct goal connection only provides early terminations without affecting the search tree, and equivalent-node pruning provides an efficient way to reject identical configurations early.

For RCS_PARA, parallelization may change the order of processing nodes, but does not change the essence of the proofs. Thus, RCS and RCS_PARA also find a motion plan as RCS_BASIC does. ■

Appendix B. Planner parameters for evaluation

In this section we describe the parameters used by each planner. For the precise definition of the different parameters, the reader is referred to the original papers describing the RRT-based algorithm [40] and AFT [41].

(i) RRT: We set goal biasing: 5% and direct goal connecting ratio: 100%. The multi-threaded version of RRT, denoted as RRT_PARA, was implemented with Motion Planning Templates (MPT) [22] and used 60 threads.
(ii) AFT: We used two tree refinements and used the cost function defined in Eq. 5. Additionally, we used five levels of increments of the fractal structure, a tree density of 17, and we rotated the tree around the root axis 10 times, each time with a step of $\frac{\pi}{5}$rad (these values were chosen according to the analysis provided by Pinzi et al. [41]).
(iii) RCS: The system cutoff resolution is computed for control frequency 40Hz, which corresponds to a time interval of 0.025s: $\delta\ell_{\min} = 5(\text{mm/s}) \cdot 0.025\text{s} = 0.125\text{mm}, \delta\theta_{\min} = 2\pi(\text{rad/s}) \cdot 0.025\text{s} \approx 0.157\text{rad}$. The value of insertion and rotation velocities are taken from [45] and the control frequency is the measurement rate of the NDI Aurora tracking system [1]. Maximum length step: $\delta\ell_{\max} = 20\text{mm}$. Distance metric weighting parameter: $\alpha = 0.05$. In addition, we empirically determined that $d_{\text{sim}} = 5.5e - 5$. We use 60 threads for RCS_PARA, the multi-threaded version of RCS.

As is mentioned in Sec. III, for RCS, we determine the finest resolution by considering the hardware's ability to measurably change the steerable needle tip's position and orientation in tissue. We use conventional constant insertion

TABLE I
PLANNER PERFORMANCE COMPARISON

	RRT_PARA	AFT	RCS_PARA
Success rate	91.2%	65.8%	97.6%
Avg. relative length	0.998	1.003	1.0
Avg. targeting error	0.053mm	0.207mm	0.051mm

and rotational velocities (as are commonly used in steerable needle robots) and the magnetic tracker reading frequency (commonly used for tracking steerable needle tips) to determine the minimal motions. These real-world minimal motions of the steerable needle tip are the minimal motions explored by the search.

APPENDIX C. ADDITIONAL EXPERIMENTS

In this section we present additional experiments evaluating the quality of the plans produced by each planner. More specifically, we focus on the trajectory length $\ell(\sigma)$ and the final targeting error $\|\sigma(1) - p_{\text{goal}}\|_2$.

For AFT, both are considered in the cost function. For RCS and RRT, although the plan quality is not explicitly optimized, as more running time is given, there is a chance to improve the plan quality. For both planners, we use the same cost function defined in Eq. 5 to pick a plan with the lowest cost from all motion plans generated. Since RCS and RRT only consider a plan to be valid if it satisfies the required targeting error, the final resulting plan is guaranteed to satisfy the targeting error. Similar to the previous comparison, RRT_PARA and RCS_PARA are allotted a running time of 100 seconds, and the planner keeps running after the first solution is found to generate more plans. We pick as the final result the solution with minimal cost among all solutions found. AFT uses two tree refinements. The results are shown in Table I. Since different test cases have different ranges of plan length, we take the best plan produced by RCS_PARA as the baseline, and compute the plan length relative to it. Values in Table I are averaged over all test cases that are successfully solved by all planners.

Due the limited insertion length and the maximum curvature constraint, all three planners produced plans with (roughly) similar lengths. RRT_PARA computed the lowest-cost trajectory on average. This may due to the steer function in RRT always trying to connect to a sampled point with the shortest arc. For the two search-based methods, RCS_PARA achieved better plan length since the resolution in RCS can be much finer than that of AFT. As for the targeting error, because RCS_PARA and RRT_PARA both try to connect to the goal point directly, they can efficiently reduce the targeting error and achieve average targeting errors much smaller than AFT.