INTEGRATING OPTIMIZATION AND SAMPLING FOR ROBOT MOTION PLANNING WITH APPLICATIONS IN HEALTHCARE

Alan Kuntz

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the College of Arts and Sciences.

Chapel Hill
2019

Approved by:

Ron Alterovitz

Nancy M. Amato

Jessica Burgner-Kahrs

Parasara Sridhar Duggirala

Marc Niethammer

ABSTRACT

Alan Kuntz: Integrating Optimization and Sampling for Robot Motion Planning with Applications in Healthcare
(Under the direction of Ron Alterovitz)

Robots deployed in human-centric environments, such as a person's home in a home-assistance setting or inside a person's body in a surgical setting, have the potential to have a large, positive impact on human quality of life. However, for robots to operate in such environments they must be able to move efficiently while avoiding colliding with obstacles such as objects in the person's home or sensitive anatomical structures in the person's body. Robot motion planning aims to compute safe and efficient motions for robots that avoid obstacles, but home assistance and surgical robots come with unique challenges that can make this difficult. For instance, many state of the art surgical robots have computationally expensive kinematic models, i.e., it can be computationally expensive to predict their shape as they move. Some of these robots have hybrid dynamics, i.e., they consist of multiple stages that behave differently. Additionally, it can be difficult to plan motions for robots while leveraging real-world sensor data, such as point clouds.

In this dissertation, we demonstrate and empirically evaluate methods for overcoming these challenges to compute high-quality and safe motions for robots in home-assistance and surgical settings. First, we present a motion planning method for a continuum, parallel surgical manipulator that accounts for its computationally expensive kinematics. We then leverage this motion planner to optimize its kinematic design chosen prior to a surgical procedure. Next, we present a motion planning method for a 3-stage lung tumor biopsy robot that accounts for its hybrid dynamics and evaluate the robot and planner in simulation and in inflated porcine lung tissue. Next, we present a motion planning method for a home-assistance robot that leverages real-world, point-cloud obstacle representations. We then expand this method to work with a type of continuum surgical manipulator, a concentric tube robot, with point-cloud anatomical representations. Finally, we present a data-driven machine learning method for more accurately estimating the shape of concentric

tube robots. By effectively addressing challenges associated with home assistance and surgical robots operating in human-centric environments, we take steps toward enabling robots to have a positive impact on human quality of life.

To my family and mentors.

ACKNOWLEDGEMENTS

First, I would like to thank my graduate advisor, Ron Alterovitz, for his invaluable support and guidance in matters both technical and otherwise throughout my time in graduate school. I would like to thank my undergraduate advisor, Lydia Tapia, for encouraging me to pursue a graduate education and setting me up for success as I continue my career. I would also like to thank my committee members for their valuable time, guidance, support, and feedback.

Next, I would like to thank the other students with whom I've shared a lab, offices, and the graduate experience. Specifically, I would not have been successful without the mentorship of Luis Torres, Jeff Ichnowski, Chris Bowen, and Kasra Manavi.

This work was made possible by the National Science Foundation (NSF) under awards IIS-1149965, CNS-1305286, and CCF-1533844, and the National Institutes of Health (NIH) under awards R21EB017952 and R01EB024864.

Finally, I would like to thank my family for their endless love, patience, and support, without whom this would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

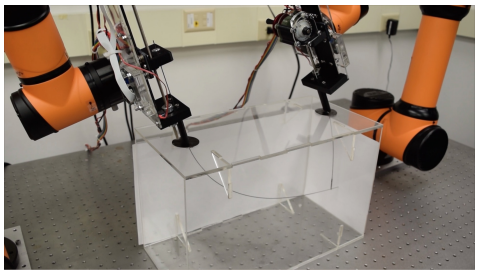
Introduction

In order for robots to have a large, positive impact on human quality of life, they must be able to safely and effectively operate in human-centric environments. One area in which robots have a large potential for positive impact is healthcare. Robots have the potential to assist people in their homes with tasks that these people may be unable to perform themselves due to age, disease, disability, or other factors. Robots in a surgical setting have the potential to increase the capabilities of surgeons, enabling the surgeons to perform new or existing procedures more safely and on a larger class of patients than is currently possible.

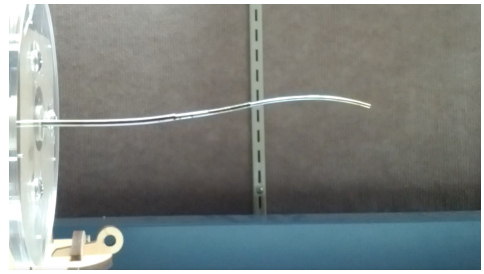
In both of these human-centric settings, robots must be capable of operating safely, avoiding unintentional collisions with obstacles in their environments. In a home assistance setting, the obstacles a robot must avoid may be the humans or objects in the humans' home. In a surgical setting, the obstacles may be inside the humans themselves—anatomical obstacles, such as bones, nerve bundles, or vasculature. In addition to avoiding unintended collision with obstacles, the robots must be effective at performing the tasks required of them. One approach to enabling robots to operate in a safe and effective manner in human-centric environments is *robot motion planning*.

Robot motion planning methods compute motions for a robot to perform a user-specified task subject to a set of constraints. These constraints can be problem and robot specific, and frequently include kinematic constraints (such as joint limits), differential constraints (such as a minimum turning radius along the robot's path), and environmental constraints (such as obstacle avoidance described above). Then, under a set of assumptions (such as perfect modeling and knowledge of the environment), any computed motion that satisfies these constraints is known to be safe, as it does not collide with the environment, as well as feasible, i.e., it conforms to the robot's kinematic requirements.

In many settings safe and feasible may not be sufficient, and it is desirable to consider some notion of cost in order to compute *high-quality* motions. One popular cost metric is the *length* of the

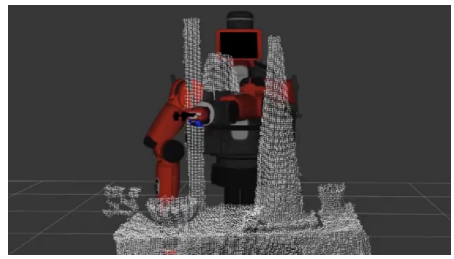


(a) CRISP Robot



(b) Concentric Tube Robot

(c) Three-stage Continuum Robot



(d) Baxter Robot

Figure 1.1: We plan safe and effective motions for a variety of different types of robots, including those shown here. (a) The Continuum Reconfigurable Incisionless Surgical Parallel (CRISP) robot is composed of multiple needle-diameter tubes that are assembled into a parallel structure using snares [1, 2]. (b) A concentric tube robot is composed of multiple telescoping precurved flexible tubes [3, 4]. (c) A three-stage continuum robot combines a tendon-actuated bronchoscope with concentric tubes and a steerable needle [5]. (d) A Baxter robot is a commercially available robot with two 7 degree of freedom (DOF) serial-link manipulator arms [6].

motion plan. A high-quality plan under such a metric would ensure that the plan avoids unnecessary motion, reducing the time and energy spent by the robot to execute the task. Another relevant cost metric is *clearance from obstacles*, wherein a high-quality plan would travel far from the obstacles in the environment, improving the safety of the plan as it is executed.

In this dissertation, we present and evaluate novel algorithms that compute safe (constraint satisfying) and effective (high-quality) motion plans for robots, such as those shown in Fig. 1.1, in healthcare applications. In order to do so, our methods leverage a combination of sampling- and optimization-based methods.

1.1 Challenges

Our contributions in this work focus on developing methods that aim to address the following specific challenges:

Computationally expensive kinematic models One class of robot that we focus on is continuum surgical robots. Continuum surgical robots are robots that are capable of taking curved shapes which allow them to safely travel deep into the human body [7], enabling minimally invasive surgical procedures (for examples of such systems, see Figs. 1.1a, 1.1b, and 1.1c). Successfully planning motions for such systems will allow for less invasive surgical procedures as they curve around sensitive anatomical structures. However, computing the anticipated shape of continuum surgical robots frequently takes orders of magnitude longer than computing the shape of traditional robotic systems such as serial-link manipulators [4, 8]. This limitation must be addressed for motion planning to be successful for such systems.

Parameters set pre-procedure which greatly impact capabilities For systems such as the Continuum Reconfigurable Incisionless Surgical Parallel (CRISP) robot (see Fig. 1.1a), decisions are made prior to the surgical procedure such as where on the patient's body the robot is inserted and the robot's parallel structure. We refer to these parameters as the robot's *kinematic design*. Decisions about the robot's kinematic design are frequently made heuristically and can have a large impact on the success of the surgical procedure.

Hybrid and highly-constrained kinematics in multistage robots The 3-stage robot shown in Fig. 1.1c consists of multiple robotic stages that are deployed in sequence and that all behave differently. Some of these stages are holonomic, while others are non-holonomic, making it difficult to efficiently integrate the stages into a single motion planning framework. Additionally, decisions made when planning motions for each of the stages dramatically impacts the stages that are deployed later in time. This property must be accounted for when planning motions for such hybrid systems.

Quickly planning motions using point cloud obstacle representations In order to make systems more reactive to changing environments, motion planning algorithms should work quickly to produce high-quality motion plans in order to move while their knowledge of the environment is relevant. Many environmental sensors, such as laser scanners, stereo cameras, and RGB-D sensors, produce point clouds to represent the robot's environment. Such sensors can provide high quality information about a robot's environment and can be deployed both in a home assistance setting and in a surgical setting (frequently through an endoscope). However, planning motions with point cloud

environments can be difficult due to the large number of points and their lack of inherent structure.

Inaccurate shape models For some continuum robot systems, such as concentric tube robots (see Fig. 1.1b) the physics-based shape models may be inaccurate due to issues such as material inhomogeneities and inconsistent and difficult to predict physical phenomena such as friction [9, 10]. Motion planning and control of these systems rely on the assumption that the shape model is accurate.

1.2 Contributions

Many methods have already been developed for planning motions for a variety of robots. However, these methods frequently do not work well in the presence of the challenges described above. As such, in this work we adapt, integrate, and develop novel algorithms specifically to address the above challenges as we plan motions for these systems. We discuss our methods briefly here and in full detail in the corresponding chapters.

1.2.1 Motion Planning for Continuum Reconfigurable Incisionless Surgical Parallel Robots

One system that suffers from the challenge of a computationally expensive kinematic model is the CRISP robot. CRISP robots consist of multiple needle-diameter flexible instruments that are assembled into a parallel structure inside the human body. With a camera placed at the tip of one of the instruments, the CRISP robot can be used to inspect anatomical sites in constrained body cavities in a minimally invasive manner. We introduce a motion planner for CRISP robots that computes manipulations of the flexible instruments outside the body such that the camera can visually inspect a user-specified site of clinical interest inside the body. Our sampling-based motion planner ensures avoidance of collisions with anatomical obstacles inside the body, enforces remote-center-of-motion constraints on the instrument's entry points into the body, and efficiently handles the expensive computation of CRISP robot kinematics. We also extend the motion planner to estimate the set of points inside a body cavity that can be visually inspected by the camera of a CRISP robot for a given kinematic design, i.e., choice of parallel structure for the robot and entry points into the body. We demonstrate our method's efficacy in a simulated endoscopic medical procedure in the pleural space around a lung. This contribution is discussed in Chapter 2 and was also presented in [8].

1.2.2 Kinematic Design Optimization of a Parallel Surgical Robot to Maximize Anatomical Visibility via Motion Planning

In order to address the challenge of effectively setting pre-procedure parameters, we introduce a method to optimize, on a patient-specific basis, the kinematic design of the CRISP robot. Our objective is to maximize the ability of the robot's tip camera to view tissue surfaces in constrained spaces. The kinematic design of the CRISP robot, which greatly influences its ability to perform a task, includes parameters that are fixed before the procedure begins, such as entry points into the body and parallel structure connection points. We combine a global stochastic optimization algorithm, Adaptive Simulated Annealing (ASA), with the motion planner designed for the CRISP robot. ASA facilitates exploration of the robot's design space while the motion planner enables evaluation of candidate designs based on their ability to successfully view target regions on a tissue surface. By leveraging motion planning, we ensure that the evaluation of a design only considers motions which do not collide with the patient's anatomy. We analytically show that the method asymptotically converges to a globally optimal solution and demonstrate our algorithm's ability to optimize kinematic designs of the CRISP robot on a patient-specific basis. This contribution is discussed in Chapter 3 and was presented in [11].

1.2.3 Motion Planning for a Three-Stage Multilumen Transoral Lung Access System

In this contribution, we develop a motion planner for a multilumen transoral lung access system (shown in Fig. 1.1c), a new system that has the potential to perform safe biopsies anywhere in the lung—enabling more effective early-stage diagnosis of lung cancer. The system consists of three cascading stages in which a bronchoscope is deployed transorally to the lung, a concentric tube robot pierces through the bronchial tubes into the lung parenchyma, and a steerable needle deploys through a properly oriented concentric tube and steers through the lung parenchyma to the target site while avoiding anatomical obstacles such as significant blood vessels. We present a sampling-based motion planner that computes actions for each stage of the system while addressing the challenge of the system's hybrid and highly-constrained kinematics and considering the coupling of the stages in an efficient manner. We demonstrate the motion planner's fast performance and ability to compute plans with high clearance from obstacles in simulated anatomical scenarios. This contribution is discussed in Chapter 4 and was originally presented in [12].

1.2.4 Toward Transoral Peripheral Lung Access: Steering Bronchoscope-Deployed Needles through Porcine Lung Tissue

In this work, we physically deploy and evaluate the efficacy of the three-stage lung tumor biopsy robot in biological tissue. We deploy the robot into inflated porcine lung and demonstrate the robot's ability to steer around anatomical obstacles to targets in the periphery of the lung with high accuracy. We run a number of trials in the lung achieving average needle-tip errors in the 1 mm to 2 mm range. We also gather CT scans of the robot's deployment in the lung to visualize the robot's placement in the anatomy. This contribution is discussed in Chapter 5 and was originally presented in [13].

1.2.5 Fast Anytime Motion Planning in Point Clouds by Interleaving Sampling and Interior Point Optimization

Robot manipulators operating in unstructured environments, such as in home assistance settings, need to plan their motions quickly while relying on real-world sensors, which typically produce point clouds. To enable intuitive, interactive, and reactive user interfaces, the motion plan computation should provide high-quality solutions quickly and in an anytime manner, meaning the algorithm progressively improves its solution and can be interrupted at any time and return a valid solution. To address these challenges, we combine two paradigms: (1) asymptotically-optimal sampling-based motion planning, which is effective at providing anytime solutions but can struggle to quickly converge to high quality solutions in high dimensional configuration spaces, and (2) optimization, which locally refines paths quickly. We propose the use of interior point optimization for its ability to perform in an anytime manner that guarantees obstacle avoidance in each iteration, and we provide a novel lazy formulation that efficiently operates directly on point cloud data. Our method iteratively alternates between anytime sampling-based motion planning and anytime, lazy interior point optimization to compute high quality motion plans quickly, converging to a globally optimal solution. This contribution is discussed in Chapter 6 and was originally presented in [14].

1.2.6 Planning High-Quality Motions for Concentric Tube Robots in Point Clouds via Parallel Sampling and Optimization

We present a method that plans motions for a concentric tube robot to automatically reach surgical targets inside the body while avoiding obstacles, where the patient's anatomy is represented by point clouds. Point clouds can be generated intra-operatively via endoscopic instruments, enabling the system to update obstacle representations over time as the patient anatomy changes during

surgery. As in the previous contribution, our motion planning method uses a combination of sampling-based motion planning methods and local optimization, via an interior point method, to efficiently handle point cloud data and quickly compute high quality plans. This ensures that the computed plan is feasible and avoids obstacles at every iteration, in an anytime fashion. Building upon the previous contribution, rather than interleaving the sampling- and optimization-based methods we instead present a framework that leverages parallelism to efficiently do both simultaneously. Additionally, rather than minimizing the motion plan's length, we instead minimize a cost metric that encourages clearance from obstacles, promoting safer motion plans. We demonstrate the method's efficacy in three anatomical scenarios, including two generated from endoscopic videos of real patient anatomy. This contribution is discussed in Chapter 7 and will appear in [15].

1.2.7 Estimating the Complete Shape of Concentric Tube Robots via Learning

In order to successfully plan motions for concentric tube robots, the motion planner needs to be able to accurately determine the shape of the robot as it moves through the world. Traditionally, this is done using physics-based mechanical models that are based on complex mechanical concepts such as Cosserat rod theory. However, these models have difficulty accurately accounting for all of the complex physical phenomenon associated with the robot in the real world, such as friction between the tubes. As a result, the mechanical models are frequently inaccurate. We propose a data-driven, machine learning based approach using a deep neural network to solve this problem.

To apply a data-driven approach to solving this problem, we need three pieces. First, we require a parameterization of the robot's joint values to be used as input to the model—we leverage a parameterization derived in recent work on learning the forward kinematics of concentric tube robots that was shown to be effective [16]. Next, a parameterization of the robot's shape in space is necessary to be used as the output of the model—we derive a 3D shape function based on linear combinations of arc-length parameterized orthonormal basis functions, allowing the neural network to output the entire shape of the robot in one pass with a relatively low dimensional output ($3 \times$ the number of basis functions, 15 dimensions in this work). Finally, we must train the model on real world data that maps between the two—we use a method from multiview computer vision, called shape from silhouette [17], to generate real-world data mapping the concentric tube robot's configuration space to its shape in the real world. This contribution is discussed in Chapter 8 and was originally presented in [18].

1.3 Thesis Statement

This dissertation proposes the following thesis:

For robots operating in human-centric environments, robot motion planning, via a combination of sampling- and optimization-based methods, enables fast and high-quality kinematic design optimization and motion computation while overcoming challenges associated with computationally expensive kinematic models, hybrid and highly-constrained kinematics, and point-cloud obstacle representations.

Each chapter of this dissertation supports this thesis statement as outlined below.

1.4 Organization

In Chapter 2 we propose a novel sampling-based motion planning algorithm to enable a robot with complex, computationally expensive kinematics—the CRISP robot—to safely move through the body. In Chapter 3 we integrate the CRISP robot's sampling-based motion planner with a global optimization method to optimize the pre-procedure kinematic design of the system on a patient-specific basis. In Chapter 4 we develop a sampling-based motion planning algorithm that plans motions for a three-stage lung tumor biopsy robot with hybrid and highly-constrained kinematics to safely steer needles through the complex anatomy of a person's lungs. In Chapter 5 we physically evaluate the three-stage robot's ability to safely and accurately steer through inflated porcine lung tissue. In Chapter 6 we present a method that interleaves sampling- and optimization-based motion planning techniques to plan motions in point cloud obstacle environments for a serial link manipulator robot. In Chapter 7 we extend the above method to work for concentric tube robots and parallelize the algorithm to plan motions more quickly than otherwise possible. Finally, in Chapter 8 we present a learned shape model for concentric tube robots that aims to help safely plan and execute motions for surgical tasks in the human body. We then conclude in Chapter 9.

CHAPTER 2

Motion Planning for Continuum Reconfigurable Incisionless Surgical Parallel Robots

The Continuum Reconfigurable Incisionless Surgical Parallel (CRISP) robot [1] is a new type of continuum robot [7, 19] that consists of multiple needle-diameter flexible instruments that are assembled together inside a body cavity to perform minimally invasive medical procedures. The CRISP robot typically includes (1) a flexible instrument with a working channel through which a tool (e.g., chip-tip camera, ablation probe, etc.) is passed, and (2) one or more additional flexible instruments that are inserted into the body cavity and attach to the first instrument via snares, creating a strong, parallel kinematic structure (see Fig. 2.1). The parallel nature of the structure provides strength to the robot, enabling the device to apply larger forces during medical procedures when required. The tool's tip can be repositioned and reoriented inside the body by robotically moving the instruments outside the body in concert.

The CRISP robot is an ideal platform for inspecting and manipulating tissues on the surface of a pleural effusion, which is a collection of excess fluid in the pleural space around the lungs. Pleural effusions can be caused by over 50 different diseases [20]. Accurate diagnosis of the disease is critical, as the underlying cause can be deadly and may have drastically different treatment paths. Thoracoscopy is the gold standard and involves insertion of endoscopic tools through the ribs [21]. Endoscopic tools give clinicians direct visualization of the pleural space. However, thoracoscopy is invasive: it requires incisions, and major complications are reported to be as high as 15% [22]. A CRISP robot, with a chip-tip camera deployed through the tool working channel, has the potential to combine the minimal invasiveness of needles with the ability of endoscopic tools to systematically inspect the interior surface of a patient's pleural effusion.

We introduce a motion planner for CRISP robots that manipulates the flexible instruments outside the body such that the tool tip camera can see a user-specified site of clinical interest inside the body. The motion planner computes motions such that all the flexible instruments inside

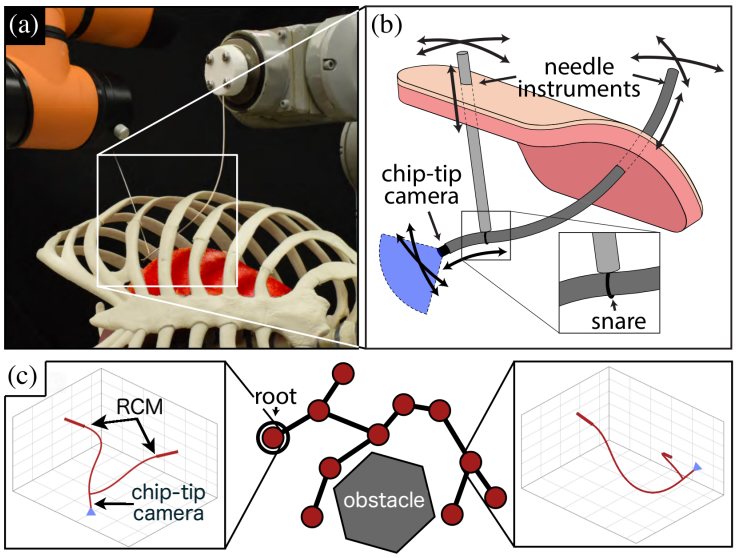


Figure 2.1: An overview of the CRISP robotic system and motion planning framework. (a) The needle-diameter flexible instruments form a parallel structure inside the body whose shape is modified by actuating the instruments outside the body. (b) The instruments can be inserted and rotated to change the view of the camera at the tip. (c) The motion planner incrementally computes a tree data structure of collision-free robot configurations, which can be used to manipulate the instruments (shown in red) outside the body to reposition and reorient the camera tip while ensuring the instruments avoid anatomical obstacles inside the body.

the body avoid collision with anatomical obstacles, including the chest wall, the lung surface, and potential connections between the lung and chest wall. The configuration of the CRISP robot is the position and orientation of the instruments outside the body, and the motion planner computes a sequence of configurations that avoids collisions with anatomical obstacles inside the body, enforces remote-center-of-motion constraints on the tube's entry points into the body, and enables visibility of the desired clinical site inside the body with the tool's camera. Motion planning for CRISP robots is challenging because evaluating their kinematics for each configuration requires modeling the elastic and torsional interactions of the robot's constituent tubes, which is computationally expensive. We introduce a sampling-based motion planner that efficiently propagates presolved state information for the kinematic model through a tree data structure in configuration space to accelerate motion plan computation.

The set of sites that can be inspected via the camera at the tool tip of a CRISP robot is heavily influenced by the robot's setup, i.e., where on the skin surface the tubes enter the body and where the snares grasp the tool instrument. We demonstrate how our motion planner can be used to both

estimate the set of points on the pleural effusion surface that can be seen by the tool tip camera of a CRISP robot for a given setup, as well as provide collision free motion plans for the robot to view the points on the pleural effusion surface. This analysis can provide physicians with insights into CRISP robot setups that are appropriate for specific clinical tasks that require pointing the camera at specific sites in the pleural effusion.

We demonstrate the speed and effectiveness of our new motion planner for CRISP robots in simulation using a pleural effusion segmented from a patient CT scan. We demonstrate both the method's ability to plan motions for the robot to view specific clinically relevant sites as well as the ability to estimate the set of points that can be seen by the CRISP robot's tool tip camera.

This chapter is based on work previously published in [8].

2.1 Related Work

Motion planning for the CRISP robot [1] is influenced by the way the shape of the robot is calculated. This influence is not unique to the CRISP robot and is a consideration present in motion planning for other continuum surgical robots.

One continuum surgical robot with related mechanics is the concentric tube robot [23]. Concentric tube robots are needle-like surgical manipulators composed of thin, nested, pre-curved nitinol tubes. Similar to the mechanics of the CRISP robot, the tubes of the concentric tube robot elastically interact in different configurations to influence the robot's shape. Using various control methods, prior work has achieved position control of concentric tube robot tips [24, 9, 25]. Sampling based motion planning has also been used to control concentric tube robots. Torres et. al. use a combination of a precomputed roadmap and an inverse kinematics controller to achieve interactive rate planning for concentric tube robots [4]. Lyons et. al. apply optimization-based motion planning using a simplified kinematics model [26].

Another related medical robotic device for interventional medical procedures is steerable needles, which are composed of a highly flexible tube and employ an asymmetric tip to steer through soft tissue [27]. Motion planning for steerable needles has been achieved in a variety of ways [28, 29, 30], including sampling-based motion planning [31, 32, 12].

Automatically controlling the motion of cameras to view specific sites can be challenging, as discussed by Christie et al. [33]. Rosell et al. plan motions of a virtual bronchoscope to view lesions

in the lung through the airway [34] but are restricted in their motion to the structure of the bronchial tree. In non-medical applications, probabilistic roadmaps have been applied to plan camera paths in virtual environments when given a specified goal position and orientation for the camera [35]. There has also been work in computer vision on how to plan new viewpoints for a camera such that object recognition is optimized [36, 37]. These works primarily consider how to plan the viewing angles of the camera, while we primarily focus on motion planning for a medical robot that contains a camera for purposes of viewing specific sites in cluttered and constrained spaces.

2.2 Problem Formulation

2.2.1 CRISP Robot

We consider a CRISP robot composed of N needle-diameter tubes. One of these tubes has a chip-tip camera affixed to its tip which we refer to as the camera tube, ρ_{c}. The remaining $N-1$ tubes are deployed with snares, and will be referred to as snare tubes, ρ_k, where k is an integer uniquely identifying a specific snare tube. In order to perform accurate mechanical modeling, we require as input each tube's inner diameter (ID) and outer diameter (OD). We also require a description of the chip-tip camera, in the form of its angular field of view, θ_v.

A CRISP robot's set of tubes can be assembled into parallel structures inside the patient's body in an infinite number of ways. We require as input a description of the CRISP robot state (illustrated in Fig. 2.2). We make a distinction between the CRISP robot's *setup* state and the robot's *actuatable* state. We define the CRISP robot's *setup* state as:

$$\{\mathbf{r}_{\mathrm{c}}, \mathbf{r}_1, \ldots, \mathbf{r}_{N-1}, s_1, \ldots, s_{N-1}\}, \tag{2.1}$$

where $\mathbf{r}_{\mathrm{c}} \in \mathbb{R}^3$ and $\mathbf{r}_k \in \mathbb{R}^3, k = 1, \ldots, N-1$ denotes the tubes' entry points into the patient's body, expressed in a global coordinate system, and the scalars s_k denote the arc length along the camera tube at which the kth snare tube attaches to the camera tube. The subscript c denotes a value corresponding to the camera tube, and an integer k denotes the value corresponding to the kth snare tube. This *setup* state is set prior to the surgical procedure and is not varied during motion execution. The entry points $\mathbf{r}$ remain fixed as remote center of motion (RCM) constraints, preventing the system's tubes from pulling laterally on the patient's body during maneuvers, and the snare grasping locations s_k remain fixed throughout the surgical procedure. We then define the

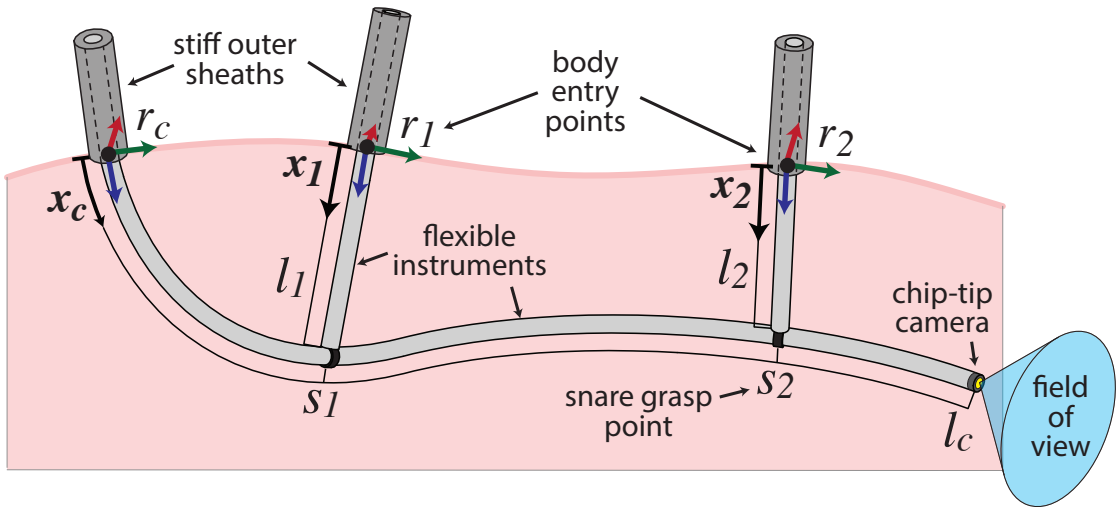


Figure 2.2: A stiff outer sheath introduces the tubes into the body entry points $\mathbf{r}_c$, $\mathbf{r}_1$, and $\mathbf{r}_2$. The snares grasp the camera tube at arc lengths s_1 and s_2. A mechanics-based model predicts the states of the camera tube $\mathbf{x}_c$ and the snare tubes, $\mathbf{x}_1$ and $\mathbf{x}_2$, in arc length.

CRISP robot's *actuatable* state as:

$$\{R_c, R_1, \ldots, R_{N-1}, \ell_c, \ell_1, \ldots, \ell_{N-1}\}, \tag{2.2}$$

where $R \in SO(3)$ denotes the tubes' orientations at their entry points as expressed in a global coordinate system, and the scalars ℓ describe how far each of the tubes are inserted beyond their respective entry points into the body (with a maximum insertion length for each tube imposed by the physical robot). This *actuatable* state represents the robot's state which will be varied during the execution of motions during the surgical procedure. The CRISP robot's total state then becomes the union of the setup and actuatable states.

2.2.2 Motion Planning

We consider the problem of planning motions for a CRISP robot. To produce motion, each tube can be actuated via changing its orientation at the entry point into the pleural effusion space and translating the tube into and out of the space.

We define an instance of the robot's actuatable state as a configuration

$$\mathbf{q} = \{R_c, R_1, \ldots R_{N-1}, l_c, l_1, \ldots, l_{N-1}\}.$$

The space of all configurations the robot can assume is then $Q \subseteq \mathcal{SO}(3)^N \times \mathbb{R}^N$.

For a given configuration $\mathbf{q} \in Q$ we define the shape of the CRISP robot as a function

$$\mathbf{P}(\mathbf{q}, \rho, s) : \mathcal{SO}(3)^N \times \mathbb{R}^N \times \mathcal{N} \times \mathbb{R} \mapsto \mathbb{R}^3.$$

Function $\mathbf{P}$ is a 3D space curve representing the backbone of tube ρ, at arc length s in the domain $[0, l_{\rho\max}]$. Function $\mathbf{P}$, combined with knowledge of the cross-sectional outer diameter (OD) of each tube allows us to calculate the shape of the entire CRISP robot. Also note, that as a special case, $\mathbf{p}_{\text{camera}}$ is the 3D location of the camera on the tip of ρ_{c}, and it has a direction of view defined by the vector $\mathbf{v}_{\text{camera}}$ which is tangent to the space curve at $\mathbf{p}_{\text{camera}}$.

We then define a motion plan $\sigma = (\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n)$ as an ordered sequence of robot configurations. We define a collision free plan as a plan for which the shape of the robot at every configuration in the plan does not collide with obstacles in the environment, and an interpolation between adjacent configurations does not collide with obstacles in the environment. We then define a valid plan as a plan that is collision free and achievable given the robot hardware.

When computing a motion plan, our method takes as input a CRISP setup, an initial configuration $\mathbf{q}_0$, a Computed Tomography (CT) scan from which we will define the environment, and a goal point $\mathbf{p}_{\text{goal}}$, the location in the anatomy that the physician is attempting to view with the camera. Our method then produces as output a plan σ, which is a collision-free sequence of configurations that will result in the robot being able to view $\mathbf{p}_{\text{goal}}$.

When evaluating the quality of a candidate setup, we require as input the setup, an initial configuration $\mathbf{q}_0$, and the CT scan. Then, instead of outputting a plan to a specific goal point, we instead output a set of cells on the interior surface of the pleural effusion which can be seen by manipulation of the CRISP robot with that specific setup, which we define as a visibility set.

2.3 Method

2.3.1 Motion Planning

To model the environment for our motion planner, we use an occupancy grid. The grid has free cells, S_{free}, in which the robot is allowed to freely move, and occupied cells, S_{obs}, which the motion planner treats as an obstacle and restricts the robot from moving into. To generate these sets, we segment the pleural effusion from the CT scan using a semi-automatic region-growing method [38].

We set S_{free} to be the cells in the CT scan consistent with the pleural effusion and S_{obs} to be the inverse segmentation. We also define a third set, S_{bound} to be the cells in S_{obs} which are adjacent to S_{free} which will contain the goal point of interest $\mathbf{p}_{\text{goal}}$ and which will be the set we attempt to visualize in the evaluation of a setup.

We solve the motion planning problem formulated in Sec. 2.2.2 using a sampling-based approach. We implement a planner based on the Rapidly-Exploring Random Trees (RRT) method [39]. The RRT method begins at a root node, the initial configuration, and iteratively and randomly constructs a tree structure where each node in the tree is a valid configuration, and an edge linking two nodes is a valid, collision free motion between them as a linear interpolation in configuration space. As the tree grows, it expands and explores the obstacle free configuration space of the robot. Once a node is found which has a camera pose with a clear view of the goal point, $\mathbf{p}_{\text{goal}}$ in the lung, the tree can be traced back to the root node, and a valid plan from the initial configuration to a goal configuration has been found.

Specifically, we begin with our root node. We then sample a randomly generated point in configuration space. We linearly interpolate between the two configurations, using spherical linear interpolation (SLERP) [40] to interpolate between the rotational degrees of freedom. We then propagate along the line segment starting at the root node for a random percentage of the line segment, computing the shape of the robot and checking it against S_{obs} at a fine discretization. If the robot collides with obstacles or reaches a configuration that the forward kinematics solver is unable to solve, we stop at the prior step of the discretization. We then add the last valid configuration and edge to the tree. This process is then repeated, but the node from which to start the propagation is chosen as the nearest neighbor in the tree to the newly sampled point.

This process continues until a time limit has passed or until a configuration has an unobstructed view of $\mathbf{p}_{\text{goal}}$, whichever comes first.

To perform collision detection, we need an accurate estimate of the robot's shape at a given configuration. To calculate the robot's shape at a given configuration, we use a modification of the mechanics-based model developed by Mahoney et al. [1] and described in greater detail in Sec. 2.3.2. Having calculated the backbone shape of each tube, and knowing each tube's radius, we are able to efficiently check for collisions between the robot's geometry and the occupancy grid. This is done by interpolating along the shape of each flexible instrument, identifying which cells the shape will

occupy, and doing an index lookup into the CT-derived occupancy grid for those cells.

To identify whether at a sampled configuration $\mathbf{p}_{\text{goal}}$ is visible from the camera on the tip of ρ_{c} we implement a ray trace. First, the camera position and direction of view, $\mathbf{p}_{\text{camera}}$ and $\mathbf{v}_{\text{camera}}$ are inferred from the tip of the shape computed for ρ_{c}. The vector between $\mathbf{p}_{\text{goal}}$ and $\mathbf{p}_{\text{camera}}$ is computed, and it is compared with $\mathbf{v}_{\text{camera}}$. To identify if $\mathbf{p}_{\text{goal}}$ lies within the field of view of the camera, we examine the planar angle between the two vectors. If the angle is larger than $\theta_v/2$, then $\mathbf{p}_{\text{goal}}$ does not lie within the field of view of the camera. If, however, the angle is less than $\theta_v/2$, then $\mathbf{p}_{\text{goal}}$ does lie in the field of view of the camera. This is not enough, however, because there must exist line of sight between the camera and $\mathbf{p}_{\text{goal}}$—the view of $\mathbf{p}_{\text{goal}}$ may be occluded by another part of the patient anatomy. To identify if there exists clear line of site, a ray is traced from $\mathbf{p}_{\text{camera}}$ to $\mathbf{p}_{\text{goal}}$. If the ray strikes an occupied cell in S_{obs} before it reaches $\mathbf{p}_{\text{goal}}$, there is not clear line of site and the motion planning continues. However, if there exists clear line of site then the plan is traced back to the root initial configuration and is returned.

2.3.2 Mechanics & Solution Seeding

One of the most computationally intensive aspects of the method is computing the forward kinematics of the CRISP robot that determines its shape. This is done for every node in the tree, and at every finely discretized point along each edge in the tree. The forward kinematics is calculated both to ensure the configuration is collision free everywhere on the CRISP robot's body and to identify the camera pose.

The forward kinematics of the CRISP robot results from its mechanics, which were initially presented in [1]. In this chapter, we assume that the flexible instruments of a CRISP robot can be physically held by robot manipulators at the point where the tubes enter the patient's body. This reduces the dimensionality of the CRISP robot's actuation space to only include orientation of each tube at the body entry point and each tubes' insertion length into the body. This assumption also simplifies the system mechanics and can be physically implemented in practice using stiff introducer sheaths through which the flexible instruments can be deployed. What follows is a summary of the simplified CRISP robot's mechanics and forward kinematics.

We model the CRISP robot using the Cosserat rod equations [41, 42], which define a system of ordinary differential equations that govern the Cosserat-rod state of each tube. This state consists of the backbone position $\mathbf{p} \in \mathbb{R}^3$, orientation expressed as a rotation matrix $R \in SO(3)$, internal

moment $\mathbf{m} \in \mathbb{R}^3$, and internal force $\mathbf{n} \in \mathbb{R}^3$ of each tube, each as a function of arc length along the tube's backbone. For instance, the Cosserat-rod state of the camera tube is defined as

$$\mathbf{x}_{\mathrm{c}}(s) = [\mathbf{p}_{\mathrm{c}}(s) \;\; R_{\mathrm{c}}(s) \;\; \mathbf{m}_{\mathrm{c}}(s) \;\; \mathbf{n}_{\mathrm{c}}(s)]\,, \;\; 0 \leq s \leq \ell_{\mathrm{c}}, \tag{2.3}$$

where s is the arc length parameter and ℓ_{c} is the length of the camera tube. We define the Cosserat-rod state of the snare tubes, $\mathbf{x}_k(s)$ for the kth snare tube, similarly. For more details on Cosserat rod theory, we direct the reader to [41] and [42].

The forward kinematics of a multi-tube system are formulated as a *multi-point boundary value differential equation*, where the Cosserat-rod states of the kth snare tube propagate along its backbone in arc length $0 \leq s \leq \ell_{\mathrm{k}}$ as

$$\mathbf{x}'_k(s) = \begin{cases} [\mathbf{p}'_k(s) \;\; R'_k(s) \;\; \mathbf{m}'_k(s) \;\; \mathbf{n}'_k(s)]\,, & s_k - \ell_k \leq s \leq s_k \\ 0, & \text{otherwise} \end{cases}, \tag{2.4}$$

where s_k is the grasp location of the kth snare tube. The states of the camera tube propagate along its backbone in arc length $0 \leq s \leq \ell_{\mathrm{c}}$, as

$$\mathbf{x}'_{\mathrm{c}}(s) = \begin{cases} [\mathbf{p}'_{\mathrm{c}}(s) \;\; R'_{\mathrm{c}}(s) \;\; \mathbf{m}'_{\mathrm{c}}(s) + \alpha \;\; \mathbf{n}'_{\mathrm{c}}(s) + \beta]\,, & 0 \leq s \leq \ell_c \\ 0, & \text{otherwise} \end{cases}, \tag{2.5}$$

where $'$ denotes the derivative with respect to arc length. These derivatives, as well as the terms α and β, can be found in [1]. The tube lengths (ℓ_{c} and ℓ_k) and the initial values of the tube position ($\mathbf{p}_k(0)$ and $\mathbf{p}_{\mathrm{c}}(0)$) and orientation ($R_k(0)$ and $R_{\mathrm{c}}(0)$) at the body entry points are given by the corresponding entry point position and orientation from the starting state, (2.1) and (2.2). The initial values of the Cosserat-rod internal moments ($\mathbf{m}_k(0)$ and $\mathbf{m}_{\mathrm{c}}(0)$) and forces ($\mathbf{n}_k(0)$ and $\mathbf{n}_k(0)$) for both the snare and camera tubes are determined later to satisfy the constraints of the multi-point boundary value problem.

The constraints of the multi-point boundary value problem include a constraint at each of the grasp points s_k on the camera tube's body. The grasp constraints enforce the tip position of the snare tube to be coincident with the camera tube's position at arc length s_k and the tip pose of the

snare tube is constrained so that there is a constant rigid body rotation that maps the snare tip orientation to the backbone pose of the camera tube's orientation at arc length s_k as

$$\mathbf{c}_k = \begin{bmatrix} \mathbf{p}_k(\ell_k) - \mathbf{p}_\mathrm{c}(s_k) \\ R_\mathrm{c}^T(s_k) R_k(\ell_k) R_\mathbf{x} - I \end{bmatrix} = \mathbf{0} \tag{2.6}$$

where $R_\mathbf{x} \in SO(3)$ is the rotation in the $[-1, 0, 0]^T$ direction by $90°$ and $I \in \mathbb{R}^{3\times3}$ is the identity matrix.

Under the assumption that the system is quasistatic and in the absence of applied forces and moments at the camera tube's tip, the force and moment at the camera tube's tip will be zero, leading to the additional constraint of

$$\mathbf{c}_\mathrm{c}(\ell_\mathrm{c}) = \begin{bmatrix} \mathbf{m}_\mathrm{c}(\ell_\mathrm{c}) \\ \mathbf{n}_\mathrm{c}(\ell_\mathrm{c}) \end{bmatrix} = \mathbf{0}. \tag{2.7}$$

The $N-1$ grasp constraints $\mathbf{c}_i$ and the tip constraint $\mathbf{c}_\mathrm{c}$ can be packed into the total constraint vector

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_\mathrm{c} & \mathbf{c}_1 & \dots & \mathbf{c}_{N-1} \end{bmatrix} = \mathbf{0}. \tag{2.8}$$

The multi-point boundary value problem is solved by varying the snare and camera tube's initial conditions of their moments ($\mathbf{m}_k(0)$ and $\mathbf{m}_\mathrm{c}(0)$) and forces ($\mathbf{n}_k(0)$ and $\mathbf{n}_\mathrm{c}(0)$) so that the total constraint equation (2.8) is satisfied. When solved, the multi-point boundary value problem yields the system's forward kinematics. We accomplish this using a numerical optimization routine known as a "shooting" method, where an initial seed of the snare and camera tube's initial internal moment are iteratively perturbed to minimize $\|\mathbf{c}\|$ [1].

The runtime of the forward kinematics computation is heavily dependent on the initial conditions seeded into the shooting method. If the initial conditions lie far from the true solution, not only will the shooting method converge more slowly, but it may not converge to a solution at all. However, if the initial conditions lie close to the true solution, then the shooting method will run much more quickly and the forward kinematics will be solved faster.

This insight is a significant motivation for our choice of an RRT based motion planner. RRT's incremental growth property means that we can always view tree expansion as a sequential small

perturbation on an already solved forward kinematics problem. More specifically, as we expand the tree, we seed the initial conditions of each subsequent shape calculation with the true values found at the state from which it is propagating. Because each step is relatively small, we are always seeding the initial moments and forces with a solution that lies close to the true solution. We note a substantial computational speedup associated with this property compared to seeding the initial moments and forces with a generic set of initial conditions, as discussed in Sec. 2.4.3.

2.3.3 Candidate CRISP Setup Evaluation

An adaptation of our method can be used to evaluate a specific candidate CRISP setup and initial configuration by generating a visibility set. Rather than attempting to find a plan from the initial configuration to a viewpoint for a specific $\mathbf{p}_{\text{goal}}$, one can ask the question "What is the total set of all points that can be viewed in the space, if we start at a specific $\mathbf{q}_0$ using a given candidate setup?" An answer to this question may be useful in evaluating how effective an initial configuration and setup is. To answer this question, we attempt to generate the set of all points in S_{bound} which can be viewed by the robot starting at $\mathbf{q}_0$ with the candidate setup. This can be viewed as the endoscopic equivalent of evaluating the reachable workspace of the robot.

We extend our method to construct a visibility set by allowing the tree to expand for a fixed and relatively long duration of time, while observing which cells in S_{bound} can be seen from each configuration in the tree. Rather than checking whether a specific $\mathbf{p}_{\text{goal}}$ can be seen, we instead ask what the set of cells is which is visible from the configuration. This is done in a similar fashion as above, but all cells in S_{bound} are evaluated and filtered by their relative angle to $\mathbf{v}_{\text{camera}}$. For each cell that lies in the field of view of the camera, a ray is then traced to a point $\mathbf{p}$ in the cell, and the cell in S_{bound} at which the ray terminates is added to the visibility set. The union of the sets for each configuration in the tree then becomes the total visibility set and is returned.

2.4 Results

We demonstrate the speed and effectiveness of our new motion planner for CRISP robots in a simulated scenario based on a pleural effusion segmented from a patient CT scan. We ran the motion planner on an Intel Xeon E5-1680 CPU with 8 cores running at 3.40GHz and 64GB of RAM.

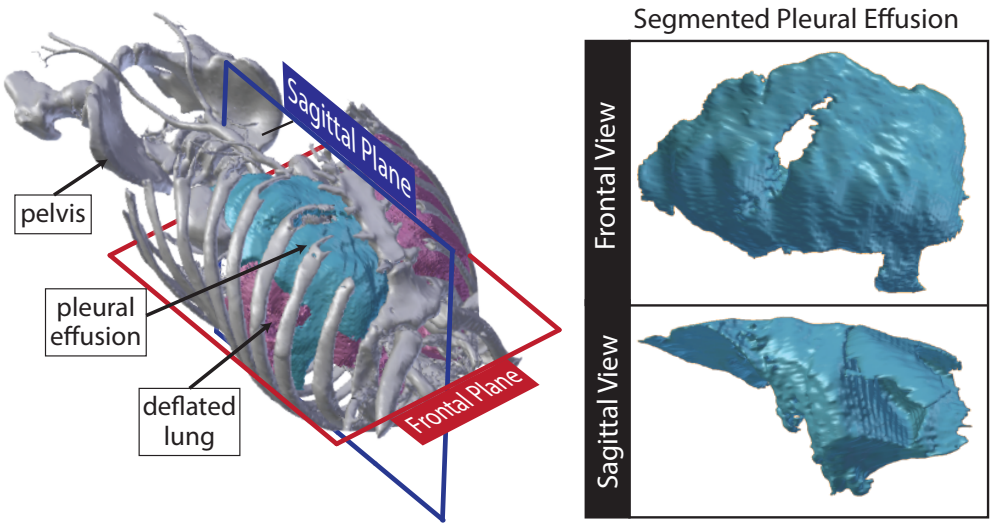


Figure 2.3: An isometric view of the segmented open pleural effusion volume, bone structure, and lung is shown on the left. Frontal and sagittal views of the pleural effusion volume are shown on the right. Note that the collapsed lung lies on the posterior side of the pleural effusion.

2.4.1 Generating Visibility Sets

We first evaluated the ability of the method to generate visibility sets for two CRISP robot setups. Each setup included one snare tube and the camera tube, where the snare tube is affixed to the camera tube 1 cm from its tip. Both tubes have an OD of 1.02 mm and an ID of 0.84 mm. The two starting setups are shown in Fig. 2.4 in the pleural effusion space. The setups initially point the camera in different directions and have differing entry points and orientations into the effusion.

For each setup, we ran the motion planner for 1 hour to explore the space and recorded cells on the pleural effusion surface that could be seen by the camera. As can be seen in Fig. 2.5, from each setup the robot is able to visualize a large portion of the interior surface of the pleural effusion. After 1 hour, Setup 1 has visualized 38% percent of the pleural effusion surface and Setup 2 has visualized 57% percent of the pleural effusion surface. The difference in the visibility sets between the two setups implies that the ability of CRISP to visually inspect a particular set of goal points is dependent on choosing a high quality setup. The computed visibility sets can provide feedback to a physician on the usefulness of each specific setup. The union of the two visibility sets covers 80% of the total effusion surface, illustrating that using multiple setups increases the size of the visibility set and can enable the physician to see most of the pleural effusion.

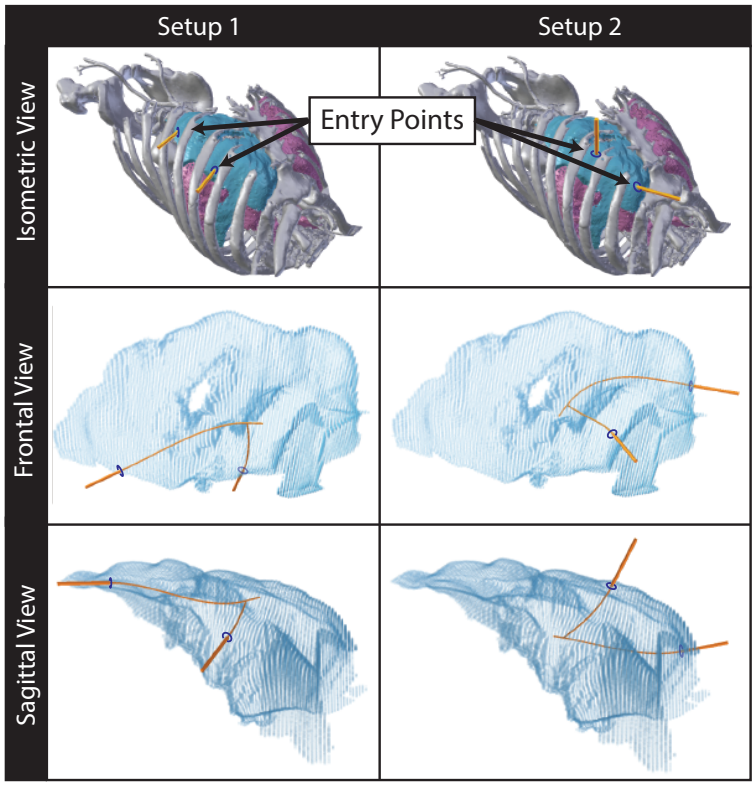


Figure 2.4: The setups and initial configurations for both Setup 1 (left column) and Setup 2 (right column). The pleural effusion is rendered transparent so the initial shape of the robot can be visualized.

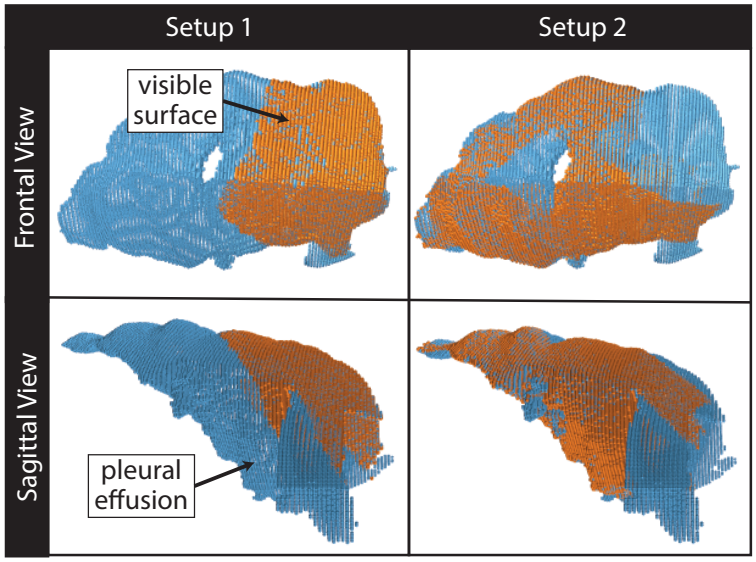


Figure 2.5: The visibility sets found by the motion planner for Setup 1 (left column) and Setup 2 (right column) after 1 hour of computation. The portion of the surface visualized is rendered in orange.

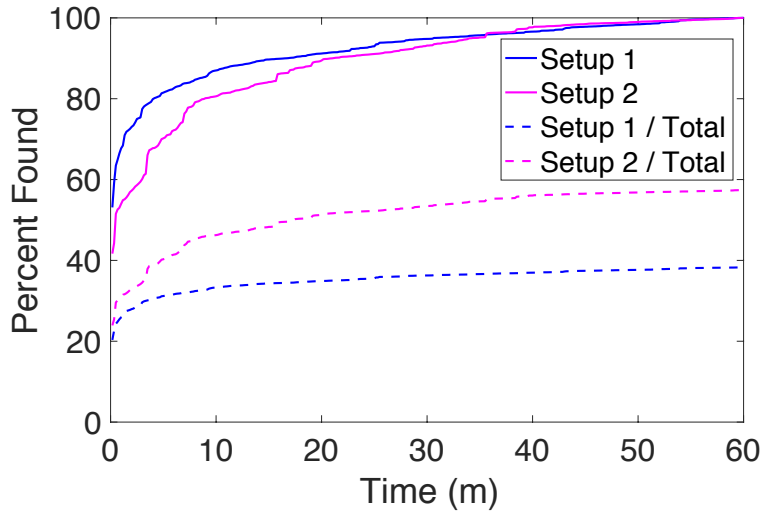


Figure 2.6: The percent of goal points seen by the camera as a function of the motion planner's exploration time for each setup. Solid lines show the percent of goal points seen by the camera with respect to each setup's visibility set. Dashed lines represent the percent of goal points seen with respect to the total number of points on the pleural effusion surface.

2.4.2 Motion Planning to View a Specific Goal Point

We next evaluate the motion planner for computing a motion to view a specific goal point using the chip-tip camera. Using $\approx 49,500$ goal points on the surface of the pleural effusion, we show in Fig. 2.6 the percentage of the goal points which have been visualized as a function of time. The percentage of points found can be viewed as the probability that the motion planner finds a plan to visualize a goal point if it were sampled uniformly from the set of visible points found by that specific setup (solid lines) or if it were sampled from all points on the interior surface of the pleural effusion (dashed lines). An example of a motion being planned to view a specific goal point can be seen in Fig. 2.7.

2.4.3 Mechanics Solution Seeding

To evaluate the efficacy of seeding the forward kinematics computation with the solution of an already known near-by shape, we repeated the experiments but without the the seeding. Instead, we initialize each shape calculation with the solved solution of only the initial configuration, corresponding to the root node in the tree.

At the end of the hour-long planning time, the motion planner for Setup 1 without the mechanics solution seeding had only planned visualizations for 30% of the total cells in S_{bounds} compared to the 38% found by the planner with the mechanics solution seeding. For Setup 2, the planner without

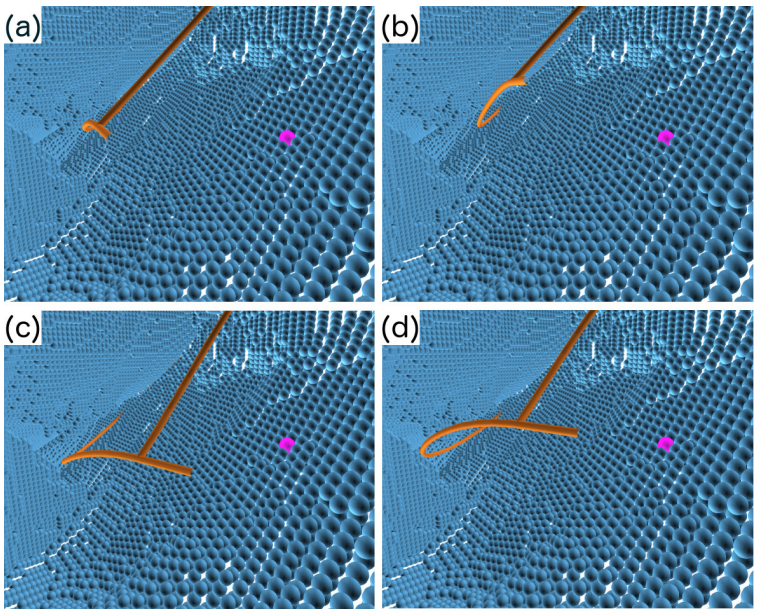


Figure 2.7: A motion plan viewed from inside the pleural effusion. Potential points of interest on the interior surface of the pleural effusion are rendered as blue spheres, with the specific goal point rendered in pink. The plan goes from the initial configuration (a), through collision free intermediate configurations (b) and (c), to a configuration in which the tip of the camera tube can view the goal point in (d).

the seeding had only computed motion plans to visualize 38% of the cells compared to the 57% found by the motion planner with the mechanics solution seeding. A stark difference was also found in the number of configurations being added to the tree. For example, from Setup 1, after 1 hour the motion planner without the seeding had only added 2,177 configurations to the tree, while the motion planner with seeding had added 37,855 configurations. For Setup 2, the motion planner without the seeding had added 2,627 configurations, while the motion planner with the seeding had added 36,610 configurations. The ≈ 14 times increase in the number of configurations when using the seeding suggests a significantly more expansive motion planning tree capable of viewing more user-specified points of interest.

2.5 Conclusion

CRISP robots provide new avenues for performing minimally invasive, incisionless medical procedures. Our motion planner for CRISP robots computes manipulations of the needle-diameter flexible instruments outside the body such that the camera can visually inspect a user-specified site of clinical interest inside the body. Our sampling-based motion planner ensures avoidance of collisions with anatomical obstacles inside the body, enforces remote-center-of-motion constraints on

the flexible instrument's entry points into the body, and efficiently handles the expensive computation of CRISP robot kinematics. We also extended the motion planner to estimate the set of points inside a body cavity that can be visually inspected by the camera of a CRISP robot for a given setup.

In future work, we plan to build upon this new motion planner to bring CRISP robots closer to clinical use. We plan to further accelerate the motion planner using sampling heuristics, precomputation, and parallelization. Additionally, we plan to extend the motion planner to account for uncertainty such as patient motion during the procedure. We also plan to integrate our motion planner with a physical robot prototype and evaluate the performance of the integrated physical system.

In Chapter 3 we utilize the motion planner presented in this chapter to automatically optimize the setup of the CRISP robot on a patient-specific basis to maximize the ability of the camera to see sites of clinical interest.

CHAPTER 3

Kinematic Design Optimization of a Parallel Surgical Robot to Maximize Anatomical Visibility via Motion Planning

A robot's kinematic design, i.e., physical parameters fixed prior to the robot's use that affect a robot's kinematics, can greatly impact the robot's ability to perform a task. The quality of a specific kinematic design can vary as the robot's task and environment changes. In medical robotics, the quality of a robot's kinematic design is influenced by the specific surgical or interventional procedure it will perform as well as the specific patient anatomy in which it will operate. A suboptimal kinematic design may negatively impact patient outcomes. In this work, we introduce a method to optimize on a patient-specific basis the kinematic design of the CRISP robot [1, 2] to maximize the ability of the robot's tip camera to view tissues in constrained spaces.

We remind the reader that the CRISP robot is a minimally invasive surgical robot composed of needle-diameter tubes which are inserted into the patient and assembled into a parallel structure (see Fig. 3.1). This assembly is performed using snares, which the tubes use to grip one another. In addition to snares, other tools can be passed through or placed at the end of the tubes, such as chip-tip cameras, biopsy needles, ablation probes, etc. As the tubes are then manipulated outside the body by robotic manipulators, the shape is changed inside the body to perform the surgical task. Due to the needle-like diameter of the tubes, the CRISP robot has the potential to further reduce invasiveness of minimally invasive surgical procedures such as abdominal, fetal, and neonatal surgery [1, 8]. When a chip-tip camera is mounted at the tip of one of the tubes, the robot can operate as an endoscope, enabling the physician to view the interior surface of an anatomical cavity, such as the abdomen or pleural space (between a lung and the chest wall) [21]. Whereas typical pleural endoscopes have diameters of 7mm and may require an incision as large as 10mm, the CRISP robot requires only needle-size (1-3mm diameter) entry points in the skin.

In this chapter we investigate kinematic design optimization for the CRISP robot with a chip-tip camera to maximize the surface area of tissues visible to the physician in a constrained space. We

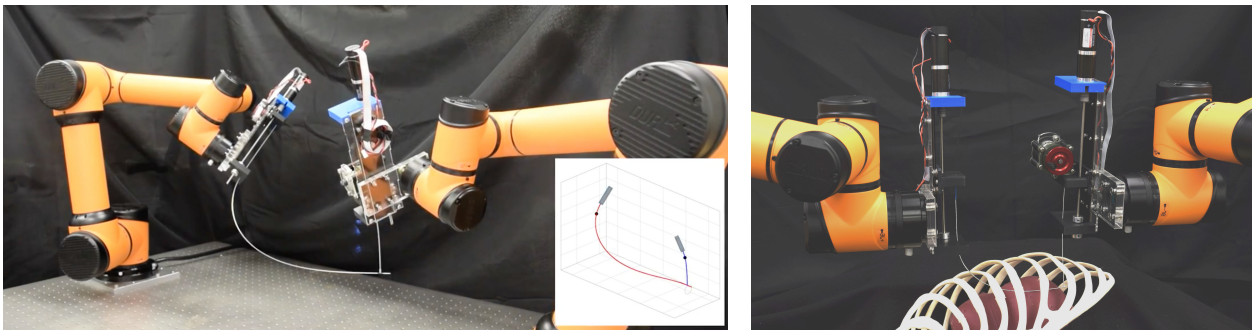


Figure 3.1: Two images of the CRISP robot, where 2 manipulators adjust the base poses of needle-diameter tubes that are inserted into the body and assembled into a parallel structure.

consider as part of the optimization the anatomical entry points of each tube into the patient's body, which may be constrained by anatomical factors, as well as the snare grasping locations which define the parallel structure of the robot. Appropriately choosing these kinematic design parameters significantly impacts the robot's ability to view the anatomical sites of interest to the physician (see Fig. 3.2).

Our design optimization approach combines a global numerical optimization method with a motion planner to evaluate the ability of candidate kinematic designs to view the anatomical sites of interest. Specifically, we use the global optimization algorithm Adaptive Simulated Annealing (ASA) [43, 44] to generate candidate designs, which we evaluate using a sampling-based motion planner specifically designed for viewing anatomical sites using the CRISP robot [8]. By evaluating candidate designs with a motion planner, we ensure that a target is considered viewable by the robot only if viewing the target can be achieved by a robot motion that avoids collision with obstacles in the patient anatomy. This guarantees that the evaluation of the design takes into account the *motions* required by the robot to view the target regions, not just the robot's theoretical ability to do so in the absence of constrained anatomy.

Whereas other previous ASA based approaches [44] optimize a design's ability to reach goal regions, we focus on optimizing a design's ability to *view* target regions with its camera, an application which introduces unique challenges in providing theoretical guarantees. We prove that for this application the optimization is asymptotically optimal, i.e., almost surely converges to a globally optimal kinematic design. We then demonstrate the ability of the design optimization algorithm to produce high quality designs in two simulated scenarios, including a pleuroscopic scenario based on patient anatomy. We show that the algorithm generates designs which over time significantly

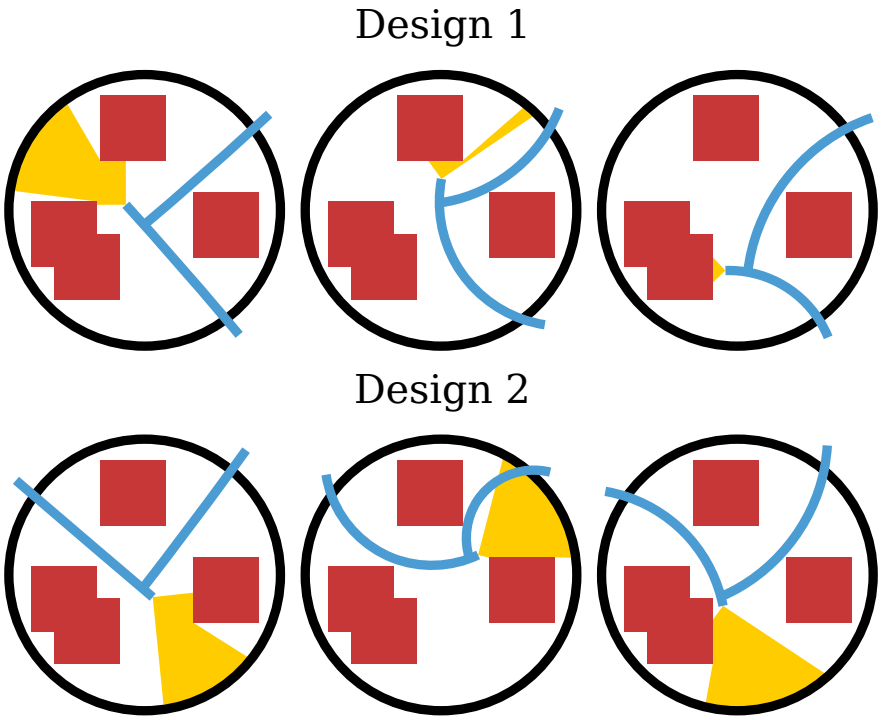


Figure 3.2: Different designs can greatly impact the robot's ability to view the interior surface of a volume via its tip camera. Here we show 2D examples of a robot (blue) and its field of view (yellow) after being occluded by obstacles (red). In Design 1 (top), the robot cannot view much of the interior surface of the volume (the black circle) due to obstacles as it is manipulated through 3 collision-free configurations. However a different design (bottom), which has different entry points into the volume and a different snare grasping location, can view a much larger percentage of the interior surface.

improve the robot's ability to view target regions.

This chapter is based on work originally presented in [11].

3.1 Related Work

Numerical optimization methods have been developed to optimize the design of various medical robots. Another continuum surgical robotic system for which design optimization methods have been developed is the concentric tube robot. Concentric tube robots are similar to the CRISP robot in that they are continuum systems with complex and computationally costly kinematics due to the elastic and torsional interactions between their tubes. The complex kinematics of these devices have motivated unique approaches to their design optimization.

Bergeles et al. provide a method that computationally optimizes concentric tube robot designs to reach a set of goal points while avoiding collisions with anatomy [45]. This method, however, does not provide a guarantee of global optimality. Burgner et al. leverage a grid-based search of the robot's configuration space with a nonlinear optimization method to optimize designs, maximizing

reachable workspace while satisfying anatomical constraints [46]. Ha et al. optimize designs to maximize the concentric tube robot's elastic stability, a problem which is of particular concern for concentric tube robots [47]. Morimoto et al. take a unique approach to concentric tube robot design by providing a human with an interface to interactively design the tubes of the robot [48]. These works focus on computing goal configurations of the robot for a given design, and as such do not provide a guarantee that there exists a collision free *path*, or sequence of configurations, which allows the computed design to achieve a goal configuration.

To accomplish optimization with start to goal path guarantees, Torres et al. integrate a motion planner into the design optimization process, ensuring that valid paths exist to each reachable target for a given design [49], but this method suffers from slow performance. Baykal et al. build upon this method to compute a minimal set of designs that reach multiple targets [50]. Niyaz et al. utilize a motion planner designed to follow surgeon specified tip trajectories [51] to optimize a concentric tube robot's starting pose [52]. Recently, Baykal et al. provide asymptotic optimality in the design of piecewise cylindrical robots, which includes concentric tube robots, for reaching workspace targets [44]. In this chapter, we build upon this prior work to optimize the kinematic design of the CRISP robot for anatomical visibility, rather than reachability, and show asymptotic optimality under this new optimization metric.

The optimization of surgical port placement, which has parallels to our optimization of needle entry points, has received previous study as well. Liu et al. optimize port placement and needle grasping locations for autonomous suturing [53], but they require as input a discrete set of candidate grasping and port placement locations. Hayashi et al. investigate abdominal port placement by examining the angular relationship between the port location and anatomical sites in the abdomen [54]. They do not, however consider the requirement of more complex motions during the procedure. Feng et al. optimize laparoscopic port placement for robotic assisted surgery by evaluating the robot's reachable workspace in the patient [55], but do not consider complex motions or obstacle avoidance during the procedure.

Design optimization has been studied outside of the surgical robotics domain as well. For instance, discrete parameter design optimization has been studied for a variety of applications, including multi-modal robots [56, 57], jumping robots [58], modular robots [59, 60], and protein chains [61]. These methods focus on optimizing over a finite, discrete set of features, whereas in this work we

optimize over continuous design parameters.

Methods for optimizing the design of serial manipulators has received a large amount of study and include grid-based approaches [62], geometric approaches [63], interval analysis [64], and genetic algorithms [65, 66, 67, 68]. Frequently these methods require simplified kinematic models or restrictive assumptions to achieve computational feasibility, and typically provide no theoretical performance guarantees.

Taylor et al. present a non-linear constrained optimization approach to the simultaneous design of shape and motion for dynamic planar manipulation tasks [69]. Ha et al. explore the relationship between design and motion by leveraging the implicit function theorem to jointly optimize robot joint and motion parameters for manipulators and quadruped robots [70]. These methods, however, do not provide global guarantees and may be subject to local minima.

3.2 Problem Definition

Similar to as in Sec. 2.2, we consider a CRISP robot composed of n needle-diameter tubes. One of the tubes, the camera tube ρ_{cam}, has a chip-tip camera attached to its tip. The rest of the $n-1$ tubes have snares with which they grip ρ_{cam} and are denoted ρ_k where k is a unique integer label.

3.2.1 Design Space

Let $\mathbf{r}_{\text{cam}}, \mathbf{r}_1, \ldots, \mathbf{r}_{n-1}, \in \mathbb{R}^3$ represent the points at which the tubes of the CRISP robot enter the patient's body expressed in a global coordinate system. Let the set of all valid such entry points be $\mathcal{R} \subseteq \mathbb{R}^3$. The entry points into the patient's body act as remote center of motion (RCM) points, preventing the tubes from applying lateral load to the patient's skin during the procedure, and are fixed during the operation of the robot.

The CRISP robot is assembled into a parallel structure where each snare tube ρ_k grips the camera tube at $s_k \in \mathbb{R}$, the scalar valued arc length along ρ_{cam}. The snare grasping locations are fixed during operation.

We define a kinematic design $\mathbf{d}$ of the CRISP robot as a vector describing each tube's entry point into the body and each snare tube's grasping location. The open set of all kinematic designs is $\mathcal{D} \subseteq \mathbb{R}^{n-1} \times \mathbb{R}^{3n}$ (see Fig. 3.3).

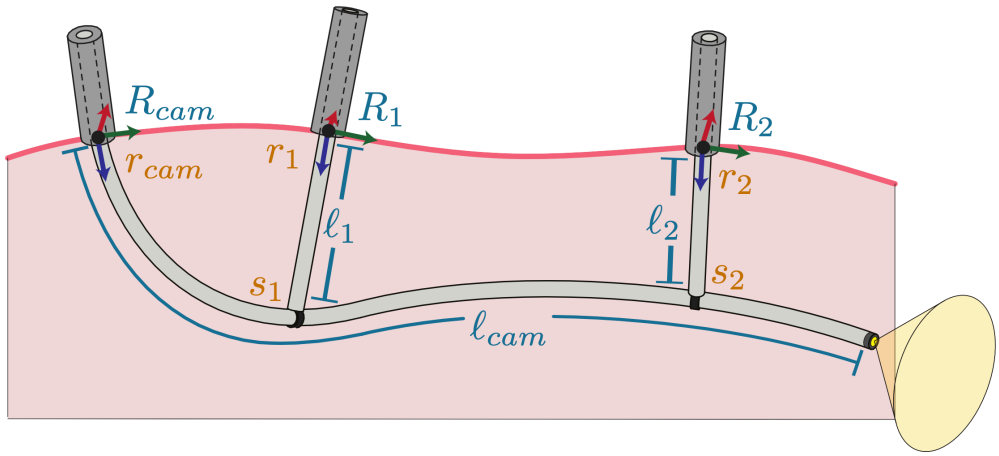


Figure 3.3: Highlighting the differences between the CRISP robot's design parameters and configuration variables: The CRISP robot's kinematic design parameters (orange), which must be set before a procedure, include the entry points into the body (r_{cam} or r_k) and the snare grasping locations (s_k). The robot's configuration variables (blue), which can be continuously modified during a procedure, include the tubes' insertion lengths into the body (ℓ_{cam} or ℓ_k) and the tubes' orientations at the entry point (R_{cam} or R_k).

3.2.2 Configuration Space

Once the kinematic design $\mathbf{d}$ is fixed and the CRISP robot is assembled inside the patient, the operation of the robot may then be performed by rotating the tubes about their RCM entry points and inserting and withdrawing the tubes from the patient. We define a configuration of the robot as

$$\mathbf{q} = (R_{\text{cam}}, R_1, \ldots R_{n-1}, \ell_{\text{cam}}, \ell_1, \ldots, \ell_{n-1}),$$

where $R \in \mathcal{SO}(3)$ represents a tube's rotation at its entry point, and $\ell \in \mathbb{R}$ represents a tube's insertion length into the patient's body. The open set of all configurations then becomes $\mathcal{Q} \subseteq \mathcal{SO}(3)^n \times \mathbb{R}^n$. We define a path as a continuous function $\sigma : [0, 1] \to \mathcal{Q}$, where $\sigma(0) = \mathbf{q}_0$, with $\mathbf{q}_0$ defined as the starting configuration of the robot, which may vary for different designs (see Fig. 3.3).

3.2.3 Workspace

We define the robot's workspace as $\mathcal{W} \subseteq \mathbb{R}^3$. We define Shape : $\mathcal{D} \times \mathcal{Q} \times \mathcal{K} \to \mathcal{W}$ as the continuous shape function of the robot, where $\mathcal{K}$ is the compact set of points representing the robot's geometry.

Given a compact obstacle set $\mathcal{O} \subseteq \mathcal{W}$, the set of points which must be avoided by the robot's geometry, we say a design-configuration pair $(\mathbf{d}, \mathbf{q}) \in \mathcal{D} \times \mathcal{Q}$ is in collision if there exists $\mathbf{k} \in \mathcal{K}$ such

that $\text{Shape}(\mathbf{d}, \mathbf{q}, \mathbf{k}) \in \mathcal{O}$.

A design-configuration pair $(\mathbf{d}, \mathbf{q})$ is said to be *reachable* if there exists a path $\sigma : [0, 1] \to \mathcal{Q}$ with $\sigma(0) = \mathbf{q}_0$ and $\sigma(1) = \mathbf{q}$ such that for all $0 \leq s \leq 1$, $(\mathbf{d}, \sigma(s))$ is not in collision.

Let $\text{View} : \mathcal{D} \times \mathcal{Q} \times \mathcal{V} \times [0, 1] \to \mathcal{W}$ be a continuous function defining the view of the robot, where $\mathcal{V}$ represents the set of rays along which the robot can see as determined by the camera's field of view, and $[0, 1]$ is the domain of a distance along the ray. A target region $T \subseteq \mathcal{W}$ is said to be *visible* from a design-configuration pair if there exists a ray in $\mathcal{V}$ that reaches any point in the target region, unoccluded by any obstacle. More formally, a target region $T \subseteq \mathcal{W}$ is visible from a design-configuration pair $(\mathbf{d}, \mathbf{q})$ if $\exists \mathbf{v} \in \mathcal{V}, r \in [0, 1] : \text{View}(\mathbf{d}, \mathbf{q}, \mathbf{v}, r) \in T \wedge \forall s \in [0, r] : \text{View}(\mathbf{d}, \mathbf{q}, \mathbf{v}, s) \notin \mathcal{O}$.

3.2.4 Maximizing Viewable Anatomy

Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ denote the set of m physician-specified target regions that we seek to view, where each target region denotes an open set of points in the workspace, $T_i \subseteq \mathcal{W}$, $\forall i \in [m]$. We define a target region as *viewable* under design $\mathbf{d}$ if the target region can be viewed from any configuration that can be reached by following a collision-free path. We then define the number of viewable target regions from a given $\mathbf{d}$ as $\Pi(\mathbf{d}) : \mathcal{D} \to [0, m]$, where

$$\Pi(\mathbf{d}) := |\text{TargetRegionsViewable}(\mathbf{d})|. \tag{3.1}$$

$\text{TargetRegionsViewable}(\mathbf{d})$ denotes the set of viewable target regions when using kinematic design $\mathbf{d}$. Our goal then becomes finding an optimal design $\mathbf{d}^*$ such that $\Pi(\mathbf{d})$ is maximized.

3.3 Method

We present a method which optimizes the kinematic design of a CRISP robot to maximize the robot's viewable target regions in a specific anatomy while avoiding collision with obstacles, and which does so in an asymptotically optimal way (see Sec. 3.4). The method, detailed in Alg. 1, is based on the global stochastic optimization algorithm Adaptive Simulated Annealing (ASA) [43, 71], which iteratively samples and evaluates candidate designs. We evaluate candidate kinematic designs with a motion planner designed specifically for the CRISP robot, and which can determine the viewable target regions for a given design [8].

Algorithm 1: CRISP Robot Design Optimization

Input:
T: target regions
$\mathcal{O}$: obstacles
i_{init}: initial number of RRT iterations
i_{Δ}: RRT iteration increment
Output: $\mathbf{d}^*$: optimal CRISP design maximizing (3.1)

$i \leftarrow i_{\text{init}}$; $\text{Temp} \leftarrow \text{Temp}_{\text{initial}}$; $\hat{\Pi}_{\text{current}} \leftarrow 0$; $\hat{\Pi}^* \leftarrow 0$
$\mathbf{d}_{\text{current}}, \mathbf{q}_0 \leftarrow$ random initial design
$\mathbf{d}^* \leftarrow \mathbf{d}_{\text{current}}$
while time allows **do**
  $\mathbf{d}', \mathbf{q}_0' \leftarrow \textsc{SampleDesign}(\mathbf{d}_{\text{current}}, \text{Temp}, i)$;
  $\text{targetsViewed} \leftarrow \textsc{CrispRRT}(\mathbf{d}', i, \mathcal{O}, \mathbf{q}_0')$;
  $\hat{\Pi}' \leftarrow |\text{targetsViewed}|$;
  if $\textsc{Accept}(\hat{\Pi}', \hat{\Pi}_{\text{current}}, \text{Temp})$ **then**
    $\mathbf{d}_{\text{current}} \leftarrow \mathbf{d}'$;
    $\hat{\Pi}_{\text{current}} \leftarrow \hat{\Pi}'$;
  end
  if $\hat{\Pi}' > \hat{\Pi}^*$ **then**
    $\mathbf{d}^* \leftarrow \mathbf{d}'$;
    $\hat{\Pi}^* \leftarrow \hat{\Pi}'$;
  end
  $i \leftarrow i + i_{\Delta}$;
  $\text{Temp} \leftarrow \textsc{UpdateTemperature}(\text{Temp})$;
end

3.3.1 Exploring Design Space

To explore the CRISP robot's design space, we leverage the ASA method. ASA is a global optimization algorithm, ensuring it does not become trapped in local optima. ASA works by iteratively improving upon the current best design. It first samples a candidate design some distance from the current design in design space (SampleDesign at line 5 of Alg. 1). It then evaluates the quality of the candidate design (Accept at line 8 of Alg. 1). If the candidate design is of higher quality than the current design, i.e., $\hat{\Pi}' > \hat{\Pi}_{\text{current}}$, ASA will accept the candidate design. However, with some probability Accept will accept an inferior design. It is this mechanism that allows the algorithm to avoid local maxima. Both the distance at which it samples and the probability of allowing an inferior design to be accepted depend upon a "temperature" value (Temp in Alg. 1). The temperature is initially set to a large value, but is decreased over time according to a cooling schedule. In this way, early in the optimization process the algorithm is much more likely to explore more distant designs or accept inferior designs, but later in the process converges to a high quality design. Our algorithm keeps track of the best design found through the course of its execution and returns the best design at its conclusion.

3.3.2 Evaluating Candidate Designs

We evaluate the number of target regions viewable by a candidate design $\mathbf{d}$, denoted $\Pi(\mathbf{d})$, using a sampling-based motion planner for the CRISP robot [8]. This approach enables our method to estimate the total number of targets which are viewable by a given design, while only considering those configurations reachable by following collision-free paths. The motion planner, CrispRRT, is based on the Rapidly-exploring Random Tree (RRT) algorithm [72] and has the property that the estimate of viewable target regions approaches the correct value as the number of motion planning iterations rises.

Sampling an Initial Configuration The motion planner requires as input a collision-free starting configuration $\mathbf{q}_0$ from which to plan motions. In the case of the CRISP robot, $\mathbf{q}_0$ depends on $\mathbf{d}$ and is non-trivial to generate.

A key challenge with planning the motions for the CRISP robot is that evaluating the forward kinematics to solve for the shape of the robot is costly and requires an initial guess as to the forces and moments being applied to the tubes of the robot. Generally, if the state of the first configuration

is known then subsequent configurations can be propagated out using the forces and moments of the previous configuration to seed that guess. This is a key insight behind the CRISP motion planner [8]. There is, however, no previous state with which to seed the initial configuration $\mathbf{q}_0$, as all subsequent configurations are grown from it, and it is unique to a given design. To solve this problem, we require $\mathbf{q}_0$ to be a load free configuration, i.e., one in which the tubes are applying no forces or moments to each other. This geometrically restricts the shape of the robot to a right triangle without any axial twist applied to the tubes. There are, however, an infinite number of right triangles that pass through the entry points defined by $\mathbf{d}$.

We address this challenge by selecting initial configurations uniformly at random from the set of possible initial configurations which satisfy the design constraints (note that the set of valid initial configurations from which to sample can be defined relatively simply geometrically). The sampled initial configuration is then collision checked against the environment. If it is found to be in collision, it is discarded and another starting configuration is sampled. This process repeats at most i times (the iteration parameter), or until a collision-free initial configuration is found, whichever comes first. If a collision-free initial configuration is not found, the design is assigned a value of 0, and the motion planner immediately returns.

Running the Motion Planner If a valid initial configuration is found, the motion planner is run to determine the design's set of viewable targets. The motion planner builds a tree in configuration space with collision-free configurations as nodes and collision-free transitions between configurations as edges. During CrispRRT (line 6 of Alg. 1), the motion planner attempts to expand its tree i times, keeping track of the set of all target regions viewable by the robot configurations represented by nodes of the tree.

A sampling-based motion planner which runs for a *finite* duration may only return an approximation of the viewable set of targets for a given design (as noted in [44] for the case of reachability). To guarantee that the approximations increasingly approach the true value, the number of iterations for which the planner is run is increased by i_Δ at each iteration of Alg. 1 (line 16). This allows the algorithm to evaluate the quality of candidate designs with increasing accuracy as the algorithm executes.

3.4 Analysis

In this section, we prove under mild assumptions the asymptotic optimality of the proposed method, which provides the user a guarantee that the method's solution will approach a globally optimal kinematic design as more computation time is allowed.

Specifically, we show that the method almost surely converges to a design under which the maximum number of target regions are viewable. This proof builds on ideas from [44] and adapts them to the case of maximizing viewable goal regions. Our approach reduces to showing that the set of optimal designs has non-zero measure and applying results from prior work on design optimization based on the motivating property of ASA. We prove this via open sets, which are either empty or have non-zero measure. In Lemma 1, we show openness of the set of design-configuration pairs for which the configuration is reachable under the design, a useful lemma in its own right. In Lemma 2, we show that the set of design-configuration pairs from which a given target region is visible is also open. We combine these results in Lemma 3 to show that the set of optimal designs is open. Straightforward application of prior work is then sufficient to prove asymptotic optimality in Theorem 1.

3.4.1 Preliminaries

Our proofs use the fact that the inverse images of open and closed sets under continuous functions are themselves open and closed respectively where the inverse image of $A \subseteq B$ under $f : C \to B$ (denoted $f^{-1}[A]$) is the set $\{c \in C \mid f(c) \in A\}$. In the following proofs, we will also frequently refer to the topological projection (or simply projection) from a Cartesian product of topological spaces $X \times Y$ to X. The projection of a set $Z \subseteq X \times Y$ to X is the set $\{x \in X \mid \exists y \in Y : (x, y) \in Z\}$. Two useful properties of projections enable the proofs below. First, the projection of an open set is itself open. Second, if Y is compact, then the projection $X \times Y \to X$ of a closed set is itself closed. This latter property is referred to as the Tube Lemma below.

Assumption 1 (Target Regions as Open Sets)**.** *Each target region $T_i \in \mathcal{T}$, is defined as an open set.*

Assumption 2 (Continuity of Shape)**.** *Shape* $: \mathcal{D} \times \mathcal{Q} \times \mathcal{K} \to \mathcal{W}$ *is continuous.*

Assumption 3 (Continuity of View)**.** *View* $: \mathcal{D} \times \mathcal{Q} \times \mathcal{V} \times [0, 1] \to \mathcal{W}$ *is continuous.*

3.4.2 Sampling Optimal Designs Infinitely Often

Lemma 1. *The set of reachable design-configuration pairs, denoted R, is open.*

Proof. Consider a reachable design-configuration pair $(\mathbf{d}, \mathbf{q})$ for which we wish to construct a reachable neighborhood. By definition of reachable, there must exist some path $\sigma \in [0,1] \to \mathcal{Q}$ with $\sigma(0) = \mathbf{q}_0$ and $\sigma(1) = \mathbf{q}$ such that $\forall s \in [0,1] : \forall \mathbf{k} \in \mathcal{K} : \text{Shape}(\mathbf{d}, \sigma(s), \mathbf{k}) \in (\mathcal{W} \setminus \mathcal{O})$. Let $\sigma_{\mathbf{q}'}(s) = \sigma(s) + s \cdot (\mathbf{q}' - \mathbf{q})$. $\sigma_{q'}$ is continuous by continuity of σ and $\sigma_{\mathbf{q}'}(1) = \mathbf{q}'$ by construction, so $\sigma_{\mathbf{q}'}$ is a path to $\mathbf{q}'$. We thus have only to show that $\sigma_{\mathbf{q}'}$ is collision-free under each design $\mathbf{d}'$ for all $(\mathbf{d}', \mathbf{q}')$ in a neighborhood of $(\mathbf{d}, \mathbf{q})$.

Observe that the mapping $L : \mathcal{D} \times \mathcal{Q} \times \mathcal{K} \times [0,1] \to \mathcal{W}$ given by

$$\mathbf{d}', \mathbf{q}', \mathbf{k}, s \mapsto \text{Shape}(\mathbf{d}', \sigma_{\mathbf{q}'}(s), \mathbf{k}) \tag{3.2}$$

is continuous by continuity of $\sigma_{q'}$ and Shape. We then have that $B = L^{-1}[\mathcal{O}] \subseteq \mathcal{D} \times \mathcal{Q} \times \mathcal{K} \times [0,1]$ is closed by closedness of $\mathcal{O}$. Let C be the projection of B to $\mathcal{D} \times \mathcal{Q}$. C is thus the set of all $(\mathbf{d}', \mathbf{q}')$ for which $\sigma_{\mathbf{q}'}(s)$ is in collision under design $\mathbf{d}'$ for some $s \in [0,1]$, and is closed by compactness of $\mathcal{K} \times [0,1]$ and the Tube Lemma.

Let $\overline{C} = (\mathcal{D} \times \mathcal{Q}) \setminus C$. Now observe that $(\mathbf{d}, \mathbf{q}) \in \overline{C}$ because $\sigma_q = \sigma$, and σ is collision-free for design $\mathbf{d}$ by definition. But $\overline{C}$ is open, so it covers some neighborhood N of $(\mathbf{d}, \mathbf{q})$. N is thus a neighborhood of $(\mathbf{d}, \mathbf{q})$ in which $\sigma_{\mathbf{q}'}$ is collision-free under design $\mathbf{d}'$ for all $(\mathbf{d}', \mathbf{q}') \in N$. □

Lemma 2. *The set of design-configuration pairs from which target region T is visible, denoted $G(T)$, is open.*

Proof. Consider $A = \text{View}^{-1}[\mathcal{O}]$, the set of all design-configuration-ray-distance tuples which yield points inside obstacles, which is closed because $\mathcal{O}$ is closed and View is continuous. We next construct $\hat{A}$, the set of all design-configuration pairs from which $\mathcal{O}$ is visible. $\hat{A}$ is the projection of A to $\mathcal{D} \times \mathcal{Q}$, and because $\mathcal{V} \times [0,1]$ is compact and A is closed, $\hat{A}$ is also closed by the Tube Lemma.

Let $L = \{(s,r) \mid s \leq r \wedge (s,r) \in [0,1]^2\}$ denote the set of all (s,r) such that if an obstacle were at distance s, it would occlude a target region at distance r and observe that this set is closed by

construction. From A, $\hat{A}$, and L, we construct

$$B = (A \times [0,1]) \cap (\hat{A} \times \mathcal{V} \times L),$$

which as the finite intersection of finite products of closed sets is, itself, closed. This is the set of all tuples $(\mathbf{d}, \mathbf{q}, \mathbf{v}, s, r)$ such that under a design $\mathbf{d}$ in configuration $\mathbf{q}$ an obstacle at distance s would occlude a target region at distance r along ray $\mathbf{v}$.

Let C be the image of B under the projection

$$\mathbf{d}, \mathbf{q}, \mathbf{v}, s, r \mapsto \mathbf{d}, \mathbf{q}, \mathbf{v}, r.$$

Again by the Tube Lemma, C is closed because B is closed and the interval $[0,1]$ is compact. C is the set of all $(\mathbf{d}, \mathbf{q}, \mathbf{v}, r)$ such that under a design $\mathbf{d}$ in configuration $\mathbf{q}$ ray $\mathbf{v}$ is occluded at or before a distance r.

Consider $U = \text{View}^{-1}[T]$, the set of all design-configuration-ray-distance tuples via which T is visible, which is open by openness of T and continuity of View. $V = U \cap \overline{C}$ is then the set of *unoccluded* design-configuration-ray-distance tuples via which target region T is visible. Furthermore, as the finite intersection of open sets, V is itself open. But $G(T)$ is simply the projection of V to $\mathcal{D} \times \mathcal{Q}$, and projections preserve openness. □

Lemma 3. *Given a set of target regions $T_1, \ldots, T_m$, the set of designs under which a maximum number of these target regions are viewable, denoted $\mathcal{D}^*$, is open.*

Proof. Let $R_i = G(T_i) \cap R$, the set of reachable design-configuration pairs from which T_i is visible. By Lemma 1, R is open, and by Lemma 2, $G(T_i)$ is open, so their intersection R_i is also open. Let $\mathcal{D}_i$ denote the projection of R_i to its designs. Projection is an open mapping, so each $\mathcal{D}_i$ is open. Let m^* be the number of target regions viewable by an optimal design. Observe that the union of all m^*-wise intersections of $\{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$ is the set of optimal designs. This is a finite union of finite intersections of open sets, and is thus open itself. □

Corollary 1. *$\mathcal{D}^*$ has non-zero measure.*

Proof. By Lemma 3, $\mathcal{D}^*$ is open, and by definition, it contains an optimal design. Every non-empty

open set has non-zero measure. □

Corollary 2. *Designs from $\mathcal{D}^*$ will be sampled and evaluated infinitely often by Alg. 1.*

Proof. This result follows readily from the fact that ASA samples from non-zero measure sets infinitely often [43, 71]. □

3.4.3 Asymptotic Optimality

We conclude the asymptotic optimality of our algorithm by invoking Corollary 2, established above, and Theorem 5 of [44] which extends directly to the objective of visibility.

Let $(\mathcal{Y})_{k\in\mathbb{N}}$ denote the sequence of random variables such that for each $k \in \mathbb{N}$, $\mathcal{Y}_k$ denotes the maximum number of viewable target regions attained over all the designs sampled in optimization iterations $1, \ldots, k$. Let m^* be the number of target regions viewable by an optimal design, as in the proof of Lemma 3.

Theorem 1 (Asymptotic Optimality)**.** *The solution generated by Alg. 1 almost surely converges to a globally optimal design $\mathbf{d}^* \in \mathcal{D}^*$, i.e.,*

$$\mathbb{P}\left(\lim_{k\to\infty} \mathcal{Y}_k = m^*\right) = 1.$$

For completeness, we note that as an implementation detail of our method, the robot's initial configuration depends upon the design. The theorem above continues to hold in this context because the initial configuration may conceptually be incorporated into the robot's design space by introducing $\mathcal{D}' = \mathcal{D} \times \mathcal{Q}$ and letting $\text{Shape}'(\mathbf{d}', \mathbf{q}, \mathbf{k}) = \text{Shape}(\mathbf{d}, \mathbf{q} + \mathbf{q}_0, \mathbf{k})$ and $\text{View}'(\mathbf{d}', \mathbf{q}, \mathbf{v}, r) = \text{View}(\mathbf{d}, \mathbf{q} + \mathbf{q}_0, \mathbf{v}, r)$ where $\mathbf{d}' = (\mathbf{d}, \mathbf{q}_0)$.

3.5 Results

We evaluate the performance of our algorithm, denoted ASA+MP, in two scenarios. Scenario 1 is an anatomically inspired but generic volume defined by an ellipsoid with a flattened side and with cylindrical obstacles in the interior (see Fig. 3.4). Scenario 2 is based on a pleuroscopic scenario using patient anatomy (see Fig. 3.5). In Scenario 2, the CRISP robot enters the volume of a pleural effusion, a serious medical condition which causes the collapse of a patient's lung [21]. The robot

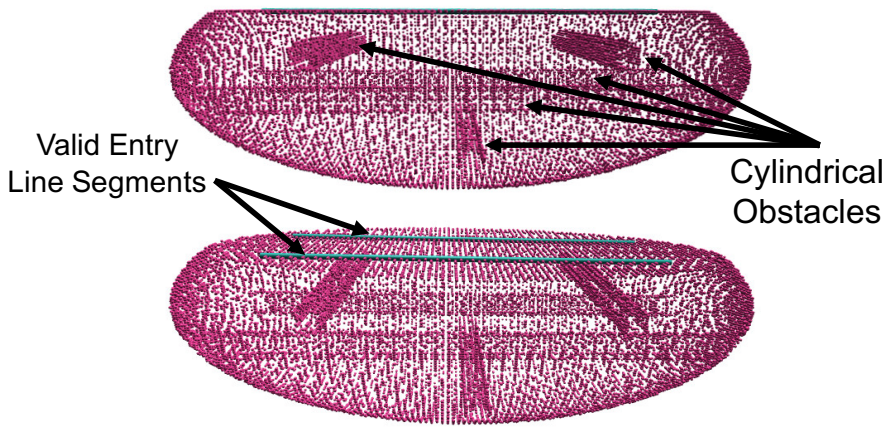


Figure 3.4: Two views of the environment for Scenario 1. The environment is a flat-topped ellipsoidal volume with 5 internal cylindrical obstacles. The line segments which define the valid points of entry are shown as green lines on the top of the volume. The pink spheres (down-sampled for ease of visualization) represent the target regions for the motion planner to view under a sampled design, as well as the obstacles which must be avoided by the robot's geometry during planning.

enters the effusion space between the patient's ribs and maneuvers inside the space to enable a physician to view the internal surface of the volume.

For both scenarios we consider a CRISP robot with a camera tube grasped by one snare tube. In the experiments we define the CRISP robot's set of valid entry points into the volume as a sequence of line segments, on the top surface of the volume for Scenario 1 and between the patient's ribs for Scenario 2. In both scenarios, the pink spheres shown in Figs. 3.4 and 3.5 were used as both the set of target regions to be viewed and as obstacles for the motion planner. All results were generated on a 3.40GHz Intel Xeon E5-1680 CPU with 64GB of RAM.

We compare our method against an implementation which uses the Nelder-Mead algorithm to explore the design space, and which has been used successfully in concentric tube robot design optimization before [46]. Even when using Nelder-Mead to explore the design space, we still use the CRISP motion planner to evaluate the candidate designs, keeping in the spirit of our desire to consider only collision-free motions. We represent this as NM+MP.

For Scenario 1 we allow both algorithms to optimize designs for 16 hours and average the results over 20 different runs. For Scenario 2 we run each algorithm for 8 hours and average over 5 runs. As can be seen in Fig. 3.6, ASA+MP performs well in both scenarios. In Scenario 1, ASA+MP goes from designs that are able to view approximately 20% of the target regions early in the optimization to designs which are able to view approximately 64% of the target regions late in the optimization. In

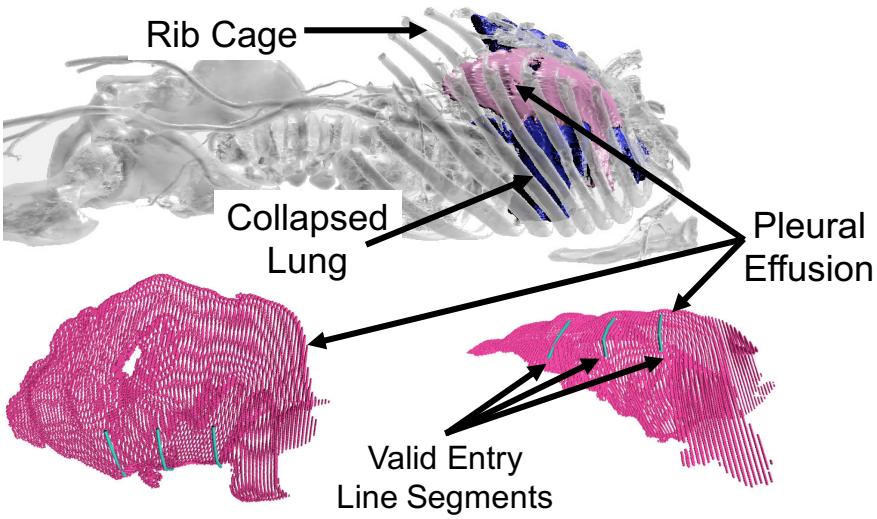


Figure 3.5: Scenario 2 is a segmented pleural effusion volume from a real patient CT scan. The pleural effusion is shown in pink, displacing the collapsed lung (blue). The valid entry line segments (green) are placed such that the robot can enter the pleural space between the ribs. The target regions are rendered as pink spheres in the bottom two images, and which serve both as target regions to be viewed under a design and as obstacles for the motion planner.

Scenario 2, ASA+MP goes from designs that can view approximately 14% to designs that can view approximately 46% of the target regions. ASA+MP also outperforms NM+MP in both Scenarios, finding kinematic designs that enable more target regions to be viewed in less computation time.

Fig. 3.7 shows a comparison between a design early in the ASA+MP optimization process and a design late in the optimization process for Scenario 2. The ability of the later design to visualize a significantly larger percentage of the target regions demonstrates the efficacy of design optimization in this scenario.

3.6 Conclusion

Design optimization can have a large impact on a robot's ability to successfully perform a task. This is especially important in surgical robotics where positive patient outcomes are so vital. In this work, we demonstrated a method for optimizing the kinematic design of the CRISP robot for endoscopic purposes on a patient-specific basis. Our method leverages Adaptive Simulated Annealing (ASA) combined with sampling-based motion planning to ensure that candidate designs are evaluated in such a way that only valid motions, i.e., motions which do not cause the robot to collide with the patient's anatomy, are considered. The results show that the method is able to significantly improve the performance of the robot.

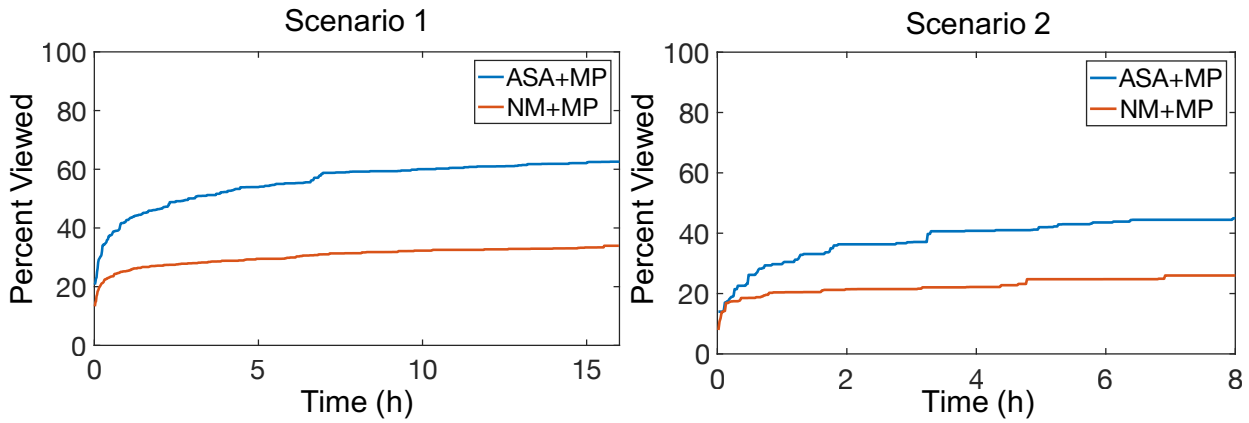


Figure 3.6: The percent of target regions viewed by the best design found over time for Scenarios 1 (top) and 2 (bottom). The blue line represents the results for the ASA algorithm with motion planning, and the red line represents results for the Nelder-Mead algorithm with motion planning. The results are averaged over multiple runs, 20 for Scenario 1 and 5 for Scenario 2.

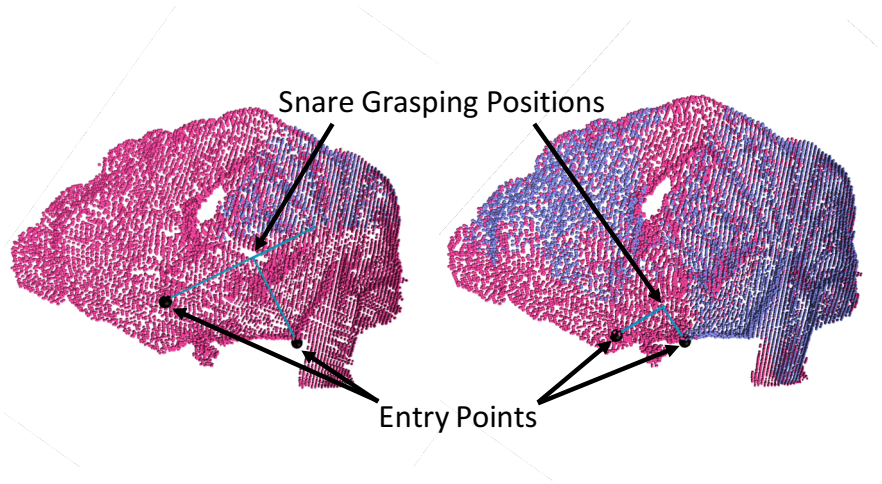


Figure 3.7: A comparison between a design generated after 2 minutes of optimization (left) and a design generated after 8 hours of optimization (right). Starting configurations for each design are overlaid in light blue, the target regions viewed by the design during motion planning are shown in purple, and target regions not viewed are shown in pink. The design generated after 2 minutes is only able to view $< 7\%$ of the target regions, while the design generated after 8 hours of optimization is capable of viewing $> 50\%$ of the target regions.

In future work, we plan to expand upon our results to help bring the CRISP robot closer to clinical use. We plan to implement and evaluate this design optimization method and the motion planner on a physical prototype of the robot. Because two designs may be near optimal but view different regions of the anatomy, we also plan to extend the work to optimize *sets* of designs to maximize the visibility of target regions, as multiple designs could potentially be employed during a single procedure. We also believe that our method and analysis can be extended for purposes of asymptotically optimal design optimization with respect to other continuous objectives and intend to pursue this idea as well. Due to the hours-long scale of the optimization, the current work would require time between imaging and the use of the robot in the clinical setting. The current time-scale would limit the use of this method to procedures for which that delay is clinically feasible, such as the pleuroscopic procedure described above. We would like to explore ways to make the optimization faster, so that it has the potential to be used in more emergent clinical procedures as well.

CHAPTER 4

Motion Planning for a Three-Stage Multilumen Transoral Lung Access System

Lung cancer, which kills over 150,000 people each year in the United States alone, is the leading cause of cancer-related death [73]. Early diagnosis is critical to survival. Medical imaging can detect nodules that are potentially cancerous, but biopsy is required for a definitive diagnosis [74]. Unfortunately, currently available biopsy techniques have significant downsides, especially when nodules are detected in the peripheral zone of the lung (i.e., near the ribs). Percutaneous approaches (i.e, needle insertion between the ribs) require puncturing the pulmonary pleura, the membrane surrounding the lung [75], and hence carry a significant risk of pneumothorax (lung collapse), a serious complication [76]. Current transoral approaches (i.e., inserting a bronchoscope through the mouth into the lungs) has a low risk of pneumothorax but cannot reach many sites in the peripheral lung due to the bronchoscope's diameter and the constraint of staying within the bronchial tubes.

To enable safe and effective lung biopsies for definitive early stage lung cancer diagnosis, a new multilumen transoral lung access system is being developed [5]. This new device provides the benefits of the transoral approach (e.g., low risk of pneuomothorax since the pulmonary pleura is not punctured) while enabling access to peripheral lung nodules. The device uses three stages, as shown in Fig. 4.1. In the first stage, a bronchoscope is inserted transorally and steered to a bronchial tube near the nodule. In the second stage, a concentric tube robot deploys through the bronchoscope, pierces the bronchial tube, enters the lung parenchyma (i.e., the lung's functional tissue), and is oriented toward the nodule. In the third stage, a bevel-tip steerable needle deploys through the concentric tube robot and steers through the lung parenchyma to the nodule while avoiding significant blood vessels. This system, shown in Fig. 4.2, has the potential to enable safe access to peripheral lung nodules, but the strong coupling between each stage of the system makes operation of the system difficult for a human operator.

In this chapter, we introduce a motion planner for this new multilumen transoral lung access system. The motion planner computes actions for each of the system's stages to enable the device

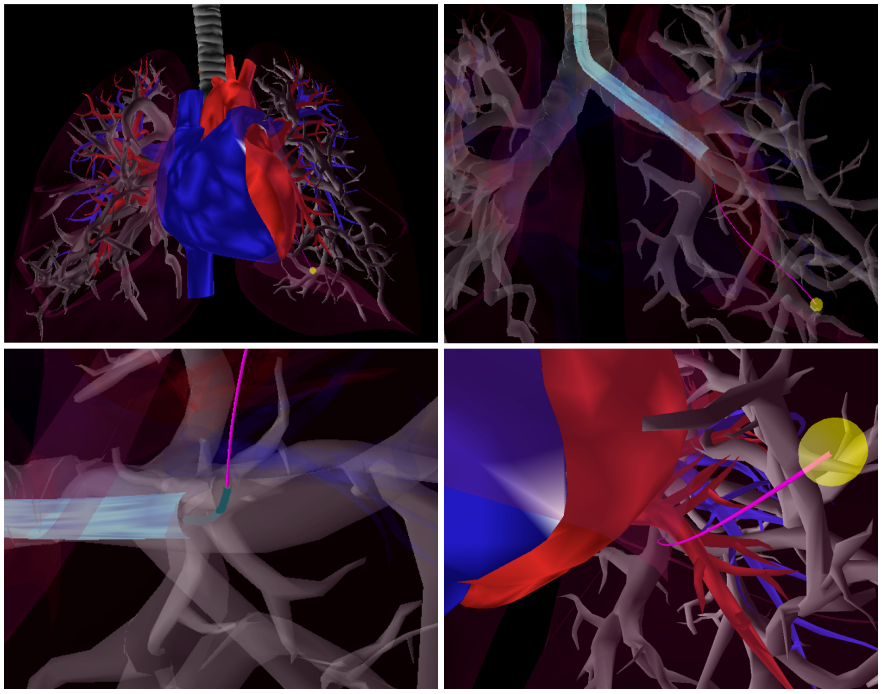


Figure 4.1: Example motion plan for the multilumen transoral lung access system reaching a lung nodule (yellow) while avoiding significant blood vessels (red and blue). Upper Left: Anatomical environment. Upper Right: The first stage of the system, a bronchoscope (light blue), is deployed through a bronchial tube. Lower Left: The second stage of the system, a concentric tube robot (green), is deployed through the bronchoscope and pierces the bronchial tube, entering the lung parenchyma. Lower Right: The third stage of the system, a bevel-tip steerable needle (pink), is deployed from the concentric tube robot and steers through the parenchyma to reach the lung nodule while avoiding anatomical obstacles. Each stage in this coupled system must be considered when computing a safe motion plan.

to safely reach a biopsy target, as shown in Fig. 4.1. The motion planner explicitly considers the coupling between the stages, e.g., the location of the tip of the bronchoscope and the configuration of the concentric tube robot both greatly affect the reachable workspace of the steerable needle and its ability to safely reach the nodule. The input to the motion planner includes specifications of the structure of the bronchial tubes, the target nodule location, and the anatomical obstacles (e.g., significant arteries and veins in the lung), all of which can be either manually or automatically segmented from medical imaging before a procedure [38]. To decrease the risk of internal bleeding, we prefer motion plans in which the steerable needle in the third stage has greater clearance from the anatomical obstacles as it maneuvers to the target nodule.

The motion planner runs in an anytime manner, computing better and better motion plans as computation time is allowed to rise. We use a sampling-based motion planner that builds a variant of

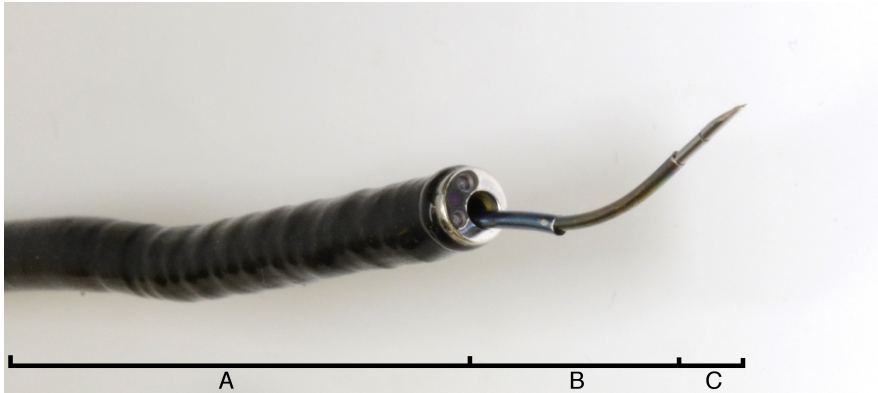


Figure 4.2: The multilumen transoral lung access system consisting of (A) a bronchoscope, (B) a 2-tube concentric tube robot, and (C) an asymmetric-tip steerable needle.

a Rapidly-Exploring Random Tree (RRT) [39] and considers the coupling of the stages in an efficient manner. The plan computed by the motion planner could be used by a physician as a guideline for controlling the device manually or could be used with a feedback controller for automatic execution. We demonstrate the performance of the motion planner in simulation for nodules in the peripheral zone of a lung and show that high quality motion plans can be computed in only a couple of minutes using a standard PC.

This chapter is based on work that was previously published in [12].

4.1 Related Work

Motion planning for the multilumen transoral lung access system requires modeling and computing plans for each of the stages of the system. We use sampling-based motion planning, an approach that has been very successful for robots with many degrees of freedom for a variety of applications [72].

The first stage of our system is a bronchoscope, an endoscopic device that is typically inserted into a patient's airway via the mouth and can maneuver through large-diameter bronchial tubes. Rosell et al. use haptic feedback and visual navigation for virtual bronchoscopy [34] and demonstrate successful guidance of a bronchoscope toward lesions in the bronchi. We are interested in positioning the bronchoscope to deploy other tools to reach points beyond the bronchi.

The second stage of our system deploys a concentric tube robot via the working channel of the bronchoscope. Concentric tube robots consist of thin, nested, pre-curved nitinol tubes. As the tubes are individually rotated and translated, the entire shape of the robot changes, enabling the robot to curve around anatomy to reach surgical sites unreachable by traditional straight tools.

Deploying concentric tube robots through tendon-actuated endoscopic devices has recently been shown to effectively increase surgical dexterity [77, 78]. Controlling the motion of a concentric tube robot is difficult due to the complicated mechanics of the tubes' interactions. Kinematic modeling of concentric tube robots has rapidly developed to consider bending and torsional interactions among the tubes [79, 9, 24]. Prior work has also achieved position control [24, 9, 25] and obstacle avoidance [4] with these robots. In our work, we use a mechanics-based kinematic model [9] to compute the concentric tube robot's shape during planning.

The third stage of our system deploys an asymmetric-tip steerable needle via the concentric tube robot. Asymmetric-tip steerable needles, such as those with a bevel tip, are flexible and exert an asymmetric force on soft tissue when inserted, causing them to achieve curved paths through the body [27]. The needle can be steered by axially rotating the needle, which changes the direction of the bevel tip. Kinematic models of varying complexity have been proposed for steerable needles [27]. Park et al. proposed a unicycle model on which our work is based [80], and various kinematic models were analyzed and experimentally validated by Webster et al. [81]. We can vary the turning radius during insertion of steerable needles using duty cycling [82].

Both control and motion planning for steerable needles have received a lot of attention [83, 84]. Park et al. developed a path-of-probability algorithm for steerable needle control based on error propagation [85]. Hauser et al. proposed a control method to account for tissue deformation during needle insertion by adapting helical trajectories [28]. Bernardes et al. achieved closed-loop control using feedback from medical imaging for planar needle steering [29]. Closed-loop control of steerable needles has also been achieved without knowledge of needle curvature properties using a sliding mode controller [86]. Seiler et al. accounted for uncertainty during insertion through fast trajectory correction [30]. Variations on the Rapidly-exploring Random Tree (RRT) algorithm have been successfully applied to steerable needles for a variety of applications. Obstacle avoidance in 3D with closed-loop feedback has been achieved through high-frequency replanning [31, 32]. We use the planning approach of Patil et al. [31] to quickly generate many different candidate paths.

The primary concern of this chapter is combining the motion planning for each of these three individual stages into a single cohesive algorithm. Computing a motion plan for the entire system can be considered a special case of a hybrid system; Branicky et al. provide a general framework for using RRTs for hybrid system planning [87]. We introduce a specialized approach that explicitly

considers the coupling across different stages of the multilumen transoral lung access system.

4.2 Problem Definition

Our motion planner requires as input a specification of the geometry of the relevant patient anatomy, including the target site and obstacles to avoid. Prior to lung biopsy procedures, physicians typically obtain a volumetric medical image (e.g., a CT scan) from which this data can be obtained via manual or automatic segmentation of the patient anatomy [38]. Specifically, we require (1) the geometry of the bronchial tubes B, (2) the geometry of significant blood vessels V in the lung, including arteries and veins, whose puncture would result in clinically significant internal bleeding, and (3) a point specifying the target site $\mathbf{p}_{\text{goal}} \in \mathbb{R}^3$. In our implementation, we assume B and V are represented using polygonal surface meshes. We define the set of anatomical obstacles O for needle steering in the parenchyma as the union of B and V, since the needle should not pierce a significant blood vessel and should also avoid bronchial tubes that would cause the needle to deflect from its intended path.

The multilumen transoral lung access system consists of three stages, and the configuration of the system encodes the degrees of freedom of each stage. The first stage of the system, the bronchoscope, is deployed via the patient's airway to a specific point $\mathbf{p}_{\text{scope}}$ inside the bronchial tubes. The second stage of the system, the concentric tube robot, is deployed through the bronchoscope, exiting at its tip, $\mathbf{p}_{\text{scope}}$. We parameterize the 2-tube concentric tube robot's configuration by the vector $\mathbf{m} = \{\beta_1, \beta_2, \theta_1, \theta_2\}^T$ where $\beta_k \in \mathbb{R}$ for $k \in \{1, 2\}$ is the insertional degree of freedom of the k'th tube and $\theta_k \in [0, 2\pi)$ is the rotational degree of freedom of the k'th tube. The third stage of the system is a steerable needle deployed through the concentric tubes. We describe the configuration of the steerable needle by the pose of the needle tip, which we represent by the 4×4 matrix

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix}$$

where $\mathbf{R} \in \mathcal{SO}(3)$ is the rotation matrix representing the orientation of the needle tip and $\mathbf{p} \in \mathbb{R}^3$ denotes the needle tip's position in the coordinate frame of the medical image. The configuration of

the entire system can then be described by the tuple

$$\mathbf{q} = (\mathbf{p}_{\text{scope}}, \mathbf{m}, \mathbf{X})$$

where $\mathbf{q} \in \mathcal{Q} = \mathbb{R}^8 \times (\mathbb{S}^1)^2 \times \mathcal{SO}(3)$.

In the third stage, the bevel-tip steerable needle is controlled by inserting the needle and axially rotating it about its base. Because the needle is flexible and has an asymmetric tip, it curves in the direction of the bevel [80, 81]. We define the control input to the steerable needle as $\mathbf{u} = \{l, \phi, \kappa\}^T$ where $l \in \mathbb{R}$ is an insertion distance, $\phi \in [0, 2\pi)$ is the axial rotation angle of the asymmetric tip (which corresponds to the angle defining the plane of the needle's curvature when inserted), and $\kappa \in \mathbb{R}^+$ is the curvature achieved using duty cycling [82]. We note κ has an upper bound of $\kappa_{\max}$, which is a property of a specific steerable needle in lung tissue.

The system is designed such that the location of the bronchoscope tip $\mathbf{p}_{\text{scope}}$ is set first and fixed, then the concentric tube robot configuration $\mathbf{m}$ is set and fixed, and finally the steerable needle is controlled such that its tip pose $\mathbf{X}$ maneuvers to the target site while avoiding obstacles. We define a motion plan σ as the sequence of configurations of the system $\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_T\}$ and the corresponding control inputs for each stage of the system.

To maximize safety, we prefer motion plans that guide the steerable needle along paths with greater clearance from the anatomical obstacles. We define the cost of a motion plan as

$$c(\sigma) = \int_{\mathbf{p}_0}^{\mathbf{p}_T} \frac{1}{\texttt{clearance}(\mathbf{p}, O)} d\mathbf{p} \tag{4.1}$$

which is the integration over the curve of the clearance between the tip of the steerable needle and the closest anatomical obstacle over the continuous path of the tip resulting from σ. This cost function encodes a balance between clearance from obstacles and path length, as both small clearance and long paths incur risk of damage to the patient. The objective of our motion planner is to compute a motion plan σ such that the final position $\mathbf{p}_T$ of the needle tip at configuration $\mathbf{q}_T$ reaches $\mathbf{p}_{\text{goal}}$, the obstacles O are avoided, and cost $c(\sigma)$ is low.

Algorithm 2: Motion planner for the multilumen lung access system.

input : T: time to execute outer loop; T_{RRT}: time allotted to an individual RRT; B: surface mesh of bronchial tubes; V: surface mesh of blood vessels; $\kappa_{\max}$: maximum needle curvature; $\mathbf{p}_{\text{goal}}$: target site position

output : σ: best motion plan computed by time T

```
σ ← ∅;
while elapsed time < T do
    p_scope ← sample_bronchial_medial_axis();
    θ1 = θ2 ← uniform_random(0, 2π);
    β1, β2 ← random_tube_translations();
    m ← {β1, β2, θ1, θ2};
    X_start ← CTR_tip_frame(p_scope, m);
    if p_goal is in workspace of needle at X_start then
        σ_needle ← RRT(X_start, T_RRT, B, V, κ_max, p_goal);
        if σ_needle ≠ null then
            σ' ← (p_scope, m, σ_needle);
            if c(σ') < c(σ) then
                σ ← σ'
            end
        end
    end
end
return σ;
```

4.3 Method

Our motion planner (Alg. 2) computes motions for each stage of the lung access system, including the bronchoscope, the concentric tube robot, and the steerable needle, to enable access to the physician-specified goal. The motion planner first searches for an optimal placement of the bronchoscope and the concentric tube robot by setting the deployment variables $\mathbf{p}_{\text{scope}}$ and $\mathbf{m}$. A desirable $(\mathbf{p}_{\text{scope}}, \mathbf{m})$ pair places the steerable needle's start state $\mathbf{X}_{\text{start}}$ such that the needle can follow a collision-free path to the goal with large clearance from anatomical obstacles. We evaluate this property of a sampled $(\mathbf{p}_{\text{scope}}, \mathbf{m})$ pair by using a sampling-based motion planner to find steerable needle controls $(\mathbf{u}_1, \mathbf{u}_2, \dots)$ that create a collision-free path to the goal point $\mathbf{p}_{\text{goal}}$. We re-execute the motion planner for the entire system repeatedly for a given amount of time, leveraging randomization to create many different motion plans and then selecting the one with lowest cost (i.e., greater clearance from obstacles).

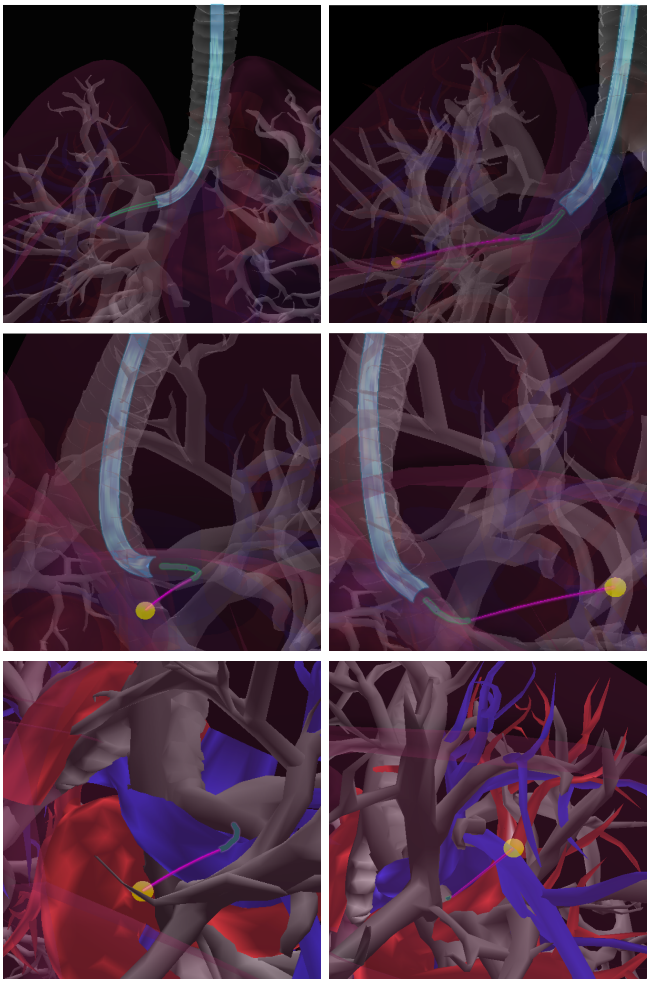


Figure 4.3: Accessing goals in the right lung, the bronchoscope utilizes different branches in the bronchial tree: the upper right branch (left images) for one goal and the lower right branch (right images) for another goal. Top to bottom show the same paths from different angles and with different transparencies for visualization.

4.3.1 Planning Deployment of the Bronchoscope

We use a sampling-based approach to create candidate placements of the bronchoscope tip $\mathbf{p}_{\text{scope}}$ within the bronchial tree. For efficiency, we leverage the fact that the bronchial tubes form a tree, which is a linear structure, and create a mapping between a parameter $y \in \mathbb{R}$ and the point $\mathbf{p}_{\text{scope}}$ in the world coordinate system. We assume that all possible bronchoscope placements must lie on the medial axis of the bronchial tree (which can be constructed automatically [88], although in our results an approximation was constructed manually). In practice, in certain wide-diameter bronchial tubes it may be possible to manipulate the bronchoscope such that its tip is not on the medial access, which will be a topic of future work. We uniformly sample points from the line

segments of the medial axis by placing the medial axis's line segments $s_i = (\mathbf{p}_i, \mathbf{p}'_i)$ in an arbitrarily ordered sequence $(s_1, \ldots, s_n)$ and then viewing this sequence of segments as a piecewise linear (and discontinuous) space curve parameterized by arc length $y \in [0, \sum_i \|\mathbf{p}_i - \mathbf{p}'_i\|]$. We can sample a parameter y from the domain of this space curve to generate a placement for the bronchoscope tip $\mathbf{p}_{\text{scope}}$ in the bronchial tree, and can then directly map y to $\mathbf{p}_{\text{scope}}$ via the mapping from the medial axis to the world coordinate system. The bronchoscope's ability to move through different branches of the bronchial tree broadens the set of reachable goals in the lung (Fig. 4.3).

4.3.2 Planning Deployment of the Concentric Tube Robot

The sampled placement $\mathbf{p}_{\text{scope}}$ of the bronchoscope tip describes the start point of deployment of the concentric tube robot. From this point, we wish to sample possible deployments of the concentric tube robot, where deployments are parameterized by the vector $\mathbf{m} = \{\beta_1, \beta_2, \theta_1, \theta_2\}$ to encode each component tube's axial rotation and translation. Due to physical constraints of our system and the desire for follow-the-leader trajectories that do not significantly deform tissues, we disallow relative rotation between the tubes. Hence, we can parameterize deployment with one rotation θ. A deployment $\mathbf{m}$ corresponds to sequentially performing the following actions: (1) rotating the tubes to required angle θ, (2) inserting both tubes at the same rate until the outermost tube reaches its stopping point β_2, and then (3) inserting the inner tube past the outer tube until it reaches its stopping point β_1.

We use a mechanics-based kinematic model [9] to compute the tip frame of the concentric tube robot after deployment. This tip frame marks the initial configuration $\mathbf{X}_{\text{start}}$ of steerable needle deployment. If the goal point $\mathbf{p}_{\text{goal}}$ lies outside of the reachable workspace of the steerable needle when deployed from $\mathbf{X}_{\text{start}}$ (see Fig. 4.4), we reject this proposed bronchoscope and concentric tube robot deployment.

4.3.3 Steering the Bevel-Tip Needle to the Goal Point

A proposed placement $(\mathbf{p}_{\text{scope}}, \mathbf{m})$ of the bronchoscope and concentric tube robot fully specifies the start state $\mathbf{X}_{\text{start}}$ of the steerable needle. We evaluate the placement $(\mathbf{p}_{\text{scope}}, \mathbf{m})$ by attempting to find a steerable needle trajectory starting at $\mathbf{X}_{\text{start}}$ that reaches the goal while avoiding anatomical obstacles. We search for this trajectory using a sampling-based motion planning algorithm for steerable needles developed by Patil et al. [31].

This motion planner is based on a Rapidly-exploring Random Trees (RRT) motion planner [72].

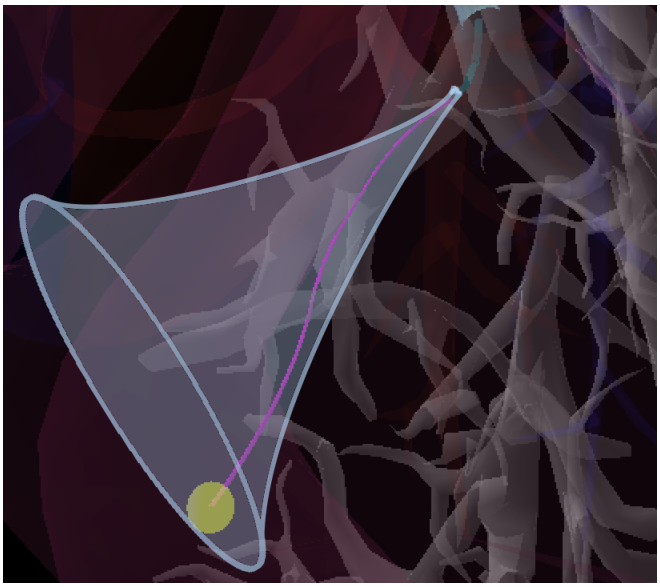


Figure 4.4: The reachable workspace of the steerable needle is a trumpet shaped volume defined by its start configuration and the maximum curvature κ_{max} it is capable of achieving in the tissue.

RRT incrementally builds a tree of robot states that are reachable from $\mathbf{X}_{\text{start}}$ by collision-free paths. At each iteration, RRT samples a state $\mathbf{X}_{\text{sample}}$ in the robot's state space, uses a distance function `distance` to select the nearest state $\mathbf{X}_{\text{near}}$ already in the tree, and uses a function `steer` to compute a control input $\mathbf{u}$ that, when applied to $\mathbf{X}_{\text{near}}$ results in a new state $\mathbf{X}_{\text{new}}$ nearer to $\mathbf{X}_{\text{sample}}$. If the motion between $\mathbf{X}_{\text{near}}$ and $\mathbf{X}_{\text{new}}$ is collision-free, $\mathbf{X}_{\text{new}}$ is added to the tree. RRT iterates these steps until either (1) the tree connects to the goal, or (2) the algorithm exceeds its time allotment.

To compute the `steer` function, we compute the unique control $\mathbf{u}_{\text{steer}}$ that directly connects the state $\mathbf{X}_{\text{near}}$ to the 3D position of $\mathbf{X}_{\text{sample}}$, and then clamp the insertion length of $\mathbf{u}_{\text{steer}}$ to a maximum of l_{max} to generate the resulting control $\mathbf{u}$. To compute the `distance` function between two states $\mathbf{X}$ and $\mathbf{X}'$, we compute the control $\mathbf{u}_{\text{steer}}$ that connects $\mathbf{X}$ to the 3D position of $\mathbf{X}'$ as above and return the insertion arc length of $\mathbf{u}_{\text{steer}}$. We check whether the motion between two states is collision-free by using the Flexible Collision Library (FCL) [89] for collision detection between the needle and the anatomical obstacles.

In order to more quickly generate motion plans to the goal $\mathbf{p}_{\text{goal}}$, we introduce a strong goal bias; after each iteration of RRT, we perform an additional RRT iteration using $\mathbf{p}_{\text{goal}}$ as $\mathbf{X}_{\text{sample}}$ in order to bias growth of the tree toward the goal.

Additionally, due to stresses on surrounding tissue, bevel-tip steerable needle insertion is impeded

if the orientation of the needle tip is oriented more than 90° from its orientation at $\mathbf{X}_{\text{start}}$. As such, controls that produce such states are discarded as invalid.

We allow the RRT to search for a collision-free path to the goal point for 0.05 seconds. This time limit is based on an intuition that the RRT either finds a path very quickly or not at all in this scenario. If a path is found, we use the path's clearance metric from Eq. 4.1 to quantify the cost of this particular placement $(\mathbf{p}_{\text{scope}}, \mathbf{m})$ of the bronchoscope and concentric tube robot. In practice, we approximate the integral in Eq. 4.1 by finely discretizing the needle's path and using the trapezoidal rule. We compute the `clearance` function using FCL [89]. If the RRT fails to find a collision-free path in the allotted time, we discard $\mathbf{X}_{\text{start}}$.

4.3.4 Motion Planning for the Entire System

After generating a placement $\mathbf{p}_{\text{scope}}$ of the bronchoscope, a deployment $\mathbf{m}$ of the concentric tube robot, and appropriate control inputs $(\mathbf{u}_1, \mathbf{u}_2, \dots)$ for the steerable needle, we concatenate these operations to form a motion plan σ of a collision-free sequence of configurations $(\mathbf{q}_1, \mathbf{q}_2, \dots)$ and associated control inputs for the entire multilumen lung access system.

The approach above generates a single plan, but we can repeat the process to generate new, different plans due to the randomization in the algorithm. We iteratively create new motion plans σ and evaluate their quality, always saving the best motion plan found so far. We repeat until we exhaust the time allotted for plan computation.

4.4 Results

We evaluated our motion planner in a simulated lung biopsy scenario. The scenario consists of a commercially available anatomical model which includes the bronchial tree, both the left and right lungs (evaluated separately), the heart, and large vasculature. We sampled valid bronchoscope access points with the manually approximated medial axis shown in Fig. 4.5. While the bronchoscope can physically access both the blue and purple segments in the figure, we did not consider the purple segments because exiting the bronchus from points along these segments would require puncturing the pulmonary pleura and increase the risk of pneumothorax. We blocked off deeper parts of the bronchial tree where a bronchoscope's diameter would typically impede further progress due to the narrowing of the bronchial tubes. We used a maximum needle steering curvature $\kappa_{\max} = (0.121\text{m})^{-1}$ in accordance with the physical prototype in prior work [90, 5]. We also used concentric tube robot design parameters that matched this prior work.

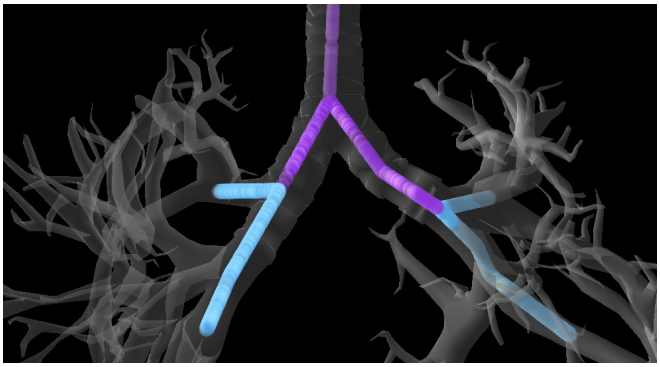


Figure 4.5: Positions from which the bronchoscope can safely deploy the latter stages of the system without puncturing the pulmonary pleura are shown in cyan. These positions can be reached via the airways shown in purple.

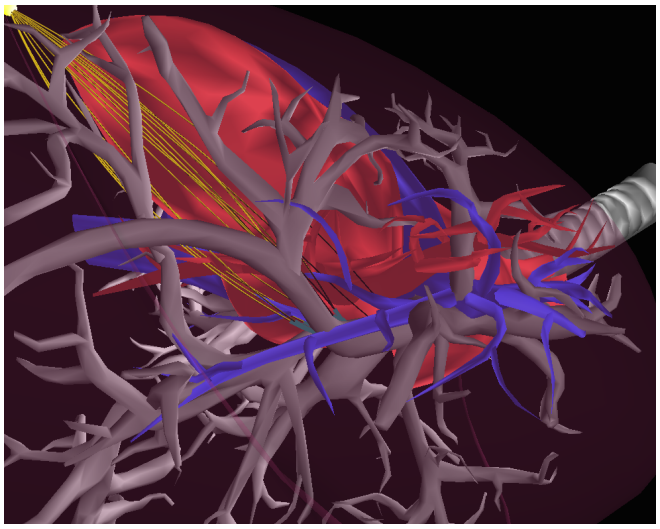


Figure 4.6: The first 15 solutions (in gold) found to a single goal spanning multiple homotopic classes.

In Fig. 4.1, we show the output of our motion planner, which computes the bronchoscope tip position, concentric tube robot configuration, and steerable needle control inputs to guide the device to a nodule while avoiding obstacles. Depending on the computation time allowed, our method may generate large numbers of motion plans to a target. In Fig. 4.6, the method computed 15 feasible motion plans to the target nodule in 5.63 seconds. This set allows our motion planner to select the best plan given the cost function. The motion plans in the set span multiple homotopic classes.

We evaluated the ability of our motion planner to compute plans to reach nodules across the peripheral zone of the lung. We randomly sampled 50 lung nodule locations in the periphery of each lung (100 total nodules) as shown in Fig. 4.7. For each query nodule, we ran our motion planner

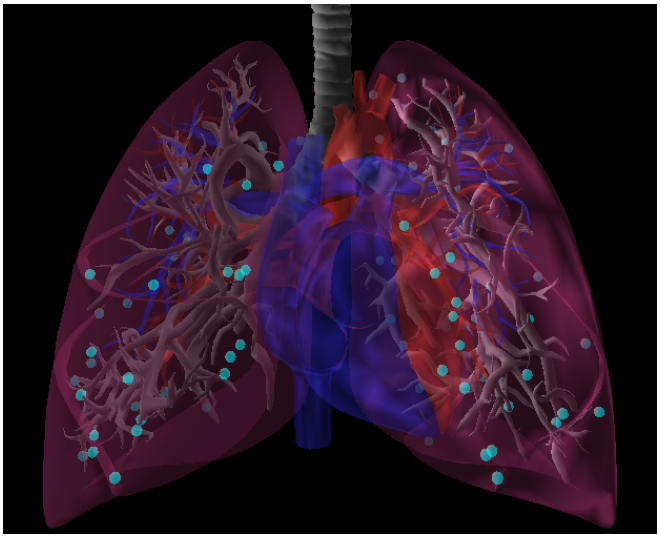


Figure 4.7: Lung nodule query locations (shown in teal) used for the reachability experiment.

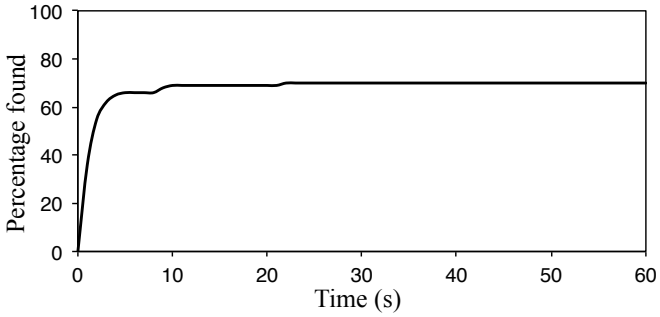


Figure 4.8: Percentage of randomly sampled peripheral lung nodules (100 nodules) to which a motion plan has been found as a function of computation time.

for 1 hour. We show the percentage of nodules reached using our motion planner as a function of computation time (first 60 seconds) in Fig. 4.8. For 36% of the lung nodules, our motion planner computed a valid motion plan to reach them in the first second of planning. By 60 seconds, our motion planner found valid plans reaching 70% of the nodules and by 1 hour valid plans were found to 75% of the nodules. We note that, with a steerable needle of tighter curvature (a subject of future work regarding the mechanical design), the percentage of nodules for which a plan is found would likely increase substantially.

We also investigated how motion plan quality increases as we increase computation time. We selected 6 of the reachable lung nodules found in the experiment above and executed 3 instances of our motion planner on each nodule for 300 seconds per instance. In Fig. 4.9, we show the best costs found by the motion planner over time for each considered nodule.

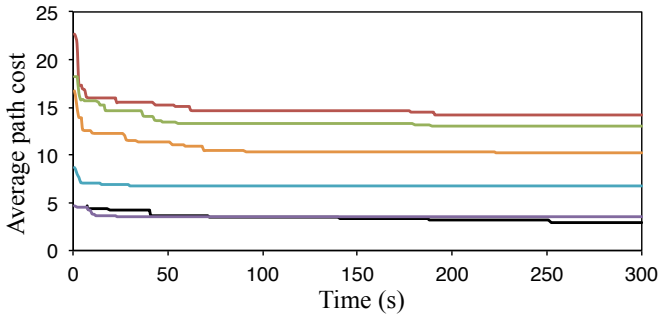


Figure 4.9: Over six distinct goals (three from each lung), the cost of the best path found by time averaged over three runs each.

Recall that we only consider placements $(\mathbf{p}_{\text{start}}, \mathbf{m})$ of the bronchoscope and concentric tube robot that place $\mathbf{p}_{\text{goal}}$ in the workspace of the steerable needle. We demonstrate the computational speedup of this algorithm design choice by leaving it out of our algorithm and comparing the results. With the placement rejection scheme, we found 15 paths to the goal shown in Fig. 4.6 in 5.63 seconds; without the placement rejection scheme, it took 42.62 seconds to find 15 paths to the goal.

To justify our choice of T_{RRT}, we ran an additional experiment where we allowed each instance of the RRT 2 seconds to find a solution. We found that a large percentage of valid RRTs were found very quickly, with more than 95% of valid plans being found within the first 0.05 seconds.

4.5 Conclusion

We introduced a motion planner for a three-stage multilumen transoral lung access system. The planner computes actions for deployment of a bronchoscope into the bronchial tubes, followed by concentric tube robot deployment into the lung parenchyma, and finally deployment of a bevel-tip steerable needle to reach a goal site while avoiding collisions with anatomical obstacles. Our sampling-based motion planner quickly computes plans with high clearance from obstacles in a simulated clinical scenario involving biopsy of lung nodules.

In future work, we will consider additional anatomical obstacles (such as lung fissures) to further decrease risk to the patient. We also plan to quantify the reachable space of the robot in the lung model and to investigate different sampling strategies and motion planning paradigms. We also plan to relax some of our restrictive assumptions on the concentric tube robot and consider approximate follow-the-leader plans that provide added control but are still safe to perform in the lung.

In Chapter 5, we integrate aspects of this motion planner with the hardware developed by Swaney et al. [5] and evaluate its efficacy in porcine lung tissue.

CHAPTER 5

Toward Transoral Peripheral Lung Access: Steering Bronchoscope-Deployed Needles through Porcine Lung Tissue

The three-stage transoral lung access system, introduced in [5] and discussed in Chapter 4 leverages the minimally invasive nature and lowered risks of the transoral biopsy approach, while having the potential to reach difficult-to-access lung nodules in the periphery of the lung with high accuracy. The system consists of three stages deployed in sequence: a bronchoscope, a concentric tube channel, and a flexure-tip steerable needle (see Figs. 5.1 and 5.2). First, the physician guides the bronchoscope to a feasible location en route to the nodule. From the working channel of the bronchoscope, the concentric tube channel is deployed, bending toward the wall of the bronchial tree. Using a pneumatic puncture mechanism, the tube then pierces through and enters the parenchyma, orienting its tip toward the lung nodule. The flexure-tip steerable needle then deploys from the tip of the concentric tube and steers through the lung parenchyma, curving around sensitive structures such as large vasculature to reach the nodule. The needle's controller uses an NDI Aurora 6-DOF magnetic tracking probe which is embedded in the tip of the flexure-tip steerable needle. The robot has previously been evaluated only in phantom gel in [5] and in simulation [12], as discussed in Chapter 4.

In this chapter, we report the first results for the deployment of the robotic lung access system inside an inflated *ex vivo* porcine lung in a CT scanner. We segmented the lung in the CT scan, including bronchial tubes and blood vessels. We steered the needle to avoid obstacles in the parenchyma (i.e., blood vessels and bronchial tubes) and achieved clinically-desirable accuracy in accessing targets near the lung periphery.

This chapter is based on work originally publish in [13].

5.1 Materials & Methods

We inflated the *ex vivo* porcine lung using a pressure regulator attached to an endotracheal (ET) tube inserted into the trachea. We connected a T-connector to the ET tube, capped with a thin

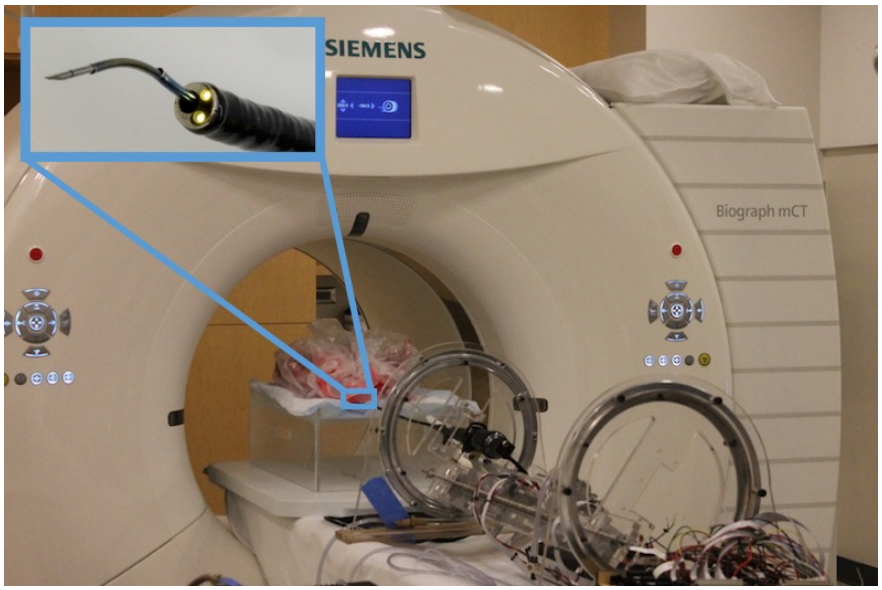


Figure 5.1: Our transoral lung access system, consisting of a concentric tube channel and a steerable needle deployed via a bronchoscope, operating in an inflated *ex vivo* porcine lung. CT scans were acquired at each stage of deployment.

membrane to maintain lung inflation, through which we inserted the robot.

We estimated the needle's maximum curvature by inserting the needle without rotation into inflated lung tissue, recording the tip's path using magnetic tracking, and fitting a circle to the path. This resulted in an experimentally determined curvature value of 0.498 m^{-1}. This curvature value was used in all future experiments.

To conduct the system experiments, we transported the system to the CT scanner, inflated the lung, and acquired a CT scan (with voxel sizes of 0.9mm×0.9mm×1.0mm). We segmented the bronchial tubes and significant vessels in the CT scan using a combination of manual segmentation and region growing-based segmentation modules in 3D Slicer [91] (see Fig. 5.3).

The needle steers through the anatomy using feedback from the electromagnetic tracking system mentioned above. As such, a transformation between the CT scan and the magnetic tracker's frame needed to be computed. To do so, we affixed 3 spherical fiducials to the surface of the lung prior to the CT scan, which we then manually segmented in the CT scan. To register the scan to the magnetic tracker we then probed the fiducials with a hand-held magnetic tracking probe and computed the rigid-body transformation between the two reference frames.

The focus of our experiments was to evaluate the ability of the steerable needle to steer through

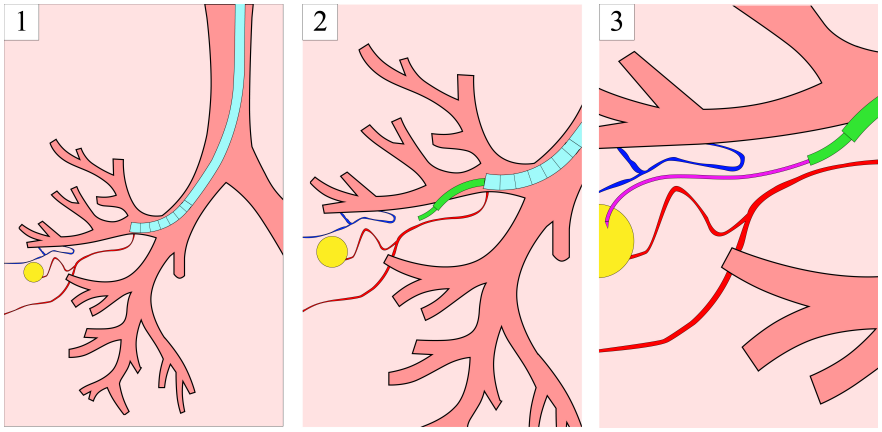


Figure 5.2: The robot's three stages: (1) the bronchoscope is inserted into the airway, (2) the concentric tube channel is deployed and exits the bronchial tube, and (3) the steerable needle travels through the parenchyma to the nodule.

lung parenchyma and accurately reach targets in the peripheral lung. We evaluated the steerable needle for 6 deployments. For each deployment, we started with the bronchoscope outside the lung and manually guided the bronchoscope into the airway. The concentric tube was then robotically deployed from the bronchoscope's tip to the bronchial tube wall where it pneumatically punctured into the lung parenchyma. Constrained to the reachable workspace of the needle, we chose a point on the surface of the lung as the target by projecting a straight line from the concentric tube and selecting a random offset. This process was aided by the hand-held electromagnetic tracking probe mentioned above, which allowed us to constrain the offset to the lung's surface. Although clinical peripheral targets would be just below the surface, we selected targets on the surface for ease of measurement. We then planned a needle path [12] that avoids the obstacles (i.e., the segmented blood vessels and bronchial tubes) and deployed the steerable needle using closed-loop control [86] to guide the tip of the needle along the planned path (see Fig. 5.3). We conducted 3 deployments with obstacle avoidance, and 3 deployments without obstacle avoidance in which we skipped motion planning and applied the automatic controller directly to the target.

5.2 Results

We show CT scan slices in Fig. 5.4 for a deployment of the system. Table 5.1 shows the accuracy results for the 6 deployments (3 with obstacle avoidance and 3 without). The average tip error was 1.97 mm with obstacle avoidance and 1.07 mm without, with average needle insertion length 64.8 mm.

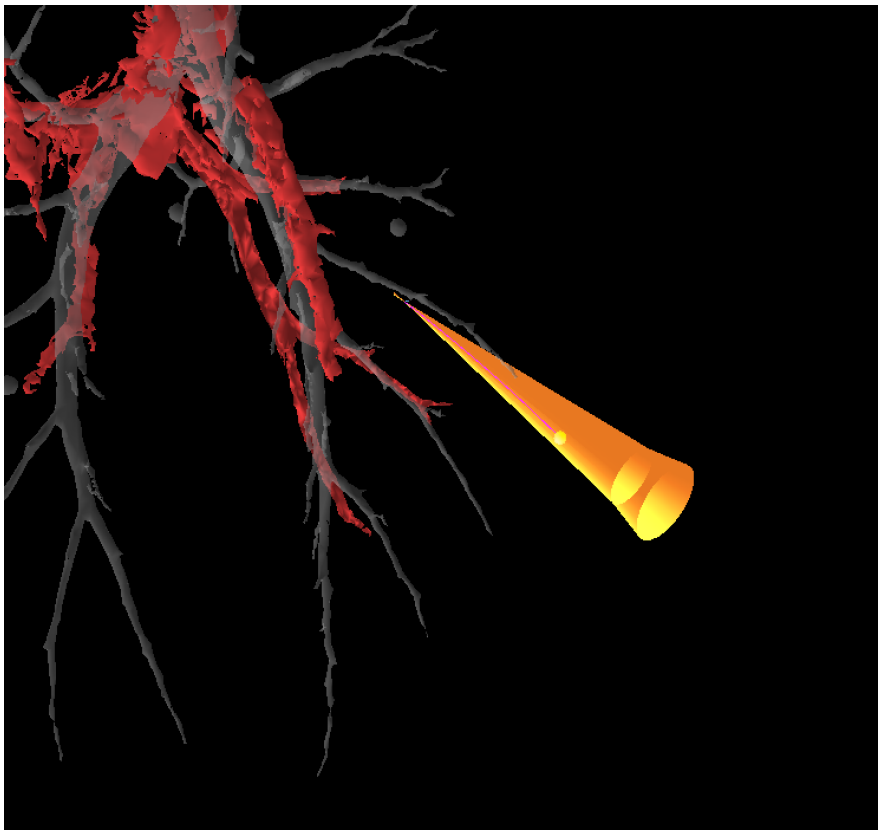


Figure 5.3: From the CT scan we segment the bronchial tree (gray), and large vasculature (red). The motion planner then computes a plan for the needle (pink) to reach a target point (yellow sphere) that obeys the maximum curvature constraint (reachable volume subject to that constraint here visualized by the orange trumpet shape).

5.3 Discussion

Our experimental results show that the device is on track to achieve accurate biopsy of clinically-relevant small suspicious nodules, which are defined by the American College of Radiology as being as small as 6 mm in diameter. The results also illustrate the trade-off of obstacle avoidance; the obstacles place constraints on the motion planner and controller that restrict the needle's feasible workspace resulting in slightly larger tip error, but avoiding significant blood vessels is important to reduce the risk of internal bleeding.

	Run 1	Run 2	Run 3	Avg
With obstacle avoidance	0.83	3.27	1.82	1.97
Without obstacle avoidance	0.19	2.47	0.55	1.07

Table 5.1: Tip error (mm) as measured by the magnetic tracker.

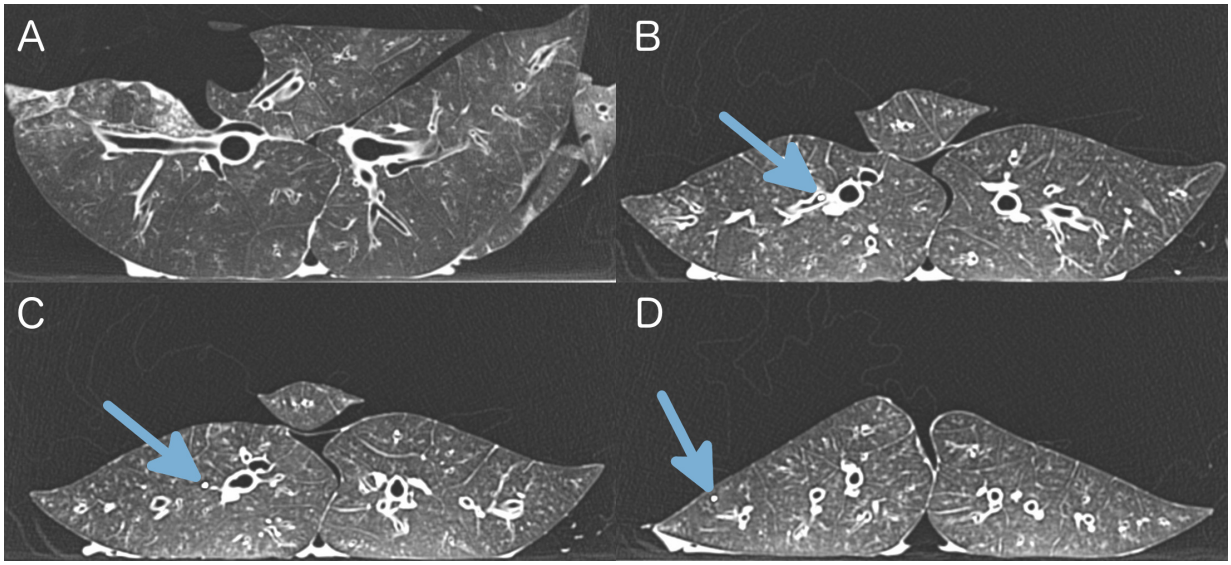


Figure 5.4: CT scans of the inflated lung, (A) prior to inserting the robot, (B) prior to the pneumatic puncture, (C) after the pneumatic puncture, and (D) after deployment of the steerable needle to the lung periphery. The blue arrows point to the robot.

The anatomical segmentation used in this work is both insufficiently detailed and the process too difficult for clinical applications. In recent work, we have demonstrated automatic segmentation of a much more extensive set of relevant lung anatomy [92]. Additionally, in this work we did not characterize the registration error between the CT scan and the electromagnetic tracker. This remains the subject of future work, but will be a necessary step prior to clinical deployment.

The early diagnosis of suspicious lung nodules is integral to combatting lung cancer. To bring definitive diagnosis to a larger percentage of the population earlier in the course of the disease, a new class of medical devices will be required. Our transoral lung access system has the potential to be such a device. While integration into clinical workflows must still be investigated, in this work we showed the ability of the bronchoscope-deployed steerable needle to reach targets with high accuracy in inflated *ex vivo* porcine lungs.

CHAPTER 6

Fast Anytime Motion Planning in Point Clouds by Interleaving Sampling and Interior Point Optimization

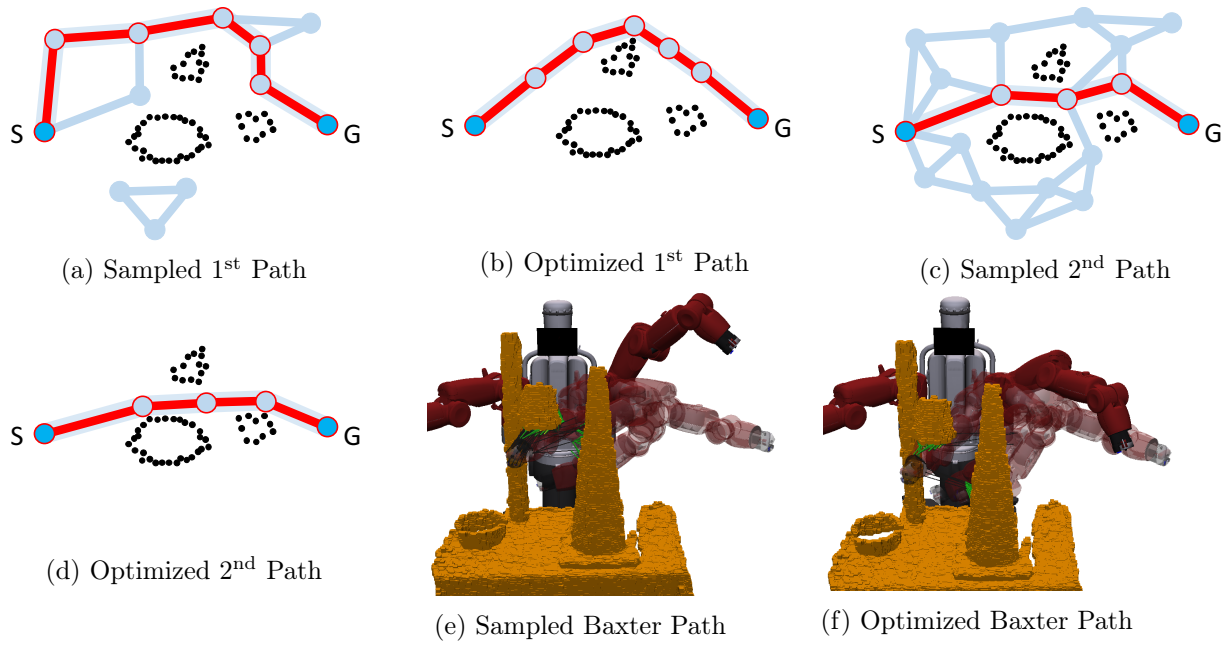
Robotic manipulators are entering unstructured environments, such as homes, offices, hospitals, and restaurants, where robots need to plan motions quickly while ensuring safety via obstacle avoidance. Motion planning in such settings is challenging in part because the robot must rely on real-world sensors such as laser scanners, RGB-D sensors, or stereo reconstruction, which typically produce point clouds. In addition, enabling intuitive, interactive, and reactive user experiences requires that the robot generate plans of high quality as quickly as possible, without necessarily knowing in advance the maximum time allocatable to motion planning. Hence, motion planning in such settings should be implemented as an anytime algorithm, meaning the algorithm progressively improves its solution and can be interrupted at any time and return a valid solution.

To enable fast, anytime, asymptotically-optimal motion planning in point clouds, we propose to blend two popular motion planning paradigms: (1) sampling-based motion planning and (2) optimization-based motion planning. Sampling-based motion planners can directly use point cloud data by incorporating appropriate collision detection algorithms [89] and can optimize a cost metric in an asymptotically optimal manner [93]. However, sampling-based motion planners in practice can be slow to converge and frequently return paths in finite time that are far from optimal, especially for problems with higher dimensional configuration spaces [94]. In contrast, optimization-based motion planners are often very fast [95, 96, 97, 98], but typically do not converge to globally optimal solutions, do not operate in an anytime manner (since, even when initialized with a feasible solution, their intermediate iterations may consider robot configurations that are in collision with obstacles), and are inefficient when using large point clouds. In this chapter, we introduce a new optimization approach to motion planning based on interior point optimization [99, 100], which has unique properties that enable anytime motion planning and efficient handling of large point clouds. By integrating our new interior point optimization formulation with a sampling-based method, our

approach quickly computes locally optimized plans in an anytime fashion, and continues to refine the solution toward global optimality for as long as time allows.

Our new method, Interleaved Sampling and Interior point optimization Motion Planning (ISIMP), interleaves global exploration with local optimization (see Fig. 6.1). The method starts with global exploration by building a graph using an asymptotically optimal motion planner, such as k-nearest Probabilistic Roadmap (PRM*) [93], until it finds a collision-free path. It then uses our lazy interior point optimization formulation to refine the path found by the sampling-based method. The algorithm then iterates between (1) resuming the sampling-based motion planner until a new better path is found and (2) running interior point optimization on this new path. The sampling-based motion planning phase of each iteration explores globally, discovering other homotopy classes in configuration space, as well as escaping local minima in the path cost landscape. The interior point optimization phase in turn allows the method to provide a high quality locally-optimized motion plan based on the best path found by the sampling-based method. Either phase can be interrupted at any time and still return a valid, collision-free result. Interleaving these phases provides higher quality motion plans earlier than asymptotically optimal sampling-based motion planning algorithms alone. In this way, our method preserves guarantees which are not provided by most optimization-based motion planning algorithms—namely completeness and asymptotic optimality—all in an anytime fashion and designed specifically to work efficiently on point cloud data.

ISIMP is based on interior point optimization, a type of local optimization that has properties critical to achieving fast, anytime motion planning for robotic manipulators. Interior point methods solve optimization problems by only considering feasible points in each iteration of the method. These methods typically incorporate constraints into the objective function by adding barrier functions, continuous functions whose value on a point increases to infinity as the point approaches the boundary of the feasible region. These methods adjust weights on the barrier functions over time so the solution converges to a locally optimal solution of the original objective function. We apply interior point optimization to refining motion plans and show that it is fast (like other motion plan optimizers, e.g., [95, 96, 97, 98, 101, 102]) while providing three additional important properties. First, our interior point optimization implementation is an anytime algorithm: each intermediate iteration of the optimization algorithm considers only collision-free robot configurations, so the optimizer always returns a feasible motion plan even when interrupted. Second, interior point optimization



(a) Sampled 1st Path (b) Optimized 1st Path (c) Sampled 2nd Path

(d) Optimized 2nd Path (e) Sampled Baxter Path (f) Optimized Baxter Path

Figure 6.1: ISIMP interleaves sampling-based and optimization-based motion planning. (a) ISIMP first performs sampling-based motion planning until a feasible motion plan is found from start to goal. (b) It then uses interior point optimization to locally optimize the plan. (c) Sampling-based motion planning resumes until a shorter plan is found. (d) The shorter plan is then optimized. The method iterates in this fashion and can be interrupted at any time after step (a) and return a feasible solution that gets better over time. (e) An example sampling-based plan found for a Baxter robot in a point cloud sensed by Microsoft Kinect. (f) That plan locally optimized.

can be formulated to efficiently and directly handle large point clouds, which we accomplish using a lazy evaluation of constraints. This eliminates the time-consuming process of transforming the point cloud into more complex geometric primitives or meshes, which is often required by other optimization-based motion planners to be efficient. Third, our interior point optimization is designed to optimize path length, a commonly desired metric which is often used by sampling-based motion planners. Many existing optimization-based motion planners [95, 96, 97, 98] are designed to optimize metrics such as smoothness, which do not satisfy the triangle inequality and hence are incompatible with cost requirements of asymptotically-optimal sampling-based motion planners.

We evaluate the efficacy of ISIMP in two settings. In the first, we apply ISIMP to plan motions for a spherical robot moving in 3D in a point cloud environment. In the second, we apply ISIMP to plan motions for a Baxter robot's 7 degree of freedom manipulator arm using a Microsoft Kinect to sense point cloud data in a cluttered environment. We demonstrate ISIMP's fast, anytime, and asymptotically optimal performance in comparison to other motion planners that only use either

sampling or local optimization.

This chapter is adapted from work originally published in [14], as well as a journal paper that is currently in preparation.

6.1 Related Work

In sampling-based motion planning, a graph data structure is constructed incrementally via random sampling providing a collision-free tree or roadmap in the robot's configuration space. These algorithms provide probabilistic completeness, i.e., the probability of finding a path, if one exists, approaches one as the number of samples approaches infinity. Examples include the Rapidly-exploring Random Tree (RRT) [103] and the Probabilistic Roadmap (PRM) [101] methods. These have been adapted in many ways, such as taking advantage of structure in the robot's configuration space or modifying the sampling strategy [104, 105].

Adaptations of these algorithms can provide asymptotic optimality guarantees in which the path cost (e.g., path length) will approach the global optimum as the number of algorithm iterations increases. Examples include RRT* and PRM* [93] where the underlying motion planning graph is either rewired or has asymptotically changing connection strategies. Other asymptotically optimal algorithms grow a tree in cost-to-arrive space [94], identify vertices likely to be a part of an optimal path [106], or investigate the distributions from which samples and trajectories are taken [107]. Lazy collision checking has been shown to substantially improve the speed of these algorithms [108], and in some cases, near optimality can be achieved while improving speed [109, 110].

Optimization-based motion planning algorithms perform numerical optimization in a high dimensional trajectory space. Each trajectory is typically encoded as a vector of parameters representing a sequence of robot configurations or controls. A cost can be computed for each trajectory, and the motion planner's goal is to compute a trajectory that minimizes cost. In the presence of obstacles and other constraints, the problem can be formulated and solved as a numerical optimization problem. Examples include taking an initial trajectory and performing gradient descent [95], using sequential quadratic programming with inequality constraints to locally optimize trajectories [96], and combining optimization with re-planning to account for dynamic obstacles [97]. These methods typically produce high quality paths but are frequently unable to escape local minima, and as such are subject to initialization concerns. To avoid local minima, some

methods inject randomness [98, 111]. The solutions of these existing optimization-based motion planners may converge to locally optimal motion plans that include robot configurations that are in collision with obstacles. This limitation can be partially circumvented through techniques such as restarting the optimization with multiple different initial paths (e.g., [96]), but such approaches are heuristic and provide no guarantee that a collision-free trajectory will be found in general.

Several methods aim to bridge the gap between sampling-based methods and optimization. Some methods use paths generated by global planners and refine them using shortcutting or smoothing methods adaptively or in post processing [102, 112]. GradienT-RRT moves vertices to lower cost regions using gradient descent during the construction of an RRT [113]. More recently, local optimization has been used on in-collision edges during sampling-based planning to bring them out of collision and effectively find narrow passages [114]. In contrast, our method is using the local optimizer not to find narrow passages, but to improve the overall quality of the paths found by the sampling-based planner, while relying on the sampling-based planner's completeness property to discover narrow passages. BiRRT-Opt [115] utilizes a bi-directional RRT to generate an initial trajectory for trajectory optimization, demonstrating the efficacy of a collision-free initial solution for local optimization. Our method differs in that interior point optimization provides collision-free iterates allowing it to work in an anytime fashion, and our method continues beyond a single local optimization to provide global asymptotic optimality.

6.2 Problem Definition

Let C be the configuration space of the robot. Let $\mathbf{q} \in C$ represent a single robot configuration of dimensionality d and $\sigma = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{n-1}\}$ represent a continuous path in configuration space in a piecewise linear manner by a sequence of n configurations. Such paths may need to satisfy generalized, user-defined inequality constraints. These constraints could include joint limits, end effector orientation requirements, etc, where each constraint may be represented by the inequality $\mathbf{g}(\sigma) \geq 0$ for some constraint specific function g. Let the set of all such user-defined constraints be J.

In the robot's workspace are obstacles that must be avoided, and which are being represented by a point cloud. We consider each point in the point cloud an obstacle and let the set of all such points be O. We also define the robot's geometry as a set of geometric primitives S. An individual geometric primitive $s \in S$ could be represented as a mesh, bounding sphere, capsule, etc.

A path is collision-free if the robot's geometry S over each edge $(\mathbf{q}_i, \mathbf{q}_{i+1}), i = 0, \ldots, n-2$, does

not intersect an obstacle $o \in O$. Formally, we require a function $\texttt{clearance}(\mathbf{q}_i, \mathbf{q}_{i+1}, s, o)$ which is a function parameterized by a path edge $(\mathbf{q}_i, \mathbf{q}_{i+1})$, a geometric primitive s, and an obstacle o. This function, defined in Sec. 6.3, is continuous and monotonically increasing, has positive value when o is not in collision with s over the edge, negative value when o penetrates into s on the edge, and 0 at the boundary. A collision-free path is then one for which each edge has non-negative clearance for all $o \in O$ and $s \in S$. This requirement can be represented as an additional set of inequality constraints, which we define as K, wherein $\texttt{clearance}(\mathbf{q}_i, \mathbf{q}_{i+1}, s, o) \geq 0$, for all $o \in O$ and $s \in S$.

Our objective is to find a collision-free path for the robot from the robot's start configuration $\mathbf{q}_{\text{start}}$ to a user specified goal configuration $\mathbf{q}_{\text{goal}}$ that satisfies all the constraints and minimizes path length. We define path length by $\texttt{length}(\sigma)$, the sum of Euclidean distances in configuration space along the path. We use the sum of Euclidean distances because it is a commonly used metric for path length and because it satisfies the triangle inequality property required by asymptotically optimal sampling-based motion planners (unlike other metrics such as sum of squared distances). This optimal motion planning problem can be formulated as a nonlinear, constrained optimization problem:

$$
\begin{aligned}
&\sigma^* = \underset{\sigma}{\operatorname{argmin}}\ \texttt{length}(\sigma) \\
&\text{Subject to:} \\
&\quad \texttt{clearance}(\mathbf{q}_i, \mathbf{q}_{i+1}, s, o) \geq 0, \quad 0 \leq i < n-1, \forall\, o \in O, \forall\, s \in S \\
&\quad \mathbf{g}(\sigma) \geq 0, \qquad\qquad\qquad\qquad \forall\, \mathbf{g} \in J \\
&\quad \mathbf{q}_0 = \mathbf{q}_{\text{start}} \\
&\quad \mathbf{q}_{n-1} = \mathbf{q}_{\text{goal}}
\end{aligned}
\tag{6.1}
$$

where σ^* is an optimal motion plan. To solve this optimization problem, we present an efficient, anytime, iterative algorithm in which the solution asymptotically approaches σ^*.

6.3 Method

ISIMP integrates ideas from both sampling-based and optimization-based motion planning. The method interleaves interior point optimization steps with sampling-based motion planning steps to achieve fast convergence. The top level algorithm, Alg. 3, runs in an anytime manner, iterating as time allows and storing the best path found as it runs.

The first step of each iteration is the global exploration step, wherein ISIMP executes a sampling-

Algorithm 3: ISIMP

Input: $\mathbf{q}_{\text{start}}$, $\mathbf{q}_{\text{goal}}$, O, optimization convergence threshold Δ_σ
Output: motion plan σ^*

```
Sampling-based motion planner path cost c_samp ← ∞
σ* ← ∅
while time remaining do
    σ_samp ← GlobalExplorationStep(O, c_samp, q_start, q_goal)
    c_samp ← cost(σ_samp)
    σ* ← minCostPlan(σ_samp, σ*)
    σ* ← LazyAnytimeInteriorPointOptimization(σ_samp, O, Δ_σ, σ*)
end
```

based motion planner until a new path is found that has cost lower than c_{samp}, the best cost found by the sampling-based motion planner up until that point. The sampling-based motion planner returns the new path σ and updates c_{samp}. In the second step of each iteration, the interior point optimization step, our method executes an interior point local optimization method to locally optimize σ. The local optimizer constantly compares against the best path found, and updates it if a better path is found during the optimization. At the end of the interior point optimization it returns the best path found. The algorithm then iterates, returning to global exploration with the sampling-based motion planner, and optimizing better paths as they are found. If the algorithm is interrupted, it returns the best path found up until that time.

6.3.1 Global Exploration using Sampling-Based Motion Planning

The global exploration step uses a sampling-based motion planner to expand a graph until a new path is found that is of lower cost than any path it has previously found. The sampling-based motion planner maintains a graph $G = (V, E)$, where V is a set of vertices which represent collision-free configurations of the robot and E is a set of edges, where an edge represents a collision-free transition between two robot configurations.

To expand the graph G, our method is designed to use an asymptotically optimal sampling-based motion planner. Although many methods would work, we implement ISIMP with both k-nearest Probabilistic Roadmap (PRM*) [93] and RRT$^{\#}$ [106]. PRM* samples random configurations in the robot's configuration space, locates their k nearest neighbors (where k changes as a function of the number of vertices in the graph), and attempts to connect the configurations to each of its neighbors. RRT$^{\#}$ is a state-of-the-art asymptotically optimal sampling-based motion planner, which

incrementally builds a tree identifying which vertices are likely to belong to an optimal path. In each global exploration step, the asymptotically optimal sampling based motion planner executes until it finds a collision-free path better than the current best, at which point the optimization step begins.

6.3.2 Lazy Interior Point Optimization

Local optimization for motion planning is typically performed by constructing a high dimensional vector out of the state variables of the problem to be optimized. In our case, this would be a vector representing the motion plan we are optimizing, i.e. if we have a path with n configurations of dimensionality d, then each motion plan can be represented as an $n \times d$ dimensional vector. Optimization can then be viewed as iteratively moving this vector through its high dimensional space to minimize the cost. In the case of constrained optimization, there are regions of this space which represent areas where the constraints are satisfied (i.e., feasible or unconstrained space) and areas where the constraints are not satisfied (i.e., infeasible or constrained space).

Many other classes of optimization methods (such as penalty methods and sequential quadratic programming) allow their intermediate solutions to move through infeasible space, i.e. collide with obstacles or violate joint limits, with the hope that the eventual locally optimal solution is feasible. Interior point methods, by contrast, require their intermediate solutions to always be feasible [99, 100]. This is typically accomplished through the use of barrier functions. A barrier function works by introducing a continuous function whose value approaches infinity at the edge of the constrained space. The intermediate iterations of optimization are then influenced by the barrier functions to avoid the constrained regions. As the optimization iterates, the width of these barrier functions is reduced such that the solution is allowed to approach the constrained space but not to enter it. This is the property that we leverage to create an anytime solution inside the optimization framework. Each intermediate solution of the optimization is always a collision-free motion plan and as such can be returned early if necessary.

We build an interior point optimization framework around a black box interior point optimizer. The optimizer is responsible for generating intermediate, feasible solutions, and our framework updates the state for the optimizer as a function of those intermediate solutions to allow for faster computation. Details of our lazy interior point optimization framework can be seen in Alg. 4.

We optimize our paths with respect to path length. The collision avoidance constraints are formulated over each *edge* in the path, i.e. the linear interpolation through configuration space

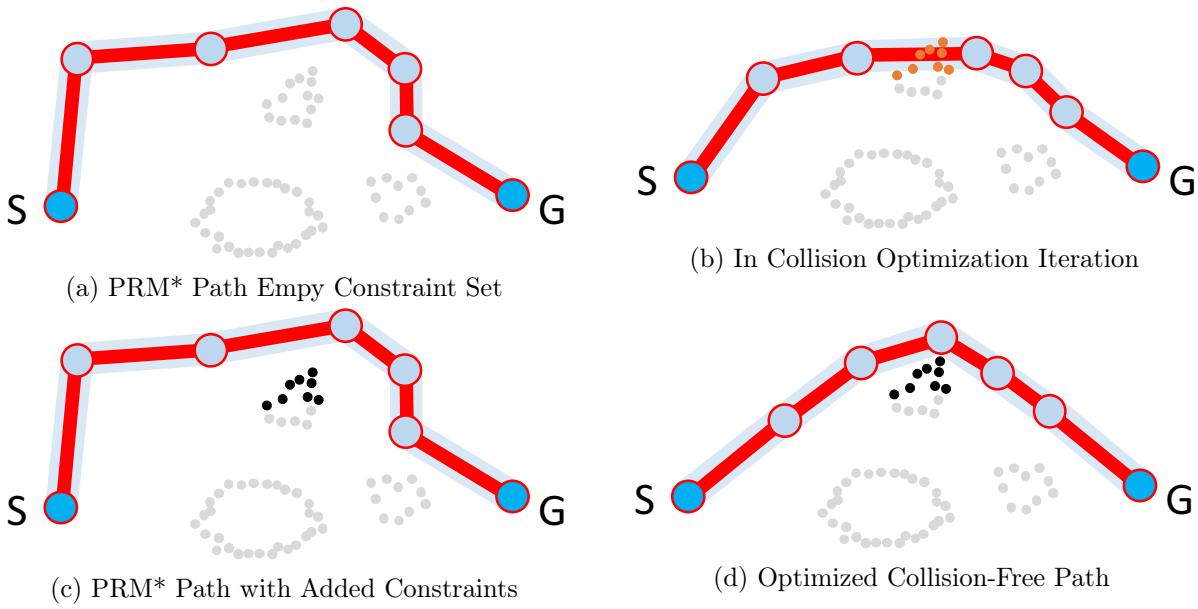
Algorithm 4: Lazy Anytime Interior Point Path Optimization

Input: initial path σ_{init}, obstacle set O, convergence threshold Δ_σ, best found path so far σ^*
Output: best path found σ^*

```
K_enab ← ∅, σ̂ ← σ_init, σ ← σ_init
while time remaining do
    σ_next ← takeInteriorPointOptimizationStep(K_enab, σ)
    O_collide ← inCollisionPoints(σ_next, O)
    if O_collide ≠ ∅ then
        K_enab ← K_enab ⋃ O_collide
        σ ← σ_init
    else
        if discontinuousConstraintDetected(K_enab, σ_next) then
            {k, k+1} = discontinuousJacobianEdge(K_enab, σ_next)
            σ_init ← bisectEdge(σ_init, k, k+1)
            σ ← σ_init
        else
            if converged(σ_next, σ̂, Δ_π) then
                return minCostPlan(σ_next, σ̂, σ*)
            end
            σ̂ ← minCostPlan(σ_next, σ̂)
            σ* ← minCostPlan(σ̂, σ*)
            σ ← σ_next
        end
    end
end
return σ*
```

between the two configurations. We define our clearance function (as in equation (6.1)) as a function of the minimum distance between the point in the point cloud and the geometric primitive interpolated through the workspace as determined by the robot's forward kinematics. This formulation generates a large number of constraints (# of robot geometric primitives × # of edges in the path × # of points in the point cloud). Because the optimization must consider each constraint, fewer constraints results in quicker optimization times. In the following paragraphs, we discuss our novel approach to reducing the number of constraints that are considered by the optimization by adding constraints in a lazy fashion.

Improving Performance through Lazy Constraint Addition Instead of using the entire set of obstacle-based inequality constraints K, which contains constraints for each point in the point cloud, we instead define a subset K_{enab} as the *enabled* constraint set. This set, initially empty,



(a) PRM* Path Empy Constraint Set

(b) In Collision Optimization Iteration

(c) PRM* Path with Added Constraints

(d) Optimized Collision-Free Path

Figure 6.2: An example of the way our method adds constraints from the point cloud in a lazy fashion, using a 2D disk robot in a point cloud (grey points are un-enabled constraint points). (a) Example unoptimized path returned by the PRM* and used to initialize the optimizer. The optimizer starts out with an empty enabled constraint set. (b) During optimization, the robot's path is found to be in collision with the point cloud (orange points). (c) The in-collision points of the cloud are immediately added to the enabled constraint set (black points) and optimization is restarted. (d) The optimizer converges to a collision-free path using the enabled constraint set, which is smaller than the set of all points.

is added to as points in the cloud become relevant to the optimization (see Fig. 6.2 and Alg. 4 lines 5-7).

We start the optimization with an empty enabled constraint set (Fig. 6.2a). We then take an optimization step, modifying the motion plan (Fig. 6.2b). At each iteration of such a step, we check the robot's path for collision with the *whole* point cloud (an operation which is computationally inexpensive compared with optimizing with each point as a constraint). If the path is found to be in collision, we identify which points are in collision, and add those points to the enabled constraint set, and restart the optimization (Fig. 6.2c). In the next iteration, the optimizer avoids collision with the previously added points (Fig. 6.2d). The process then repeats until a collision-free convergence is achieved. In this way, only points which prove to be relevant over the course of the optimization are taken into account by the optimizer. This improves the computational speed of the optimization algorithm drastically, as in practice relatively few points turn out to be relevant to collision detection in an otherwise large point cloud.

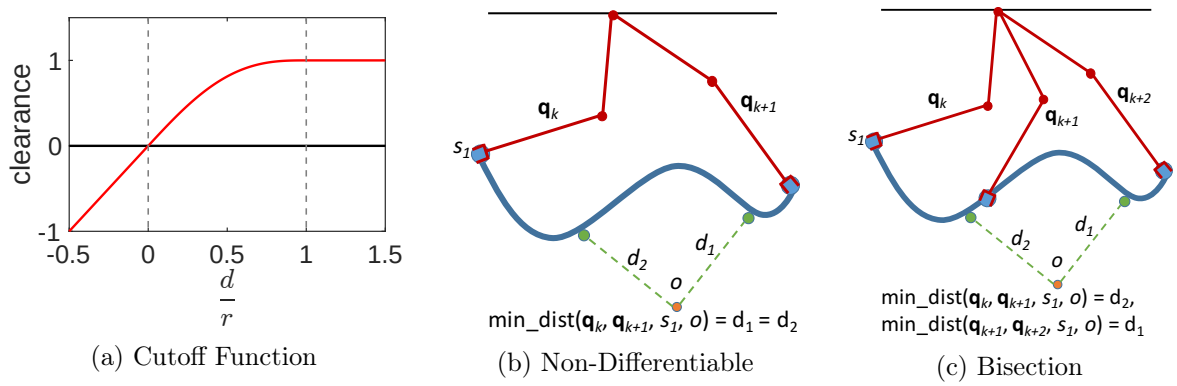
Enabled Constraint Cutoff Function One of the most time consuming parts of the optimization is computing the constraint Jacobian for the constraints in K_{enab}, which has a row for each constraint and a column for each state variable, i.e. $d \times (n-1)$ columns. Even if a constraint is relevant to the optimization as a whole, and as such included in K_{enab}, sometimes the point represented by the constraint is far away from the specific robot geometric primitive or edge representing an entry in the Jacobian. To leverage this intuition and further speed up the evaluation of the constraints in K_{enab}, we use a cutoff function (6.3) as our clearance function (see Fig. 6.3a) rather than the distance itself, allowing the user to set a radius, $r > 0$, as a "look-ahead" distance. Let

$$d = \texttt{min\_dist}(s, \mathbf{q}_i, \mathbf{q}_{i+1}, o), \tag{6.2}$$

$$\texttt{clearance}(d, r) = \begin{cases} 2\left(\frac{d}{r}\right) & \left(\frac{d}{r}\right) < 0 \\ \left(\frac{d}{r}\right)^4 - 2\left(\frac{d}{r}\right)^3 + 2\left(\frac{d}{r}\right) & 0 \leq \left(\frac{d}{r}\right) \leq 1 \\ 1 & \left(\frac{d}{r}\right) > 1, \end{cases} \tag{6.3}$$

where `min_dist` is the minimum distance between the point o and the geometric primitive s when s is swept along the space curve defined by the transition between configurations $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$, i.e. as the robot arm is moving from configuration $\mathbf{q}_i$ to $\mathbf{q}_{i+1}$. Equation 6.3 sends the clearance function's derivative to zero at r, allowing us to sparsify the constraint Jacobian and only consider points in K_{enab} which are within distance r from the robot's geometry at a given time. This allows even greater speedup to the optimization, as the constraint set K_{enab}, which is already greatly reduced in size from the full point cloud, is still larger than needs to be considered in many entries of the constraint Jacobian.

Another challenge of our constraint formulation is that the minimum clearance between a given robot geometric primitive along an edge and a point in the point cloud is not guaranteed to be differentiable. This is because the robot's arm is tracing a nonlinear, complex curve through the workspace, even though the edge is linear in configuration space. This means that in some places, the closest position on that curve to an obstacle point can be discontinuous as the edge is perturbed (see Fig. 6.3b), i.e., there are multiple closest positions on the curve which are equidistant to the point cloud point. Using distance over the nonlinear edge allows us to ensure obstacle avoidance,



(a) Cutoff Function (b) Non-Differentiable (c) Bisection

Figure 6.3: (a) The cutoff function used as the clearance function for our obstacle avoidance constraints. The function flattens out after a certain distance so as to allow us to only consider points in K_{enab} which are within a specified radius r. (b-c) An example with a manipulator (shown in red). (b) If a sphere bounding the robot geometry is swept between two configurations and produces a curve such as the blue curve here, then when d_1 and d_2 are equal, the minimum distance between sphere s_1 swept from $\mathbf{q}_k$ to $\mathbf{q}_{k+1}$ and point cloud point o is not necessarily differentiable. (c) When such a case is detected, an intermediate configuration is added into the path with its own constraint. In this way, both d_1 and d_2 are used in different constraints and are both locally differentiable.

but can cause numerical problems during the optimization. To address this issue, when we are evaluating the derivative of a constraint and find it to be discontinuous over an edge, we bisect that edge and start the optimization anew (see Fig. 6.3c, and Alg. 4 lines 9-12). In this way, after enough bisections the discontinuity is avoided. In practice, we observed relatively few bisections.

6.3.3 Asymptotic Optimality

Asymptotic optimality of our method follows naturally from the use of an asymptotically optimal motion planner as the underlying sampling-based motion planner, as long as three conditions are met. First, the sampling-based motion planner adds at least one node to its roadmap between adjacent interior point optimization calls. Second, the interior point optimization does not make the roadmap's solution worse. Third, the interior point optimization completes in finite time.

The first condition is dependent on the implementation of the sampling-based motion planner. The implementations we use guarantee this property. The second condition can be handled by discarding any solutions which are worse than the best that has been found at any point in time. The third condition can be guaranteed with an external time limiter on the optimization.

6.4 Results

We implement the sampling-based motion planning portion of our method using the OMPL framework [116], including OMPL's PRM* and RRT$^{\#}$ implementations. For the purposes of this

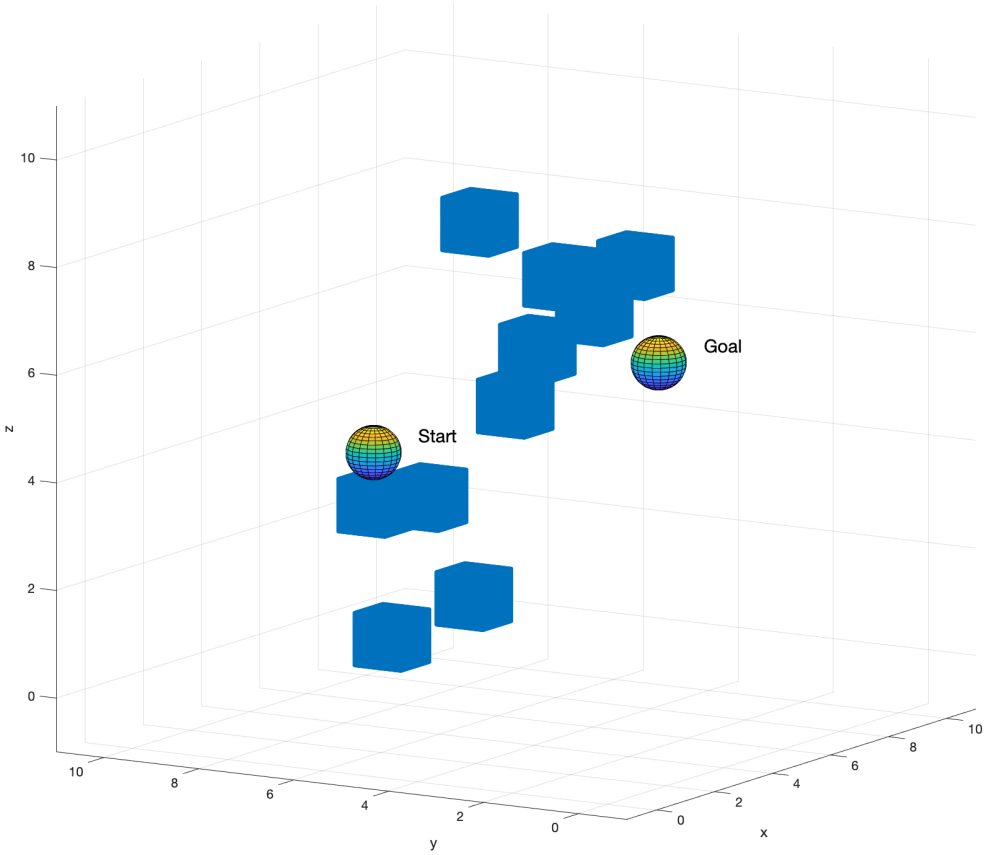
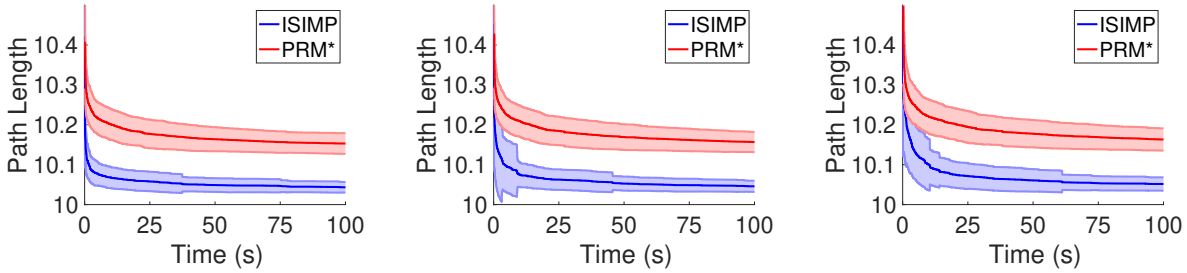


Figure 6.4: Randomly generated 3D environment. The environment contains points on the surface of 10 boxes.



(a) 15,000 points in point cloud (b) 37,500 points in point cloud (c) 150,000 points in point cloud

Figure 6.5: Average path length over time for the 3D point cloud scenario with (a) 15,000 points, (b) 37,500 points, and (c) 150,000 points in the point cloud. Results are averaged over 100 runs with the shaded region denoting 1 standard deviation.

section, we will refer to ISIMP with PRM* as ISIMP, and ISIMP with RRT$^{\#}$ as ISIMP-#. For the black box local optimizer, we use the IPOPT software framework, an open source interior point optimization library[1] [100].

We evaluate ISIMP in two scenarios. In the first we consider a spherical robot moving in a 3D point cloud environment (see Fig. 6.4). In the second, we plan motions for the 7-DOF left arm of a Baxter robot (see Fig. 6.8) in a point cloud environment generated by a real-world sensor. All results were generated on an 3.40GHz Intel Xeon E5-1680 CPU with 64GB of RAM.

6.4.1 Motion Planning for a 3D Spherical Robot

We consider a spherical robot of radius 0.5 units with 3 degrees of freedom (x, y, and z) moving in a 3D environment. In the environment we have randomly placed 10 cubes with sides of length 1 unit. We represent the cubes using a point cloud of 3D points densely placed on the surfaces of the cubes (see Fig. 6.4). We then task ISIMP with planning a motion for the spherical robot from one side of the environment to the other. We evaluate this task with three separate point cloud sizes. The first involves the full point cloud, which contains $150,000$ points. We downsample this point cloud, using $37,500$ points for the second and $10,000$ for the third.

We generate results for each of these three environments using 100 different random seeds. The average path length over time for each of the three point cloud sizes is shown in Fig. 6.5. We compare ISIMP with PRM*, plotting the shortest path length found at any given time for each of the methods averaged over the 100 runs, with 1 standard deviation shown in the shaded regions. As can be seen, ISIMP outperforms PRM*, on average finding shorter paths earlier in the planning process.

We also evaluate the efficacy of the lazy constraint addition aspect of ISIMP. For each of the 3 point cloud sizes, we keep track over all 100 runs of the total number of points that were added to the constraint set at the convergence of each interior point path optimization loop (i.e. execution of Alg. 4). These results are shown in Table 6.1. The results show that it is necessary to add as constraints only a very small percentage ($\approx 1.2\%$) of the overall number of points in the point cloud

[1] IPOPT requires definitions of a cost function to be minimized (path length in our case), constraint functions (J and K or K_{enab}), and cost and constraint Jacobians. It then handles the optimization iterations itself. Note that default settings will allow intermediate solutions to potentially exist in infeasible space. We set the settings to prevent infeasible intermediate solutions.

Total Points	Average Points Used	Percentage	Instances
$15,000$	188.67 ± 160.69	1.26%	1323
$37,500$	438.21 ± 381.58	1.17%	1357
$150,000$	$1,755.05 \pm 1,710.01$	1.17%	1319

Table 6.1: Statistics on point cloud points added to the constraint set by executions of the lazy interior point optimization algorithm

in order for the interior point optimization to converge to a high-quality solution that is collision-free with respect to the entire point cloud. For example, in the $150,000$ point environment, averaged over 1319 instances of converged interior point optimization, only 1.17% of points were used by the optimization.

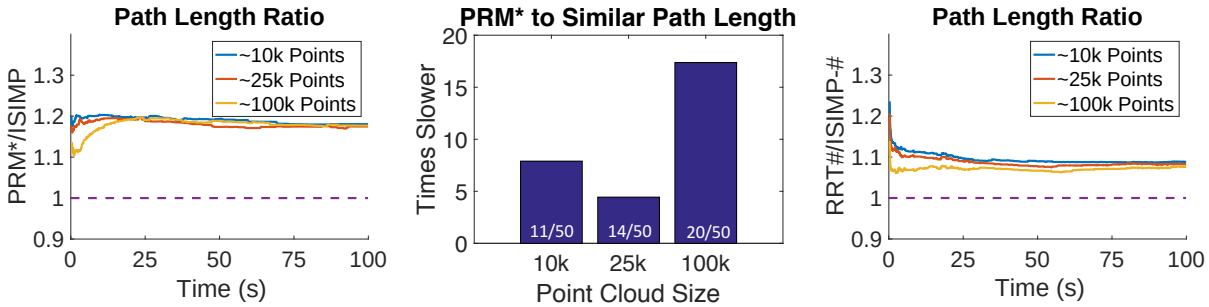
6.4.2 Motion Planning for the Arm of a Baxter Robot

We consider a scenario involving motion planning for the left arm of a Baxter robot (see Fig. 6.8). For the purposes of computing constraints in the optimization and obstacle avoidance in the sampling-based motion planner, we represent the geometry of the robotic manipulator arm as a set of 9 bounding spheres along the links of the robot's arm. The choice of bounding spheres allows us to conservatively represent the robot's geometry while enabling fast distance calculations between the geometric primitives and the points in the point cloud.

We evaluate ISIMP's performance in a real-world point cloud obtained from a Microsoft Kinect sensor (see Fig 6.8). We evaluate our method utilizing point clouds of differing sizes, $\approx 10,000$ points, $\approx 25,000$ points, and $\approx 100,000$ points, the smaller of which were generated via downsampling the original point cloud. We generated 50 motion planning scenarios at random in the scene using random start and goal configurations, using rejection sampling to remove trivial scenarios for which a straight-line naive trajectory would not collide with the point cloud.

We compare our method to the popular anytime asymptotically-optimal sampling based motion planners PRM* [93] and RRT$^{\#}$ [106] using OMPL [116], and to a popular trajectory optimization method, Traj-Opt [96], using their distributed source code. All results were generated on an 3.40GHz Intel Xeon E5-1680 CPU with 64GB of RAM.

Comparison to Asymptotically-Optimal Sampling-Based Motion Planners To evaluate how well ISIMP performs in an anytime manner, we compare our method to PRM* and RRT$^{\#}$ with the three point cloud sizes. In Fig. 6.6a, we compare the best path found by ISIMP, up until a given



(a) PRM* Ratio over time (b) Time to length (c) RRT# Ratio over time

Figure 6.6: (a) The path length ratio over time between PRM* and ISIMP, averaged over 50 runs with random start and goal configurations. Results are shown for various sizes of point clouds. A value greater than one indicates ISIMP has a shorter path than PRM*. (b) Another way to visualize anytime results. ISIMP is stopped after 1 second, and displayed is how many times longer PRM* ran to achieve a comparable result. In many cases, PRM* had still not found a comparable result after 5 minutes, in which case we did not include that run in this analysis. The included runs were averaged. The number of included runs is shown in white overlaid on each bar. (c) The path length ratio over time between RRT# and ISIMP-#.

time, to the best path found by PRM* at that same time. We average this over 50 runs, and plot the ratio of the PRM*'s path length to our method's path length. A value greater than one indicates that our method has a shorter path, and a value less than one indicates that PRM* has a shorter path. Similarly, in Fig. 6.6c, we compare ISIMP-# with RRT#.

As another way of evaluating anytime performance, we stop ISIMP's execution at 1 second and then measure how long PRM* requires to produce a solution of equal or better cost. In the majority of cases, PRM* in 5 minutes failed to produce a path of shorter length than ISIMP's solution at 1 second. For the cases where PRM* did produce a better solution in less than 5 minutes, we average the results and show them in Fig. 6.6b.

As can be seen, ISIMP outperforms PRM* on path length by a large percentage for a long duration (Fig. 6.6a), and ISIMP-# outperforms RRT# similarly, but by a smaller percentage (Fig. 6.6c).

Comparison to an Optimization-Based Motion Planner We also compare ISIMP's performance to a state-of-the-art optimization-based motion planner, Traj-Opt [96], for which open-source code compatible with the Baxter robot is available. The Traj-Opt distribution recommends several ways to handle point cloud data input, including: (1) converting the points in the point cloud into down-sampled cubic boxes, and (2) constructing a polygonal mesh out of the point cloud, and then

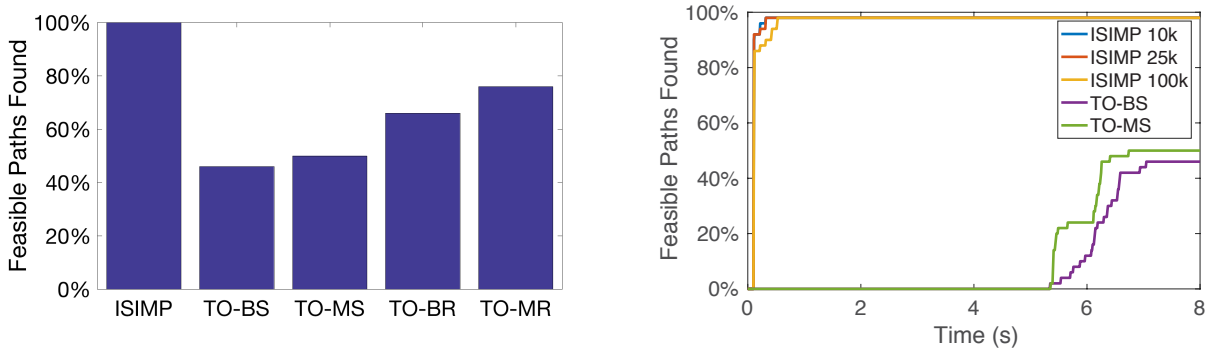


Figure 6.7: (a) The percent of trials for which a feasible path has been found over 50 trials. (b) The percent of trials for which feasible, collision-free, paths have been found over time. The ISIMP variations find feasible paths for 98% of the trials within 0.5 seconds (and 100% of trials within 30 seconds). The Traj-Opt variations take significantly longer ($\approx$ 5.5–7 seconds) to find feasible paths for some trials, and fail to find feasible paths at all for the others. (TO-BR and TO-MR are not shown in (b) because they are seeded with already feasible solutions from a sampling-based motion planner.)

simplifying the mesh through decimation.

As with most trajectory optimization methods, the question arises of how to generate an initial trajectory. We evaluate Traj-Opt using two initialization approaches recommended in the optimization-based motion planning literature. First, we initialize with a straight line trajectory in configuration space with 10 configurations (the number of configurations used for initialization in the example code), which starts in collision. Second, we seed Traj-Opt with the first path found by our method from the sampling-based motion planner, which is augmented with duplicate configurations if it starts with fewer than 10 total. This initialization starts out collision-free. TO-BS refers to Traj-Opt with a box environment representation and a straight line initialization, TO-MS refers to a mesh environment with a straight line initialization, TO-BR refers to a box environment with a sampling-based motion planner initialization, and TO-MR refers to a mesh environment with a sampling-based motion planner initialization.

The comparison between our method and Traj-Opt reveals three things. First, Traj-Opt can fail to converge to a collision-free path in a significant percentage of cases in our scenario. See Fig. 6.8 for an example. Traj-Opt returned locally optimal paths that were infeasible likely due to the complex objective function landscape with many in-collision local minima induced by the real-world point cloud data. Fig 6.7a shows the percentage of problems for which each method

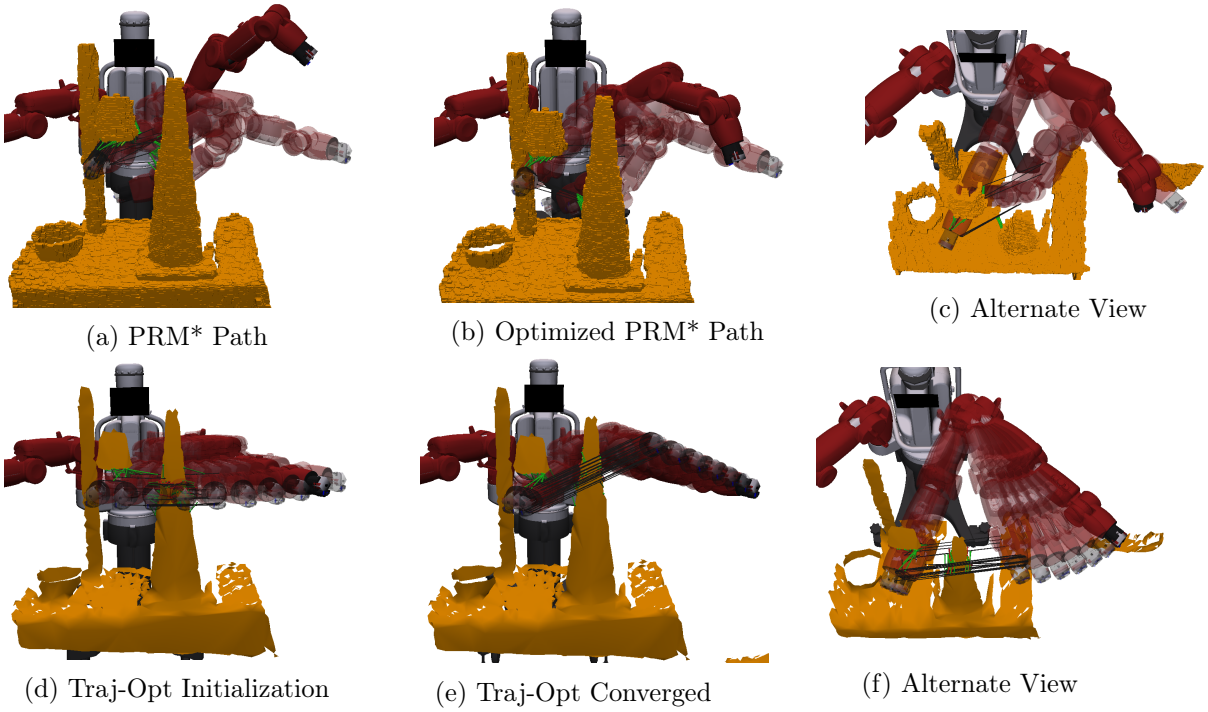


Figure 6.8: (a) ISIMP's collision-free initialization found by the sampling-based motion planning algorithm. (b-c) ISIMP's collision-free trajectory after interior point optimization, which is still collision-free and passes behind the obstacles. The path length has been reduced by more than 36%. (d) Traj-Opt is initialized with a straight line trajectory, passing through the point cloud obstacles. (e-f) Traj-Opt's converged trajectory which is still in collision due to being trapped in a poor quality local minimum.

found a feasible solution. Our method found feasible solutions to 100% of planning problems within 30 seconds, while variations on Traj-Opt returned locally optimal paths that were infeasible in a significant percentage of planning problems. This is even true in the case where the initialization was collision-free (TO-BR and TO-MR).

Second, Traj-Opt, when it did produce a valid collision-free solution, takes between 5.3 and 7.1 seconds to initialize the problem, load the scene, and optimize the trajectory. By contrast, our method initializes the problem, loads the scene, and produces its first valid collision-free solution in an average of 0.41s for 10k points, 0.43s for 25k, and 0.63s for 100k (See Fig. 6.7b). Traj-Opt's slower relative speed is mostly due to the time required to initialize the problem and pre-process the point cloud data, with Traj-Opt's optimization itself taking on average less than 1 second.

Third, when Traj-Opt converged to a collision-free solution it produced high quality paths. For the cases where Traj-Opt produced a collision-free solution, the path lengths of its solutions were

comparable to the path lengths produced by ISIMP, with path lengths varying by less than 5% across the methods.

6.5 Conclusion

We presented a method designed to achieve the benefits of both local optimization and global sampling-based motion planning when planning motions for a robotic manipulator arm with point cloud sensor data. ISIMP interleaves asymptotically-optimal sampling-based motion planning with anytime interior point local optimization. Designed to work with point cloud data directly, our novel lazy interior point optimization formulation brings the path quality benefits of local optimization to the anytime performance of sampling-based methods, while providing completeness and global asymptotic optimality guarantees not present in current optimization-based motion planning methods. The results demonstrate that our method outperforms asymptotically-optimal sampling-based motion planning alone, producing higher quality motion plans earlier. Our method also outperforms an optimization-based method, Traj-Opt, in the percentage of collision-free solutions found and in time to first valid solution.

In the future, we plan to evaluate ISIMP on a variety of other systems of varying dimensionality. We also plan to investigate integrating more complex constraints, such as task and non-holonomic constraints into the method.

In Chapter 7 we extend this method to plan motions for a concentric tube robot, using a cost metric based on clearance from obstacles, and integrate the optimization- and sampling-based methods into a parallel structure to do both simultaneously.

CHAPTER 7

Planning High-Quality Motions for Concentric Tube Robots in Point Clouds via Parallel Sampling and Optimization

Motion planning can enable surgical robots such as concentric tube robots [3] to automatically reach a desired surgical target while avoiding anatomical obstacles. Composed of nested, pre-curved tubes, concentric tube robots can curve around anatomical obstacles to reach targets in highly constrained environments such as the skull base, the lungs, and the heart [7]. Enabling the robot to safely avoid anatomical obstacles (such as blood vessels, critical nerves, sensitive organs, and bones) requires a fast and effective motion planner, as well as an accurate model of the patient anatomy. In previous work, such an anatomical model is typically generated from the segmentation of preoperative 3D volumetric imaging, such as Computed Tomography (CT) [117, 118, 8]. However, for surgical procedures that modify the anatomy, anatomical models created from preoperative images may quickly become out-of-date and inaccurate, significantly hindering safe motion planning. By contrast, a variety of intra-operative endoscopic sensors can be used to quickly produce point cloud representations of the anatomy. In this work, we introduce a new motion planning method, Parallel Sampling and Interior point optimization Motion Planning (PSIMP). PSIMP quickly produces high-quality motion plans for concentric tube robots operating in point cloud anatomical representations.

Point clouds that represent patient anatomy can be generated during minimally-invasive surgery in a variety of ways, including via small laser scanners [119], structured light sensors [120], and generated directly from endoscopic video (see Fig. 7.1) using computer vision techniques [121, 122, 123]. In contrast to CT imaging, such sensors and techniques can be repeatedly used intra-operatively, acquiring point clouds in seconds rather than minutes, and do not rely on ionizing radiation (i.e., x-rays), which may be harmful to patients and clinical staff when used repeatedly. Such techniques also have advantages over fluoro imaging due to the reduced radiation exposure and are much cheaper than MRI. As the anatomy changes, new point clouds can be generated and used by the motion

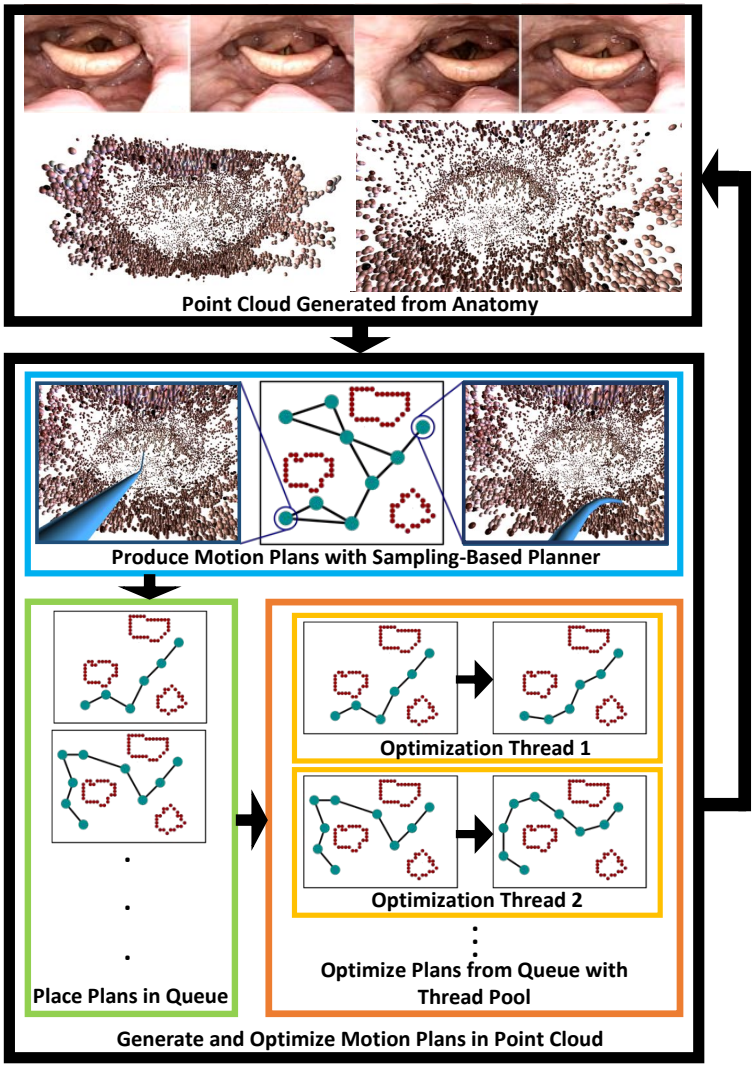


Figure 7.1: Our method, PSIMP, takes as input a point cloud representing patient anatomy (top). PSIMP generates and optimizes motion plans for the robot to move safely through the point cloud (bottom). A sampling-based motion planner runs constantly in its own thread (blue box), generating motion plans over time—represented as collision-free sequences of configurations (2D cartoon representations of the plans are included here for illustrative purposes). As the motion plans are generated they are placed in a queue (green box). A thread pool (orange box) then takes each of the motion plans, and optimization threads (yellow boxes) perform interior point local optimization on the plans, improving their quality according to a cost based on clearance from obstacles. If the anatomy has not changed significantly, multiple motion planning queries can be solved for the initial point cloud, in real time, allowing the physician to move the robot through the anatomy safely. If the anatomy changes significantly (e.g., due to the surgical procedure), a new point cloud can be generated to be used by the motion planner in subsequent queries.

planner. In this work, we evaluate our method using point clouds generated via endoscopic video using a technique called "structure from motion" [124, 125], but the method can be used directly with point clouds from any source.

To enable high-quality, fast motion planning for concentric tube robots when obstacles are represented using point clouds, PSIMP combines sampling-based motion planning with local optimization (see Fig. 7.1). This combines the benefits of sampling-based motion planning, such as the exploration of multiple homotopic classes (see Fig. 7.2), with the benefits of local optimization, namely the ability to produce high-quality plans very quickly. We introduce a parallel, multi-threaded framework, that combines a sampling-based motion planning thread with a pool of local optimization threads. As the sampling-based motion planner generates motion plans, those solutions are placed in a queue of motion plans that are then locally optimized by a pool of threads that are running the local optimization method in parallel. This allows the sampling-based motion planner to run uninterrupted, ensuring that it continues exploring globally, while the local optimizers are iteratively improving the quality of the motion plans more quickly than the sampling-based motion planner is capable of doing on its own.

PSIMP plans motions at rates suitable for interactive use, returning the first valid motion plan in a fraction of a second on average, and rapidly improving upon that initial solution in fractions more. Our local optimization uses interior point optimization, a class of optimization techniques that guarantee intermediate solutions satisfy constraints (e.g., obstacle avoidance and joint limits) during optimization [99]. This property enables the method to run in an *anytime* fashion, i.e., the method can be stopped at any time and the best solution found up until that point will be returned. Our method leverages a cost function, similar to that in [126], based on the robot's clearance from the point cloud that encourages the avoidance of anatomical obstacles and helps to produce motions robust to incomplete obstacle knowledge.

We demonstrate the efficacy of PSIMP in three scenarios in which obstacles are represented via point clouds: an upper airway environment (near the epiglottis), a colon environment, and a skull base environment. The upper airway and colon environments are generated from real endoscopic video of real patients, while the skull base environment is generated from synthetic data. We show that combining local optimization with sampling-based motion planning outperforms sampling-based motion planning by itself in each of the anatomical settings. We also demonstrate the ability of PSIMP to react to a change in the point cloud when the anatomy is modified during a surgical procedure, which is not feasible when obstacle representations are generated solely from pre-operative imaging such as CT scans.

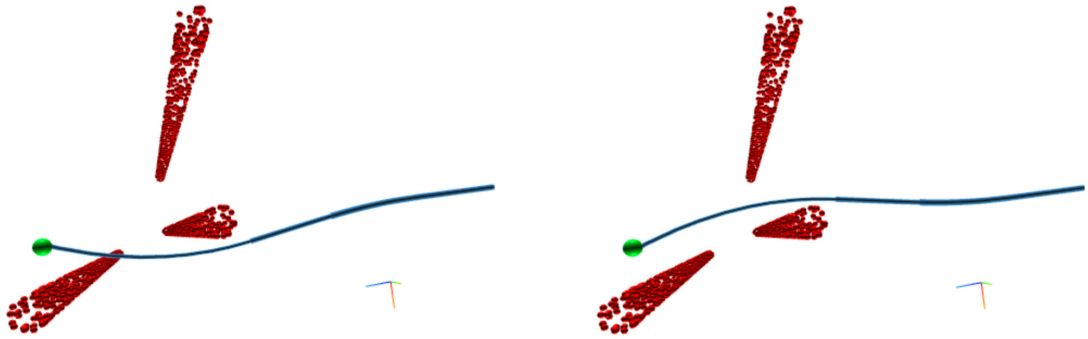


Figure 7.2: A top down view of an example scenario wherein the robot is tasked with passing between vertical columns represented as point clouds. The sampling-based motion planning phase of the planner allows for the discovery of different homotopic classes. (Left) The planner may initially find a solution to the goal point (green) for the robot (blue) that passes very close to the point cloud (red). (Right) The planner may later in the planning process find a better homotopic class in which the robot is able to reach the goal in a safer way, further from points in the point cloud.

This chapter is based on work that will appear in [15].

7.1 Related Work

Point clouds have been used in medical procedures in a variety of ways. Point clouds from stereoscopic cameras have been used for virtual fixtures in haptic interfaces [127] and for the registration of a digital overlay for teleoperation [128]. Soft tissue deformation has been tracked using 3D plenoptic imaging during autonomous suturing [129]. Point clouds generated by structure from motion in nasal endoscopy have been used for registering endoscopic images to CT data and overlaying areas of interests on the endoscope images [121]. We propose the use of point clouds as the anatomical representation during motion planning, enabling the obstacle representation to be regularly updated during a surgical procedure and hence enabling the motion planner to adapt to changes in the anatomy during surgery.

Concentric tube robots have been proposed for a variety of surgical tasks [7]. The control of concentric tube robots has primarily considered computing controls based on desired tip movements. This includes methods that compute controls based on the robot's Jacobian [9, 25] and a Fourier series based approximation of the robot's kinematics [24].

Motion planning can enable robots to automatically move in an environment while avoiding obstacles. A popular motion planning paradigm is sampling-based motion planning, which includes methods such as Probabilistic Roadmaps (PRM) [101] and Rapidly-exploring Random Trees (RRT)

[103], in which a collision-free graph or tree data structure is incrementally constructed. Many such algorithms provide a property called probabilistic completeness which states that the probability the algorithm finds a valid motion plan, if one exists, approaches 1 as the number of samples approaches infinity. Extensions to these methods have the stronger guarantee of asymptotic optimality, i.e., the method will converge to a globally optimal motion plan under some objective function as the number of samples approaches infinity. Such methods include RRT*, PRM* [93], BIT* [130], and FMT* [94].

Optimization-based motion planning methods work by numerically locally optimizing plans in a high dimensional trajectory space. Such methods include CHOMP [95], ITOMP [97], and Traj-Opt [96]. In Chapter 6 we presented ISIMP, a method that combines local optimization with sampling-based motion planning in point clouds for serial link manipulator arms [14]. The motion planner presented in this chapter, PSIMP, differs from these methods in a few ways. These methods optimize either for path smoothness [95, 97, 96] or for path length [14]. By contrast, here we introduce an anatomical clearance cost function that encourages motions that avoid anatomical obstacles by larger distances, increasing plan safety. This also allows us to simplify the constraint set by leveraging the new cost function for obstacle avoidance. Additionally, as in [14], PSIMP employs both sampling-based motion planning and interior point local optimization, but rather than interleaving the two, we utilize parallelism to perform both optimization and sampling simultaneously.

When computing motions for concentric tube robots that avoid obstacles, a few approaches have been studied. This includes simplifying the kinematics for fast computation [26, 131]. Sampling-based motion planning for concentric tube robots has been studied for skull base surgery. However, until recently, the previous methods either provided planning rates that were much slower than required for an intra-operative setting [132], or required preoperative imaging and extensive precomputation of a roadmap over the course of many hours prior to motion planning [118, 4]. The desire for a reactive anatomical representation precludes the use of methods that require extensive precomputation. Recently, a template-based, fast kinematic model was developed for concentric tube robots, based on an unloaded torsionally compliant kinematics model, in conjunction with a PRM-style motion planner which achieves much faster planning rates [133]. PSIMP uses this new kinematic model to enable fast shape computations during motion planning. PSIMP efficiently solves motion planning problems in environments represented by point clouds by leveraging local optimization to improve

upon PRM-style motion planning alone.

7.2 Problem Definition

We consider a point cloud P, where $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_j\}, \mathbf{p}_k \in \mathbb{R}^3$ for $k \in \{1, \ldots, j\}$, is an unordered set of j 3D points in the global coordinate frame lying on the surface of patient anatomy. In the anatomical environment represented by this point cloud, we consider a concentric tube robot. The concentric tube robot consists of N telescoping pre-curved tubes numbered in order of increasing cross-sectional radius, such that T_1 is the innermost tube, and T_N is the outermost tube. Each tube consists of a straight segment followed by a pre-curved segment reaching to its tip. We define the location of the robot in the global coordinate frame, i.e., the position from which the tubes extend, as $\mathbf{x}_{\text{start}} \in \mathbb{R}^3$ with orientation $\mathbf{v}_{\text{start}} \in SO(3)$. Each tube can be inserted linearly starting at $\mathbf{x}_{\text{start}}$ and rotated axially at its base. We define the length of insertion for tube k as $\beta_k \in \mathbb{R}$ and the rotational value as $\theta_k \in [-\pi, \pi)$. A configuration for the robot then becomes $\mathbf{q} = (\theta_i, \beta_i : i = 1, \ldots, N)$ with configuration space $\mathcal{Q} = (S^1)^N \times \mathbb{R}^N$.

Given a configuration $\mathbf{q} \in \mathcal{Q}$, we define the robot's backbone shape function as $\texttt{backbone}(\mathbf{q}, s) : (S^1)^N \times \mathbb{R}^N \times \mathbb{R} \mapsto \mathbb{R}^3$. This function describes the centerline of the robot as a space curve parameterized by $s \in [0, 1]$ where $\texttt{backbone}(\mathbf{q}, 0) = \mathbf{x}_{\text{start}}$ and $\texttt{backbone}(\mathbf{q}, 1)$ maps to the 3D position of the tip of the robot in the global frame for configuration $\mathbf{q}$. This function, combined with knowledge of the cross-sectional radii of the tubes, allows us to estimate the shape of the robot $\texttt{shape}(\mathbf{q})$ as the volume in space occupied by the robot in configuration $\mathbf{q}$. To compute `backbone` we use the mechanics-based model developed by Leibrandt et al. [133].

We define a path as a continuous function $\sigma : [0, 1] \to \mathcal{Q}$. The motion planning problem then becomes one of finding a path such that $\sigma(0) = \mathbf{q}_{\text{start}}$ and $\texttt{backbone}(\sigma(1), 1) = \mathbf{x}_{\text{goal}}$, the 3D location of the goal point in the global frame. Intuitively, this states that $\sigma(0)$ is the starting configuration of the robot and $\sigma(1)$ is a configuration for which the tip of the robot is at the goal point. We then define a collision-free path as a path such that $\mathbf{p} \notin \texttt{shape}(\sigma(s)), \forall \mathbf{p} \in P, \forall s \in [0, 1]$, i.e., one such that the shape of the robot along the entire path does not contain any points in P. Conceptually, this defines each point in the point cloud as an obstacle, precluding the need for a more complex geometric representation of the anatomy. Defining the kinematic constraints, such as joint limits, of the robot as the general inequality $\mathbf{g}(\sigma) \geq 0$, we can then combine these to define a *valid* motion

plan as one such that the following constraints are satisfied:

$$\begin{aligned}
&\mathbf{p} \notin \texttt{shape}(\sigma(s)), \forall \mathbf{p} \in P, \forall s \in [0,1] \\
&\mathbf{g}(\sigma) \geq 0 \\
&\sigma(0) = \mathbf{q}_{\text{start}} \\
&\texttt{backbone}(\sigma(1), 1) = \mathbf{x}_{\text{goal}}.
\end{aligned} \tag{7.1}$$

To enable the computation of high quality motion plans, we introduce the notion of cost. We choose a cost function that facilitates safe motion planning by favoring plans that move the robot's geometry far from the patient's anatomy. This has the benefit of increasing the safety of the plans by making them robust to actuation noise and to the possibility of small holes and gaps in the point cloud. Specifically, we define a function $\texttt{clear}(\mathbf{q}) = \min_{\mathbf{p} \in P} \texttt{sd}(\texttt{shape}(\mathbf{q}), \mathbf{p})$, where $\texttt{sd}(\texttt{shape}(\mathbf{q}), \mathbf{p})$ is the signed distance between the shape of the robot at configuration $\mathbf{q}$ and point cloud point $\mathbf{p}$. The function $\texttt{sd}$ is defined as the positive distance if $\mathbf{p}$ is external to the robot's geometry, and negative penetration depth if $\mathbf{p}$ is internal to the robot's geometry. $\texttt{clear}(\mathbf{q})$ is the minimum such value over all the points in the point cloud, for a specific configuration. We define the cost of a configuration $\mathbf{q}$ as:

$$\texttt{cost}(\mathbf{q}) = \begin{cases} \dfrac{1}{\texttt{clear}(\mathbf{q})}, & \texttt{clear}(\mathbf{q}) > 0 \\ \infty, & \texttt{clear}(\mathbf{q}) \leq 0. \end{cases} \tag{7.2}$$

The cost of a path σ then becomes

$$\texttt{Cost}(\sigma) = \int_0^1 \texttt{cost}(\sigma(s)) ds. \tag{7.3}$$

We next define a motion planning *query* as the tuple $(P, \mathbf{q}_{\text{start}}, \mathbf{x}_{\text{goal}}, t_{\max})$, where $t_{\max}$ is the maximum time allotted for the motion planner to solve the query. The goal is then to produce a valid motion plan that solves the query, satisfies (7.1), and has as low of a cost as possible, as defined by (7.3).

Multiple queries can be performed, as the physician desires, and P can be updated appropriately

as the patient's anatomy changes during the procedure. We compute motion plans with respect to the most recent P.

7.3 Method

To plan motions for the concentric tube robot in the point cloud representing the patient's anatomy, we propose Parallel Sampling and Interior point optimization Motion Planning (PSIMP). PSIMP combines sampling-based methods (to globally explore different routes around anatomical obstacles) with local optimization (to facilitate fast computation of high quality plans).

7.3.1 Method Overview

We begin by using the sampling-based motion planner PRM* [93], which allows for the discovery of multiple homotopic classes (see Fig. 7.2) while planning using an objective function, (7.3) in our case, returning better and better motion plans as computation time increases. It does so while enforcing that paths obey obstacle avoidance constraints as well as other kinematic constraints. For the concentric tube robot model we use, we have kinematic constraints, i.e., $\mathbf{g}(\sigma) \geq 0$ in (7.1), associated with maximum and minimum insertion values for each tube.

At a high level, our method works by continuously running a PRM* thread which discovers better-and-better paths over time. PRM* generates paths as a discretized sequence of n configurations $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{n-1})$, such that $\mathbf{q}_0 = \mathbf{q}_{\text{start}}$ and $\texttt{backbone}(\mathbf{q}_{n-1}, 1) = \mathbf{x}_{\text{goal}}$, and n can vary depending on the path. A continuous motion plan is derived from such a representation by moving from one configuration to the next, via linear interpolation in configuration space. Each time PRM* discovers a better path than it had previously, that path is placed in a queue of paths that are awaiting local optimization. Simultaneously, a thread pool of local optimization threads are continuously taking motion plans from the queue and improving the plans in parallel via interior point optimization. Interior point optimization iteratively optimizes a path generated by PRM* by moving the configurations in configuration space to lower the overall cost of the motion plan. For example, consider Fig. 7.3, which shows a discretized motion plan as a set of configurations, pre-optimization (Fig. 7.3, top) and post-optimization (Fig. 7.3, bottom). Interior point optimization is used due to its property of maintaining a solution that is collision-free and satisfies the kinematic constraints *during* optimization. This process continues for as long as time allows (i.e., until $t_{\max}$) and the best path found up to any given time is retained (see Fig. 7.1).

We next describe the method's two submethods, *global exploration through sampling* and *interior*

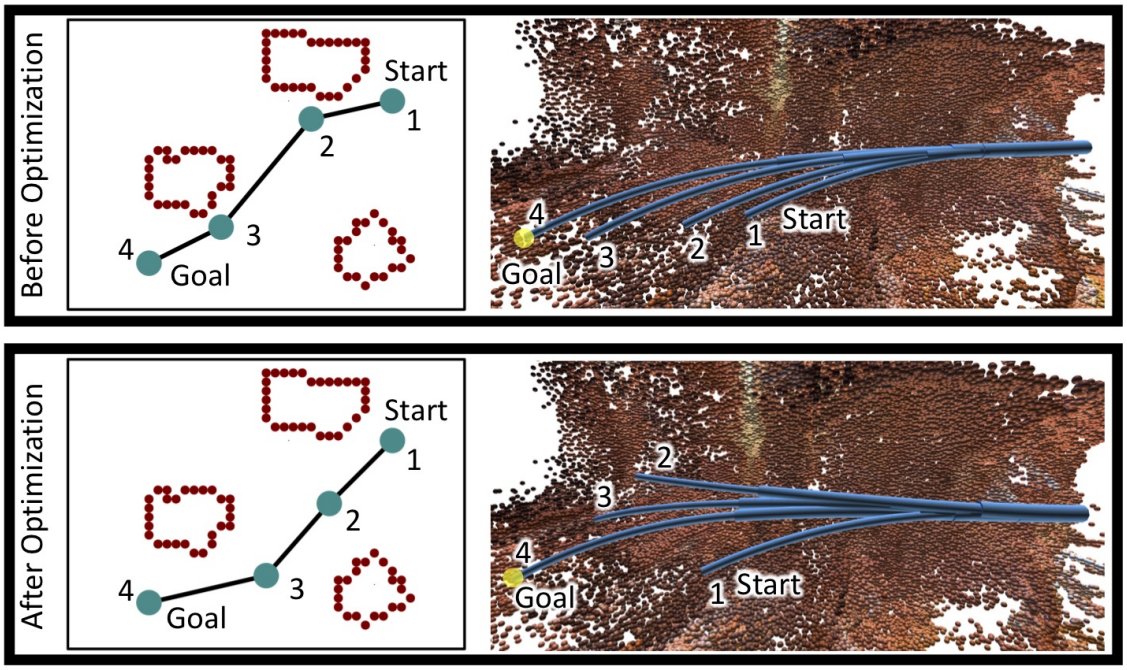


Figure 7.3: Top: A plan produced early on by the sampling-based planner passes close to obstacles on its way to a configuration who's tip touches the goal point (yellow). Observe that during the motion the robot's tip stays near the point cloud (the configurations are overlayed on eachother in the image on the right). Bottom: The intermediate configurations of that plan after optimization will travel further from obstacles as the robot moves toward the goal point. Observe that the robot's tip moves toward the center of the point cloud, far away from the anatomy, as it travels toward the goal point. For demonstrative purposes, to the left of each plan visualization is a 2-dimensional drawing illustrating the concepts. Note, however, that the actual plan exists as a sequence of configurations in the robots 6-dimensional configuration space.

point local optimization.

7.3.2 Global Exploration through Sampling

In our method, we utilize a constantly running PRM* motion planning thread to explore the configuration space and discover paths in multiple homotopic classes. Specifically, we utilize the k-nearest variation of PRM*. PRM* works by iteratively constructing a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a motion-planning roadmap, embedded in $\mathcal{Q}$, where $\mathcal{V}$ is a set of vertices which represent collision-free configurations of the robot and $\mathcal{E}$ is a set of edges, where an edge represents a valid transition between two robot configurations. PRM* randomly samples configurations in $\mathcal{Q}$, and adds the collision-free configurations to $\mathcal{V}$ (we explicitly limit the sampling to configurations that respect the kinematic constraints). It then attempts to connect each newly sampled configuration to its k nearest neighbors in $\mathcal{V}$, where k is a parameter that scales with $|\mathcal{V}|$ as in [93]. If two configurations can be

connected in a collision-free way via linear interpolation in configuration space, an edge between the two configurations is added in $\mathcal{E}$. Because the goal is defined as a location in $\mathbb{R}^3$ and not as a configuration, multiple configurations can satisfy the goal. In order to discover such a configuration quickly, our method performs goal biasing by attempting to ensure that a user-specified percentage of the samples are configurations that touch the goal with their tip and connect them to $\mathcal{G}$. This is done using a damped least-squares inverse kinematics (DLS-IK) controller [134, 135, 9], and allows the method to find an initial solution quickly.

For each motion planning query, PRM* starts by running until it finds the first path in $\mathcal{G}$ that connects the start configuration to a configuration that places the tip of the robot within some radius around the goal point. When that first valid path is found, it is placed in the optimization queue. It then continues sampling configurations, adding to $\mathcal{G}$, discovering lower cost paths and other configurations who's tips reach the goal point. Each time it finds a path with lower cost than it has found before, that path is placed in the optimization queue and sampling continues. This process continues as time allows, i.e., until $t_{\max}$ has been reached. It is worth noting that although we use PRM* in our implementation, many sampling-based motion planning algorithms could be used in a similar way instead.

7.3.3 Interior Point Local Optimization

We maintain a queue of motion plans generated by the PRM* thread. This queue is being operated on by a pool of local optimization threads in parallel. Anytime one of the local optimization threads is available, it retrieves the next motion plan from the queue and performs local optimization on it. When the thread completes the optimization of a plan, it returns to the pool and optimizes the next plan in the queue, if the queue is non-empty. These threads locally optimize plans generated by the PRM* thread using an interior point constrained optimization method [99]. At a high level, this works by taking the initial path, representing it in a high dimensional path space, and performing gradient descent with respect to the path's cost, defined by (7.3), while keeping each iteration interior to the feasible set, i.e., respects the kinematic constraints and is collision-free.

More formally, given a discretized path generated by the PRM*, $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{n-1})$, we concatenate $\mathbf{q}_1$ through $\mathbf{q}_{n-2}$ into a vector of dimensionality $2 * N * (n-2)$, which represents the intermediate configurations in the path. This is done in order to ensure that the start and goal configurations remain unchanged by the optimization. We then perform gradient descent with backtracking line

search using the Armijo condition [99] on this vector with respect to the cost in (7.3). Note that we use only first-order gradient information, computed numerically, as computing higher-order derivatives can be computationally expensive for concentric tube robots. In order to compute the cost of the intermediate configurations vector, the first and last configurations must be added back into the path. In order to ensure there is at least one intermediate configuration, if an initial path has only a start and goal configuration, a third is added halfway between the two.

We use a cost metric that assigns infinite cost to paths that collide with the environment, and which encourages paths to be as far from collision as possible. For this reason, we do not need to formulate the point cloud into the constraint set, enabling us to consider a much simpler set of constraints. This allows us to use a simpler interior point method than in [14]. At each step of descent, if the configurations violate the kinematic constraints, they are projected back into the convex feasible set by clamping the configurations between their maximum and minimum values defined by the robot's tube lengths, ensuring $\mathbf{g}(\sigma) \geq 0$ is satisfied. In this way, a path is locally optimized with respect to the path cost, and the constraints are enforced at each iteration. This frequently results in a large improvement in path cost compared to the pre-optimized path.

7.3.4 Keeping Track of the Best Plan Found

The PRM* thread and the optimization threads are working in parallel to generate better and better plans as time allows, i.e. until $t_{\max}$ has elapsed, at which time the best plan found is returned for execution on the robot. However, even prior to $t_{\max}$, the best plan found at any given time, by any of the threads, is maintained. In this way, if for any reason the algorithm must be stopped early, the best plan found up until that time can be used.

It is worth noting that the choice of an interior point optimization strategy augments the ability to stop the method early and return a high-quality solution. The interior point optimization is an iterative process, i.e., the optimization is occurring as a sequence of small steps in the intermediate configuration's vector space described above. Unlike many other constrained optimization methods, interior point methods have the property that the plan at each of its iterations are valid and collision-free. This implies that we do not need to wait for the interior point optimization to complete before we can leverage the improvements it has found. If PSIMP must stop in the middle of an optimization, the last iterate of the optimization is itself a valid plan, and as such can be used if it is of the lowest cost found by any of the threads up until that time.

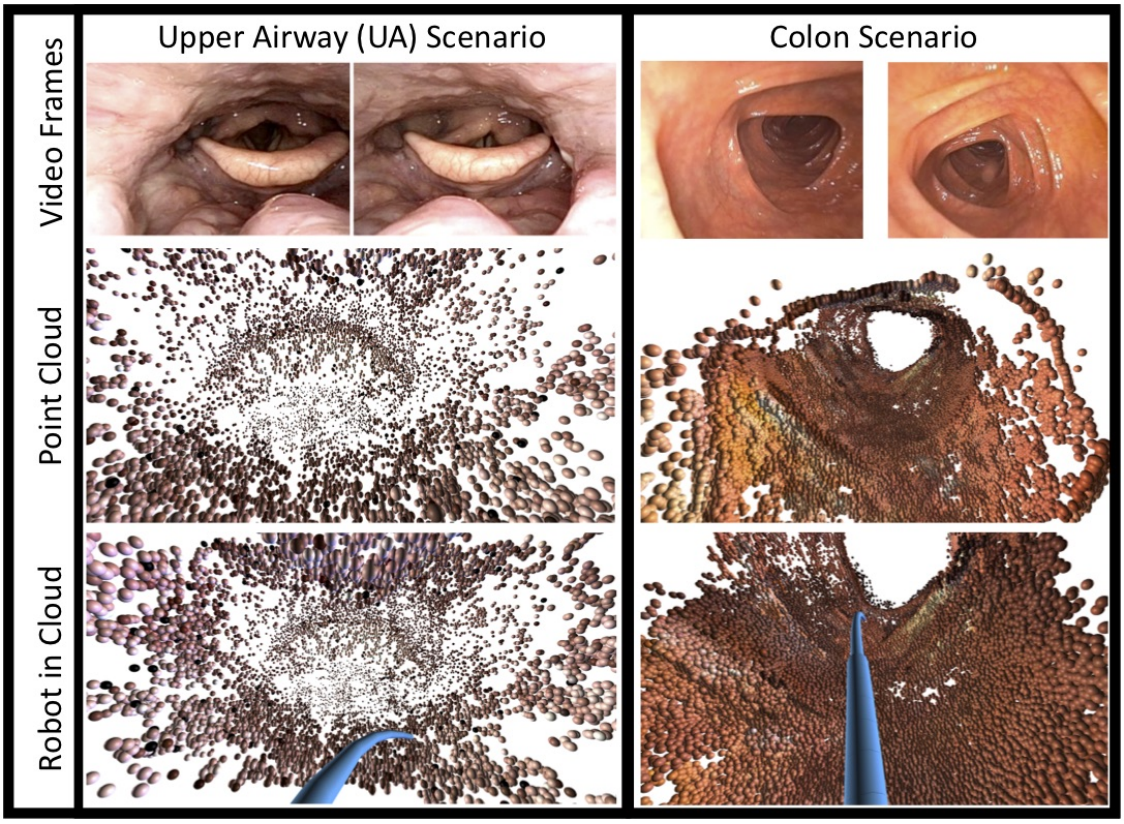


Figure 7.4: We evaluate our method using two point clouds generated from real patient anatomy. Left: The first is generated from an endoscopic video of the upper airway (UA). Right: The second is generated from an endoscopic video of a colon. A simulated version of the concentric tube robot is shown in blue in both point clouds in the bottom row. Both point clouds are generated from endoscopic video using COLMAP [124, 125]. In addition to these two point clouds generated from real patient data, we also evaluate in a point cloud generated from simulated skull base anatomy, which is shown in Fig. 7.7.

7.4 Results

We evaluate PSIMP in two ways. First, we compare it to a pure sampling-based motion planner PRM* by itself, in three anatomical scenarios, and demonstrate that PSIMP is able to find motion plans with significantly lower cost in a fraction of a second, and which continue to improve as time allows. Second, we demonstrate the method's ability to adapt to changes in the anatomy during the surgical procedure.

All results were generated on an 3.40GHz Intel Xeon E5-1680 CPU with 64GB of RAM, and 4 threads were allocated for the local optimization thread pool in all experiments.

7.4.1 Comparison and Analysis

We evaluate our method in three point cloud scenarios, two generated from real patient data using the structure from motion library COLMAP [124, 125] (see Fig. 7.4) and one generated from a synthetic skull model (see Fig. 7.7). The real patient point cloud scenarios are generated from endoscopic video of a patient's upper airway (UA) near the epiglottis, and a patient's colon. Note that although the point clouds used for evaluation are generated via structure from motion, point clouds generated by other methods or sensors can be used by the method and the method remain unchanged.

In all three scenarios, 100 random queries are generated with different collision-free start configurations, and goal points within the reachable workspace of the robot, for 300 total. To ensure we are evaluating queries that simulate surgical tasks, we construct goal points near the point cloud by randomly sampling collision-free configurations that place the tip of the robot within 3 mm of the point cloud, and set the goal point to be the robot's tip position in that configuration. The configuration that generated the goal point is not recorded, only the $\mathbb{R}^3$ goal point and we do not ensure that a valid motion plan exists between the randomly generated start configuration and the tip position prior to evaluating the methods. We evaluate PSIMP's ability to compute high-quality paths over time, and compare the cost of the best computed paths to those generated by a PRM* algorithm without our interior point optimization added. The motion planners were able to successfully plan motions in 98 queries in the UA scenario, 99 queries in the colon scenario, and 98 queries in the skull base scenario. The results presented here are averaged over the successful queries for each scenario.

First, we demonstrate how the quality of the paths improve as computation time increases. In Fig. 7.5, we compare the cost of the best plan found by PSIMP and PRM* up until a given time, with the cost of the best path found after 100 seconds. Because the queries have different start configurations and goal points, the costs between the queries in each scenario can vary greatly making them difficult to compare directly. As such, for each query we compute the ratio of cost at a given time over the best cost found after 100 seconds, and plot the average over all the queries. In Fig. 7.5 we show the ratios as they improve over time. We present the time axis in log scale to provide more detail in the early timescales. As shown, on average the best plans found by PSIMP start improving upon those found by PRM* after ≈ 0.1 seconds, and are $10\% - 30\%$ better by ≈ 0.5

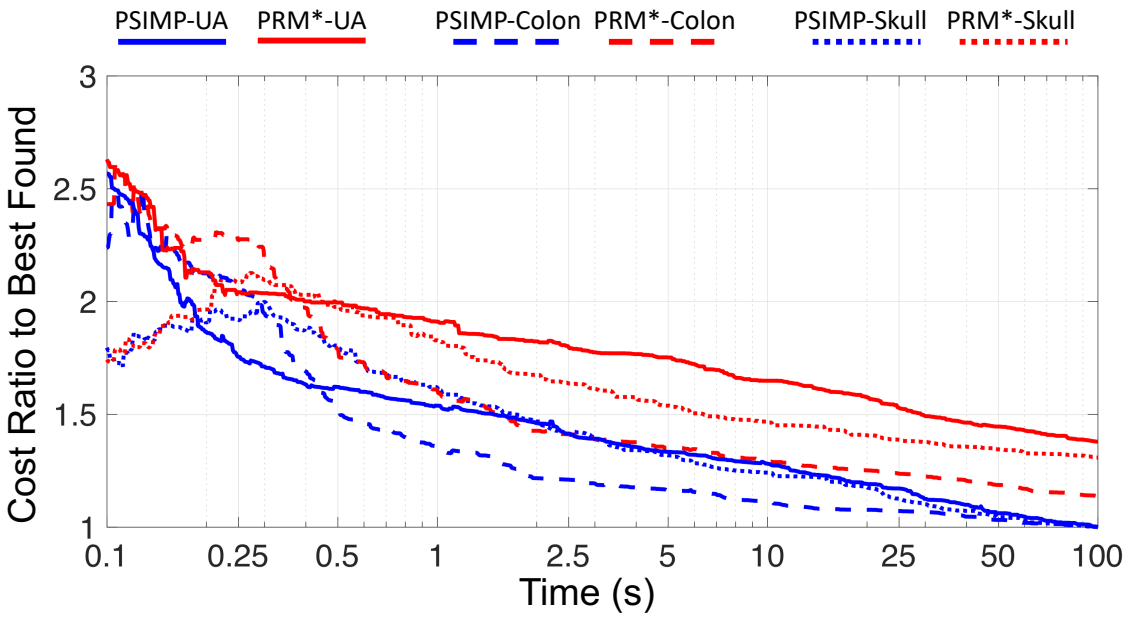


Figure 7.5: We evaluate the performance of PSIMP (blue) and PRM* (red) over time for all three scenarios, UA (solid lines), Colon (dashed lines), and Skull base (dotted lines). Shown here is the ratio of the cost of the best plan found by each method up to a given time divided by the cost of the best plan found at any time, by either method, shown for 100 total seconds of computation. At any time after ≈ 0.25 seconds, the cost of the best plan found by PSIMP is significantly lower than that of PRM*. The results are averaged over 99 different queries for the Colon scenario, and 98 different queries for the UA and Skull base scenario. Note that the time axis is plotted on a log scale to show more detail in the earlier timescales.

seconds, depending on the scenario. Prior to ≈ 0.1 seconds, the methods are comparable as the first plan found by both methods will be identical, and the first optimization of PSIMP has yet to occur.

In Fig. 7.6, we compare the costs of the plans found by the methods directly to each other, plotting the ratio of the cost of the best plan found by PRM* divided by the cost of the best plan found by PSIMP, for any point in time. Similar to the previous results, this shows that PSIMP has found, on average, a plan that is between 10 and 30 percent better than that found by PRM* at any point in time after ≈ 0.5 seconds, with improvements beginning around ≈ 0.1 seconds. Prior to that time, the results between the two methods are very similar due to the reasons described above. This demonstrates the efficacy of the local optimization performed by PSIMP to improve the quality of solutions found at very short timescales.

In order to be effective in a surgical setting, the motion planner used must produce a valid path quickly. This is the case for PSIMP. The first path is found by PSIMP in a fraction of a second, with a median value of 0.12 seconds. This is the time required by the sampling-based thread in

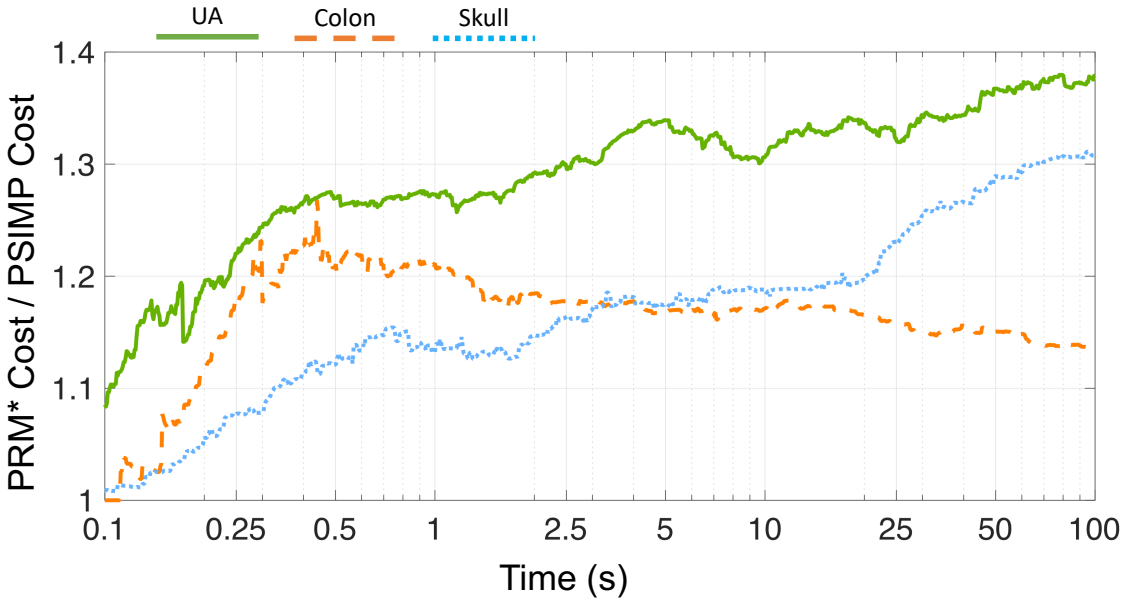


Figure 7.6: We compare the two methods directly. Here we show the cost of the best plan found by PRM* divided by the cost of the best plan found by PSIMP over time. At any time after ≈ 0.5 seconds, PSIMP has found a plan that is 10-30 percent lower in cost than the best plan found by PRM*. The results are averaged over 99 queries for the Colon scenario and 98 queries for the UA and Skull base scenario. Note that the time axis is plotted on a log scale to show more detail in the earlier timescales.

PSIMP to find its first solution. Furthermore, our results demonstrate that if you are willing to allow a small amount of extra computation, PSIMP will significantly improve the plan via local optimization, making it safer to execute. For instance, given 0.15 seconds longer of computation time, PSIMP improves the cost of the best plan found by 13.2% on average across all queries in all scenarios. Further, as more time is allowed to plan, the quality will continue to improve, allowing for safer motion plans.

7.4.2 Adapting to Changing Anatomy

To demonstrate our method adapting to changing anatomy, we use a point cloud constructed from an anatomically inspired model of the human skull base and nasal passageways. For visualization of the entire process, see Fig. 7.7. We generate the point cloud from the model by moving a virtual camera through the nasal passageways, and recording the points on the surface of the model visible to the virtual camera. These points are then concatenated into a point cloud.

We consider an example surgical scenario in the skull base, wherein the robot must move through the sinuses, remove an obstruction in the sinus passageways on the patient's left side, and then move

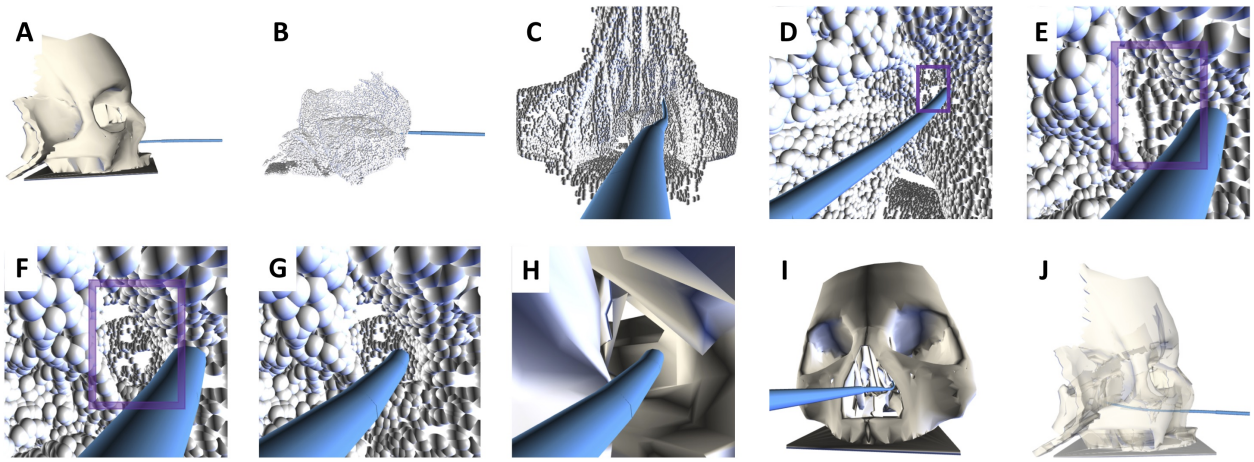


Figure 7.7: PSIMP plans collision-free motions for a concentric tube robot in anatomy represented using a point cloud, and it can quickly replan motions based on a new point cloud when the anatomy changes. (A) An anatomical model of the skull base from which we derive the initial point cloud. (B) The point cloud derived from the model using a virtual camera moving through the sinuses and skull base. (C) Motions are planned for the robot (blue) deep into the sinuses using our method. (D) In the model, an obstruction exists in the sinuses. The point cloud reflects the obstruction with points on its surface (in the purple window). We plan a motion for the robot to a point at the obstruction. (E) A closer view of the obstruction. (F) The obstruction is removed from the anatomy, and we generate a new point cloud which reflects the new opening (in the purple window). (G) Our method generates a motion plan for the robot to move through the new opening in the point cloud to a point behind the opening. (H,I) Two views of the anatomical model that has the obstruction removed, with the robot passing through the obstruction. (J) The anatomical model rendered semi-transparent for visualization of the final configuration. Our method was able to find a collision-free solution both to the obstruction and beyond.

deeper into the skull base to continue the procedure. The point cloud anatomical model initially reflects this blockage by containing points on the surface of the blockage, and not containing points behind it (due to occlusion caused by the blockage). We first plan a motion for the concentric tube robot that brings the tip of the robot up to the blockage. Next we remove the blockage from the model, as if it were done during the surgical procedure. We then generate a new point cloud with the blockage removed, adding points that can be viewed from near the robot's tip, as if it were carrying a small chip tip camera. This new cloud is then used as the obstacle representation for the robot, and a motion is planned for it to proceed beyond where the blockage was previously.

Planning the second motion through the opening would not be possible in the case where the obstacle representation remains static, such as is the case when segmenting obstacles only from pre-operative imaging. By updating the model, as we do through updating the point cloud, the obstacle representation accurately represents the patient's changed anatomy and safe motions can

be planned with respect to the changed anatomy.

This is a demonstrative example showing the value of using an obstacle representation that can be generated quickly intra-operatively, compared with using pre-operative imaging to generate the obstacle representation for planning.

7.5 Conclusion

Planning motions for concentric tube robots in point clouds allows for adaptation with respect to changing patient anatomy during the course of a surgical procedure. We presented PSIMP, a method that effectively and safely plans motions in point cloud representations of anatomy using a combination of sampling-based global exploration and interior point local optimization. We evaluated our method in three anatomical scenarios, an upper airway scenario and colon scenario generated from endoscopic video of real patients, and a skull base scenario with point clouds generated from simulated images. We evaluated the efficacy of our method, showing that it succeeds in quickly finding collision-free motion plans and significantly improves upon the initial motion plans in fractions of a second. We demonstrated the ability of PSIMP to react to changing point clouds and to plan motions based on the updated information.

We note that as an open-loop method, this work is subject to uncertainty. As such, we plan to evaluate PSIMP on a physical robot in ex vivo or phantom anatomy using closed-loop, sensed tip position control in order to follow motion plans computed by PSIMP. We also plan to consider accounting for the uncertainty associated with concentric tube robot mechanical models during the motion planning process.

CHAPTER 8

Estimating the Complete Shape of Concentric Tube Robots via Learning

In order to safely control and plan motions for concentric tube robots that automatically prevent unintended collisions with the patient's anatomy, an accurate shape model of the entire robot's shaft is required. Accurate prediction of the entire shape of a concentric tube robot from its control inputs is challenging, and current state-of-the-art shape models are often unable to accurately account for complex and unpredictable physical phenomena such as inconsistent friction between tubes, non-homogenous material properties, and imprecisely shaped tubes [9, 10]. In this work, we present a data driven, deep neural network-based approach for learning a more accurate model of a concentric tube robot's entire shape.

Machine learning enables a data driven approach to the shape estimation of concentric tube robots. Neural network models have been successfully used to more accurately model the forward kinematics and inverse kinematics of concentric tube robots [136, 16], and an ensemble method has been applied to learn and adapt a forward kinematics model online [10]. However, these models only consider the pose of the robot's tip. In order to successfully plan and execute motions that avoid unwanted collisions between the robot's shaft and patient anatomy, a model must accurately predict the *entire shape* of the robot.

In this chapter, we present a deep neural network approach that learns a function that accurately models the entire shape of the concentric tube robot, for a given set of tubes, as a function of its configuration (see Fig. 8.1). The neural network takes as input the robot's configuration, and the network outputs coefficients for orthonormal polynomial basis functions in x, y, and z parameterized by arc length along the robot's tubular shaft. In this way, a function representing the entire shape of the robot can be produced by one feed forward pass through the neural network.

The key insight behind our parameterization is that the uncertainty in the physics-based shape models is due mainly to uncertainty in curvature and torsion. The *arc length* of the robot's shape, however, is independent of these and as such is generally not subject to the same sources of uncertainty.

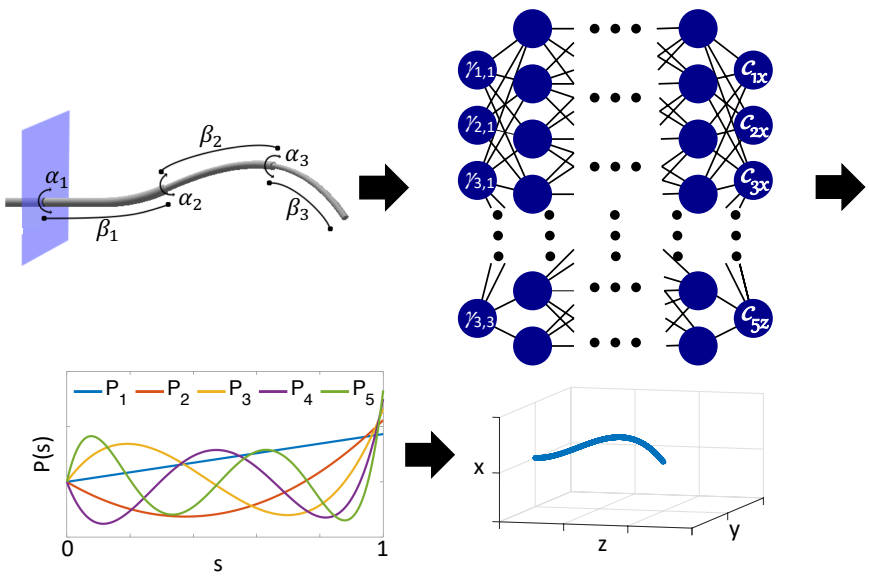


Figure 8.1: Given a concentric tube robot configuration defined by the translations and rotations of the tubes (upper left), our neural network (upper right) outputs coefficients for a set of polynomial basis functions (lower left) that are combined to model the backbone of the robot's 3D shape (lower right).

We can leverage this known state by parameterizing our shape function by arc length.

This chapter is based on work previously published in [18].

8.1 Materials & Methods

In order to learn a shape function for the concentric tube robot, data representing the robot's shape as a function of its configuration must be gathered. To gather shape data, we utilize a multi-view 3D computer vision technique called shape from silhouette [17], in which multiple images of the robot's shape for a given configuration are collected from cameras with known position (see Fig. 8.2). We then automatically segment the robot's shaft in each image using color thresholding and for each pixel in the segmentation a ray is traced out from the camera's position through its image plane. These rays then pass through a voxelized representation of the world, and voxels that are intersected by rays from every camera represent the robot's shape in the world. We then fit a 3D space curve to the voxels using ordinary least squares, resulting in a curve that represents the true, sensed backbone of the robot. We then train the neural network using the sensed backbone as ground truth.

Our neural network architecture consists of a feed forward, fully connected network, with 5 hidden layers of 30 nodes each. We utilize the parametric rectified linear unit as our non-linear

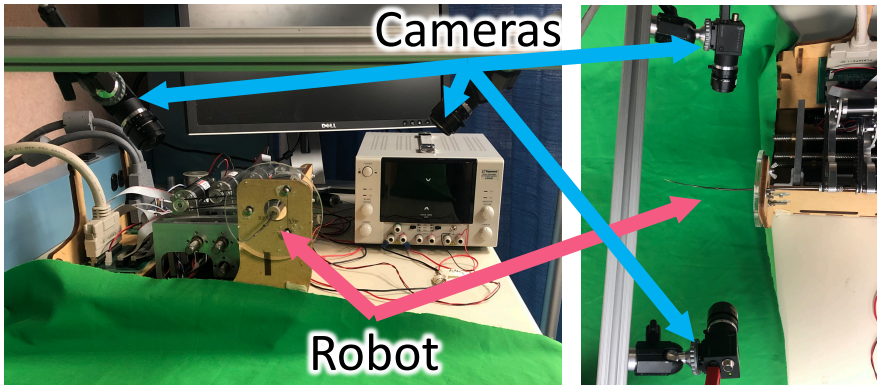


Figure 8.2: We train the neural network using data from a physical robot. By taking images from multiple cameras (blue arrows), the shape of the robot's shaft (pink arrows) can be reconstructed in 3D using shape from silhouette.

activation function between layers, which we noted provided a slight performance improvement over the standard rectified linear unit.

For a robot consisting of k tubes, we parameterize the ith tube's state as $\boldsymbol{\gamma}_i := \{\gamma_{1,i}, \gamma_{2,i}, \gamma_{3,i}\} = \{\cos(\alpha_i), \sin(\alpha_i), \beta_i\}$ where $\alpha_i \in (-\pi, \pi]$ is the ith tube's rotation and $\beta_i \in \mathbb{R}$ is the ith tube's translation, as in [16]. We then parameterize the robot's configuration as $\mathbf{q} = (\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_k)$. This serves as the input to the neural network.

The network outputs 15 coefficients, $c_{1x}, c_{2x}, \ldots c_{5x}, c_{1y}, c_{2y}, \ldots c_{5y}, c_{1z}, c_{2z}, \ldots c_{5z}$, which serve as coefficients for a set of 5 orthonormal polynomial basis functions in x, y, and z parameterized by arc length, shown in Table 8.1. This results in three functions, $x(\mathbf{q}, s)$, $y(\mathbf{q}, s)$, and $z(\mathbf{q}, s)$, where $x(\mathbf{q}, s) = \text{len}(\mathbf{q}) \times (c_{1x}\text{P}_1(s) + c_{2x}\text{P}_2(s) + \cdots + c_{5x}\text{P}_5(s))$, where s is a normalized arc length parameter between 0 and 1, and $\text{len}(\mathbf{q})$ is the total arc length of the robot's backbone in configuration $\mathbf{q}$. Then $y(\mathbf{q}, s)$ and $z(\mathbf{q}, s)$ are defined similarly with their respective coefficients. The resulting shape function is $\texttt{shape}(\mathbf{q}, s) = < x(\mathbf{q}, s), y(\mathbf{q}, s), z(\mathbf{q}, s) >$. To evaluate the shape of the robot at a given configuration, the neural network can be evaluated at $\mathbf{q}$, and the resulting coefficients define a space-curve function that can then be evaluated at any desired arc length. This, combined with knowledge of the robot's radius as a function of arc length, results in a prediction of the robot's geometry in the world.

We first pretrained our model on $100,000$ data points (configuration and backbone pairs), where the configuration was sampled uniformly at random from the robot's configuration space and the backbone was generated by the physics-based model presented in [9]. Such pretraining allows us to

	s	s^2	s^3	s^4	s^5
$P_1(s)$	1.7321	0	0	0	0
$P_2(s)$	-6.7082	8.9943	0	0	0
$P_3(s)$	15.8745	-52.915	39.6863	0	0
$P_4(s)$	-30.0	180.0	-315.0	168.0	0
$P_5(s)$	49.7494	-464.33	1392.98	-1671.6	696.4912

Table 8.1: Coefficients for the orthonormal polynomial basis functions generated using Gram-Schmidt orthogonalization. $P_1(s) := 1.7321s$, $P_2(s) := -6.7082s + 8.9943s^2$, etc., plotted in Fig. 8.1 (lower left).

Tube	Outer Diameter (m)	Inner Diameter (m)	Straight Length (m)	Curved Length (m)	Curvature (m^{-1})
Inner	0.0013	0.0010	0.2459	0.0666	9.3354
Middle	0.0019	0.0016	0.1631	0.0456	4.6270
Outer	0.0025	0.0022	0.0952	0.0364	7.4184

Table 8.2: Tube parameters for the 3-tube concentric tube robot.

prevent overfitting on the smaller amount of sensed, real world data.

We then trained our network on $8,000$ randomly sampled data points, and we evaluate the network on $1,000$ different randomly sampled test data points (both sets generated via shape from silhouette). We utilize a pointwise sum-of-squared-distances loss function and the ADAM [137] optimizer during training.

8.2 Results

The specifications of the robot's component tubes used in the experiments are shown in Table 8.2.

First, in order to evaluate how well the polynomial basis functions are able to approximate the shape, we computed the optimal set of coefficients for the 100,000 pre-training data points using ordinary least squares. We then evaluated how well the resulting shape representation approximated the training data at 20 equally-spaced points along the backbone of the shape, as in the training process. We then calculated the maximum L^2 distance over the 20 points along the backbone of the robot between the polynomial representation and the ground truth for each of the 100,000 configurations. Over the 100,000 configurations, the mean of the maximum L^2 distance along the backbone was 0.044 ± 0.037 mm. This demonstrates that the polynomial basis functions are capable of representing the shape of a concentric tube robot with accuracy well into the sub-millimeter range.

Next, we evaluate how well our model is able to learn the shape of real-world concentric tube

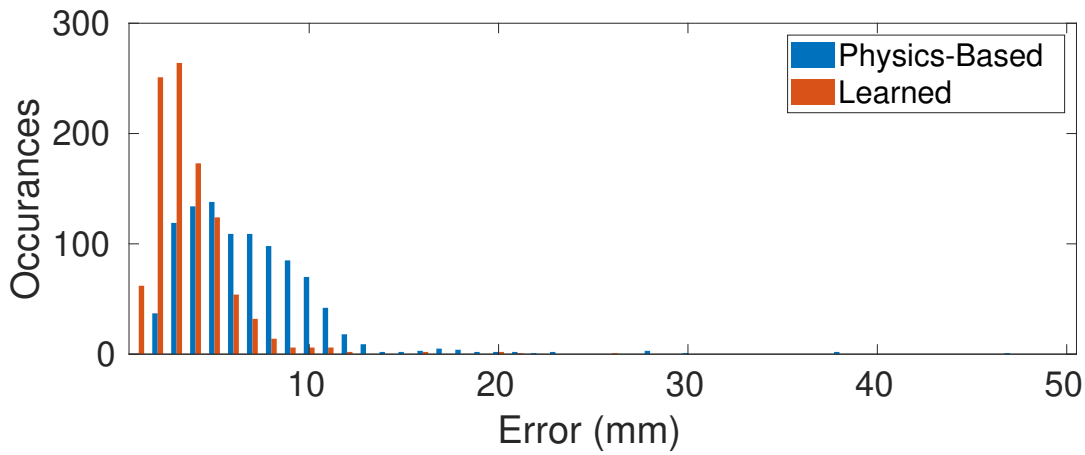


Figure 8.3: A histogram of the maximum error along the robot's shaft for the learned model and the physics-based model, for each of the $1,000$ test points. The distribution is shifted to the left in the learned model, indicating that it is more likely to produce lower error values.

	Physics-Based (mm)	Learned (mm)
Minimum	1.11	0.49
Maximum	46.68	25.32
Mean	6.32 ± 3.95	3.16 ± 2.20

Table 8.3: Error value statistics for the physics-based model and our learned model across the 1000 test configurations.

robot configurations. Specifically, we compare our neural network's shape computation to that of the physics-based model presented in [9]. In Fig. 8.3 we plot a histogram of the errors across the $1,000$ test configurations. For each configuration of our 3-tube robot we evaluate the shape of the physics-based model, the learned model, and the ground truth from the vision system at 20 evenly spaced points along the robot's shaft. We then present the maximum error—the L^2 distance of the point that deviates from the ground truth the most. This error value presents us with a maximum deviation between the predicted shape and the ground truth, a useful metric when considering safety related to anatomical obstacle avoidance. In Table 8.3, we present further statistics for the maximum error values over the 1000 test configurations for both the physics-based model and our learned model. In the histogram in Fig. 8.3, it can be seen that the error distribution of the learned model is shifted to the left compared with that of the physics-based model, indicating that the learned model is more likely to produce lower error values than the physics-based model. Additionally, the learned model produces lower minimum, maximum, and mean error values.

8.3 Discussion

In this work we present a learned, neural network model that outputs an arc length parameterized space curve. This allows us to take a data driven approach to modeling the shape of the concentric tube robot and improve upon a physics-based model. This may allow for safer motion planning and control of these devices in surgical settings that require avoiding anatomical obstacle, as a shape that deviates less from the shape predicted in computation will be less likely to unintentionally collide with the patient's anatomy. The model is only trained on cases where the robot is operating in free space. Accounting for interaction with tissue is the subject of future work. We also plan to investigate using different network architectures, i.e., a different number of hidden nodes and layers, as the architecture used in this work was chosen heuristically. We also intend to augment the learned model to account for other sources of uncertainty in concentric tube robot shape modeling, including hysteresis, and plan to integrate the learned model with a motion planner and evaluate its use in automatic obstacle avoidance during tele-operation or automatic execution of surgical tasks.

CHAPTER 9

Conclusion

In order to deploy robots in human-centric environments, they must be able to move in these environments safely. Robot motion planning allows for the computation of safe, efficient motions that obey robot specific constraints while avoiding collision with obstacles in the robot's environment, be they anatomical obstacles in a surgical robotics setting or more traditional obstacles in a home assistance setting. In this dissertation, we presented methods that improve upon the state-of-the-art in robot motion planning in such settings.

Specifically, in this dissertation we addressed the following thesis statement:

For robots operating in human-centric environments, robot motion planning, via a combination of sampling- and optimization-based methods, enables fast and high-quality kinematic design optimization and motion computation while overcoming challenges associated with computationally expensive kinematic models, hybrid and highly-constrained kinematics, and point-cloud obstacle representations.

To support this thesis, we presented novel motion planning methods that incorporate random sampling as well as numerical optimization. We first presented a sampling-based robot motion planning algorithm for the CRISP robot that accounts for its expensive shape computation. We then leveraged this motion planning algorithm to demonstrate that an efficient motion planner can be used to optimize the pre-procedure design parameters of this robot. Next, we presented a motion planning algorithm for a three-stage transoral lung tumor biopsy robot that accounted for its hybrid system kinematics and evaluated its efficacy in both simulated settings and *ex vivo* porcine tissue in the real world. We then expanded our motion planning contributions beyond the surgical setting by presenting a novel method that incorporates sampling and optimization in order to efficiently plan motions for a serial-link manipulator robot in a home assistance setting with point cloud obstacle representations. We expanded upon this motion planning concept and presented a parallel version of this algorithm applied to concentric tube robots. Finally, we presented a machine-learning based neural shape model for concentric tube robots that will enable more accurate shape computation

during motion planning.

In order to bring our work closer to real-world deployment, more evaluation in the real world is required. For instance, while our evaluation of the three-stage lung tumor biopsy robot in *ex vivo* tissue is a step in the right direction, this work must be evaluated in *in vivo* settings to move it toward clinical deployment. Similarly, much or our work in this dissertation focusses on algorithmic development and as such is primarily evaluated in simulation. Moving this evaluation into the real world will help us to identify areas to improve upon in the future, such as accounting for sources of uncertainty that we do not yet model.

Many natural extensions also exist that would allow for a broader impact of our methods. For instance, expanding ISIMP and PSIMP in order to work with non-holonomic systems, such as steerable needles, is a promising area for future work. Similarly, incorporating the learned, neural shape model of concentric tube robots into a recurrent framework would allow the method to account for hysteresis in the shape computation.

Broadly speaking, in the future we intend to take steps that both broaden the applicability of the methods, expanding their impact to other domains, as well as improving the efficacy of our methods in their current domain, by identifying and accounting for short-comings in our current models through real-world evaluation.

REFERENCES

[1] A. W. Mahoney, P. L. Anderson, P. J. Swaney, F. Maldonaldo, and R. J. Webster III, "Reconfigurable parallel continuum robots for incisionless surgery," in *IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 4330–4336, Oct. 2016.

[2] P. L. Anderson, A. W. Mahoney, and R. J. W. III, "Continuum reconfigurable parallel robots for surgery: Shape sensing and state estimation with uncertainty," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1617–1624, July 2017.

[3] H. B. Gilbert, D. C. Rucker, and R. J. Webster III, "Concentric tube robots: The state of the art and future directions," in *Int. Symp. Robotics Research (ISRR)*, Dec. 2013.

[4] L. G. Torres, A. Kuntz, H. B. Gilbert, P. J. Swaney, R. J. Hendrick, R. J. Webster III, and R. Alterovitz, "A motion planning approach to automatic obstacle avoidance during concentric tube robot teleoperation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 2361–2367, May 2015.

[5] P. J. Swaney, A. W. Mahoney, A. A. Remirez, E. Lamers, B. I. Hartley, R. H. Feins, R. Alterovitz, and R. J. Webster III, "Tendons, concentric tubes, and a bevel tip: three steerable robots in one transoral lung access system," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 5378–5383, May 2015.

[6] Rethink Robotics, "Baxter research robot," 2013.

[7] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Trans. Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.

[8] A. Kuntz, A. W. Mahoney, N. E. Peckman, P. L. Anderson, F. Maldonado, R. J. Webster, and R. Alterovitz, "Motion planning for continuum reconfigurable incisionless surgical parallel robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6463–6469, Sept. 2017.

[9] D. C. Rucker, *The mechanics of continuum robots: model-based sensing and control.* PhD thesis, Vanderbilt University, 2011.

[10] G. Fagogenis, C. Bergeles, and P. E. Dupont, "Adaptive nonparametric kinematic modeling of concentric tube robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4324–4329, Oct. 2016.

[11] A. Kuntz, C. Bowen, C. Baykal, A. W. Mahoney, P. L. Anderson, F. Maldonado, R. J. Webster, and R. Alterovitz, "Kinematic design optimization of a parallel surgical robot to maximize anatomical visibility via motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 926–933, IEEE, May 2018.

[12] A. Kuntz, L. G. Torres, R. H. Feins, R. J. Webster III, and R. Alterovitz, "Motion planning for a three-stage multilumen transoral lung access system," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 3255–3261, Sept. 2015.

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

[14] A. Kuntz, C. Bowen, and R. Alterovitz, "Fast anytime motion planning in point clouds by interleaving sampling and interior point optimization," in *Proc. International Symposium on Robotics Research (ISRR)*, Dec. 2017.

[15] A. Kuntz, M. Fu, and R. Alterovitz, "Planning high-quality motions for concentric tube robots in point clouds via parallel sampling and optimization," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Nov 2019. to appear.

[16] R. Grassmann, V. Modes, and J. Burgner-Kahrs, "Learning the forward and inverse kinematics of a 6-dof concentric tube continuum robot in se(3)," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Madrid, Spain), pp. 5125–5132, Oct. 2018.

[17] S. Baker and T. Kanade, "Shape-from-silhouette across time part I: Theory and algorithms," *International Journal of Computer Vision*, vol. 62, no. 3, pp. 221–247, 2005.

[18] A. Kuntz, A. Sethi, and R. Alterovitz, "Estimating the complete shape of concentric tube robots via learning," in *Hamlyn Symposium on Medical Robotics*, June 2019.

[19] G. Chirikjian, "Conformational modeling of continuum structures in robotics and structural biology: A review," *Adv. Robot.*, vol. 29, no. 13, pp. 817–829, 2015.

[20] R. W. Light, *Pleural Diseases.* Lippincott Williams & Wilkins, 2007.

[21] M. Noppen, "The utility of thoracoscopy in the diagnosis and management of pleural disease," in *Seminars in Respiratory and Critical Care Medicine*, vol. 31, pp. 751–759, Thieme Medical Publishers, 2010.

[22] R. J. Harris, M. S. Kavuru, A. C. Mehta, S. V. Medendorp, H. P. Wiedemann, T. J. Kirby, and T. W. Bice, "The impact of thoracoscopy on the management of pleural disease," *Chest*, vol. 107, no. 3, pp. 845–852, 1995.

[23] A. W. Mahoney, H. B. Gilbert, and R. J. Webster III, "A review of concentric tube robots: Modeling, control, design, planning, and sensing," *Encyclopedia of Medical Robotics*, no. Minimally Invasive Surgical Robotics, 2016.

[24] P. E. Dupont, J. Lock, B. Itkowitz, and E. Butler, "Design and control of concentric-tube robots," *IEEE Trans. Robotics*, vol. 26, pp. 209–225, Apr. 2010.

[25] R. Xu, A. Asadian, A. S. Naidu, and R. V. Patel, "Position control of concentric-tube continuum robots using a modified jacobian-based approach," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 5793–5798, May 2013.

[26] L. A. Lyons, R. J. Webster III, and R. Alterovitz, "Motion planning for active cannulas," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 801–806, Oct. 2009.

[27] N. J. Cowan, K. Goldberg, G. S. Chirikjian, G. Fichtinger, R. Alterovitz, K. B. Reed, V. Kallem, W. Park, S. Misra, and A. M. Okamura, "Robotic needle steering: design, modeling, planning, and image guidance," in *Surgical Robotics: System Applications and Visions* (J. Rosen, B. Hannaford, and R. M. Satava, eds.), ch. 23, pp. 557–582, Springer, 2011.

[28] K. Hauser, R. Alterovitz, N. Chentanez, A. M. Okamura, and K. Goldberg, "Feedback control for steering needles through 3D deformable tissue using helical paths," in *Robotics: Science and Systems (RSS)*, June 2009.

[29] M. C. Bernardes, B. V. Adorno, P. Poignet, and G. A. Borges, "Robot-assisted automatic insertion of steerable needles with closed-loop imaging feedback and intraoperative trajectory replanning," *Mechatronics*, vol. 23, pp. 630–645, July 2013.

[30] K. M. Seiler, S. P. Singh, S. Sukkarieh, and H. Durrant-Whyte, "Using lie group symmetries for fast corrective motion planning," *Int. J. Robotics Research*, vol. 31, pp. 151–166, Dec. 2011.

[31] S. Patil, J. Burgner, R. J. Webster III, and R. Alterovitz, "Needle steering in 3-D via rapid replanning," *IEEE Trans. Robotics*, vol. 30, pp. 853–864, Aug. 2014.

[32] W. Sun, S. Patil, and R. Alterovitz, "High-frequency replanning under uncertainty using parallel sampling-based motion planning," *IEEE Trans. Robotics*, vol. 31, pp. 104–116, Feb. 2015.

[33] M. Christie, P. Olivier, and J.-M. Normand, "Camera control in computer graphics," *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197–2218, 2008.

[34] J. Rosell, A. Perez, P. Cabras, and A. Rosell, "Motion planning for the virtual bronchoscopy," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 2932–2937, May 2012.

[35] D. Nieuwenhuisen and M. H. Overmars, "Motion planning for camera movements," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 3870–3876, 2004.

[36] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3583–3592, 2016.

[37] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg, "A dataset for developing and benchmarking active vision," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1378–1385, May 2017.

[38] H. J. Johnson, M. McCormick, L. Ibáñez, and Insight Software Consortium, "The ITK software guide." Available: http://www.itk.org/ItkSoftwareGuide.pdf, Dec. 2013.

[39] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, pp. 293–308, Natick, MA: AK Peters, 2001.

[40] K. Shoemake, "Animating rotation with quaternion curves," *Computer Graphics (Proc. SIGGRAPH 1985)*, vol. 19, no. 3, pp. 245–254, 1985.

[41] M. B. Rubin, *Cosserat Theories: Shells, Rods and Points.* Springer Science & Business Media, 2000.

[42] S. S. Antman, *Nonlinear Problems of Elasticity.* Springer, 1995.

[43] L. Ingber, "Very fast simulated re-annealing," *Mathematical and Computer Modelling*, vol. 12, no. 8, pp. 967–973, 1989.

[44] C. Baykal and R. Alterovitz, "Asymptotically optimal design of piecewise cylindrical robots using motion planning," in *Robotics: Science and Systems (RSS)*, Robotics: Science and Systems Foundation, July 2017.

[45] C. Bergeles, A. H. Gosline, N. V. Vasilyev, P. Codd, P. J. del Nido, and P. E. Dupont, "Concentric tube robot design and optimization based on task and anatomical constraints," *IEEE Trans. Robotics*, vol. 31, pp. 67–84, Feb. 2015.

[46] J. Burgner, H. B. Gilbert, and R. J. Webster III, "On the computational design of concentric tube robots: Incorporating volume-based objectives," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1185–1190, May 2013.

[47] J. Ha, F. C. Park, and P. E. Dupont, "Achieving elastic stability of concentric tube robots through optimization of tube precurvature," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 864–870, Sept. 2014.

[48] T. K. Morimoto, J. D. Greer, M. H. Hsieh, and A. M. Okamura, "Surgeon design interface for patient-specific concentric tube robots," in *Proc. IEEE Int. Conf. Biomedical Robotics and Biomechatronics (BioRob)*, pp. 41–48, 2016.

[49] L. G. Torres, R. J. Webster III, and R. Alterovitz, "Task-oriented design of concentric tube robots using mechanics-based models," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 4449–4455, Oct. 2012.

[50] C. Baykal, L. G. Torres, and R. Alterovitz, "Optimizing design parameters for sets of concentric tube robots using sampling-based motion planning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 4381–4387, Sept. 2015.

[51] S. Niyaz, A. Kuntz, O. Salzman, R. Alterovitz, and S. S. Srinivasa, "Following surgical trajectories with concentric tube robots via nearest-neighbor graphs," in *Proc. International Symposium on Experimental Robotics (ISER)*, Nov. 2018.

[52] S. Niyaz, A. Kuntz, O. Salzman, R. Alterovitz, and S. S. Srinivasa, "Optimizing motion-planning problem setup via bounded evaluation with application to following surgical trajectories," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Nov 2019. to appear.

[53] T. Liu and M. C. Cavusoglu, "Needle grasp and entry port selection for automatic execution of suturing tasks in robotic minimally invasive surgery," *IEEE Transactions on Automation Science and Engineering*, vol. 13, pp. 552–563, Apr. 2016.

[54] Y. Hayashi, K. Misawa, and K. Mori, "Optimal port placement planning method for laparoscopic gastrectomy," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1677–1684, Mar. 2017.

[55] M. Feng, X. Jin, W. Tong, X. Guo, J. Zhao, and Y. Fu, "Pose optimization and port placement for robot-assisted minimally invasive surgery in cholecystectomy," *The International Journal of Medical Robotics and Computer Assisted Surgery*, 2017.

[56] Y. Mulgaonkar, B. Araki, J.-s. Koh, L. Guerrero-Bonilla, D. M. Aukes, A. Makineni, M. T. Tolley, D. Rus, R. J. Wood, and V. Kumar, "The flying monkey: a mesoscale robot that can run, fly, and grasp," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 4672–4679, 2016.

[57] C. Wright, A. Buchan, B. Brown, J. Geist, M. Schwerin, D. Rollinson, M. Tesch, and H. Choset, "Design and architecture of the unified modular snake robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 4347–4354, May 2012.

[58] M. M. Plecnik, D. W. Haldane, J. K. Yim, and R. S. Fearing, "Design exploration and kinematic tuning of a power modulating jumping monopod," *Journal of Mechanisms and Robotics*, vol. 9, no. 1, p. 11009, 2017.

[59] G. Jing, T. Tosun, M. Yim, and H. Kress-Gazit, "An end-to-end system for accomplishing tasks with modular robots.," in *Robotics: Science and Systems (RSS)*, 2016.

[60] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[61] L. Denarie, K. Molloy, M. Vaisset, T. Siméon, and J. Cortés, "Combining system design and path planning," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Dec. 2016.

[62] J. Park, P. Chang, and J. Yang, "Task-oriented design of robot kinematics using the grid method," *J. Advanced Robotics*, vol. 17, no. 9, pp. 879–907, 2003.

[63] R. Vijaykumar, K. J. Waldron, and M. J. Tsai, "Geometric optimization of serial chain manipulator structures for working volume and dexterity," *Int. J. Robotics Research*, vol. 5, no. 2, pp. 91–103, 1986.

[64] J.-P. Merlet, "Optimal design of robots," in *Robotics: Science and Systems (RSS)*, 2005.

[65] O. Chocron, "Evolutionary design of modular robotic arms," *Robotica*, vol. 26, no. 3, pp. 323–330, 2008.

[66] L. K. Katragadda, *A Language and Framework for Robotic Design.* PhD thesis, Carnegie Mellon University, 1997.

[67] C. Leger, *Automated Synthesis and Optimization of Robot Configurations: An Evolutionary Approach.* PhD thesis, Carnegie Mellon University, 1999.

[68] D. Salle, P. Bidaud, and G. Morel, "Optimal design of high dexterity modular MIS instrument for coronary artery bypass grafting," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1276–1281, Apr. 2004.

[69] O. Taylor and A. Rodriguez, "Optimal shape and motion planning for dynamic planar manipulation," in *Robotics: Science and Systems (RSS)*, July 2017.

[70] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Joint optimization of robot design and motion parameters using the implicit function theorem," in *Robotics: Science and Systems (RSS)*, July 2017.

[71] M. Locatelli, "Simulated annealing algorithms for continuous global optimization," in *Handbook of Global Optimization*, pp. 179–229, Springer, 2002.

[72] S. M. LaValle, *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006.

[73] American Cancer Society, "Cancer facts & figures 2010," tech. rep., American Cancer Society, 2010.

[74] G. Krishna and M. K. Gould, "Minimally invasive techniques for the diagnosis of peripheral pulmonary nodules," *Current Opinion in Pulmonary Medicine*, vol. 14, pp. 282–286, 2008.

[75] L. M. Perlmutt, W. W. Johnston, and N. R. Dunnick, "Percutaneous transthoracic needle aspiration: a review," *American Journal of Roentgenology*, vol. 152, pp. 451–455, 1989.

[76] R. S. Wiener, L. M. Schwartz, S. Woloshin, and H. Gilbert Welch, "Population-based risk for complications after transthoracic needle lung biopsy of a pulmonary nodule: An analysis of discharge records," *Annals of Internal Medicine*, vol. 155, pp. 137–144, Aug. 2011.

[77] R. J. Hendrick, S. D. Herrell, and R. J. Webster III, "A multi-arm hand-held robotic system for transurethral laser prostate surgery," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 2850–2855, May 2014.

[78] E. J. Butler, R. Hammond-Oakley, S. Chawarski, A. H. Gosline, P. Codd, T. Anor, J. R. Madsen, P. E. Dupont, and J. Lock, "Robotic neuro-endoscope with concentric tube augmentation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 2941–2946, Oct. 2012.

[79] R. J. Webster III, J. M. Romano, and N. J. Cowan, "Mechanics of precurved-tube continuum robots," *IEEE Trans. Robotics*, vol. 25, pp. 67–78, Feb. 2009.

[80] W. Park, J. S. Kim, Y. Zhou, N. J. Cowan, A. M. Okamura, and G. S. Chirikjian, "Diffusion-based motion planning for a nonholonomic flexible needle model," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 4611–4616, Apr. 2005.

[81] R. J. Webster III, J. S. Kim, N. J. Cowan, G. S. Chirikjian, and A. M. Okamura, "Nonholonomic modeling of needle steering," *Int. J. Robotics Research*, vol. 25, pp. 509–525, May 2006.

[82] D. Minhas, J. A. Engh, M. M. Fenske, and C. Riviere, "Modeling of needle steering via duty-cycled spinning," in *Proc. Int. Conf. IEEE Engineering in Medicine and Biology Society (EMBS)*, pp. 2756–2759, Aug. 2007.

[83] K. B. Reed, A. Majewicz, V. Kallem, R. Alterovitz, K. Goldberg, N. J. Cowan, and A. M. Okamura, "Robot-assisted needle steering," *IEEE Robotics and Automation Magazine*, vol. 18, pp. 35–46, Dec. 2011.

[84] N. Abolhassani, R. V. Patel, and M. Moallem, "Needle insertion into soft tissue: a survey," *Medical Engineering & Physics*, vol. 29, pp. 413–431, May 2007.

[85] W. Park, Y. Wang, and G. S. Chirikjian, "The path-of-probability algorithm for steering and feedback control of flexible needles," *Int. J. Robotics Research*, vol. 29, pp. 813–830, June 2010.

[86] D. C. Rucker, J. Das, H. B. Gilbert, P. J. Swaney, M. I. Miga, N. Sarkar, and R. J. Webster III, "Sliding mode control of steerable needles," *IEEE Trans. Robotics*, vol. 29, pp. 1289–1299, July 2013.

[87] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan, "Rrts for nonlinear, discrete, and hybrid planning and control," in *Proc. IEEE Conf. Decision and Control*, pp. 657–663, Dec. 2003.

[88] A. P. Kiraly, J. P. Helferty, E. A. Hoffman, G. Mclennan, and W. E. Higgins, "Three-dimensional path planning for virtual bronchoscopy," *IEEE Trans. Medical Imaging*, vol. 23, no. 9, pp. 1365–1379, 2004.

[89] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 3859–3866, May 2012.

[90] P. J. Swaney, J. Burgner, H. B. Gilbert, and R. J. Webster III, "A flexure-based steerable needle: high curvature with reduced tissue damage," *IEEE Trans. Biomedical Engineering*, vol. 60, pp. 906–909, Apr. 2013.

[91] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J.-C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, J. Buatti, S. Aylward, J. V. Miller, S. Pieper, and R. Kikinis, "3d slicer as an image computing platform for the quantitative imaging network.," *Magnetic Resonance Imaging*, vol. 30, no. 9, pp. 1323–1341, 2012.

[92] M. Fu, A. Kuntz, R. J. Webster, and R. Alterovitz, "Safe motion planning for steerable needles using cost maps automatically extracted from pulmonary images," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 4942–4949, IEEE, Oct. 2018.

[93] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robotics Research*, vol. 30, pp. 846–894, June 2011.

[94] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robotics Research*, 2015.

[95] N. D. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 489–494, May 2009.

[96] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems (RSS)*, June 2013.

[97] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Int. Conf. Automated Planning and Scheduling (ICAPS)*, June 2012.

[98] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 4569–4574, May 2011.

[99] S. Wright and J. Nocedal, *Numerical Optimization*. Springer Science, 1999.

[100] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[101] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robotics and Automation*, vol. 12, pp. 566–580, Aug. 1996.

[102] R. Luna, I. A. Şucan, M. Moll, and L. E. Kavraki, "Anytime solution optimization for sampling-based motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 5068–5074, May 2013.

[103] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robotics Research*, vol. 20, pp. 378–400, May 2001.

[104] J. Denny and N. M. Amato, "Toggle PRM: Simultaneous mapping of C-free and C-obstacle - a study in 2D," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2632–2639, Sept. 2011.

[105] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 2, pp. 1618–1623, Sept. 2004.

[106] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 2421–2428, May 2013.

[107] M. Kobilarov, "Cross-entropy motion planning," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[108] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 2951–2957, May 2015.

[109] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Trans. on Robotics*, vol. 32, pp. 473–483, June 2016.

[110] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *Int. J. Robotics Research*, vol. 33, no. 1, pp. 18–47, 2014.

[111] W. Carriker, P. Khosla, and B. Krogh, "The use of simulated annealing to solve the mobile manipulator path planning problem," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 204–209, May 1990.

[112] R. Geraerts and M. Overmars, "Creating high-quality paths for motion planning," in *Int. J. Robotics Research*, vol. 26, pp. 845–863, 2007.

[113] D. Berenson, T. Siméon, and S. S. Srinivasa, "Addressing cost-space chasms in manipulation planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 4561–4568, May 2011.

[114] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, (Stockholm, Sweden), pp. 4207–4214, May 2016.

[115] L. Li, X. Long, and M. A. Gennert, "BiRRTOpt: A combined sampling and optimizing motion planner for humanoid robots," in *IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids)*, pp. 469–476, Nov. 2016.

[116] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics and Automation Magazine*, vol. 19, pp. 72–82, Dec. 2012.

[117] C. Bergeles and P. E. Dupont, "Planning stable paths for concentric tube robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 3077–3082, Nov. 2013.

[118] L. G. Torres, C. Baykal, and R. Alterovitz, "Interactive-rate motion planning for concentric tube robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1915–1921, May 2014.

[119] M. Hayashibe, N. Suzuki, and Y. Nakamura, "Laser-scan endoscope system for intraoperative geometry acquisition and surgical robot safety management," *Medical Image Analysis*, vol. 10, no. 4, pp. 509–519, 2006.

[120] C. Schmalz, F. Forster, A. Schick, and E. Angelopoulou, "An endoscopic 3d scanner based on structured light," *Medical Image Analysis*, vol. 16, no. 5, pp. 1063–1072, 2012.

[121] S. Leonard, A. Reiter, A. Sinha, M. Ishii, R. H. Taylor, and G. D. Hager, "Image-based navigation for functional endoscopic sinus surgery using structure from motion," in *Medical Imaging 2016: Image Processing*, vol. 9784, p. 97840V, International Society for Optics and Photonics, 2016.

[122] R. Wang, T. Price, Q. Zhao, J.-M. Frahm, J. Rosenman, and S. Pizer, "Improving 3d surface reconstruction from endoscopic video via fusion and refined reflectance modeling," in *Medical Imaging 2017: Image Processing*, vol. 10133, p. 101330B, International Society for Optics and Photonics, 2017.

[123] Q. Zhao, T. Price, S. Pizer, M. Niethammer, R. Alterovitz, and J. Rosenman, "The endoscopogram: A 3d model reconstructed from endoscopic video frames," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, Lecture Notes in Computer Science, pp. 439–447, Oct. 2016.

[124] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[125] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, Oct. 2016.

[126] P. K. Agarwal, K. Fox, and O. Salzman, "An efficient algorithm for computing high-quality paths amid polygonal obstacles," *ACM Transactions on Algorithms (TALG)*, vol. 14, no. 4, p. 46, 2018.

[127] T. Yamamoto, N. Abolhassani, S. Jung, A. M. Okamura, and T. N. Judkins, "Augmented reality and haptic interfaces for robot-assisted surgery," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 8, no. 1, pp. 45–56, 2012.

[128] T. Yamamoto, B. Vagvolgyi, K. Balaji, L. L. Whitcomb, and A. M. Okamura, "Tissue property estimation and graphical display for teleoperated robot-assisted surgery," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 4239–4245, May 2009.

[129] A. Shademan, R. S. Decker, J. D. Opfermann, S. Leonard, A. Krieger, and P. C. W. Kim, "Supervised autonomous robotic soft tissue surgery," *Science Translational Medicine*, vol. 8, no. 337, pp. 337ra64—-337ra64, 2016.

[130] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 3067–3074, May 2015.

[131] K. Trovato and A. Popovic, "Collision-free 6d non-holonomic planning for nested cannulas," in *Proc. SPIE Medical Imaging*, vol. 7261, Mar. 2009.

[132] L. G. Torres and R. Alterovitz, "Motion planning for concentric tube robots using mechanics-based models," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 5153–5159, Sept. 2011.

[133] K. Leibrandt, C. Bergeles, and G.-Z. Yang, "Concentric tube robots: Rapid, stable path-planning and guidance for surgical use," *IEEE Robotics & Automation Magazine*, vol. 24, no. 2, pp. 42–53, 2017.

[134] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, pp. 1–19, 2004.

[135] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[136] C. Bergeles, F. Y. Lin, and G. Z. Yang, "Concentric tube robot kinematics using neural networks," in *Hamlyn Symposium on Medical Robotics*, pp. 1–2, June 2015.

[137] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.