Title

Online Robotic Skill Learning and Adaptation: Integrating Real-time Motion Planning, Model Learning, and Control

Author

Wang, Changhao

Publication Date

2023

Peer reviewed|Thesis/dissertation

Online Robotic Skill Learning and Adaptation: Integrating Real-time Motion Planning, Model Learning, and Control

By

Changhao Wang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Roberto Horowitz
Professor Koushil Sreenath
Professor Pieter Abbeel

Fall 2023

Online Robotic Skill Learning and Adaptation: Integrating Real-time Motion Planning, Model Learning, and Control

Abstract

Online Robotic Skill Learning and Adaptation: Integrating Real-time Motion Planning, Model Learning, and Control

By

Changhao Wang

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Robots are designed to perform tasks with precision and efficiency, often surpassing human capabilities. They not only enhance productivity in manufacturing but also work in areas inaccessible or hazardous to humans. In homes, robotic vacuum cleaners and assistants make daily life more convenient. In healthcare, they assist in complex surgeries and patient care. Moreover, the integration of machine learning and large models further equips robots with intelligence, enabling them to learn from their environments and make informed decisions without additional human commands. However, as those models or policies are learned offline, it is still an open question of whether the robots can reliably interact in the new scenarios online. To address this question, this dissertation delves into the development of efficient and robust methods that empower robots to learn and plan manipulation trajectories in real-time, and to self-adjust their skills based on live sensor feedback.

The core of this dissertation is anchored in three principal challenges:

Efficient Trajectory Optimization: Model predictive control (MPC) plays an important role in online robotic planning. In these scenarios, a robot will optimize its future actions based on the dynamics model. Therefore, the development of efficient formulations and algorithms for MPC and trajectory optimization is crucial for real-time robot learning and skill adaptation. In Part I of the dissertation, we introduce algorithms and formulations tailored for real-time robotic trajectory optimization. Our focus is on enhancing the computational speed of the optimization process while ensuring the resulting trajectory maintains high quality. These advancements can equip robots with the ability to swiftly plan and adjust their movements in novel environmental conditions.

Adaptive Model Learning for Deformable Object Manipulation: Building on the efficient MPC formulation presented in Part I, the second part of the dissertation shifts its focus to the real-time enhancement of robotic behaviors. This enhancement involves dynamically adjusting

the robot's dynamics model based on real-world sensor data. Part II specifically addresses online model learning for the manipulation of deformable objects, such as ropes and fabrics. These tasks, characterized by their complex and non-linear state changes, present substantial challenges for model-based methods, yet are critically important for everyday applications. In this part of the dissertation, we introduce algorithms capable of learning the dynamics of these objects and effectively manipulating them. Our research emphasizes the integration of real-time sensory feedback with machine learning algorithms. This synergy is crucial for the continuous refinement and updating of the dynamics model. We illustrate how such real-time adjustments are vital for enabling robotic systems to modify their manipulation strategies effectively.

Real-time Control Policy Adaptation for Contact-Rich Manipulation Tasks: While the MPC and adaptive model learning can provide robots with more reliable trajectories, another crucial aspect for real-time execution is low-level force control, which calculates the motor torque for reliable environmental interaction. Part III of the dissertation concentrates on optimizing control policies for robotic systems engaged in contact-rich manipulation tasks such as assembly, pivoting, and screwing. The main emphasis is on the ability of robotic systems to dynamically adjust their control policies in response to force and torque feedback. This level of adaptation is critical for the successful completion of tasks requiring delicate and precise physical interactions. By doing so, we demonstrate the improvement of the robot's motion under massive contacts.

To My Family

Contents

List of Figures

List of Tables

Acknowledgments

It's been an incredible five and a half years at Berkeley. This journey has been about learning, discovery, and meeting amazing people.

I'm extremely thankful to my Ph.D. advisor, Professor Masayoshi Tomizuka. His support was key to my dissertation. He was there for me during moments of uncertainty and placed his trust in me from the beginning. Professor Tomizuka has been much more than just an academic mentor; he's been a role model in both my professional and personal growth. My journey as his Ph.D. student has been about more than just learning and research; it's been a path toward becoming a better person.

I'm grateful to Professor Roberto Horowitz, Professor Koushil Sreenath, and Professor Pieter Abbeel for their roles on my dissertation committee. Thanks also to Professor Kameshwar Poolla, Professor Somayeh Sojoudi, Professor Koushil Sreenath, and Professor Mark Mueller for their insights during my qualifying exam. Their advice greatly influenced my studies and life at Berkeley.

Special thanks to the Chin Leung Shui Chun Fellowship, the ME department fellowship, and the FANUC Corporation for funding my research. The discussions with researchers at FANUC Advanced Research Laboratory, especially Mr. Tetsuaki Katou, Mr. Kaimeng Wang, Dr. Yongxiang Fan, Dr. Yu Zhao, Dr. Hsien-chung Lin, and Dr. Te Tang, have been insightful throughout these years.

I'm also deeply thankful for the opportunity to spend two incredible summers at Google X, working on the Everyday Robots Project alongside Dr. Jeffrey Bingham. His extensive knowledge, insightful vision, passion for work, and sense of humor have greatly inspired me.

A big thanks to my collaborators for their invaluable contributions and inspiring discussions. Special thanks go to Dr. Te Tang, Dr. Hsien-chung Lin, Dr. Liting Sun, Dr. Jeffrey Bingham, Dr. Yu Sun, Dr. Shiyu Jin, Xiang Zhang, Xinghao Zhu, Linfeng Sun, Zheng Wu, Zhian Kuang, Yuyou Zhang, Mingrui Yu, Wu-Te Yang, Ting Xu, and Joohwan Seo. Their insights have significantly enriched my research experience.

I am also grateful to my friends at the Mechanical System Controls (MSC) lab. The support from both current and former lab members has been immense. I'd like to acknowledge Professor Changliu Liu, Dr. Daisuke Kaneishi, Dr. Wei Zhan, Professor Jianyu Chen, Dr. Zining Wang, Dr. Kiwoo Shin, Professor Jiachen Li, Dr. Yeping Hu, Dr. Zhuo Xu, Dr. Yujiao Cheng, Dr. Saman Fahandezhsaadi, Dr. Jessica Leu, Dr. Chen Tang, Dr. Yiyang Zhou, Dr. Huidong Gao, Dr. Hengbo Ma, Jining Li, Catherine Weaver, Ran Tian, Chengfeng Xu, Akio Kodaira, Chenran Li, Qiyang Qian, Jen-Wei Wang, Yichen Xie, Wei-Jer Chang, Keita Kobashi, Yiheng Li, Yuxin Chen, Boyuan Liang, Yixiao Wang, Chensheng Peng, Shuqi Zhao, Yutaka Shimizu, and Mingyu Ding for their support. I would also like to express my appreciation to all the friends I met at Berkeley, Google, and HRI. The time we spent together was truly memorable and enjoyable.

In the end, my deepest love goes to my parents and my girlfriend Xinyu Li for your unconditioned love, support, and encouragement.

Chapter 1

Introduction

1.1 Background and Motivations

The evolution of robots from pre-programmed machines to sophisticated, decision-making systems marks a significant leap in robotics. Today, robots are vital in various aspects of our lives, notably in manufacturing, healthcare, and logistics. The integration of machine learning algorithms into these systems has further expanded their capabilities, enabling robots to autonomously interpret and adapt to their surroundings without the need for continuous human supervision. This level of autonomy is especially crucial in our daily lives.

A main challenge in the current landscape of robotics is enabling real-time decision-making and adaptability. While significant progress has been made in offline learning of models and policies within controlled settings, it is still challenging for robots to reliably interact with new scenarios. The critical gap is the robot's ability to not just execute pre-learned models but to actively make decisions, plan, and adapt their actions in real-time in response to new and unforeseen situations.

Take the peg-in-hole insertion task as an example: the objective is to accurately align a peg with a hole and then apply the appropriate force to insert it fully. This task poses considerable challenges for robotic manipulators both from a planning and control perspective. Firstly, it involves determining the manipulation trajectory to complete the task. Secondly, it requires the generation of precise low-level force/torque commands for safe and efficient interaction. Although data can be pre-collected and used to train the robot offline, this alone is insufficient for handling the variability and unpredictability encountered in real-world scenarios. Offline training data might not encompass all possible variations in alignment, force application, and environmental conditions the robot will encounter. Therefore, the robot must be capable of online learning and adaptation. It needs to learn from each insertion attempt, refining its trajectory planning and force control strategies in real time. This approach not only enhances the robot's performance in repetitive tasks but also equips it with the flexibility to handle new, unanticipated scenarios effectively.

This dissertation is motivated by the need to address this gap. It focuses on developing

methodologies that enable robots to perform online planning and make decisions efficiently. The objective is to advance robotic capabilities beyond robust offline learning, towards dynamic, real-time adaptability in various operational contexts. The emphasis is on enabling robots to not only follow predetermined paths but to intelligently navigate, adjust their strategies, and learn from their environment.

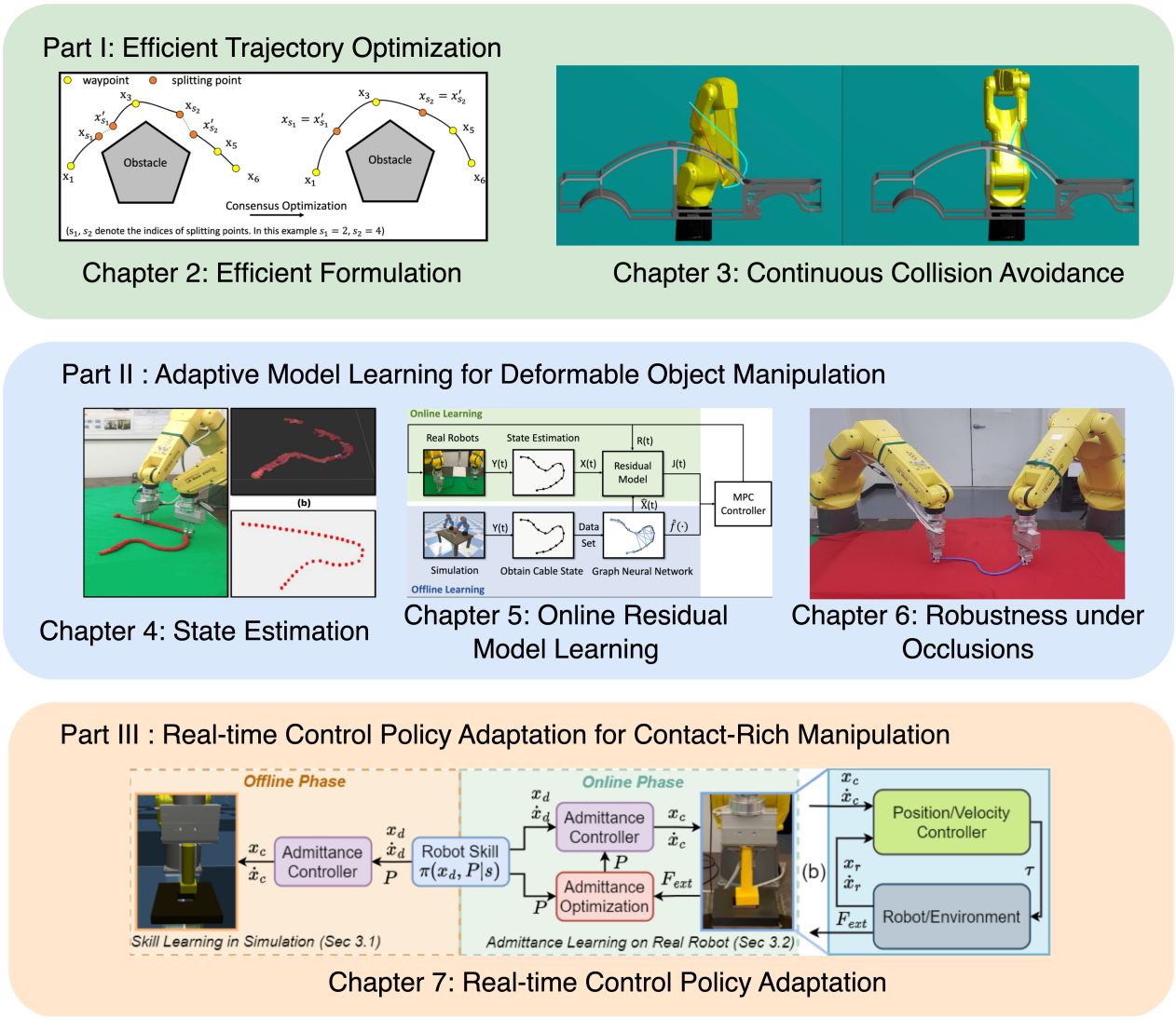


Figure 1.1: Overview of the dissertation. This dissertation presents methodologies aimed at enabling robots to interact reliably with their environment and objects in real time. Our focus was distributed across three key aspects: robotic motion planning (Chapters 2 and 3), adaptive model learning (Chapters 4, 5, and 6), and online control policy adaptation (Chapter 7).

1.2 Dissertation Outlines

The overall goal of this dissertation is to develop methodologies that enable robots to adapt in real-time to novel scenarios. Specifically, we focus on three main challenges: efficient robotic motion planning, online model learning for manipulating deformable objects, and real-time control policy adaptation for contact-rich manipulation tasks.

Part I: Efficient Trajectory Optimization

Robotic motion planning that meets dynamics, kinematics, and collision avoidance constraints is fundamental for autonomous robotic systems. Developing an efficient algorithm to compute optimal trajectories is crucial for robots operating in new and unstructured environments. This part of the dissertation discusses methodologies developed for effective trajectory optimization.

Efficient Trajectory Optimization via Trajectory Splitting

Achieving efficient trajectory optimization, especially in unstructured environments, is challenging due to the need for both speed and quality in the solutions. A significant factor is that second-order optimality demands Hessian matrix calculations, which grow quadratically with the number of waypoints. Reducing waypoints can decrease computation time but often at the cost of trajectory quality, potentially leading to collisions. Our approach, inspired by recent consensus optimization studies, proposes a distributed formulation of collocated trajectory optimization. It segments a long trajectory into smaller parts, each solved in parallel and then unified into a single trajectory with consensus constraints. This method distributes the quadratic complexity and enables the creation of high-quality trajectories with denser waypoints. We demonstrate the improved efficiency of this method against leading motion planning algorithms. Part of this work was published in [102].

Bilevel Trajectory Optimization Formulation for Efficient Collision Avoidance

Considering the collision along the continuous trajectory is extremely challenging. Standard formulations focus on individual waypoints, but a comprehensive approach must account for potential collisions along the entire path. This dissertation introduces a bilevel path optimization formulation (BPOMP), which constrains the closest position along the path to the obstacles. If this position is free of collision, the entire path is likely collision-free. We formulate this as a bilevel optimization problem, then simplify it to nonlinear programming (NLP). Experimental and simulation comparisons validate the effectiveness of BPOMP. Part of this work was published in [103].

Part II: Adaptive Model Learning for Deformable Object Manipulation

In addition to classic motion planning problems, interacting with various objects is another essential challenge in many robotic applications, and having an accurate dynamics model is beneficial for achieving manipulation skills. However, obtaining and aligning these models with real-world objects is challenging. This section explores online learning and refinement of dynamics models for better object manipulation.

A Unified Framework for Deformable Objects State Estimation and Task Planning

The manipulation of deformable objects like ropes or clothes involves complexities due to their infinite-dimensional configuration spaces. In Chapter 4, we introduce a comprehensive framework using Coherent Point Drift (CPD) for state estimation and we utilize it for task planning. This approach effectively captures the dynamic states of deformable objects by aligning observed data points with a reference model in simulation, enabling precise real-time tracking even under occlusions. This accuracy is crucial for robotic systems to adapt their manipulation strategies dynamically, especially in scenarios involving significant deformation or movement. The CPD-based state estimation method is not only robust in various manipulation contexts but also essential for achieving high precision and reliability in tasks requiring complex interactions with deformable materials. Part of this work was published in [94, 30].

Offline-online Learning of Deformation for Robotic Cable Manipulation

Precisely manipulating deformable linear objects like cables has applications in manufacturing and surgery. An accurate model predicting deformation is crucial for control. We propose a hybrid offline-online method to learn cable dynamics using a Graph Neural Network (GNN) for simulation data and a real-time linear residual model to bridge the sim-to-real gap. This model is integrated into a Model Predictive Controller (MPC) for optimal robot movement. Comparative results show the method's effectiveness and robustness. Part of this work was published in [104].

Robust Deformation Model Learning under Uncertainties

In Chapter 6, we expand upon the methodologies discussed in previous chapters, addressing more complex scenarios characterized by significant sensor noise and extensive occlusion. We introduce a novel framework, SPR-RWLS, that synergizes the real-time cable tracking algorithm from Chapter 4 with the online model learning methods outlined in Chapter 5. This integration is designed to enhance the robustness of object manipulation under challenging conditions. Specifically, for cable tracking, we employ Structure Preserved Registration (SPR) to accurately estimate the movement of key points on a cable, even amidst sensor

noise, outliers, and occlusions. Subsequently, Robust Weighted Least Squares (RWLS) is utilized to compute the local deformation model of the cable, factoring in various uncertainties. This combined SPR-RWLS approach significantly improves the ability of dual-arm robots to manipulate cables of different thicknesses and lengths into various desired curvatures across multiple scenarios. Moreover, we demonstrate that the real-time implementation of the SPR-RWLS method can be streamlined through parallel computation, enhancing its efficiency and applicability. Part of this work was published in [28].

Part III: Real-time Control Policy Adaptation for Contact-Rich Manipulation Tasks

In addition to the manipulation tasks that do not require extensive force interactions, contact-rich manipulation, found in tasks like assembly and grasping, requires precise control over manipulation trajectories and force parameters. Incorrect parameters can cause damage due to oscillations or excessive forces. This section addresses the challenge of online control adaptation for efficient and safe contact-rich manipulation.

Efficient Control Policy Adaptation with Online Admittance Residual Learning

Learning contact-rich manipulation skills is vital, but challenging due to data inefficiencies and the sim-to-real gap. We introduce a hybrid offline-online framework for robust skill learning, using model-free reinforcement learning in the offline phase for motion and compliance control parameters. In the online phase, we adapt these parameters in real time based on force sensor data. We provide comparative results to demonstrate our approach's effectiveness in tasks such as table wiping, assembly, pivoting, and screwing. Part of this work was published in [105, 114].

1.3 Summary of Contributions

This dissertation's contributions are summarized as follows:

- Part I: Efficient Trajectory Optimization
  - Chapter 2 introduces a distributed optimization algorithm that segments the trajectory into smaller parts, effectively distributing problem complexity for enhanced computational efficiency.
  - Chapter 3 proposes a bilevel optimization formulation for motion planning, focusing on efficiently resolving potential mid-waypoint collisions.
- Part II: Adaptive Model Learning for Deformable Object Manipulation

  - Chapter 4 presents a comprehensive framework using non-rigid registration to estimate the states of deformable objects and devise corresponding manipulation strategies.
  - Chapter 5 details a hybrid offline-online learning approach, combining an offline Graph Neural Network (GNN) model with online residual learning to accurately predict dynamics for manipulating deformable linear objects.
  - Chapter 6 advances these concepts into more unstructured environments, addressing challenges like significant sensor noise and occlusions.

- Part III: Real-time Control Policy Adaptation for Contact-Rich Manipulation Tasks
  - Chapter 7 introduces an online admittance learning approach, optimizing the residual of admittance control parameters in real time. This method adapts previously learned manipulation policies to new scenarios efficiently.

Part I

Efficient Trajectory Optimization

Chapter 2

Efficient Trajectory Optimization via Trajectory Splitting

As discussed in Chapter 1, to enable real-time decision-making capability, one of the fundamental problems is to develop efficient trajectory optimization algorithms that can let the robot navigate through constrained environments and avoid obstacles. In this chapter, we introduce a novel trajectory optimization algorithm designed to decompose a complex, long trajectory planning problem into several segments. Each sub-trajectory is processed in parallel, enhancing computational efficiency. Then the solutions are fused into a single trajectory with a consensus constraint that enforces continuity of the segments through a consensus update. With this scheme, the complexity is distributed to each segment and enables solving for higher-quality trajectories much more efficiently.

2.1 Introduction

Finding optimal, collision-free trajectories is important for robots to interact with people and the environment. Sampling and optimization are two of the most powerful ways to achieve the goal. Optimization allows defining the problem in terms of constraints and finding solutions that optimize performance. Currently, collocation-based optimization methods, such as TrajOpt [80], solve a non-linear program (NLP) using non-linear optimization algorithms like sequential quadratic programming (SQP) [7]. The structure of the NLP leads to at least $O(N^2)$ time complexity concerning the number of waypoints to satisfy second-order optimality criteria. Therefore, increasing the density of waypoints quadratically increases the computation time. In comparison, sampling-based planners, such as rapidly exploring random trees (RRT) [39], are able to be parallelized to achieve a higher computation efficiency.

Our goal is to combine the expressiveness and quality of optimization-based techniques with the computational advantages of parallelization in sampling-based methods. We propose a method that separates the trajectory optimization problem into a set of subproblems

that can be solved in a distributed manner.

Prevailing methods for parallel or distributed optimization mainly focus on multi-agent planning [77, 44] with limited research applying the techniques to single-agent planning. Brendan. et al [68] pioneered in this field by applying the operator splitting method to convex optimal control problems. They proposed splitting an optimal control problem into subproblems by constraint. This idea is further studied in [87] and applied to more challenging control scenarios. If it were applied to a collision-avoiding trajectory problem for a robot, two subproblems would be created. One subproblem would satisfy only the robot dynamics and the other would only avoid the collision. These problems would then be fused using an iterative consensus update to find the optimal trajectory. However, even with this splitting scheme, the number of variables for each subproblem remains the same, and it still suffers from the $O(N^2)$ time complexity. Furthermore, splitting by constraint may create subproblems with different complexity. Some subproblems may dominate the computation time, and others should wait until the most complicated subproblem is finished in order to begin the next consensus iteration.

To deal with those problems, we propose to split the problem by the path variables, creating subproblems for segments of the trajectory. In this way, the problem complexity can be equally distributed. These segments are then fused together using a consensus update scheme similar to the method described above. This proposed method of "trajectory splitting" exploits the observation that complexity decreases as path length decreases and in this way offers a more efficient approach to solving trajectory optimization for certain classes of problems. Moreover, the proposed splitting scheme is amenable to be incorporated with any existing trajectory optimizers to solve the subproblem. To the authors' knowledge, this is the first approach to parallelizing optimization-based trajectory planning by splitting the path variables in order to equally distribute the problem complexity to each trajectory segment. The contributions of the proposed method are listed as follows:

- A novel formulation of trajectory optimization problems via splitting the trajectory into segments that can equally distribute the problem complexity.
- A distributed optimization algorithm to solve the proposed formulation for better computational efficiency.
- An implementation of the trajectory splitting algorithm with a state-of-the-art collision checker and optimization solver.
- Comprehensive comparisons of the proposed trajectory splitting algorithm against existing methods using both simulation and real-world experiments.

2.2 Related Works

Motion planning algorithms are nominally classified into two broad categories, sampling, and optimization. Sampling methods are well suited for problems where any feasible solution

is acceptable and gradient information is difficult or expensive to compute. Probabilistic roadmap (PRM) [33] solves the planning problem by constructing a complete map of the workspace and then using a search algorithm to find a feasible path from the map. One major problem of PRM is the computation time. It is time-consuming to build a complete roadmap, especially in high-dimensional space. RRT [39] deals with this problem by incrementally building a graph and checking feasible paths at the same time. In practice, RRT and its variants [37, 39] are still some of the most powerful ways to deal with the planning problem. They are efficient and can easily be parallelized [39] for even better performance. However, since sampling-based planners are stochastic, they may find different solutions for the same problem, and the solution quality and the computation time may also have a large variance. This problem has been reported in several papers [123, 56]. It is still an active research area to improve the robustness of sampling planners.

Though sampling planners are effective in finding feasible paths, it is often preferable to obtain 'optimal' paths that satisfy an objective. RRT* [39], an optimal variant of traditional RRT, can obtain optimal paths with the help of an additional 'rewire' operation. Exploiting the probabilistic completeness property provides a theoretical guarantee of globally optimal solutions. However, in practice, the planner is only given finite time to find a path and thus, RRT* performance declines sharply as the dimensionality of the scenario increases [56].

Optimization, on the other hand, provides a way to go beyond finding a feasible solution and offers a means to seek 'better' solutions based on an objective. State-of-the-art trajectory optimization algorithms start with an infeasible solution and evolve a trajectory to minimize a defined cost and satisfy all constraints. CHOMP [123] pioneered this approach for collision avoidance planning problems in robotics, proposing a covariant gradient descent update rule to optimize the trajectory. To deal with non-differentiable constraints, STOMP [32] proposed a stochastic update rule. TrajOpt [80] introduced an SQP formulation to solve the planning problems. Recently, there has been considerable progress in this field [61, 49, 90], and these efforts suggest new approaches to optimization-based planning have the potential to be both computationally efficient and retain high-quality solutions.

2.3 Mathematical Background

In this section, we introduce the preliminaries of the proposed method. The classic trajectory optimization formulation by collocation is introduced in Section 2.3. The mathematical background of consensus optimization and alternating direction method of multipliers (ADMM) is illustrated in Section 2.3.

Trajectory Optimization Formulation

A common discrete form of trajectory planning can be formulated as an optimization problem with the following form:

$$\begin{aligned} \min_{\tau} \quad & c(\tau) \\ \text{s.t.} \quad & x_{i+1} = f(x_i) \\ & g(x_i) \leq 0 \\ & i = 1, \cdots, N-1 \end{aligned} \tag{2.1}$$

where $\tau = \{x_1, \cdots, x_N\}$ denotes a discretized robot trajectory, and x_i denotes the robot state (position, velocity, and acceleration). $c(\cdot)$ is a human designed cost function, $f(\cdot)$ may include the robot kinematics or dynamics constraints, and $g(\cdot)$ is the robot state constraint. In this chapter, we assume the cost function and constraints are separable (or block-separable) $c(\tau) = \sum_{i=1}^{N} c_i(x_i)$.

ADMM and Consensus Optimization

Consensus optimization [8] considers the problem with separable objectives:

$$\begin{aligned} \min_{x_1, \cdots, x_N, z \in \mathbb{R}^N} \quad & \sum_{i=1}^{N} c_i(x_i) \\ \text{s.t.} \quad & x_i = z \quad i = 1, \cdots, N \end{aligned} \tag{2.2}$$

where x_i is called a local variable, and z is a global value that each local variable tries to achieve.

ADMM [8], an augmented Lagrangian method [6], is able to parallelize the consensus problem and solve it efficiently. The augmented Lagrangian of (2.2) is shown in (2.3), where y_i denotes the Lagrange multiplier of the corresponding consensus constraint. Similar to penalty methods, the augmented Lagrangian adds an additional constraint term to the original Lagrangian in order to penalize the constraint violation, and ρ is a weight that controls the constraint violation.

$$L = \sum_{i=1}^{N} [c_i(x_i) + y_i^T (x_i - z) + (\rho/2)\|x_i - z\|^2 \tag{2.3}$$

Based on the augmented Lagrangian in (2.3), ADMM can solve the original problem in a distributed way as shown in Fig. 2.1. First, ADMM initializes N separate agents (solver) in order to update each local variable x_i. Then, a central unit collects the solution from the

agent and updates the dual variable accordingly y_i. The update rule is then given by:

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i} \quad [c_i(x_i) + y_i^{kT}(x_i - z^k) + (\rho/2)||x_i - z^k||^2] \\ y_i^{k+1} &= y_i^k + \rho(x_i^{k+1} + z^{k+1}) \\ z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N} x_i^{k+1} \end{aligned} \tag{2.4}$$

The convergence and optimality condition of ADMM is illustrated in Theorem 1. For the convex consensus optimization problem, a globally optimal solution is guaranteed.

The ADMM algorithm has also been extensively applied to non-convex, coupling optimization problems [106, 50]. Recently, convergence criteria for general non-convex problems have been studied and we refer readers to [106] for further details. We will discuss convergence for our implementation and similar problems in Section 2.4. In summary, ADMM algorithms show amazing practical success in solving both convex and general nonlinear optimization problems even while the theoretical proof of convergence is forthcoming.

Assumption 1. *$c_i(x), i = 1, \cdots, N$ are closed, proper, and convex.*

Assumption 2. *The augmented Lagrangian L_0 contains a saddle point*

Theorem 1. *Under Assumptions 1 and 2, the ADMM iteration satisfies the following [8]:*

- *Residual convergence: $x_i - z \to 0$ as $k \to \infty$*
- *Objective convergence: $\sum_{i=1}^{N} c_i(x_i) \to f^*$ as $k \to \infty$*
- *Dual convergence: $y_i^k \to y_i^*, i = 1, \cdots, N$ as $k \to \infty$*

It is worth noticing that (2.2) is a simplified form of the consensus problem. For the general form [8], each local variable, or even each element, can have its own consensus constraints. The above update rule can be easily transformed to the general case by replacing the consensus constraint, and the optimality and convergence analysis still hold for the general formulation.

2.4 Trajectory Splitting Algorithm

Trajectory optimization aims at finding a smooth and collision-free trajectory between two predefined states. Similar to previous methods [80], we formulate the motion planning problem as an NLP as introduced in Section 2.3. While in contrast with other methods, which directly apply optimization algorithms (such as, covariant gradient descent or SQP) to solve the entire problem, we propose splitting the trajectory into several segments and solving them in a distributed manner for better computational efficiency. The splitting formulation as a consensus optimization will be introduced in Section 2.4, 2.4. The collision

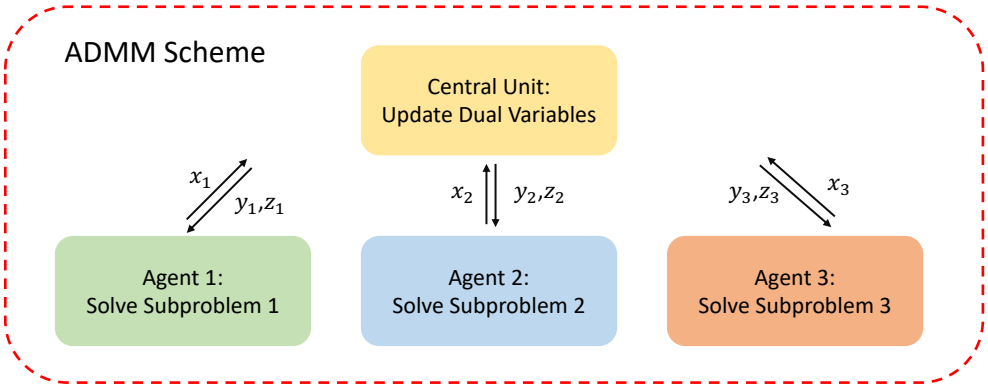


Figure 2.1: An illustration of the ADMM update rule to solve consensus problems. ADMM solves the problem in a distributed manner, where it creates N separate agents to solve each subproblem. A central unit then collects the results obtained from each agent and updates the dual variable until a stopping criterion is met.

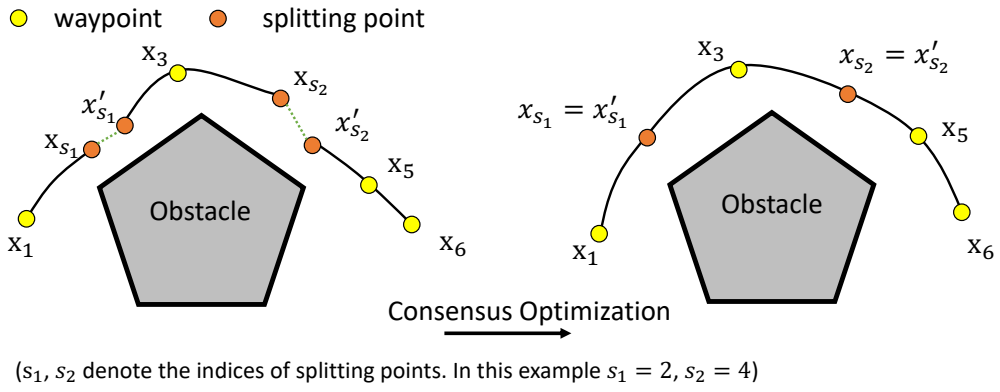


Figure 2.2: Illustration of trajectory splitting algorithm. The waypoints are denoted by the yellow dots, and the orange dots are the splitting points x_{s_i}. The trajectory splitting algorithm is able to find feasible trajectory pieces in parallel and then connect them together.

avoidance constraints will be formalized in Section 2.4. The optimization update rule will be explained in Section 2.4. In the end, we provide the stopping criterion and convergence analysis of the proposed algorithm.

Intuition for Trajectory Splitting: A Three Waypoint Example

Let us begin with an example with three waypoints x_1, x_2, x_3. Consider we want to split the trajectory at x_2 creating a leading trajectory of $\tau_1 = \{x_1, x_2\}$, and a trailing trajectory of $\tau_2 = \{x_2', x_3\}$. Here x_2' is a slack variable that is defined to be equal to x_2. According to the consensus formulation in (2.2), in order to achieve the consensus between x_2 and x_2', a global variable z is introduced to enforce this constraint. Therefore, we can rewrite the three-waypoint trajectory optimization problem in (2.1) as follows:

$$\begin{aligned} \min_{x_1,x_2,x_2',x_2,z} \quad & c_1(x_1) + \frac{1}{2}c_2(x_2) + \frac{1}{2}c_2(x_2') + c_3(x_3) \\ \text{s.t.} \quad & x_2 = z, \quad x_2' = z \\ & x_2 = f(x_1), \quad x_3 = f(x_2') \\ & g(x_i) \leq 0, \quad i = 1, 2, 3 \\ & g(x_2') \leq 0 \end{aligned} \tag{2.5}$$

Notice that the problem is separable between these two trajectory pieces, except for the first two consensus constraints. According to the consensus optimization update rule in (2.4), each trajectory segment τ_1, and τ_2 can be updated by solving the following optimization where the dual variable update follows the same manner as in (2.4).

$$\begin{aligned} \min_{x_1,x_2} \quad & c_1(x_1) + \frac{1}{2}c_2(x_2) + y_1^T(x_2 - z) + (\rho/2)\|x_2 - z\|^2 \\ \text{s.t.} \quad & x_2 = f(x_1) \\ & g(x_1) \leq 0, \quad i = 1, 2 \end{aligned} \tag{2.6}$$

$$\begin{aligned} \min_{x_2',x_3} \quad & \frac{1}{2}c_2(x_2') + c_3(x_3) + y_2^T(x_2' - z) + (\rho/2)\|x_2' - z\|^2 \\ \text{s.t.} \quad & x_3 = f(x_2') \\ & g(x_2') \leq 0 \\ & g(x_3) \leq 0 \end{aligned} \tag{2.7}$$

Remark 1. *Constraints can be incorporated into the objective function via indicator functions. Therefore, it does not affect the update rule for the ADMM formulation.*

General Trajectory Splitting Formulation

Consider we split the trajectory into $M + 1$ segments, and the position that splitting happens is denoted by x_{s_i}, where s_i is the index of the i th splitting point on the original

Algorithm 1: Trajectory Splitting

Require: Trajectory: τ_i, Dual variables: $y_{i,1}$, $y_{i,2}$

```
while splitting tolerance (2.16) not satisfied do
    for i = 1, ··· , M + 1 do
        τ_i ← Solve Eq. 2.10
    end for
    for i = 1, ··· , M do
        z_i = ½(x_{s_i} + x'_{s_i})
        y_{i,1} = y_{i,1} + ρ(x_{s_i} − z_i)
        y_{i,2} = y_{i,2} + ρ(x'_{s_i} − z_i)
    end for
end while
return τ_i   i = 1, ··· , M + 1
```

$$
\begin{aligned}
L_1 &= \sum_{j=1}^{s_1-1} c_j(x_j) + \frac{1}{2}c_{s_1}(x_{s_1}) + y_{1,1}^T(x_{s_1} - z_1) + (\rho/2)\|x_{s_1} - z_1\|^2 \\
&\cdots \\
L_i &= \frac{1}{2}c_{s_{i-1}}(x'_{s_{i-1}}) + \sum_{j=s_i+1}^{s_{i+1}-1} c_j(x_j) + \frac{1}{2}c_{s_i}(x_{s_i}) + y_{i-1,2}^T(x'_{s_{i-1}} - z_{i-1}) + (\rho/2)\|x_{s_{i-1}} - z_{i-1}\|^2 + y_{i,1}^T(x_{s_i} - z_i) + (\rho/2)\|x_{s_i} - z_i\|^2 \\
&\cdots \\
L_M &= \sum_{j=s_M+1}^{N} c_j(x_j) + \frac{1}{2}c_{s_M}(x'_{s_M}) + y_{M,1}^T(x'_{s_M} - z_M) + (\rho/2)\|x'_{s_M} - z_M\|^2
\end{aligned}
\tag{2.9}
$$

trajectory as shown in Fig. 2.2. Similar to the previous example, we introduce the slack variable x'_{s_i} as a copy of the splitting point. In order to enforce that the trajectories are connected with each other, a global variable z_i is used for this constraint:

$$
\begin{aligned}
x_{s_i} &= z_i \\
x'_{s_i} &= z_i, \quad i = 1, \cdots, M
\end{aligned}
\tag{2.8}
$$

and we use $y_{i,1}$ and $y_{i,2}$ to denote the Lagrangian multipliers of the consensus constraints above. Thus, we can formulate the augmented Lagrangian L_i for each trajectory piece as shown in (2.9).

Then the primal update rule for each trajectory segment is given in (2.10).

$$
\begin{aligned}
\tau_i^{k+1} = \min_{\tau_i} \quad & L_i \\
\text{s.t.} \quad & x_{j+1} = f(x_j) \\
& g(x_j) \leq 0 \\
& \forall x_j \in \tau_i
\end{aligned}
\tag{2.10}
$$

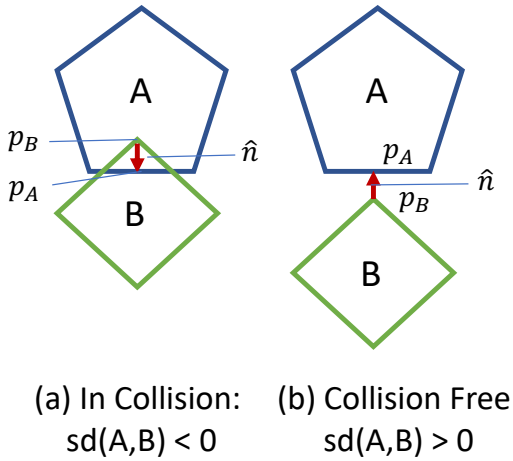


Figure 2.3: Illustration of signed distance function. The signed distance value is positive if objects are collision-free, and the value is negative if they are in collision.

The overall algorithm is illustrated in Alg. 1. In practice, the trajectory τ can be initialized as a straight line that goes from the initial point to the target point, or it can be given as a feasible solution from sampling-based planners. The Lagrange multipliers $y_{i,1}$, $y_{i,2}$ can be initialized as zero vectors. The number of splitting M could be any integer that is smaller than the number of waypoints, and the waypoints are then uniformly separated into each subproblem.

Collision Avoidance Constraints

We consider collision-avoidance constraint $g(x_i) \leq 0$ as a function of signed distance [80]. As shown in Fig. 2.3, the signed distance function denotes the minimum distance between two objects, where the sign is determined by whether the surfaces of the objects interpenetrate. We mathematically define the signed distance function between two objects A and B by:

$$\text{sd}(A, B) = \text{dist}(A, B) - \text{penetration}(A, B) \tag{2.11}$$

where the 'dist' function is defined as the minimum translation distance to just cause contact between the surfaces of a pair of objects:

$$\text{dist} = \inf\{\|T\| : \exists p_A \in A, p_A + T \in B\} \tag{2.12}$$

and similarly, the 'penetration' function denotes the minimum translation that moves the two objects out of contact:

$$\text{penetration} = \inf\{\|T\| : \forall p_A \in A, p_A + T \notin B\} \tag{2.13}$$

In our implementation, we approximated the signed distance using methods from the Flexible Collision Library (FCL) [70]. In turn, this signed distance method was used to formulate a collision-avoidance constraint where the signed distance between the robot and each obstacle had a strictly positive value along the entire trajectory. An additional approximation was adopted by utilizing the Jacobian for the contact point from the signed distance, as introduced in [80]. With this technique, assume the object A is a robot link and its position is determined by the robot state x. Then, the signed distance is defined by contact points denoted by p_A for the robot link and p_B for an obstacle. A static assumption is made to arrive at the approximation given in (2.14) by assuming the contact point p_A is not a function of x:

$$sd_{AB}(x) \approx \hat{n}(F_A^w(x)p_A - F_B^w p_B) \tag{2.14}$$

where F_A^w and F_B^w are the homogeneous transformation from A, and B frames to the world coordinate, and $\hat{n}$ is the direction that points from p_B to p_A.

In addition, the gradient of this constraint can be computed as a Jacobian of the signed distance sd_{AB} with the following approximation:

$$\nabla \mathrm{sd}_{AB}(x) \approx \hat{n}^T J_{p_A}(x) \tag{2.15}$$

where J_{p_A} denotes the robot translation Jacobian at p_A.

Optimization

The key to this algorithm is the separation of the trajectory optimization problem into several subproblems that can be solved independently. For each subproblem in (2.10), it is expressed in a standard form for trajectory optimization; therefore, existing optimization algorithms, such as gradient descent, SQP [7], CHOMP [123], and TrajOpt [80] can be directly applied.

In our implementation, we use the optimization library IPOPT [100] as the basis of our subproblem solver. IPOPT implements a primal-dual interior-point filter line search algorithm to solve the general nonlinear optimization problems, and the convergence of this algorithm is proved in [99].

Stopping Criterion and Convergence Analysis

Similar to [120], we choose to use the primal residuals as the stopping criterion:

$$\|r^k\|_2 \leq \epsilon \tag{2.16}$$

where ϵ is a predefined positive scalar, and splitting tolerance r^k is the average of constraint violations:

$$\|r^k\|_2 = \frac{1}{M}(\sum_{i=1}^{M} \|q_{s_i}^k - {q'_{s_i}}^k\|^2)^{\frac{1}{2}} \tag{2.17}$$

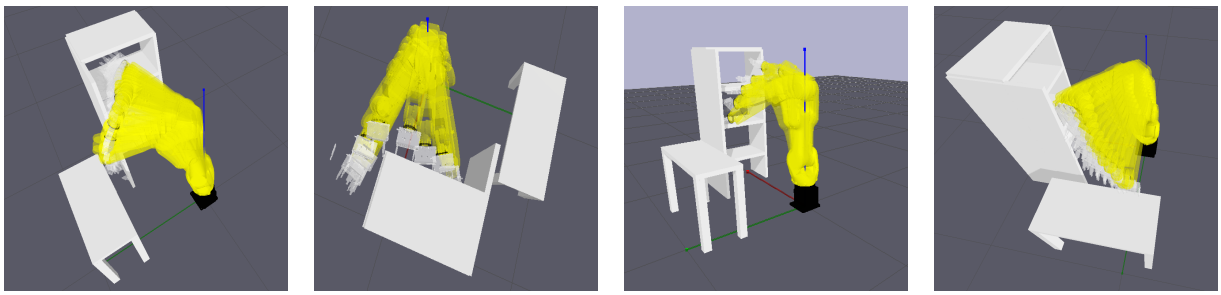


Figure 2.4: Simulation benchmark results showing four different planning problems and the results obtained by the trajectory splitting algorithm in the bookcase scenario. The algorithm can find the local optimal solution for each trajectory segment efficiently and connect them together in a smooth trajectory.

The convergence of the proposed trajectory splitting algorithm can be divided into three cases:

- **Convex**: According to Theorem 1, if the objective functions and constraints are convex (the equality constraint should be linear), the proposed trajectory splitting algorithm is guaranteed to converge to a globally optimal solution. An example of this type of problem is the linear quadratic regulators (LQR). For the LQR problem, the trajectory splitting algorithm is guaranteed to find a global optimal solution.

- **Restricted prox-regular[106]**: Recent studies provide the convergence of ADMM on non-convex, non-smooth problems. [106] proves the condition that if the objectives and constraints satisfy the restricted prox-regular condition, then the ADMM algorithm will converge to a stationary point. In our scenario, if the dynamics constraint $f(\cdot)$ is linear, and the nonlinear collision constraint $g(\cdot)$ is restricted prox-regular, then the solution (τ, y, z) obtained from Alg. 1 will converge to a stationary point (τ^*, y^*, z^*) for the augmented Lagrangian in (2.9). According to [106], restricted prox-regular is a weaker condition than prox-regular and semi-convex[16]. As studied in [49], the signed distance function between two convex objects is semi-convex; therefore, a large set of trajectory optimization problems fall into this category. This includes mobile robot planning with convex obstacles and linear kinematics (as shown in Section 2.5).

- **General non-convex**: For the general non-convex case, the convergence of ADMM is still an active field of optimization research. The multi-joint robotic motion planning problem falls into this category. Though the signed distance function of convex obstacles satisfies the semi-convex condition, the nonlinear robot forward kinematics violates these conditions and makes the planning problem extremely challenging. This is a common problem encountered by all the existing trajectory optimization

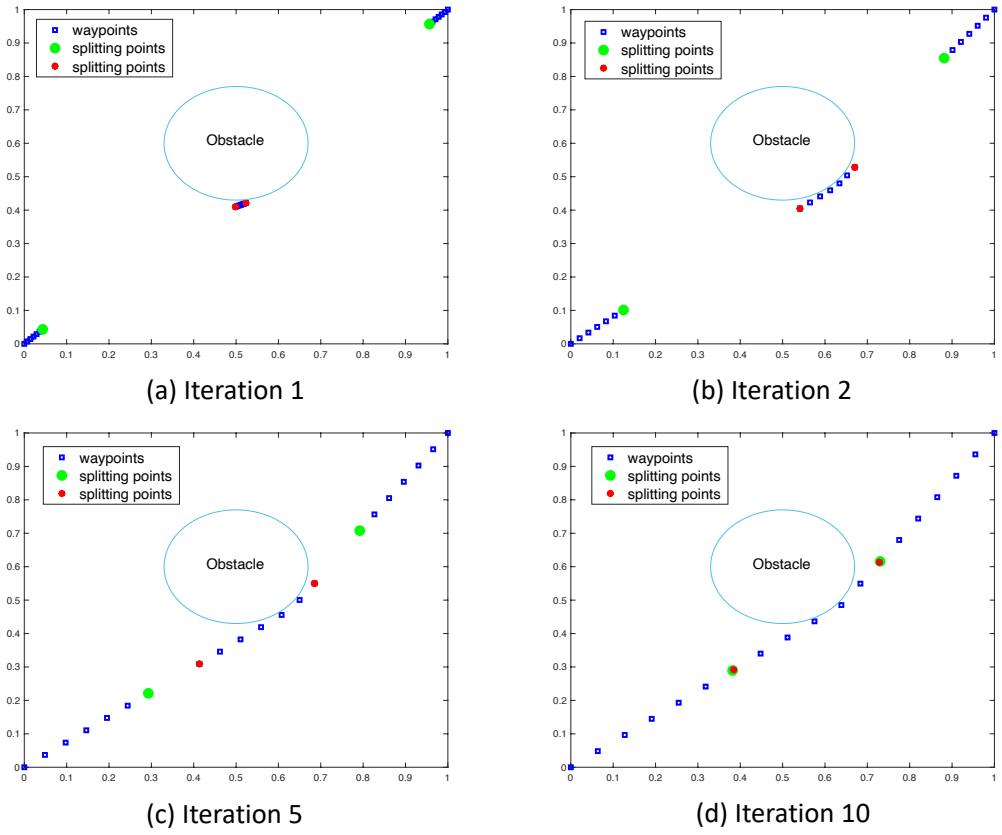


Figure 2.5: A 2D example of trajectory splitting. The planning problem is split into three segments and solved in a distributed manner. The trajectory is initialized using linear interpolation. As the optimization begins, each segment is very short and there is a large splitting error since the individual problems have an objective that minimizes segment distance. As the optimization progresses, the consensus update reduces the splitting error, and a continuous, smooth trajectory is obtained.

algorithms. Though there is no theoretical proof of convergence yet, our practical observation from simulation and experiments show that the trajectory splitting algorithm converges reasonably for this challenging scenario (as shown in Section 2.5).

2.5 Simulation and Experiments

Robot Model

The proposed trajectory splitting algorithm is tested in two scenarios. The first scenario is a simple 2D case with a sphere obstacle (as shown in Fig. 2.5), and the second is on a 6-DoF

Table 2.1: Simulation Benchmark of 6-DOF Robot Planning

Algorithm	RRT	LBKPIECE	RRT*	CHOMP	IPOPT	**TrajSplit_5**	**TrajSplit_3**
Average Time (s)	0.254	0.218	5.001	0.267	0.489	0.381	0.178
Path Length (rad)	11.24	10.97	9.03	7.15	7.23	7.82	7.43
Success Rate	25/25	25/25	25/25	21/25	23/25	21/25	23/25

FANUC LRMATE 200iD robot (as shown in Fig. 3.9).

For the 2D example, the robot state x is defined as the Cartesian position, and velocity in 2D: $x = [p_x, p_y, \dot{p}_x, \dot{p}_y]^T$, where p_x, and p_y denote the position in x and y axes.

For the multi-jointed robot case, the robot state x is selected to include the robot joint angle $\theta \in \mathbb{R}^6$, joint velocity $\dot{\theta} \in \mathbb{R}^6$, and joint acceleration $\ddot{\theta} \in \mathbb{R}^6$. For optimization, we use linear double-integrator dynamics as the equality constraint:

$$\begin{bmatrix} \theta_{i+1} \\ \dot{\theta}_{i+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & T\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \theta_i \\ \dot{\theta}_i \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ T\boldsymbol{I} \end{bmatrix} \ddot{\theta}_i \tag{2.18}$$

where $\boldsymbol{0} \in \mathbb{R}^{6\times6}$, and $\boldsymbol{I} \in \mathbb{R}^{6\times6}$ are the zero and identity matrices, and T is a positive scalar that denotes the robot travel time in-between each consecutive waypoint pair. The objective function is selected to minimize the summation of joint velocities in (2.19) for the minimum path length trajectory.

$$c(x) = \sum_{i=1}^{N} \|\dot{\theta}_i\|^2 \tag{2.19}$$

Simulation in 2D

We first evaluate the effectiveness of the proposed trajectory splitting algorithm in a 2D scene with a sphere obstacle. The collision avoidance constraint is formulated as:

$$|| \begin{bmatrix} p_x \\ p_y \end{bmatrix} - \begin{bmatrix} p_{ox} \\ p_{oy} \end{bmatrix} ||^2 \geq r_o^2 \tag{2.20}$$

where p_o and r_o denote the obstacle center position and its radius. As we can see from the simulation results in Fig. 2.5, the trajectory is split into three segments. The evolution of the solution initially produces short segments with a large splitting error that is refined through the consensus iteration to produce the final smooth trajectory.

Simulation on Multi-jointed Robot

We benchmarked the proposed trajectory splitting algorithm against existing state-of-the-art motion planners and optimization solvers, which include sampling-based methods RRT, RRT*, LBKPIECE in OMPL/Moveit, and an optimization-based method CHOMP in MoveIt using default parameters. We also implemented an optimization baseline using IPOPT with

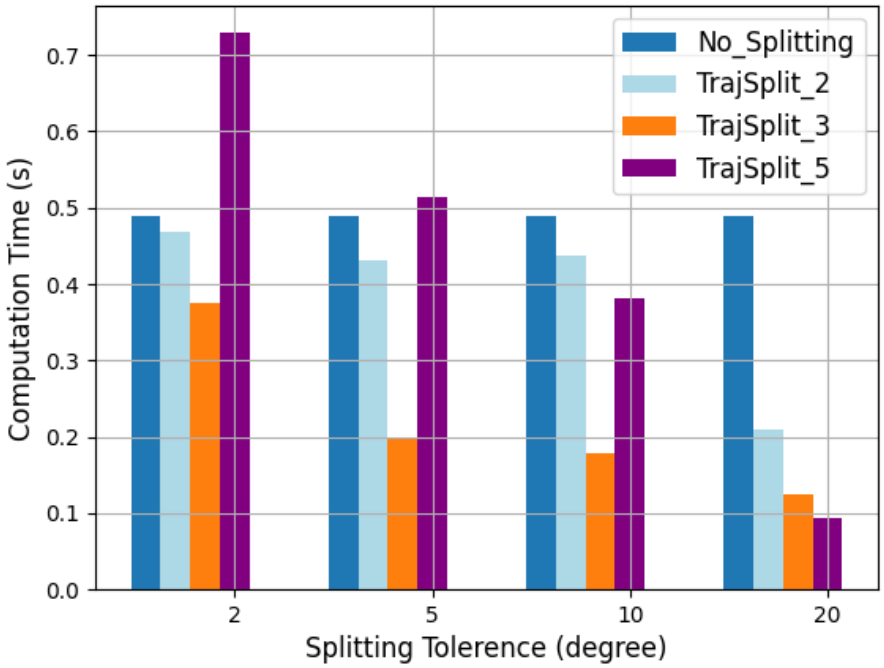


Figure 2.6: Parameter sweeps for the trajectory splitting algorithm, where Trajsplit_2, Trajsplit_3, Trajsplit_5 denote using the splitting algorithm to separate a trajectory into two, three, and five pieces. Computation time decreases as splitting error tolerance in the stopping criterion is increased. In general, increasing the number of split segments does not guarantee a shorter computation time. For these problems, three segments provide a nice balance between the number of ADMM iterations and subproblem complexity.

FCL as the collision checker. Our implementation of the trajectory splitting planner is based on the IPOPT baseline and uses the Python multiprocess library to achieve parallelization.

Fig. 3.9 shows the 4 of the benchmark results obtained from the proposed trajectory splitting algorithm. We manually selected 5 start poses and 5 end poses to construct 25 unique planning problems. The planning time limit is set to 5 seconds. Since MoveIt planners can only compute paths instead of trajectories, to make the comparison fair, we also only compute the joint path and ignore the dynamics constraint in (2.18) for this benchmark. The initialization for the IPOPT baseline and the trajectory splitting algorithms is set to be a straight line.

Table. 6.1 shows the benchmark results. TrajSplit_3 and TrajSplit_5 denote the proposed splitting methods that split a trajectory into 3 and 5 segments respectively. Sampling-based planners, such as RRT, and LBKPIECE perform well in these tasks. They all achieve a 100%

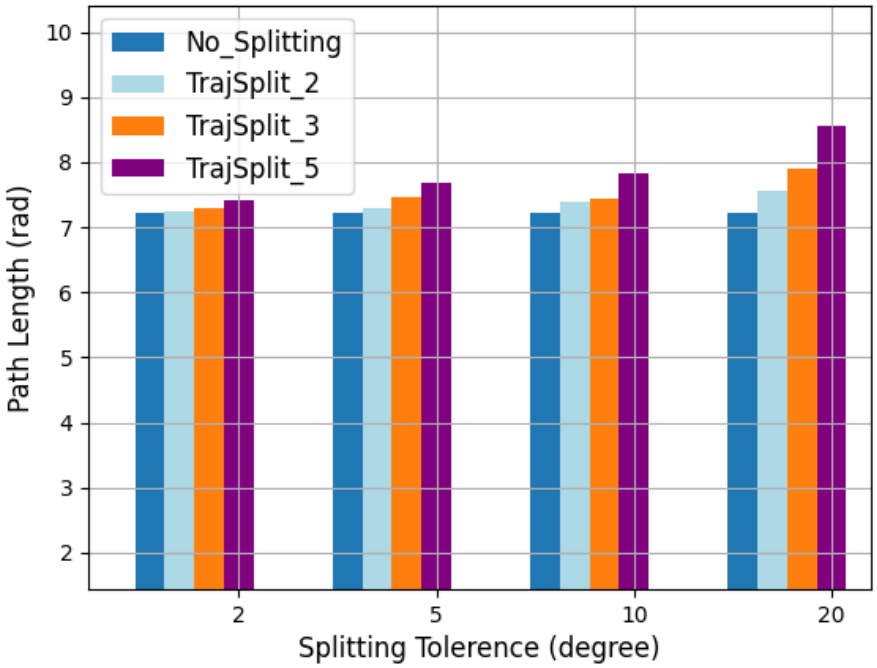


Figure 2.7: Parameter sweeps for the trajectory splitting algorithm, where Trajsplit_2, Trajsplit_3, Trajsplit_5 denote using the splitting algorithm to separate a trajectory into two, three, and five pieces. Reducing the splitting error tolerance results in smoother trajectories at the cost of greater computation time. These problems show a reasonable trade-off in quality and performance by setting the splitting error tolerance to 10 degrees with two or three split segments.

success rate in this setting. However, since those sampling planners are stochastic, they may generate different paths for the same problem, and the computation time and solution quality also have a very large variance (for example, RRT is normally efficient in dealing with those problems, but sometimes it may take over 3.7 seconds to obtain a sub-optimal solution). We noticed in our benchmarking, that the sampling-based planners sometimes generated non-intuitive motion with much larger average path lengths than the optimization-based planners. RRT* is a sampling-based method that tries to deal with the optimality problem. However, for all our tasks, RRT* was not able to find an optimal solution within the 5 seconds limit and the solution quality was qualitatively poor relative to the other optimization-based planners we compared. Similar results have been reported in other papers [56]. These data corroborate that RRT* is not as competitive in terms of computation efficiency.

The CHOMP algorithm in MoveIt was able to obtain optimal paths efficiently. However,

the robustness of the optimization-based planner was low. In our benchmark, the planner was often caught in a local minimum and incapable of finding a feasible solution when the start pose was close to the obstacle. In contrast, the IPOPT baseline was slower but more robust, and able to solve most of the problems, which was expected as IPOPT is built for general NLP and not for speed. Our proposed trajectory splitting method demonstrated a nice balance between speed and solution quality when the splitting error tolerance was set to 10 degrees and split into three segments. The splitting error is negligible considering the robot has 6 DOF. Trade-offs in quality and speed were straightforward by adjusting the number of splits and tolerance of the splitting error. However, the most efficient setup required balancing the number of segments that may benefit the computation time for a single primal update with additional ADMM iterations that may be required to meet the stopping criterion. The typical failure case of the proposed trajectory splitting algorithm comes from the misalignment of the splitting point that penetrates the obstacle. Due to the splitting of the trajectory, each segment tends to keep away from the obstacle. When the obstacle is thinner than the distance that results from the splitting error tolerance, the intermediate trajectory may penetrate the obstacle. We will address this problem in future work.

We further tested the behavior of our method by tuning hyperparameters: number of segments and splitting tolerance. We illustrate the variation of the computation time and path length for different splitting tolerances and different splitting segment numbers. As shown in Fig. 2.7, splitting a trajectory into more pieces resulted in a longer path length due to the splitting residual. In practice, there was a trade-off between the computation time and the splitting residual tolerance as shown in Fig. 2.6. To balance the solution quality and the efficiency, people may need to carefully tune these two parameters for different types of requirements.

Experiments

We tested the effectiveness of the trajectory splitting algorithm in a real-world scenario using a 6 DOF FANUC LRMATE 200id robot. As shown in Fig. 2.8, the robot picks a bottle from a wooden chair and places it on a bookshelf. The trajectory was initialized with linear interpolation in joint space and the problem was split into two segments with a splitting tolerance of 10 degrees. The solution was obtained from Alg. 1, and the assembled trajectory was sent as a list of joint space positions to the robot for execution. The snapshots from the video show the continuity of the complete trajectory, free from any collision. Note that with this splitting error tolerance, a small motion artifact is observable between the two split segments.

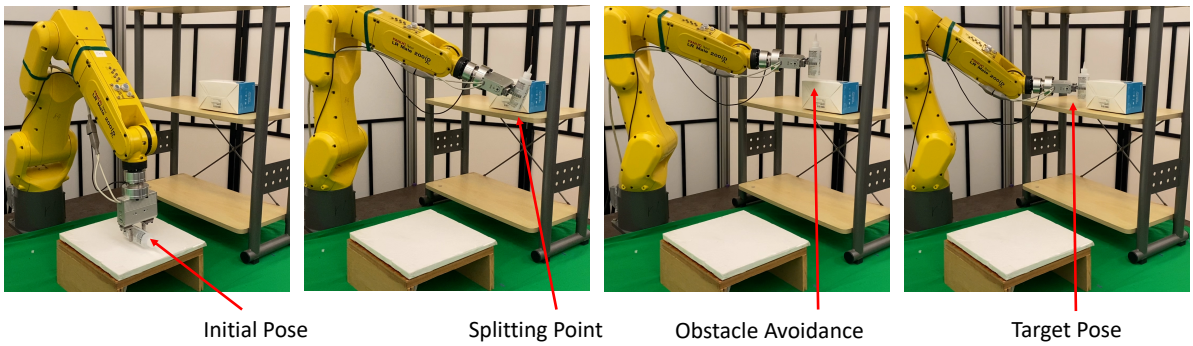


Figure 2.8: Experiment snapshots. The robot picks a bottle onto the bookshelf. Using the proposed trajectory splitting framework, the problem is split into two pieces and solved in a distributed manner. The splitting happens in the middle of the trajectory, and we can observe a tiny splitting residual during the execution in the experiment video.

2.6 Chapter Summary

Modern computational hardware is replete with opportunities for parallel computation, e.g. multi-core CPUs, GPUs, and TPUs. Our algorithm offers a novel scheme to exploit parallelism in trajectory optimization and a framework for balancing trajectory quality with computational speed. Planning tasks that require fast solve times can choose fewer segments and looser splitting tolerances to obtain quick, reasonable quality solutions. Conversely, high-accuracy trajectories can be obtained with increased segments and tighter splitting tolerance with modest increases in planning time.

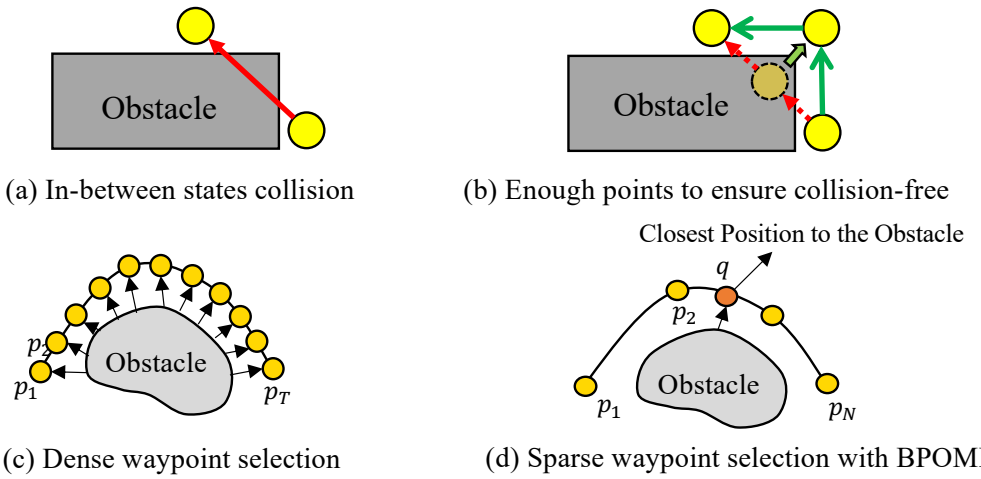
Chapter 3

Bilevel Trajectory Optimization Formulation for Efficient Collision Avoidance

Building on the efficient trajectory optimization algorithm, 'trajectory splitting', introduced in Chapter 2 to enhance computational efficiency in motion planning, this chapter addresses another crucial aspect: solution quality, with a specific emphasis on continuous obstacle avoidance. We present the Bilevel Path Optimization Formulation for Motion Planning (BPOMP), an advanced approach designed to ensure collision avoidance along the entire continuous trajectory. Different from standard formulations that typically focus on avoiding collisions at discrete waypoints, BPOMP extends this consideration to potential collisions occurring along the entire path. This is achieved by calculating the closest position to any obstacle along the trajectory. The underlying intuition is that if this closest point is free from collision, it implies that the entire path is likely to be collision-free. This optimization problem is structured as a bilevel problem and subsequently converted into a canonical form of nonlinear programming (NLP), enabling it to be resolved using classical solution methods. The chapter delves into a comprehensive analysis and demonstration of BPOMP's effectiveness, showcasing its ability to produce higher-quality solutions in motion planning, particularly in ensuring robust and continuous obstacle avoidance.

3.1 Introduction

Motion planning aims to find a sequence of configurations that moves the robot from an initial pose to a target pose without collisions. Sampling methods, such as probabilistic roadmap (PRM) [33] and rapidly exploring random tree (RRT) [40] are well suited for finding feasible solutions. Moreover, their variants, such as RRT* [39] and PRM*, can obtain optimal paths according to an objective. Exploiting the probabilistic completeness property provides a theoretical guarantee of global optimality. However, in practice, the planner is only given a



(a) In-between states collision

(b) Enough points to ensure collision-free

(c) Dense waypoint selection

(d) Sparse waypoint selection with BPOMP

Figure 3.1: (a) illustrates the collision that happens between waypoints. One way for existing methods to deal with the problem is to add more waypoints, as shown in (b). However, dense waypoint selection (shown in (c)) would significantly increase the computation time and may result in a jerky motion. The proposed BPOMP formulation can deal with that problem efficiently by finding and optimizing the worst state without dense waypoint selection.

limited time to find a path, and thus, their performance declines rapidly as the dimensionality of the scenario increases [56].

In this chapter, we focus on optimization-based planners. These methods involve defining a set of waypoints along the path/trajectory and checking for collisions for each waypoint. It is reported in several papers that optimization-based methods are efficient in moving the robot out of the collision [123, 80, 102]. However, a drawback of the existing optimization formulation is that collisions may still exist between those waypoints, as illustrated in Fig. 3.1(a).

Increasing the density of waypoints, i.e., making the spacing between waypoints small enough to detect obstacles as shown in Fig. 3.1(b), is an effective way to avoid potential collisions. Using dense waypoints, however, brings new challenges. First, it significantly increases the computation time since the equality constraints (e.g., the forward or inverse kinematics) and nonlinear constraints (e.g., collision avoidance) have to be evaluated for each waypoint. Moreover, the number of waypoints is a hyper-parameter that has to be carefully tuned for different scenarios. These issues limit the achievable performance, efficiency, and generalizability.

Instead of discrete waypoints, some works proposed to penalize collisions of the continuous path via a convex hull approximation. Authors of TrajOpt[80] validate the optimization formulation of continuous penalty in various scenarios. However, the convex hull approximation has no guarantee when rotational movements are involved, particularly in high di-

mensional spaces. Besides, the convex hull approximation is conservative in many scenarios. The method is not able to obtain feasible solutions when the feasible set is narrow.

Another way to handle the collisions that happen in the middle of waypoints is via parameterization, such as B-splines [12, 57]. The convex hull property of the B-spline can be applied to reduce the decision space from infinite to finite dimensions. Such relaxation, however, introduces conservatism to the problem and narrows the feasible region. Recently, researchers proposed to use the Gaussian process for path/trajectory representation[60, 19, 61]. Nevertheless, those frameworks still only check the collisions on some predefined points.

This chapter introduces a bilevel optimization-based path planning formulation called BPOMP. With fewer waypoints, BPOMP is able to achieve a similar collision checking accuracy as dense waypoint selection but with better computation efficiency. As shown in Fig. 3.1(d), BPOMP works with a set of sparsely spaced waypoints by finding and optimizing the closest position to the obstacle along the continuous path. If that position on the path is not in collision with the obstacles, then any other point should also be collision-free. We formulated the BPOMP as a bilevel optimization and relaxed it as canonical nonlinear programming (NLP) that standard NLP solvers can directly solve.

The contributions of the chapter are listed as follows:

- A bilevel formulation of path optimization problems that consider the collisions that happen in the middle of waypoints.
- An implementation of BPOMP with the state-of-the-art optimization solver and collision checker for better quality and efficiency.
- Comparisons of the proposed formulation against existing methods in both simulation and real-world experiments.

The remainder of the chapter is organized as follows. Section 3.2 introduces the mathematical background of path optimization and bilevel optimization. Section 3.3 proposes the BPOMP framework including the problem formulation, its relaxation as an NLP, and the algorithm to solve the bilevel problem. Section 3.4 shows the simulation and experiment results. Section 3.5 concludes the chapter. Supplementary videos can be found on our website: https://changhaowang.github.io/BTOMP/BTOMP.html.

3.2 Mathematical Background

The Path Optimization Problem

The planning problem can be naturally formulated as an optimization problem as shown below:

$$\begin{aligned} \min_{\tau} \quad & F(\tau) \\ \text{s.t.} \quad & h_i(\tau) = 0, \quad i = 1, \cdots, n_{eq} \\ & g_i(\tau) \leq 0, \quad i = 1, \cdots, n_{ineq} \end{aligned} \tag{3.1}$$

where, $\tau = \{p_1, \cdots, p_N\}$ denotes a discretized robot path, where p_i is the robot position at ith waypoint. $F(\cdot)$ is a manually designed objective function that reflects the robot performance, $f(\cdot)$ includes the robot kinematic constraints, and $g(\cdot)$ is the nonlinear state constraints, such as obstacle avoidance.

Sequential Quadratic Programming

Assumption 3. *The cost function $F(\cdot)$, equality constraints $h_i(\cdot)$, and the inequality constraints $g_i(\cdot)$ are twice continuously differentialable.*

Sequential Quadratic Programming (SQP) is one of the most powerful methods to solve constrained nonlinear optimization problems in the form of (3.1). Namely, SQP solves the problem by iteratively constructing a Quadratic Program (QP) around the current iterate τ^k until convergence:

$$\begin{aligned} d^k = \arg\min_{d} \quad & \nabla F(\tau^k)^T d + \frac{1}{2} d^T H^k d \\ \text{s.t.} \quad & h_i(\tau^k) + \nabla h_i(\tau^k)^T d = 0, \quad i = 1, \cdots, n_{eq} \\ & g_i(\tau^k) + \nabla g_i(\tau^k)^T d \leq 0, \quad i = 1, \cdots, n_{ineq} \end{aligned} \tag{3.2}$$

where d is the descent direction of the trajectory, ∇ denotes the gradient of scalar functions, and H^k is a positive semi-definite matrix. Then the trajectory can be updated using the following update rule. The solution $\{\tau^k\}$ obtained by this update rule is proved to converge to a local optimal solution of the original problem in (3.1) with a proper step length α^k [7].

$$\tau^{k+1} = \tau^k + \alpha^k d^k \tag{3.3}$$

Definition 1. *The Lagrangian function of (3.1) is defined as follows:*

$$L(\tau, \mu, \lambda) = F(\tau) + \sum_{i=1}^{n_{eq}} \mu_i \cdot h_i(\tau) + \sum_{i=1}^{n_{ineq}} \lambda_i g_i(\tau) \tag{3.4}$$

where μ and λ are the Lagrangian multipliers of the equality and inequality constraints.

In practice, there are multiple ways to select H^k. Newton SQP algorithm [7] suggests using the Hessian of the Lagrangian function as H^k. At each iteration, along with the primal descent direction d^k, the optimal Lagrange multipliers will be calculated. Letting the optimal multipliers be denoted by μ^{*k} and λ^{*k}, and setting $d_\mu^k = \mu^{*k} - \mu^k$, and $d_\lambda^k = \lambda^{*k} - \lambda^k$, we can have the update rules for the multipliers:

$$\begin{aligned} \mu^{k+1} &= \mu^k + \alpha^k d_\mu^k \\ \lambda^{k+1} &= \lambda^k + \alpha^k d_\lambda^k \end{aligned} \tag{3.5}$$

For the selection of the step size α^k, it should produce a decrease in a merit function $\phi(\tau)$, which can measure the progress of the algorithm. $l1$ penalty form is commonly used as the merit function [7]. With the line search algorithm or any other rules that can find a proper step size, the overall SQP algorithm is guaranteed to converge to a local optimal solution [7, 6].

Bilevel Optimization

Bilevel optimization is defined as a mathematical program where an optimization problem contains another optimization as a constraint. A canonical formulation of bilevel optimization problems can be expressed as follows:

$$\begin{aligned} \min_{x \in \mathbb{X}, y \in \mathbb{Y}} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0 \\ & y \in \arg\min_{z \in \mathbb{Y}} \quad \{f(x, y) : g(x, z) \leq 0\} \end{aligned} \tag{3.6}$$

where F and x represent the upper-level objective and the decision variable, whereas f and y denote the cost function and the decision variable for the lower-level problem.

There are several approaches [3, 88, 16] to solve the bilevel problem. One well-developed method replaces the lower-level problem with its KKT conditions and then applies a gradient-based method to solve the resulting optimization problem. We will discuss this more in detail in Section 3.3.

3.3 BPOMP Formulation

In this section, we introduce the proposed BPOMP formulation for optimization-based path planning. In the beginning, we introduce the intuition and formulation of the proposed collision constraint. Then, we mathematically formulate it as a bilevel optimization. In the end, its relaxation to NLP and the corresponding update rule is explained.

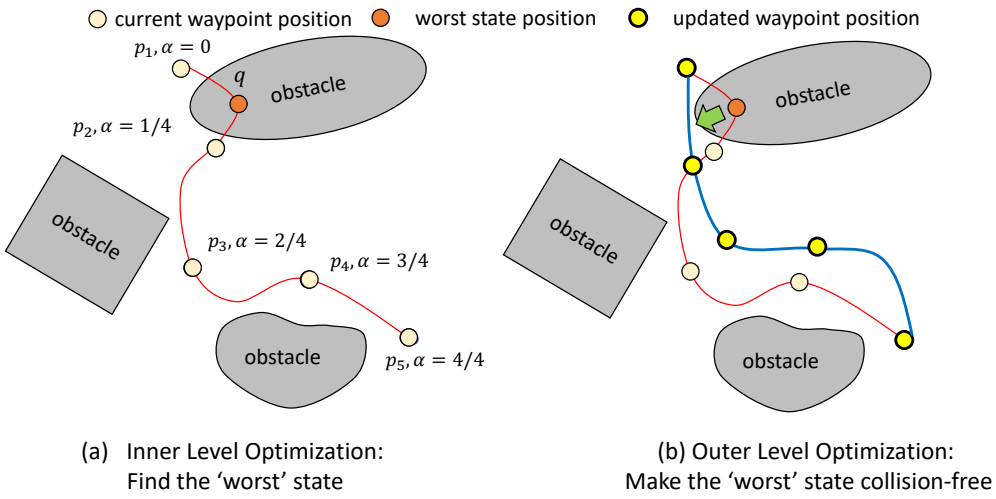


Figure 3.2: Illustration of the bilevel formulation of BPOMP. At each iteration, the inner-level optimization aims to find the position that is closest to the obstacle, and the outer-level optimization tries to update each waypoint to make that position collision-free.

BPOMP Formulation

The obstacle avoidance constraint in (3.1) can be defined by $\tau \in \Gamma_c = \{\tau : \phi_c(\tau) \geq 0\}$, where ϕ_c is a distance function between the robot and the obstacles. Since τ is a discretized path, this constraint only accounts for each waypoint p_i, and collisions may be ignored in the middle. To alleviate this issue, we propose an additional constraint on the closet position along the interpolated path to the obstacle as shown below:

$$\phi_c(q) \geq 0 \tag{3.7}$$

where q denotes the minimum distance position to the obstacles on the interpolated trajectory.

Adding the (3.7) together with the original optimization constraint in (3.1), we have the proposed BPOMP formulation. Intuitively, if the closest position to obstacles satisfies the collision constraint, every other position should satisfy the collision avoidance constraint as well. In the next subsection, we will introduce how to formulate the closest position q as a function of the optimization variable $\tau = \{p_1, \cdots, p_N\}$.

BPOMP as a Bilevel Optimization

Assume the robot's continuous path can be determined by curve fitting on the waypoints. That is, given a set of waypoints $\tau = \{p_1, \cdots, p_N\}$, every point on the continuous path can be obtained by an interpolation function $g(\alpha, \tau)$, where $\alpha \in [0, 1]$ is a ratio specifying the

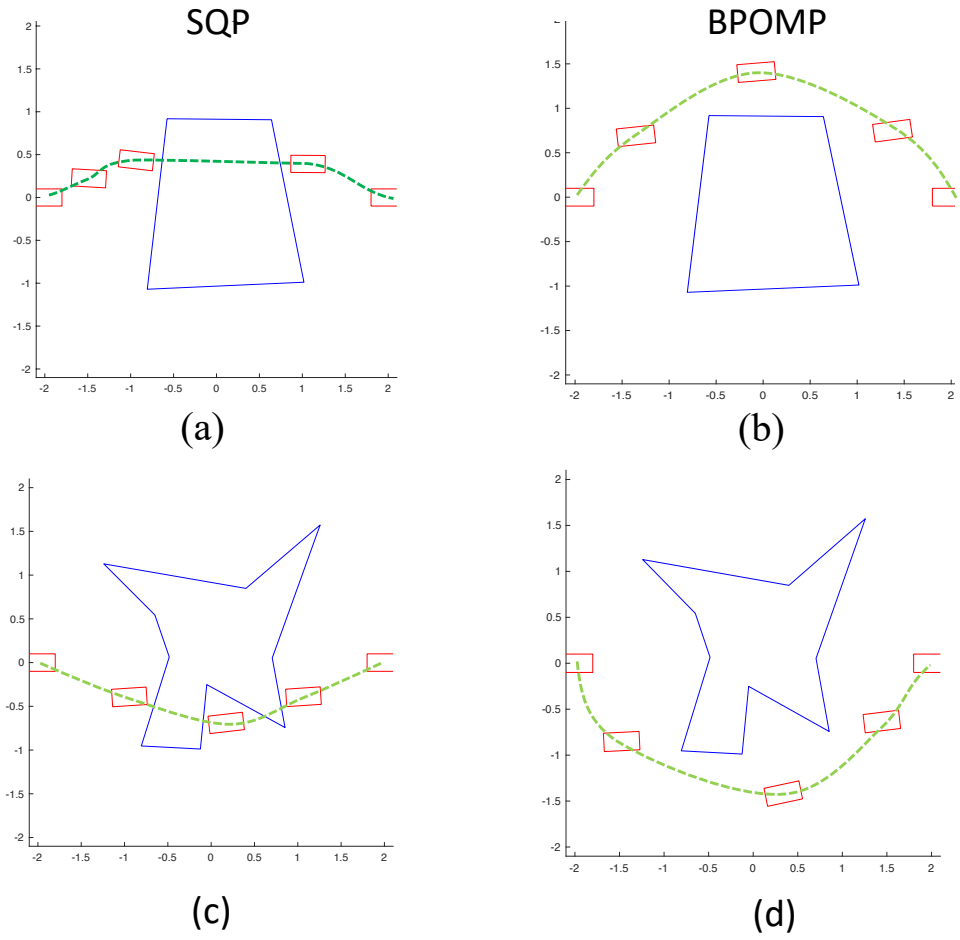


Figure 3.3: Simulation results on a mobile robot in a 2D space. The SQP-based trajectory optimizer is able to find collision-free waypoints but fails to make the overall trajectory collision-free. The proposed BPOMP formulation is able to find a feasible trajectory with sparse waypoint selection.

location of the point with respect to two ends of the trajectory. Fig. 3.2(a) demonstrates an example of a continuous path in a 2D space, where the five waypoints (yellow dots) are equally spaced along the continuous path (orange line), so that $\alpha = 0$ at p_1, $\alpha = \frac{1}{4}$ at p_2, and etc. Given a set of waypoints and the interpolation function, we can use a single parameter α to represent an arbitrary point on the continuous path. Therefore, the closest position q

to the obstacle can then be defined as follows:

$$\begin{aligned} \alpha^* &= \underset{\alpha\in[0,1]}{\operatorname{argmin}}\ \phi_c(g(\alpha,\tau)) \\ q &= g(\alpha^*,\tau) \end{aligned} \tag{3.8}$$

We can substitute (3.8) into the standard optimization formulation and reformulate it into a bilevel optimization given in (3.9). The optimization problem (3.9) contains two levels of optimization, where the lower level finds the closest position, and the outer level improves such position and other waypoints to be collision-free.

$$\begin{aligned} \min_{\tau,\alpha^*} \quad & F(\tau) \\ \text{s.t.} \quad & \phi_c(p_i) \geq 0 \quad i = 1,\cdots,N \\ & \phi_c(g(\alpha^*,\tau)) \geq 0 \\ & \alpha^* \in \underset{\alpha\in[0,1]}{\operatorname{argmin}}\ \phi_c(g(\alpha,\tau)) \end{aligned} \tag{3.9}$$

Remark 2. *For simplicity, we only keep the collision constraint and omit other constraints like kinematics. Those constraints can be considered without modification in the outer-level optimization.*

BPOMP Relaxation as a NLP

There exist several methods [3, 88] to transform a bilevel optimization to canonical nonlinear programming (NLP). One effective way is to replace the lower-level optimization problem with its optimality conditions [88]. Following this strategy, we could replace the lower level problem of (3.9) with its first-order necessary condition and its second-order sufficient condition [6]:

$$\begin{aligned} \frac{\partial \phi_c(g(\alpha^*,\tau))}{\partial \alpha^*} &= 0 \\ \frac{\partial^2 \phi_c(g(\alpha^*,\tau))}{\partial {\alpha^*}^2} &> 0 \end{aligned} \tag{3.10}$$

Replacing the inner level problem with (3.10), we can transform the BPOMP formulation into a standard NLP. Compared with the standard formulation of path optimization in (3.1), the proposed BPOMP formulation considers the collision on the entire path. Therefore it can have better performance to avoid the robot tunneling through thin obstacles. Since BPOMP does not require dense waypoints for collision checking, it can achieve a better computation efficiency that is empirically observed from the simulation and experiment results.

Most of the existing nonlinear optimization solvers can be directly utilized to solve the BPOMP formulation without modification. In our implementation, we adopt the update

rule of SQP in (3.2) to solve it. The equality constraint is

$$h(\tau) = \frac{\partial \phi_c(g(\alpha_w, \tau))}{\partial \alpha_w} \tag{3.11}$$

and inequality constraints include

$$\begin{aligned} g_{1:N}(\tau) &= -\phi_c(p_i) \quad i = 1, \cdots, N \\ g_{N+1}(\tau) &= -\phi_c(g(\alpha_w, \tau)) \\ g_{N+2}(\tau) &= -\frac{\partial^2 \phi_c(g(\alpha_w, \tau))}{\partial \alpha_w^2} \end{aligned} \tag{3.12}$$

In practice, we formulate the collision avoidance constraint $\phi_c(\cdot)$ using signed distance [123]. The distance values are negative inside obstacles, positive outside, and zero at the boundary. As we demonstrated in the simulation and experiment section, the obstacles we consider are general non-convex shapes. We approximate the derivative of the distance function with a finite difference.

Similar to most of the existing optimization-based planners, due to the non-convexity, the algorithm can only provide local optimal solutions, and the robot may still penetrate the obstacle after the optimization. However, we empirically observe from the simulation and experiment results that BPOMP can effectively increase the success rate and provide the solutions more efficiently. In the future, we plan to address the local optimal problem by providing a good initial reference via sampling planners, such as RRT and PRM.

3.4 Simulation and Experimental Results

In this section, we present the simulation and experimental results of BPOMP. We demonstrate the efficiency and robustness of our optimization formulation on a set of representative tasks in factories. Comparisons with other state-of-the-art optimization-based planners are also provided to validate the effectiveness of the proposed formulation. Videos of the experiments can be found at https://changhaowang.github.io/BPOMP/BPOMP.html.

Simulation in 2D Environment

The proposed BPOMP is first implemented on a mobile robot in a 2D space for validation. The robot has three degrees of freedom $[x, y, rotation]$. The initial trajectory is selected as a straight line and discretized into 5 waypoints. Collision checking is done via the Gilbert-Johnson-Keerthi (GJK) algorithm [21] and the Expanding Polytope Algorithm (EPA) [5].

As shown in Fig. 3.3, the SQP-based trajectory optimizer with the standard formulation in (3.1) is able to move the waypoints out of the obstacle but fails to make the overall trajectory collision-free. In contrast, using the BPOMP formulation, the same SQP solver is able to find a feasible solution. As we expected, the standard trajectory planning formulation only considers collision on waypoints, where BPOMP continues to update the waypoints until all collisions are eliminated as Fig. 3.3(b) shows.

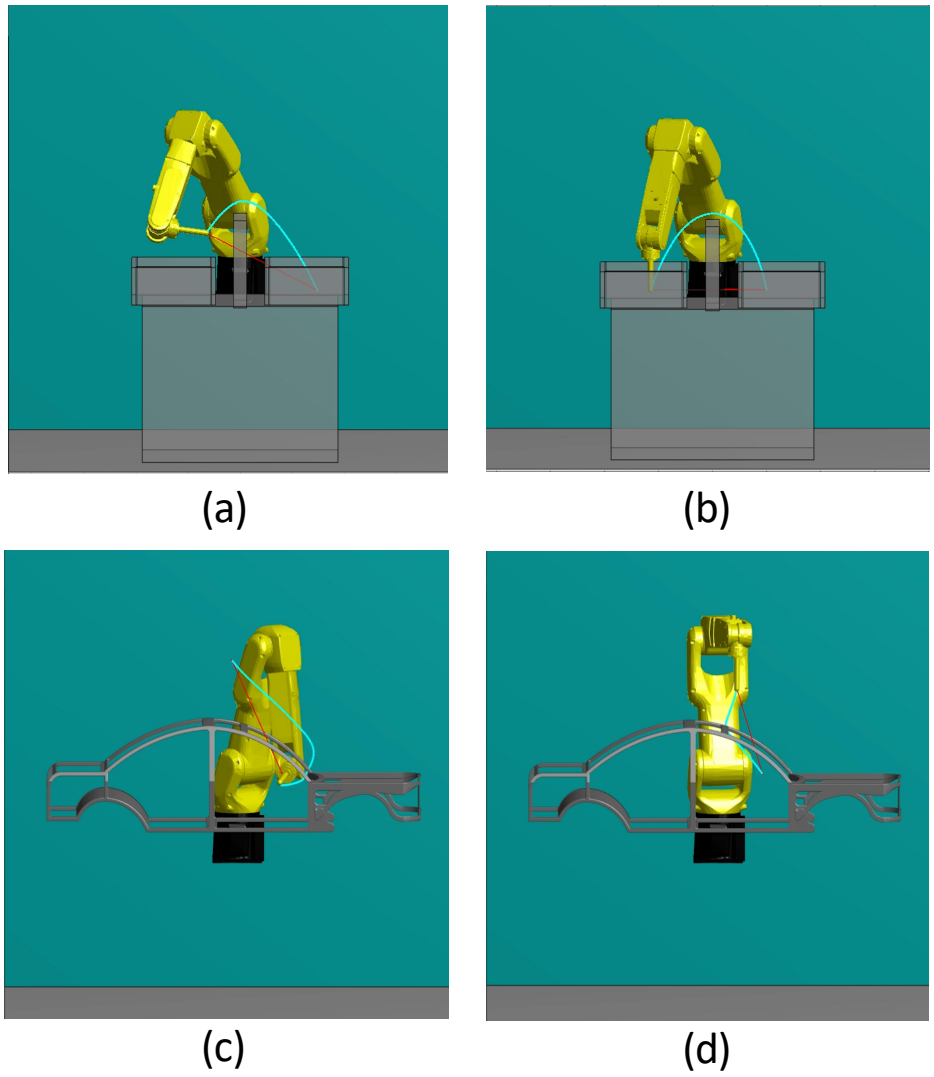


Figure 3.4: Simulation results of the proposed BPOMP algorithm in the bin picking and car frame painting scenarios. The FANUC robot is able to finish the task and avoid all the thin obstacles.

Benchmark on a 6-DOF Robot

The proposed algorithm is also tested on a 6-DOF robot in various scenarios. A FANUC LR Mate 200iD/7L robot is used. For collision checking, each link is modeled by several spheres with a different radius that covers the whole link (Fig. 2 of [48]). Then the collision avoidance constraint can be constructed using spheres. The environment is voxelized into a

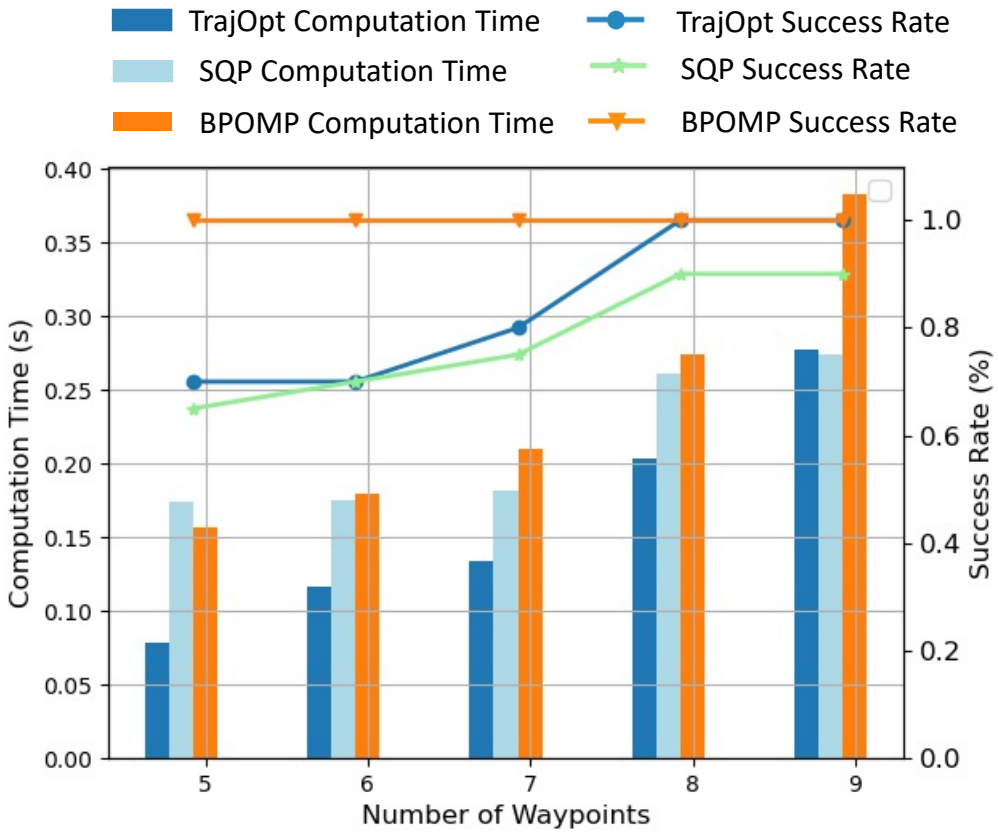


Figure 3.5: Performance comparison of optimization-based planners in the bin-picking scenario. With the proposed BPOMP formulation, the optimization algorithm is able to achieve a 100% success rate with only 5 waypoints and the lowest computation time.

Table 3.1: Comparisons in bin picking scenarios

Algorithm	Success Rate	Solve Time (s)	Path Length (rad)	Minimum Required Waypoints
RRT	20/20	0.4138 ± 0.2323	3.9777 ± 0.9546	NA
RRT*	20/20	0.7033 ± 0.1863	3.7180 ± 0.5942	NA
TrajOpt	20/20	0.2035 ± 0.0337	3.7279 ± 0.6386	8
SQP	19/20	0.2746 ± 0.0594	3.7324 ± 0.6479	8
BPOMP	20/20	0.1567 ± 0.0251	3.7344 ± 0.6021	5

$200 \times 200 \times 200$ grid. The MATLAB function *bwdist* is used for the distance field computation. Fmincon with the SQP solver in MATLAB is utilized as our optimization method.

We first test the BPOMP formulation in a bin-picking scenario, where the initial and target positions are randomly selected on each side. As shown in Fig. 3.4(a)(b), a robot picked a bin from one box, avoided all the potential collisions, and put the bin into the other box.

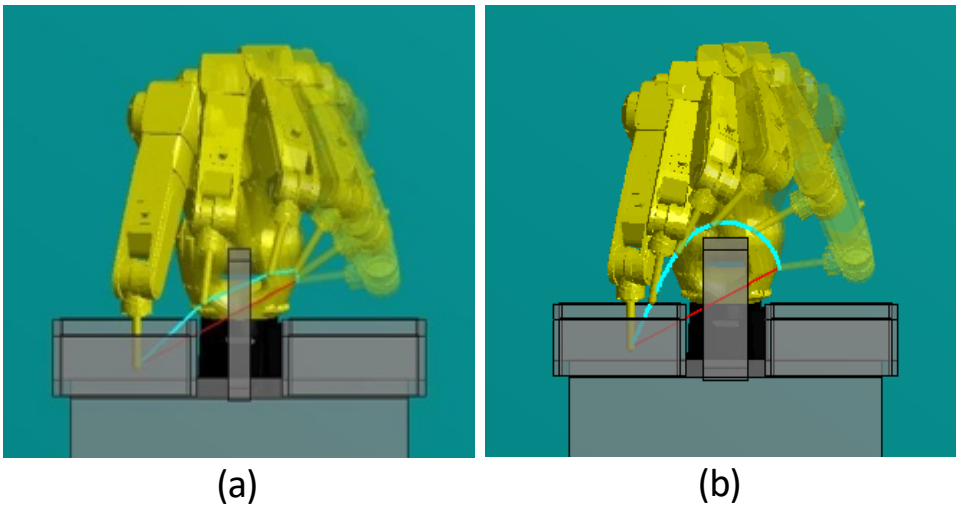


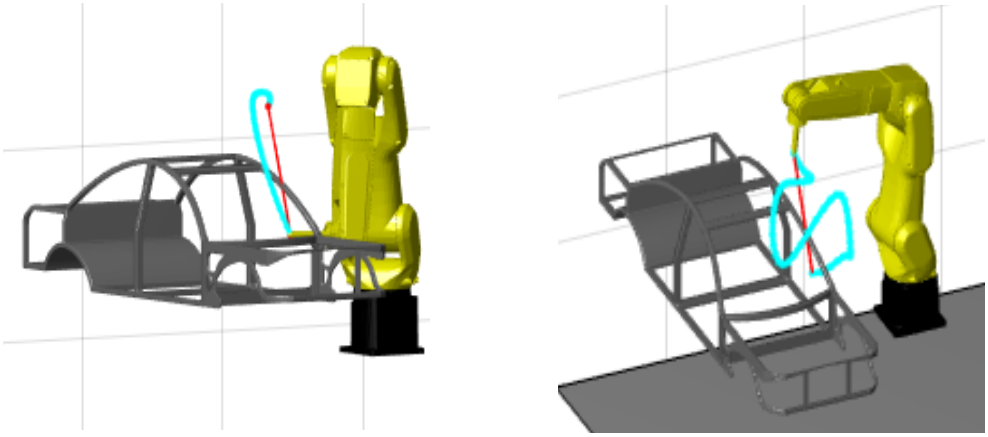
Figure 3.6: (a) shows a typical failure case of TrajOpt when the number of waypoints is not enough to detect the collision happening in the middle. (b) shows that BPOMP is able to avoid the potential collision. The red line denotes the initial path, and the green line is the solved path via optimization.

We benchmarked the proposed formulation with existing optimal path planners. Table 6.1 shows the comparison between RRT, RRT*, TrajOpt, SQP, and BPOMP (solved by SQP). The sampling planners are adopted from the MATLAB implementation in [43], with a goal sampling probability of 50%.

Observed from the result, sampling planners RRT, and RRT* are able to find feasible paths robustly. However, due to the stochastic characteristic, there exists a large variance in the computation time. Compared with RRT, RRT* is able to obtain the optimal path via the additional rewire operation with some sacrifice on the computation time.

Optimization-based planners, such as TrajOpt, and SQP are able to achieve a high success rate with better computation efficiency. Utilizing the gradient information of the signed distance field, they are more efficient than sampling planners to some extent.

For the proposed BPOMP formulation, it is able to achieve a 100% success rate and the best computation efficiency. The reason is revealed in Fig. 3.5, where we showed the curve of computation time and success rate with respect to different numbers of waypoints. Even though for the same amount of waypoints, BPOMP may have the highest computation time due to the additional constraint. In general, benefit from the special formulation, BPOMP is able to consider the collision happens on the entire path with minimum waypoints, and therefore, reduces the dimension of the optimization that is required and results in the best efficiency. Fig. 3.6(a) demonstrates a typical failure case without using the BPOMP formulation with sparse waypoints, where the algorithm was not able to detect collisions in



(a) Failure case of SQP with standard formulation

(b) Failure case of BPOMP

Figure 3.7: The red line denotes the initial path, and the green line is the solved path via optimization. (a) shows a failure case of SQP that the robot moves up inside the car and neglects the collision with the front pillar. (b) shows a failure case of BPOMP, where it detects the collision but fails to make the trajectory smooth enough.

between waypoints.

We further evaluated the performance of optimization-based planners in a complicated car frame dispensing scenario, where car frames are thin and non-convex. Fig. 3.8 shows the curve of the computation time and success rate with respect to different numbers of waypoints for optimization-based planners. Fig. 3.4(c) and Fig. 3.4(d) demonstrated the trajectories that BPOMP found to complete the task without collisions. Due to the narrow feasible region and non-convexity, the optimization-based planners worked poorly in this scenario. Since SQP and TrajOpt are not able to detect the collision and terminate earlier, their computation time is not comparable with BPOMP. Fig. 3.7(a) demonstrated a failure case using the SQP algorithm. The thin car frames are often neglected for collision checking and result in infeasible solutions for existing methods.

With the BPOMP formulation, the SQP solver is able to achieve a higher success rate. The optimization will not terminate earlier and ignore the collision that happens in the middle of those waypoints. The results corroborate the effectiveness of the BPOMP formulation. Fig. 3.7(b) showed a failure of BPOMP, where it detects the collision but fails to make the overall trajectory smooth enough within the 5-second time limit. The problem is due to the non-convexity of the problem. When the optimizer stocks in a local optimal solution, it is challenging to get out and find the global optimal solution.

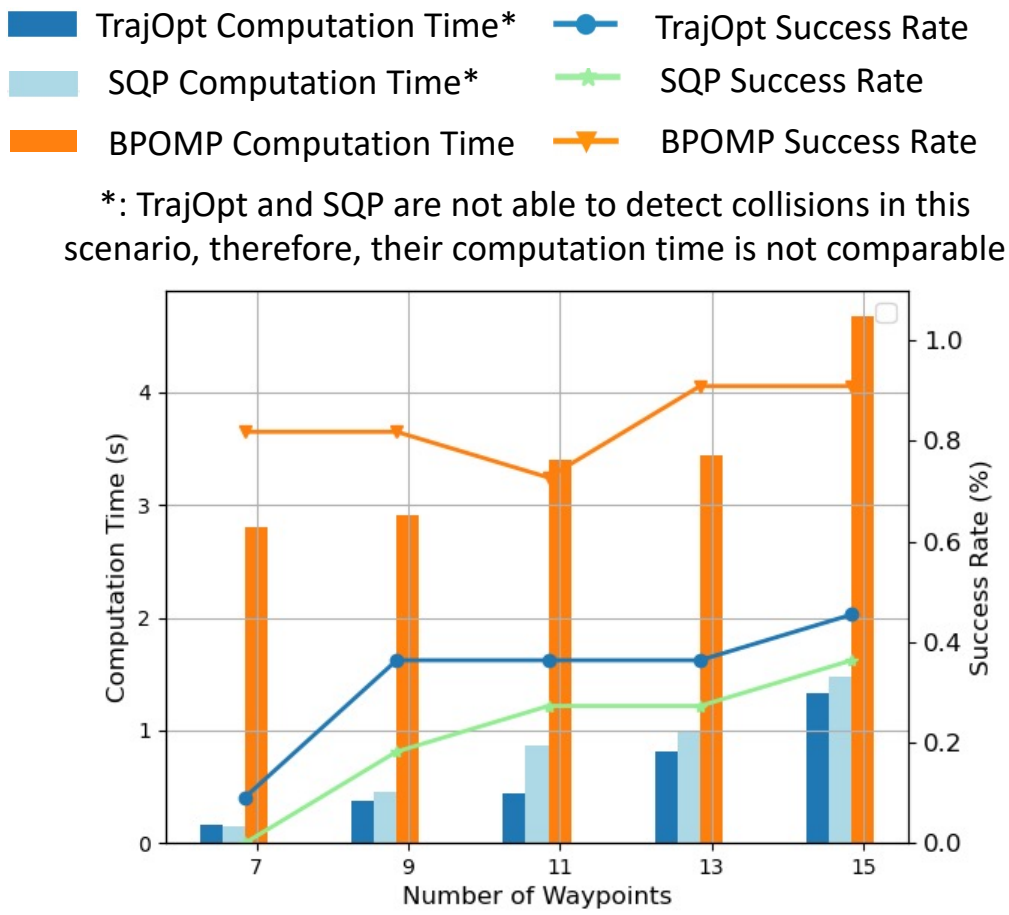


Figure 3.8: Performance comparison of optimization-based planners in the car frame dispensing scenario. With the proposed BPOMP formulation, the optimization algorithm is able to achieve a higher success rate.

Experiments on a 6-DOF Robot

The proposed BPOMP formulation is experimentally validated with a FANUC M-20iA robot. Similar to the above car frame dispensing scenario, in the workspace, a cage with thin frames is placed in front of the robot. The robot is set to put a bottle inside the cage with several different poses. Due to the geometric structure of the cage, the robot has to go through different faces of the cage to reach a particular target pose. For example, if the robot wants to perform a top-down grasp, the only feasible trajectory is passing through the top of the cage. The thin frames of the cage block the straight trajectories to reach the goal and result in a narrow feasible set in the joint space. This problem is challenging for existing optimization-based path/trajectory planning algorithms due to thin obstacles

Figure 3.9: Sequences of snapshots of two different tasks. A FANUC robot is able to grasp a bottle and place it inside the cage with different final poses without collision.

as illustrated in the previous simulation results. Fig. 3.10 showed the trajectories obtained by BPOMP, and Fig. 3.9 showed sequences of motions that the FANUC robot completed the pick-and-place tasks. From the experiments, we can observe that BPOMP can provide a larger chance to avoid potential obstacles.

3.5 Chapter Summary

This chapter proposed BPOMP, a novel path optimization formulation that can efficiently eliminate potential collisions with sparse waypoints. To achieve that, we introduced an additional collision constraint on the closest position to the obstacle on the continuous path. The problem is then formulated as a bilevel optimization problem and relaxed to canonical nonlinear programming (NLP). Comparisons with state-of-the-art path optimization/sampling algorithms corroborate that the proposed formulation could enable optimization-based planners to achieve higher efficiency and success rates.

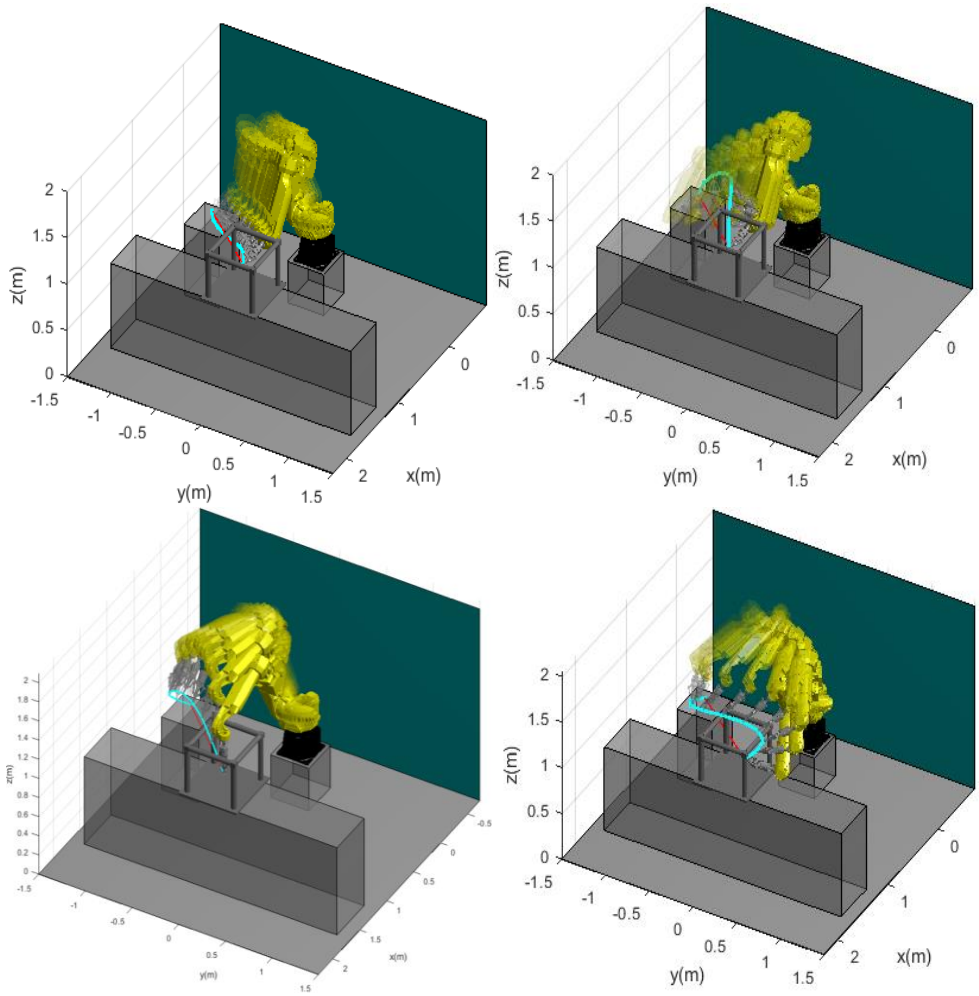


Figure 3.10: Paths obtained by BPOMP for the real-world experiments. Snapshots of the tasks can be found in Fig. 3.9.

Part II

Adaptive Model Learning for Deformable Object Manipulation

Chapter 4

A Unified Framework for Deformable Objects State Estimation and Task Planning

In Part I of the dissertation, we built the foundation for autonomous robotic systems with the introduction of efficient online trajectory optimization. Building upon that, Part II shifts the focus to the application of model-based planning and control methodologies, specifically aimed at achieving robust manipulation of deformable objects. This chapter is dedicated to discussing the framework we have developed for real-time tracking and state estimation of such objects.

Deformable objects pose unique challenges due to their variable shapes and behaviors under manipulation. Our approach utilizes a unified framework to accurately track and predict the state changes of these objects in real time. This capability is crucial for the robotic system to make informed decisions and adapt its manipulation strategies effectively. We will provide a detailed description of how it integrates with the overall control system of the robot to enable precise and reliable manipulation of deformable objects in a variety of settings.

4.1 Introduction

While a great amount of work is focused on the manipulation of rigid objects, manipulating deformable objects, especially deformable linear objects (DLO), remains under-explored. There are many applications involving the manipulation of DLO, such as cable harnessing in factories, thread packing in production lines, suturing in medical surgeries, etc. These tasks are usually labor-intensive and have not been automated for many years. The major difficulty lies in the fact that these objects have high degrees of freedom which are expensive to model and control.

Take rope knotting as an example (Fig. 4.1). The objective is to manipulate the rope from

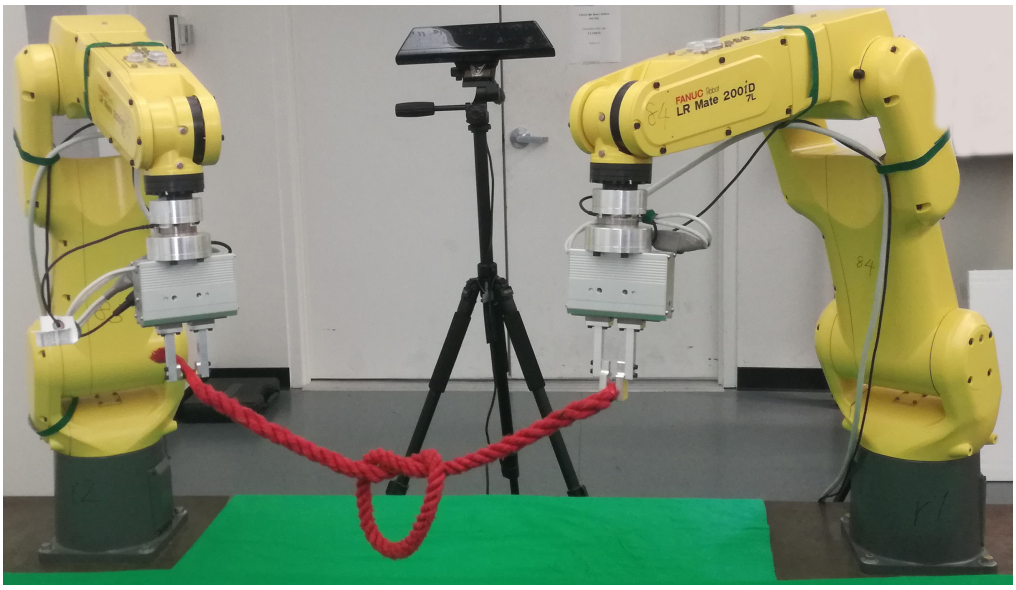


Figure 4.1: Two Fanuc LR Mate 200iD/7L robots are knotting a rope

a random initial state to a desired knotted state. Robots need to generate corresponding motions to manipulate the rope based on the observation of current rope states. This task has many challenges in several aspects, especially in state estimation, task planning, and trajectory planning.

First, for state estimation, the position of each rope segment needs to be identified from 3D camera measurements (point clouds). As shwon in Fig. 4.2, the rope we are tracking is featureless. In other words, there are no distinguishable markers or features to recognize each segment, and it is unknown which segment on the rope generates the measured points in the point clouds. This missing correspondence makes traditional visual tracking algorithms, for instance Kalman filter, unable to execute. Besides, since the rope is occluded by robot arms or self-occluded by itself frequently during manipulation, the state estimator should be specially designed to handle occlusion robustly. Moreover, considering the curse of dimensionality, running high dimensional state estimations in real time is also a challenging problem.

Second, regarding task planning, robots need to take several sequential steps to knot the rope gradually. Based on the state estimation result, a task planner needs to be developed to classify at which step the manipulation is and determine what actions each robot should take. Meanwhile, failure detection and recovery mechanisms should be included in the task planner in case a failure occurs.

Third, for trajectory planning, the difficulty lies in that the system is underactuated. Limited numbers of grippers (two in our case) are actuating the rope with high degrees of freedom. It is also observed that the rope always runs to unrepetitive shapes during manip-

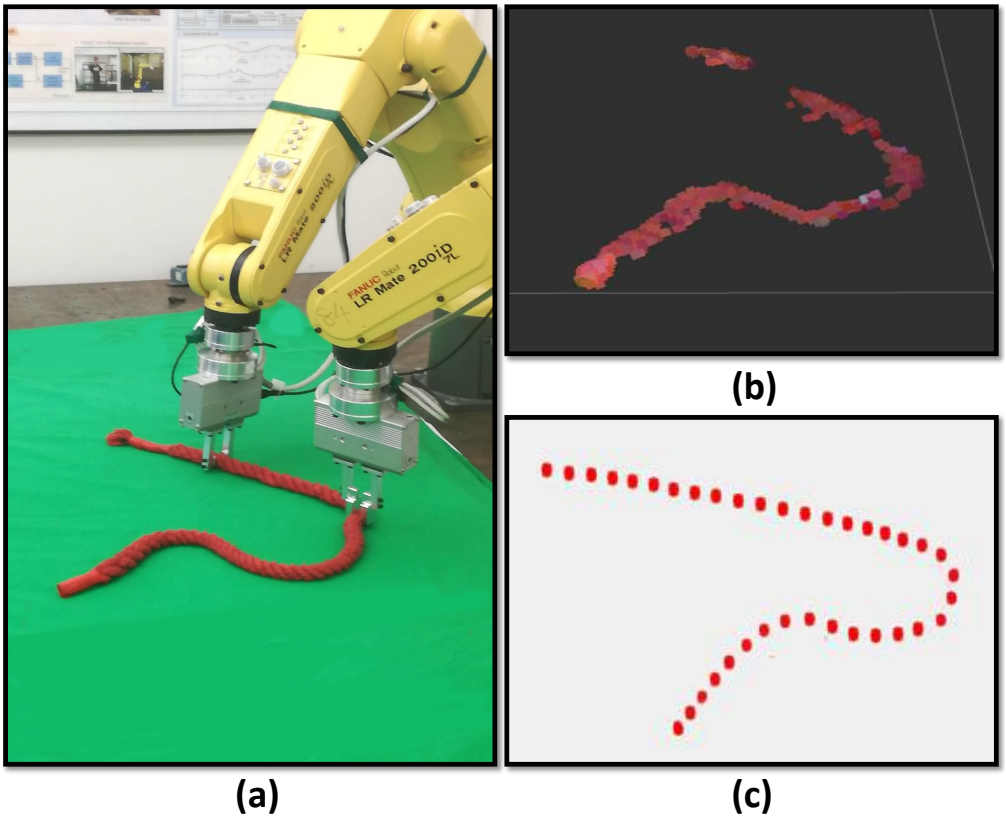


Figure 4.2: Illustration of state estimation for deformable objects. (a) shows the real rope, (b) shows the point cloud directly gained by the camera, and (c) shows a chain of connected nodes with uniform distance which we used to represent the rope.

ulation, i.e., shape differences always exist between training and test scenarios. Therefore, simply replaying the predefined trajectory for training easily fails for the test stages. An online trajectory planner should be developed to refine the trajectory with high efficiency.

In this chapter, a uniform framework for manipulating deformable linear objects is proposed, which aims to address all the challenges discussed above. The core technique we are using is called coherent point drift (CPD), a registration method for mapping one point set to another one non-rigidly. For state estimation, the position of each node on the object is acquired by registering the previous estimation results to the new point cloud measurements. The object states can be estimated robustly in real time under noise, outliers, and occlusions. For task planning, CPD is introduced to check the similarity between current object states and pre-recorded training states, then the manipulation step can be determined by finding the maximum similarity. Operation failure can also be detected if the similarity value is below some threshold. For trajectory planning, the learning from the demonstration

approach is introduced in this chapter. In training scenarios, human operators pre-program the corresponding trajectories for some specific rope shapes. During the test, a mapping function from the training scenario to the test scenario is constructed by CPD. The training trajectory is warped by the mapping function to obtain a new trajectory that is feasible for the test scenario.

The remainder of this chapter is organized as follows. Section 4.2 introduces related works on manipulating deformable objects. Section 4.3 describes the coherent point drift method for point registration. Section 4.4 explains the design of the framework in detail, which includes state estimation, task planning, and trajectory planning modules. Section 4.5 tests the performance of the proposed framework by a series of experiments. Section 4.6 concludes the chapter and proposes future work.

4.2 Related Works

Manipulation of soft ropes has been a subject of robotics research for decades. Researchers have addressed this problem with many different methods, such as modeling, knot theory, deep learning, etc in order to let the robot manipulate the rope just like what human does. Morita et al. [59] developed a Knot Planning from Observation (KPO) system which estimated the states of the rope from the vision feedback by knot theory. The primitive movements were also predefined to change the states of the rope. Kudoh et al. [36] built a multi-finger hand and programmed skill motions by observing human knotting procedures. They realized the three-dimensional in-air knotting with different types of knots. Many of these methods, however, require empirical laws and are developed for a specific task, which is not easy to generalize for other tasks. Ashvin Nair et al [63] built a learning-based system to train the robot to manipulate the rope. However, for each manipulation task, thousands of training data is required, besides, the success rate is not high.

Human teaching together with non-rigid registration has emerged as a practical and effective approach for the manipulation of soft ropes. Schulman et al. [81] proposed a non-rigid registration method to teach the robot to manipulate ropes by human demonstration. Their method used TPS-RPM to transfer the original trajectory (taught by human demonstration) to get a new manipulation trajectory that was suitable for the test scene. Since then, many follow-up works improved the non-rigid registration-based method. Lee et al. [42] extended Schulman's approach by jointly optimizing the non-rigid registration and the trajectory optimization into a single optimization framework, such that the resulting trajectory is smoother given obstacles. Lee et al. [41] then incorporated the normals into the objective function to find a better registration to ensure that the robot gripper is vertical to the operation surface. But all of these works use the point cloud of the rope as feedback. When they apply their methods, there are many limitations like poor performance under occlusion and outliers.

Compared with others, we are the first to propose a uniform framework by non-rigid registration for robotic manipulation of soft ropes. Together with state estimation, trajec-

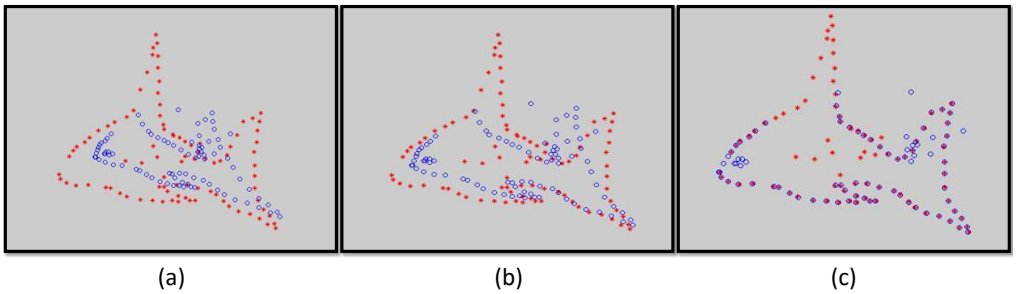


Figure 4.3: A sequence of pictures of CPD registration. The blue point set registers with the red point set by a smooth transformation.

tory planning, and scenario recognition, we greatly improve the robustness and efficiency of manipulation tasks.

4.3 Non-Rigid Registration by Coherent Point Drift

Non-rigid registration, i.e., registration source points to target points non-rigidly, is the core technique in this chapter to deal with the problem of state estimation, trajectory planning, and scenario recognition during the task of rope manipulation. Coherent Point Drift (CPD) is a popular method of non-rigid registration for its robustness towards outliers and missing points [15], and we will introduce it in detail in this section.

Fig. 4.3 provides an example of CPD registration. The blue point set registers with the red point set by a smooth transformation.

Assume there are two sets of point cloud, source point set $X = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^{N \times D}$ and target point set $Y = \{y_1, y_2, \ldots, y_M\} \in \mathbb{R}^{M \times D}$. N and M are the point numbers in X and Y respectively. D is the dimension of each point. The registration objective is to find a smooth transformation function $v : \mathbb{R}^D \to \mathbb{R}^D$ that maps X to a new point set $\bar{X} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N\} \in \mathbb{R}^{N \times D}$, such that $\bar{X}$ is similar to Y.

In the above objective, there are two aspects needing mathematical formulation: (1) how to measure the similarity between $\bar{X}$ and Y and (2) how to quantitatively describe the smoothness of the transformation function v.

For measuring similarity, the CPD algorithm takes a Gaussian mixture model (GMM) point of view, where the transformed points $\bar{X}$ are regarded as the centroids of multiple Gaussians and points in Y are random samples from the Gaussian mixtures.

Assume that each Gaussian has equal membership probability $\frac{1}{N}$ and the same isotropic covariance $\sigma^2\mathbf{I}$. Then the probability of each point y_m sampled from the Gaussian mixture

model can be described as

$$\begin{aligned} p(y_m) &= \sum_{n=1}^{N} \frac{1}{N} \mathcal{N}(y_m; \bar{x}_n, \sigma^2 \mathbf{I}) \\ &= \sum_{n=1}^{N} \frac{1}{N} \frac{1}{(2\pi\sigma^2)^{D/2}} \exp(-\frac{\|y_m - \bar{x}_n\|^2}{2\sigma^2}) \end{aligned} \tag{4.1}$$

To account for noise and outliers, an additional uniform distribution is added to the mixture model. The complete mixture model takes the form:

$$p(y_m) = \sum_{n=1}^{N+1} p(n) p(y_m|n) \tag{4.2}$$

with

$$p(n) = \begin{cases} (1-\mu)\frac{1}{N}, & n = 1, \ldots, N \\ \mu, & n = N+1 \end{cases} \tag{4.3}$$

$$p(y_m|n) = \begin{cases} \mathcal{N}(y_m; \bar{x}_n, \sigma^2 \mathbf{I}), & n = 1, \ldots, N \\ \frac{1}{M}, & n = N+1 \end{cases} \tag{4.4}$$

where μ denotes the weight of the uniform distribution.

The complete log-likelihood function Q is constructed,

$$\begin{aligned} Q &= \sum_{m=1}^{M} \sum_{n=1}^{N+1} p(n|y_m) \log\left(p(n) p(y_m|n)\right) \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N} p(n|y_m) \left(\log(\frac{1-\mu}{N(2\pi\sigma^2)^{D/2}}) - \frac{\|y_m - \bar{x}_n\|^2}{2\sigma^2} \right) \\ &+ \sum_{m=1}^{M} p(N+1|y_m) \log(\frac{\mu}{M}) \end{aligned} \tag{4.5}$$

The larger the value of Q, the more likelihood that points in Y are sampled from the Gaussian mixtures created by $\bar{X}$, i.e., the more similar between the two point sets $\bar{X}$ and Y.

The transformed set $\bar{X}$ is achieved from X by transformation function v:

$$\bar{x}_n = x_n + v(x_n) \tag{4.6}$$

v generates globally rigid transformation while also allowing locally non-rigid deformation. According to the regularization theory [22] , the function smoothness can be quantitatively measured by norm $\int_{\mathbb{R}^D} \frac{|V(s)|^2}{G(s)} ds$, where $V(s)$ is the Fourier transform of v and $G(s)$ is a symmetric filter with $G(s) \to 0$ as $s \to \infty$. The Fourier-domain norm definition basically passes v by a high-pass filter, then measures its remaining power at high frequency. Intuitively, the larger the norm, the more "oscillation" v will behave, i.e., less smoothness.

A modified likelihood function $\tilde{Q}$ is constructed by involving function v and penalizing its oscillation:

$$\begin{aligned}\tilde{Q}(v) &= Q - \frac{\lambda}{2}\int_{\mathbb{R}^D}\frac{|V(s)|^2}{G(s)}ds \\ &= \sum_{m=1}^{M}\sum_{n=1}^{N} p(n|y_m)\log(\frac{1-\mu}{N(2\pi\sigma^2)^{D/2}}) \\ &- \sum_{m=1}^{M}\sum_{n=1}^{N} p(n|y_m)\frac{\|y_m - x_n - v(x_n)\|^2}{2\sigma^2} \\ &+ \sum_{m=1}^{M} p(N+1|y_m)\log(\frac{\mu}{M}) - \frac{\lambda}{2}\int_{\mathbb{R}^D}\frac{|V(s)|^2}{G(s)}ds \end{aligned} \tag{4.7}$$

$\lambda \in \mathbb{R}^+$ is a trade-off weight that balances the data fitting accuracy (from $\bar{X}$ to Y) and the smoothness constraints (from X to $\bar{X}$). The negative sign before λ indicates a smaller norm, or a smoother transformation from X to $\bar{X}$, is preferred.

It can be proved by variation theory that the maximizer of (4.7) has the form of the radial basis function [62]:

$$v(z) = \sum_{n=1}^{N} w_n g(z - x_n) \tag{4.8}$$

where kernel $g(\cdot)$ is the inverse Fourier transform of $G(s)$, and $w_n \in \mathbb{R}^D$ is the kernel weights. In general, kernel $g(\cdot)$ can take many formulations, as long as it's symmetric, positive definite, and $G(s)$ behaves like a low-pass filter. For simplicity, a Gaussian kernel $g(\cdot)$ is chosen, with $g(z - x_n) = \exp(-\frac{\|z-x_n\|^2}{2\beta^2})$. $\beta \in \mathbb{R}^+$ is a manually tuned parameter that controls the rigidity of function v, where large β corresponds to rigid transformation, while small β produces more local deformation.

Substitute (4.8) to (4.7), then we get

$$\begin{aligned}\tilde{Q} &= \sum_{m=1}^{M}\sum_{n=1}^{N} p(n|y_m)\log(\frac{1-\mu}{N(2\pi\sigma^2)^{D/2}}) \\ &- \sum_{m=1}^{M}\sum_{n=1}^{N} p(n|y_m)\frac{\|y_m - x_n - \sum_{k=1}^{N} w_k g(x_n - x_k)\|^2}{2\sigma^2} \\ &+ \sum_{m=1}^{M} p(N+1|y_m)\log(\frac{\mu}{M}) - \frac{\lambda}{2}trace(\mathbf{W}^T\mathbf{G}\mathbf{W}) \end{aligned} \tag{4.9}$$

where $\mathbf{G} \in \mathbb{R}^{N\times N}$ is a symmetric positive Gramian matrix with element $\mathbf{G}_{ij} = g(x_i - x_j)$. $\mathbf{W} = [w_1, \ldots, w_N]^T \in \mathbb{R}^{N\times D}$ is the vectorization of kernel weights in (4.8).

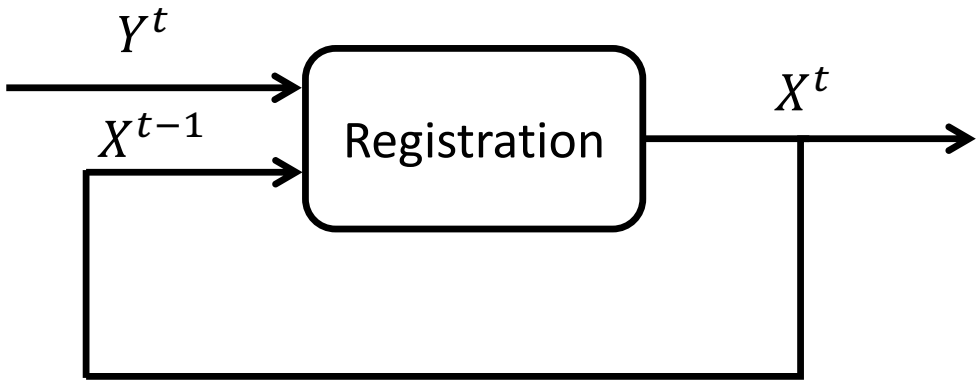


Figure 4.4: The framework of point set registration. Y_t is the perceived point cloud at time step t. X_{t-1} is the state estimation at the previous step. A new estimation X_t is achieved by registering X_{t-1} to Y_t

$\tilde{Q}$ is now parameterized by $(\mathbf{W}, \sigma^2)$ in (4.9). The EM algorithm can be performed to maximize the value of $\tilde{Q}$ and to estimate the parameter $(\mathbf{W}, \sigma^2)$ iteratively. In the E-Step, the posterior probability distribution $p(n|y_m)$ is calculated using the estimated $(\mathbf{W}, \sigma^2)$ from the last M-step. In M-Step, take $\frac{\partial \tilde{Q}}{\partial \mathbf{W}} = 0$ and $\frac{\partial Q}{\partial \sigma^2} = 0$ to achieve a new estimation of $(\mathbf{W}, \sigma^2)$. The closed-form solution for M-step requires some linear algebra derivation, and more details can be found in [62].

When $\tilde{Q}$ is converged, transfer function v can be calculated by (4.8), and the transformed set $\bar{X}$ is:

$$\bar{X} = X + \mathbf{GW} \tag{4.10}$$

4.4 Robotic Manipulation of Deformable Linear Objects

This section introduces the framework for robotic manipulation of deformable linear objects. Three major modules, state estimation, task planning, and trajectory planning, are introduced in sequence. Each of them uses CPD as a primary tool. For ease of illustration, an example of rope manipulation will be discussed in the following sections. However, the proposed framework should be general for other types of linear objects as well.

State Estimation of Deformable Linear Objects

During the process of rope manipulation, since the soft rope easily deforms into unscheduled shapes, it is necessary to close the execution loop by monitoring the rope states in real time.

Tracking infinite-dimensional configuration space is important. Therefore, we first discretized the rope to a chain of connected nodes with uniform distance (Fig. 4.9). Our objective is to estimate the position of each node at each time step from the dense, noisy, and occluded point clouds (Fig. 4.9) perceived by stereo cameras.

Suppose at the initial time step, the rope state is noted as $X^{t=0} = \{x_1^{t=0}, x_2^{t=0}, \ldots, x_N^{t=0}\} \in \mathbb{R}^{N\times 3}$, where $x_n^{t=0} \in \mathbb{R}^3$ is the nth node's initial position in the three-dimensional Cartesian space. N is the number of nodes. At time step $t = 1$, the rope moves to a new shape, and its point cloud $Y^{t=1} = \{y_1^{t=1}, y_2^{t=1}, \ldots, y_M^{t=1}\} \in \mathbb{R}^{M\times 3}$ is perceived by cameras. $y_m^{t=1} \in \mathbb{R}^3$ denotes the position of a single point in the cloud. M is the total number of points and usually $M >> N$. Using CPD, the node positions $X^{t=0}$ can be non-rigidly transformed to $\bar{X}$, which is well aligned to the point cloud $Y^{t=1}$. $\bar{X}$ serves as the state estimation of rope nodes at time step $t = 1$, i.e., $X^{t=1} \triangleq \bar{X}$.

Running the above procedure iteratively, the state estimation X^t at time step t can be achieved by registering the previous step estimation X^{t-1} to the current point cloud Y^t. Fig. 4.4 shows the structure of this state estimator. Note that since tracking is performed in sequences, and the rope shapes between adjacent time steps should not deviate much, only a few iterations of EM updates will register X^{t-1} to Y^t. Therefore, the proposed state estimator can be run efficiently in real-time. Besides, the estimator is robust to occlusion. During robot manipulation, the robot arm easily occludes the stereo camera, which results in missing points in the rope point cloud. However, since the transformation function, v is applied on source points coherently, X^{t-1} can still be registered to the missing point area in Y^t, i.e., the node position in the occluded area is still able to be obtained.

Task Planning

A complete manipulation task is usually composed of multiple procedures. For example, as shown in Fig. 4.5, there are two major steps to move the rope from a straight line to a 'Z' shape. At each procedure, a corresponding trajectory can be programmed by human operators to guide the robot to successfully manipulate the rope.

For autonomous manipulation, it is necessary for the robot to recognize at which procedure the current state is so that the most relevant trajectory from the human-programmed 'trajectory pools' can be selected for the following manipulation.

The CPD registration is utilized again to design the task planner. Suppose there are S procedures in total to manipulate the rope, and the initial state of the rope at each procedure is recorded as $X_s \in \mathbb{R}^{N\times 3}, s = 1, \cdots, S$. The current state of the rope, X^t, is estimated by the proposed observer (Section IV.A). CPD is then applied to register each recorded state X_s to the current state X^t. The log-likelihood function Q_s^t can be calculated after each registration (4.9). Note that Q_s^t is negative, and the less negative Q_s^t is, the more similar

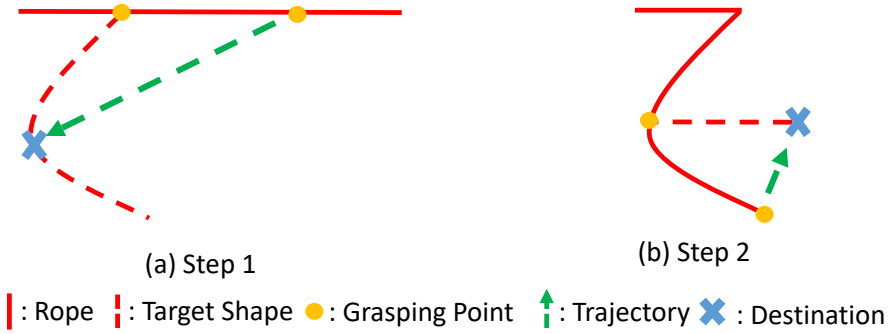


Figure 4.5: Two steps to move a straight line to a 'Z' shape.

between the states X_s and X^t. To normalize the similarity within the 0-100% range, the similarity matrix η_s^t is defined as follows:

$$\eta_s^t = \frac{Q_s^s}{Q_s^t} \tag{4.11}$$

where Q_s^s is the log-likelihood between X_s and itself. When the source point set and the target point set overlay,

$$p(y_m|n) = \begin{cases} 1, & n = m \\ 0, & otherwise \end{cases} \tag{4.12}$$

$$p(n|y_m) = \begin{cases} 1, & n = m \\ 0, & otherwise \end{cases} \tag{4.13}$$

Therefore,

$$Q_s^s = N \log(\frac{1}{N}) \tag{4.14}$$

and

$$\eta_s^t = \frac{N \log(\frac{1}{N})}{Q_s^t} \tag{4.15}$$

η_s^t approaching to 100% indicates stronger similarity between X_s and X^t. The most possible step that the current manipulation lies in can be determined by finding maximum similarity:

$$s^* = \arg\max_s \ \eta_s^t, \quad s = 1, \cdots, S \tag{4.16}$$

Moreover, the task planner can be applied to detect failures during manipulation. If the maximum similarity $\eta_{s^*}^t$ is smaller than a pre-defined threshold, η_{thre}, it indicates that the current rope state differs from all the scheduled steps. Rope manipulation runs into some

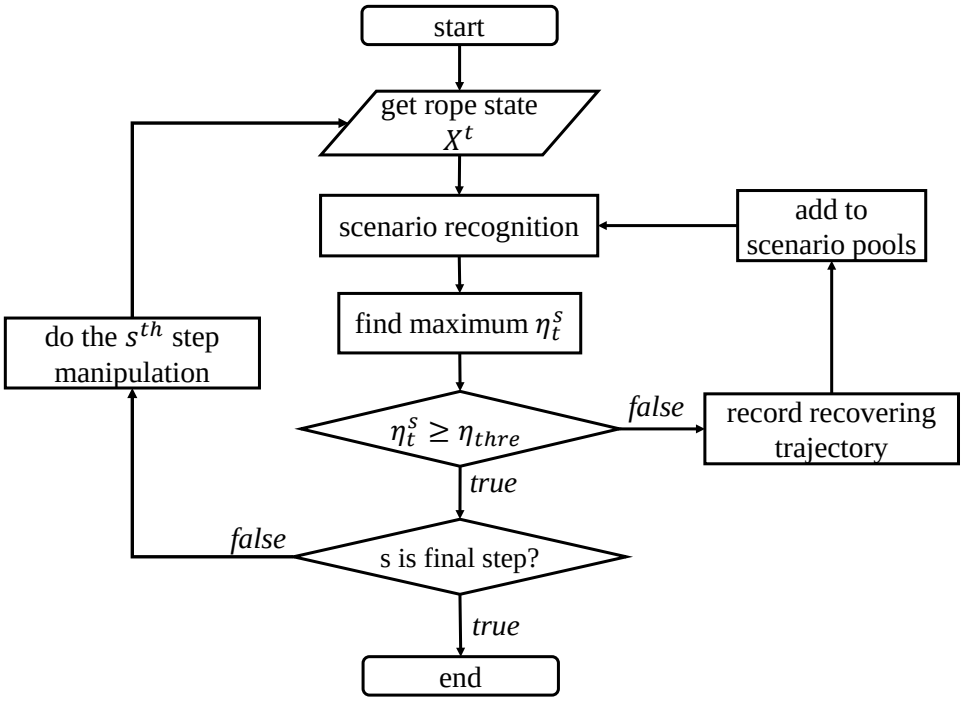


Figure 4.6: The framework of task planning. η_t^s represents the similarity between current state X^t and recorded steps. If the maximum η_t^s is smaller than a lower-bound η_{thre}, an unknown failure occurs. Human is asked to help robots recover the rope. The scenarios pools will be augmented so that when running into the same failure next time, it will be recognized and the taught recovering trajectory will be traced back.

unknown failures. Human operators need to interfere and teach robots some recovering trajectories to move the rope back to one of the recorded states. Then from that recorded state, manipulation can be continued autonomously. The failure state will also be augmented to the scenario pools X_s. If this similar failure occurs again in the future, no human interference is required, instead, the planner will use the taught trajectory for recovering as last time. The framework of the proposed task planning module is shown in Fig. 4.6.

Trajectory Planning

During Training, human operators program the end-effector's trajectory $T^s_{\text{train}},\ s = 1, \cdots, S$ for each of the S procedures. At the test, robots succeed in recognizing that the current rope is at sth step by the task planning module (Section IV.B). However, the sth step's corresponding trajectory T^s_{train} cannot be directly applied for the test scenario, since no matter how similar, there is always some minor shape difference between the rope at training

and that at test. This minor difference makes the exact replay of the training trajectory fail during the test, such as failing to grasp the rope. Therefore, T^s_{train} only serves as an approximate reference, while some trajectory replanning based on T^s_{train} is required to achieve a feasible manipulation at test time.

Note that when registering X_s to the current rope shape X^t during task planning, besides the likelihood function Q^t_s, the transformation function $v : \mathbb{R}^3 \to \mathbb{R}^3$ is also constructed (see (4.6),(4.8)). As shown in Fig. 4.13, v transforms X_s to align to X^t by twisting the original Cartesian space. v can also be utilized here to transform the end-effector trajectory T^s_{train} to get a new trajectory T^s_{test} that is feasible for the rope at test.

The trajectory T of a robot end-effector can be regarded as a sequence of poses $\{p, R\}$, where $p \in \mathbb{R}^3$ is the position vector of the end-effector, and $R \in \text{SO}(3)$ is the orientation matrix. With this observation, the feasible trajectory T^s_{test} can be achieved by applying the following transformation on T^s_{train}:

$$p_{test} = p_{train} + v(p_{train}) \tag{4.17}$$

$$R_{test} = orth(J_v(p_{train}) \cdot R_{train}) \tag{4.18}$$

$J_v(p)$ is the Jacobian matrix of v evaluated at position p, and $orth(\cdot)$ is a function that orthogonalizes matrices. If v is a rigid transformation T_v, this trajectory transformation procedure is equal to left-multiplying each end-effector pose T_{train} by T_v.

4.5 Experiments and Results

A series of experiments were performed to test the proposed state estimation, task planning, and trajectory planning algorithms for manipulating soft ropes. The task is that two FANUC LR-Mate 200iD robots collaboratively knot a 1-meter-long soft rope from a random initial shape.

State Estimation

As shown in Fig. 4.7, the Microsoft Kinect was utilized to monitor the environment at 10Hz. The captured 640×480 RGB and depth images were sent to a Ubuntu 14.04 desktop (Intel i7@3.60 GHz + RAM 16GB) synchronously and then synthesized to get the environmental point cloud. Since object segmentation is not our focus in this work, we simply placed a red rope on a green or white color background and implemented a color-based filter to segment out the rope from the environment. The rope's point cloud was then downsampled to 200 points uniformly by the VoxelGrid filter.

For state estimation, the 1-meter rope was discretized and modeled by 50 linked capsules. A CPD toolbox implemented with C++ was utilized to register the 50 nodes' positions to the rope's point cloud. The point sets were first normalized to zero mean and unit variance before registration. The weight μ for uniform distribution was chosen to be 0.1.

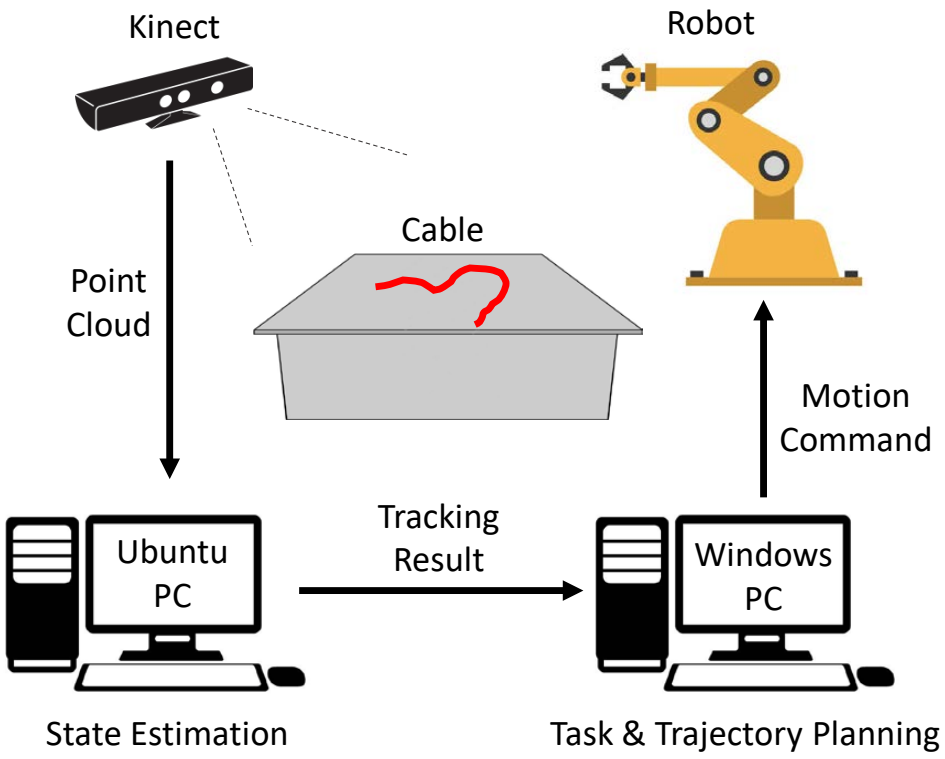


Figure 4.7: The testbed setup.

Table 4.1: Execution time of State Estimation

Segmentation	7 ms
Downsampling	2 ms
CPD Registration	10 ms
Total	∼20 ms

Smoothness regularization parameter λ and Gaussian kernel's variance β were set as 3.0 and 2.0 respectively. All the data points were denormalized after registration.

Fig. 4.9 and Fig. 4.8 show the real-time tracking results when the two robots were manipulating the rope. Note that the rope's point cloud was noisy, containing outliers and occluded by the robot arms from time to time, but the proposed state estimator could still track the rope robustly and efficiently. Table 4.1 lists the major execution time of the state estimator. The overall running time is less than 20ms.

As shown in Fig. 4.10, to analyze the tracking accuracy, 11 markers with distinct colors were attached to the rope with $10cm$ intervals. These markers were distinguished by a color-based filter and their ground-truth positions were measured directly from Kinect. Note that

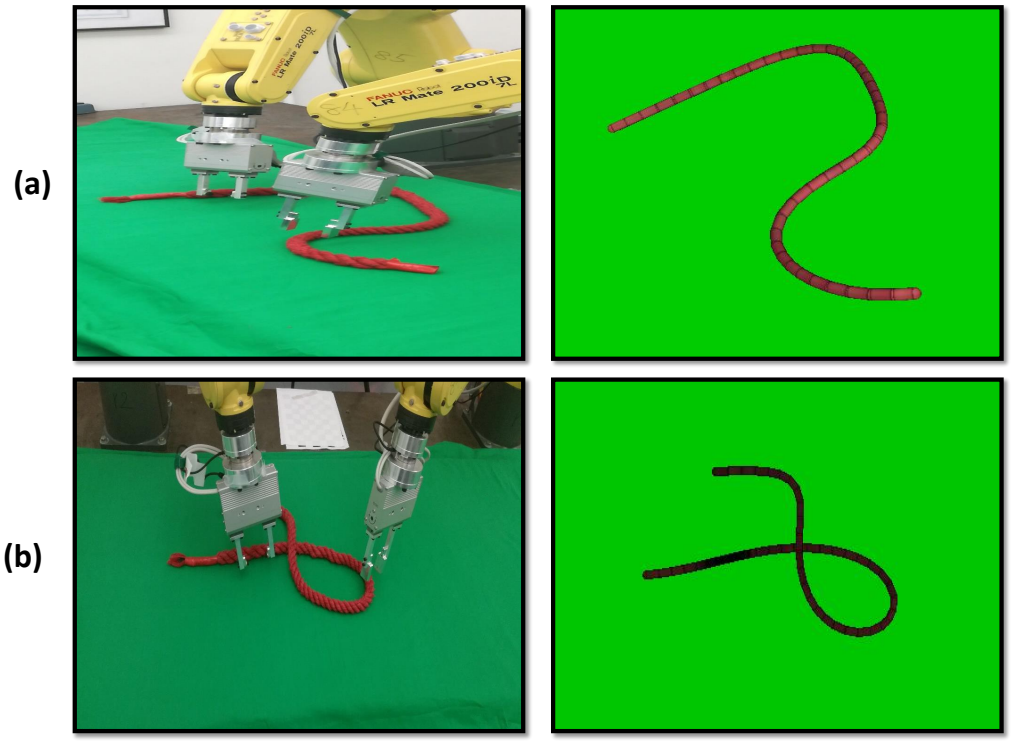


Figure 4.8: Snapshots during the real-time tracking experiments.

Table 4.2: Estimation Error of State Estimation

Rope configuration	Estimation Error (cm)
Straight Line	0.63 ± 0.11
L-Shape	1.49 ± 0.37
Circle	2.15 ± 1.06

these markers were only used for the purpose of ground truth. They were not utilized in our state estimation algorithm. The estimation results from our proposed method were compared with these ground-truth values. Fig. 4.11 shows the average tracking error and standard deviation at six different rope configurations. In general, the tracking error is less than 2.2cm, smaller than the jaw width (6cm) of the robot gripper. Therefore, even if the gripper went to an inaccurate grasp pose because of the tracking error, the rope was still located between the gripper's fingers and could be successfully grasped. We also compared our tracking with the MEM method. Fig. 4.11 shows that their tracking performance is similar. However, our proposed framework advances in extendibility since its application is not limited to state estimation, but can also be applied to task planning and trajectory planning for deformable object manipulation.

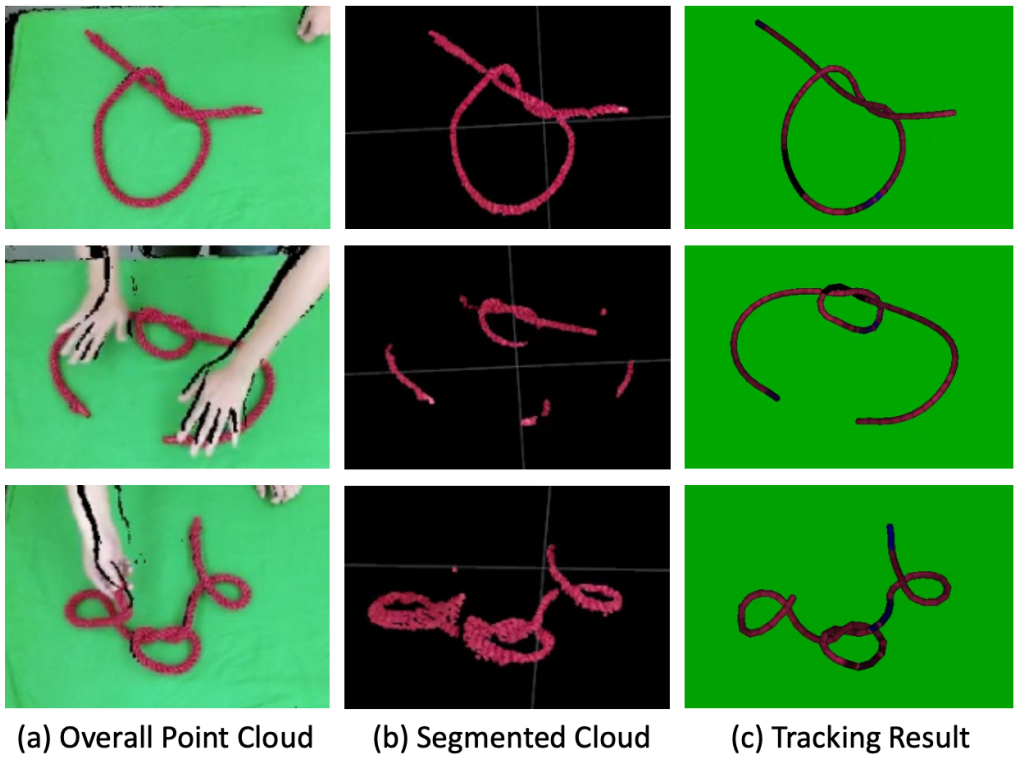


Figure 4.9: Snapshots during the real-time tracking experiments.

Task Planning

Following the pipeline in Fig. 4.7, the state estimation result was then sent to a Windows desktop (Intel i7@3.60 GHz + RAM 8GB), which ran the task planning and trajectory planning algorithms. ROS was utilized to serve as the interface to communicate between the Kinect, the Ubuntu PC, and the Windows PC.

As shown in Fig. 4.13(a), four major steps were predefined by human operators for the task of rope knotting. At each step, human operators placed the rope into one initial shape which was recorded by the state estimator. Then the corresponding manipulation trajectory was demonstrated by lead through teaching. To be specific, operators guide the two robots' end-effectors through some key poses, and then the training trajectory is obtained by linear interpolation between neighbor poses.

At the test, the current rope states were estimated and compared by CPD to each of the four recorded templates. The similarity level is calculated by (4.15). As shown in Fig. 4.12, the red point set (current state) and the blue point set (recorded state) in the second image had the largest similarity (90%), which indicated that the manipulation process was at the second step at that moment. The similarity lower-bond η_{thre} was set as 80%. If all the similarity check is below 80%, a warning message will be shown to ask the human operator

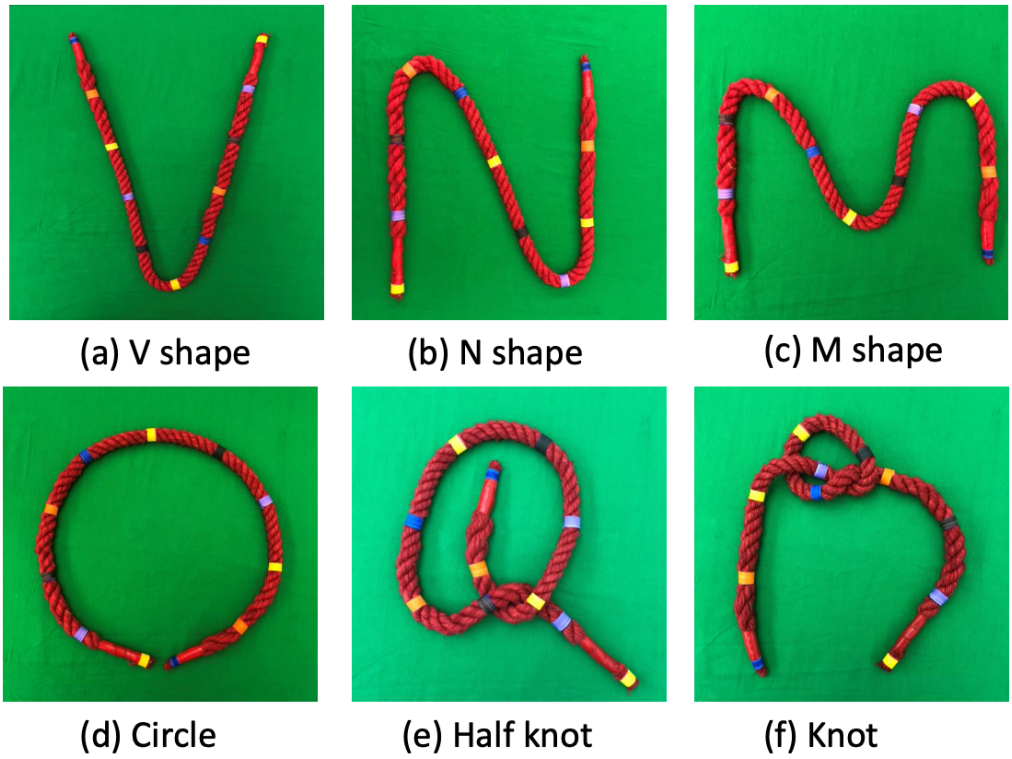


Figure 4.10: Benchmark setting: Six configuration shapes with markers attached.

to demonstrate a recovering trajectory. The failure states and recovering trajectories were then augmented into the task planning sample pools. The robot's ability to detect and recover from failures is shown in the attached videos.

Trajectory Planning

After identifying the task step, the manipulation trajectory was generated by the proposed trajectory planning algorithm.

To represent the rope positions and the manipulation trajectory under the same coordinate system, the relative translation (extrinsic parameter) between the Kinect. The rope states were all translated to the robot world frame afterward.

As shown in Fig. 4.13, the rope state during the test was different from that in training. With CPD registration, the training trajectory was transformed by (4.17) and (4.18) at each step to achieve the test trajectory. The end-effector poses were then transformed to robot joint command by robotic inverse kinematics. The joint command was finally sent to the robot controller for execution.

Fig. 6.7 shows the snapshots of autonomous rope knotting by two robot arms. Three types of knotting were designed, with each type tested 15 times. The overall success rate

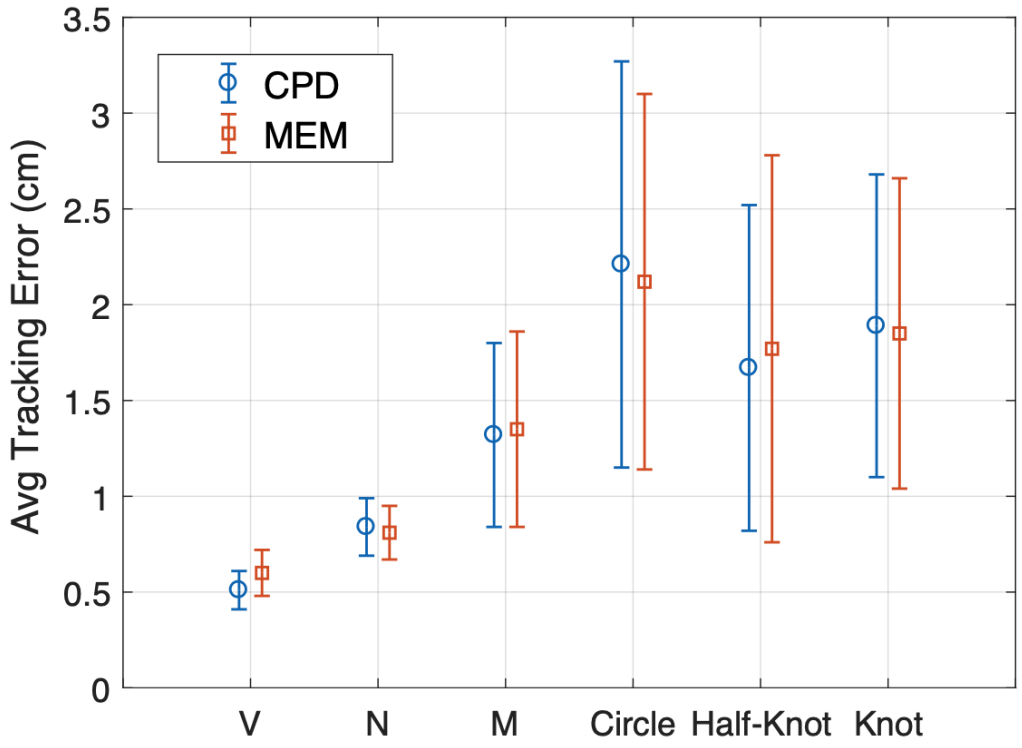


Figure 4.11: Average tracking errors and standard deviations at the marker positions.

was 40/45. Most of the failure was miss-grasping, which might have resulted from the relatively low accuracy of Kinect and calibration errors between the camera frame and robot base frame.

To quantify the accuracy of the calculated trajectory, the similarity between the rope's final shape in training and at test was achieved by (4.11) to evaluate the distortion. In addition, the execution time of the trajectory planning was also recorded to test the efficiency. Since the accuracy and execution time highly rely on the shape of the rope and the complexity of the manipulation step, a benchmark is designed as shown in Fig. 4.15. In training, humans demonstrated a trajectory to manipulate the red rope into a desired shape. At the test, the initial state of the 1-meter-long rope was placed by the following rules:

(1) the positions P_0, P_1 (in both the x and y directions) remain the same as the positions in training.

(2) the distance from the root point P_0 to the corner point P_2, L_0, is changed from 40cm to 60cm.

(3) the angle θ_0 is changed from $+30\,\text{deg}$ to $+70\,\text{deg}$

According to the above rules, 100 different initial states of the rope were placed and manipulated by the robot. Table 4.3 lists the results of the test. For each trial, the average execution time of the trajectory planning algorithm is about 11ms. The primary time-

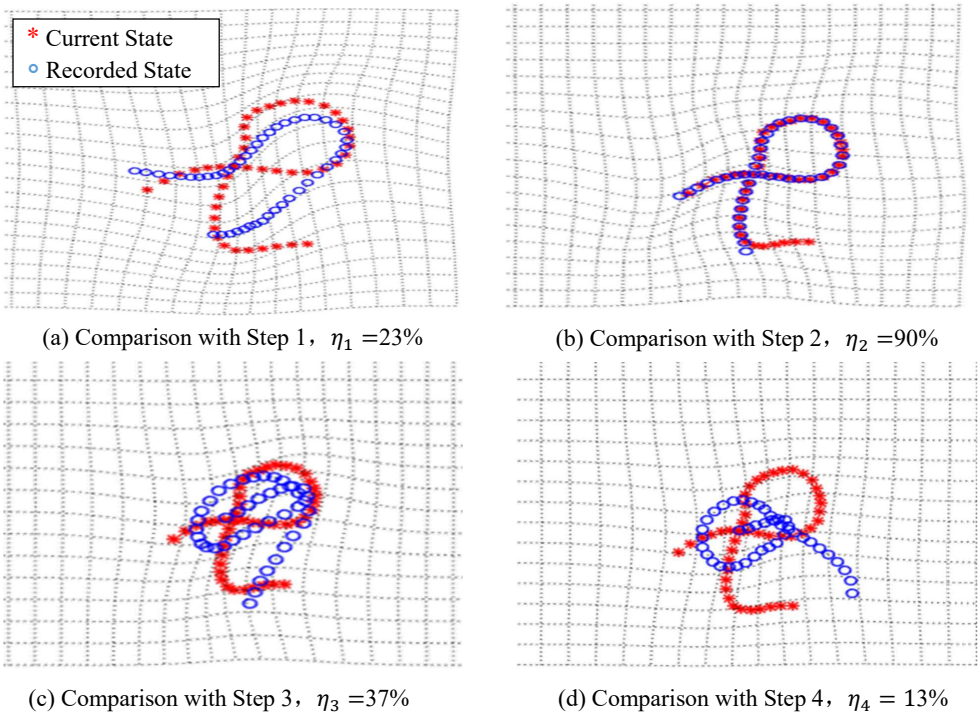


Figure 4.12: Similarity check between the current rope state (red dots) and the four recorded training states (blue dots). The second scenario is most similar to the current state with a 90% similarity.

Table 4.3: Result of Rope Manipulation Benchmark

Average Execution Time (ms)	Average Similarity (%)
11	90

consuming component of this part is the registration step. If following the entire framework, registration will be done in the scenario recognition step, hence, the execution time of trajectory planning will decrease to about 2ms. For shape distortion, the trajectory planning algorithm is able to guarantee about a 90% of similarity between the final state of the rope at test and in training. Even though the similarity may decrease when dealing with more difficult tasks, considering the high tolerance of distortion during rope manipulation, we can still conclude that this trajectory planning algorithm is able to generate a suitable trajectory efficiently.

To conclude, the experimental results show the proposed algorithms are able to manipulate ropes autonomously. The state estimation algorithm enables robots to track the rope states in real time; the task planning algorithm lets robots recognize the manipulation

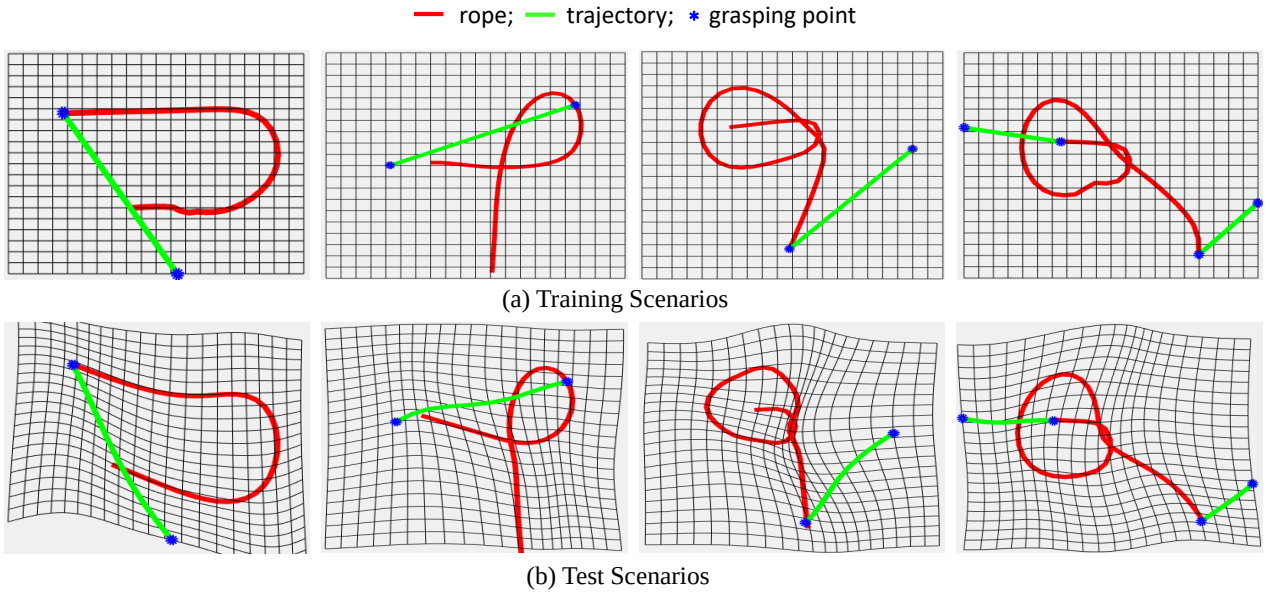
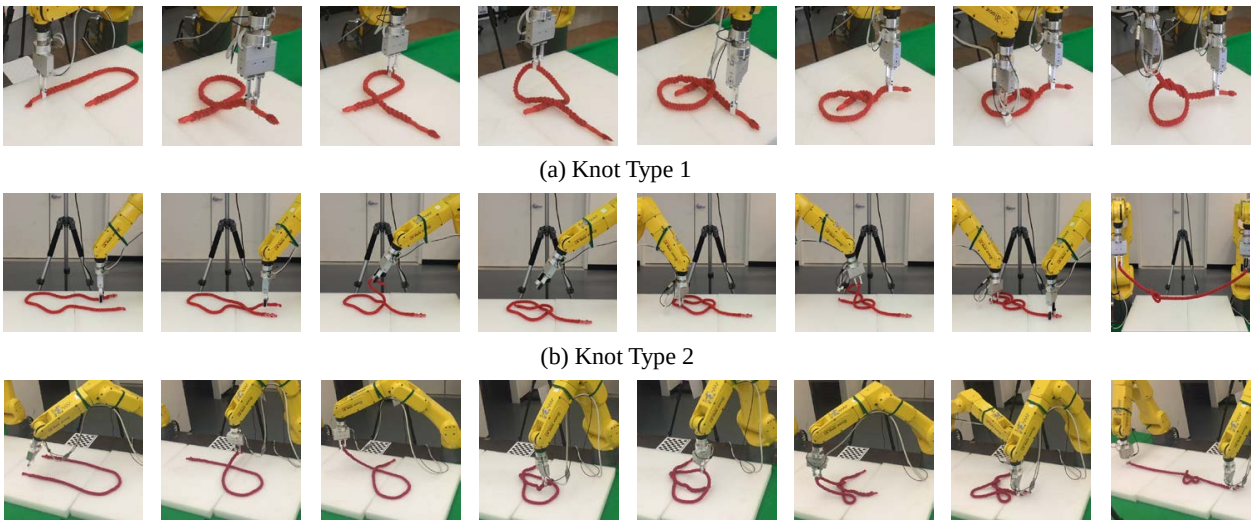


Figure 4.13: Four major steps for rope knotting manipulation. The red line is the rope state, and the green line is the trajectory of the robot end-effector. Blue dots are the grasping/releasing positions. Black grids show that the original Cartesian space is twisted so as to map the training scenarios toward test scenarios.

procedure and detect failure cases robustly; the trajectory planning algorithm can generate feasible trajectories for new scenarios by warping the old trajectories from training scenarios.

4.6 Chapter Summary

A uniform framework, which includes state estimation, task planning, and trajectory planning, is proposed and implemented in this chapter for manipulating deformable linear objects. Based on coherent point drift (CPD), a robust real-time observer is developed to estimate the position of each node on the rope from noisy and occluded point clouds. A task planner is then developed to let robots recognize which procedure the current manipulation is and decide which action to take. Finally, by registering the training scenario and the test scenario, a transformation function can be constructed and utilized to warp the training trajectory to achieve a feasible test trajectory. A series of experiments on ropes knotting tasks are implemented, which indicate the effectiveness of the proposed methods.



(a) Knot Type 1

(b) Knot Type 2

(c) Knot Type 3

Figure 4.14: Snapshots of the rope knotting experiments. Two robot arms were collaborating to knot the rope based on the transformed trajectory. (a)(b)(c) show three different types of knotting.

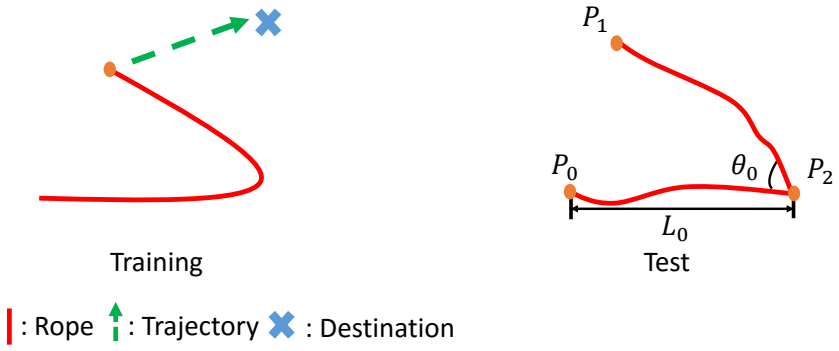


Figure 4.15: Test benchmark for trajectory planning.

Chapter 5

Offline-online Learning of Deformation for Robotic Cable Manipulation

In this chapter, following the introduction of our framework for real-time tracking and state estimation of deformable objects, we shift our focus to their reliable manipulation, achieved through a combination of offline and online learning methods. We address the challenge of manipulating objects such as cables by proposing a novel hybrid offline-online approach for learning their dynamic behaviors efficiently and robustly. In the offline phase, we utilize Graph Neural Networks (GNN) to learn the deformation dynamics of these objects from simulated data. This phase lays the foundational understanding of the object's behavior under various conditions. The online phase then builds on this foundation, where a linear residual model is developed in real-time. This model acts as a bridge, effectively closing the gap between the simulated environment and real-world scenarios. The learned model is then utilized as the dynamics constraint of a trust region-based Model Predictive Controller (MPC) to calculate the optimal robot movements. This integration enables the calculation of optimal movements by the robotic manipulator, adapting to the continuously updated model. The entire process, encompassing online learning and MPC, operates in a closed-loop system, ensuring robust and accurate task execution.

5.1 Introduction

Manipulating deformable linear objects, especially cables, has a wide range of applications. For example, in factory manufacturing, both stator winding and cable harnessing [31] require cables to be assembled in a precise manner. In medical surgery, steering needles or catheters are essential for vascular pathology treatment and minimally invasive surgery. To complete the above tasks automatically, robustly manipulating cables to the desired shape remains a fundamental problem for robotics. However, due to the high dimensionality and nonlinearity,

it is challenging to obtain an accurate dynamics model for precise planning and control.

The finite element method (FEM) serves as a powerful tool to model cables. Through discretizing the objects into small elements, the deformation can be derived by solving a set of partial differential equations. FEM can accurately represent the dynamics of deformable objects with fine tessellation [9]. However, it is usually computationally expensive to solve, thus posing a challenge to use it for real-time robotic manipulation tasks.

Aside from the FEM models, data-driven approaches are often utilized to learn the non-linear deformation dynamics [112, 26, 110]. Recently, many works have attempted to use deep neural networks (DNN) to learn such dynamics. However, the characteristic of deformable objects is complex, and it is found challenging for those general-purpose network structures to capture the cable behaviors.

Recent studies on graph neural networks (GNN) bring a new structure for model learning. By viewing particles of the object as the graph vertices, the dynamics are modeled as the interaction between the vertices pairs [79]. The future object state can be predicted through a sequence of 'message passing' blocks, which mimic the transmission of the interaction from one graph vertex to another. Compared with those general-purpose network structures, the GNN encodes the prior knowledge on how the interaction may transmit inside the deformable objects. However, as most of the machine learning-based approach does, the GNN model relies on simulation for data generation. Thus, the sim-to-real gap still poses a significant challenge for the model to be adopted in robotic manipulation tasks.

To tackle the above issues, we propose a dynamics learning framework that consists of an offline GNN model and an online residual model, as shown in Fig. 5.1. In the offline phase, we learn rough global graph dynamics from the simulation data. In the online phase, we further refine the local predictions through a time-varying linear residual model that estimates the error of the GNN output. In contrast with existing sim-to-real approaches, our method does not require pre-collecting real-world data. The linear residual model can be learned simultaneously as the robot executes. With this framework, we can efficiently capture the global deformation behaviors without sacrificing local accuracy. For manipulation, a trust region-based MPC formulation with the learned model is proposed to optimize the robot movement. In summary, the main contributions are listed as follows:

- Combine the GNN with an online linear residual model for robust model learning.
- Formulate a trust region-based model predictive controller to optimize the robot movements.
- Demonstrate the effectiveness of the proposed method through comparative simulations and experiments.

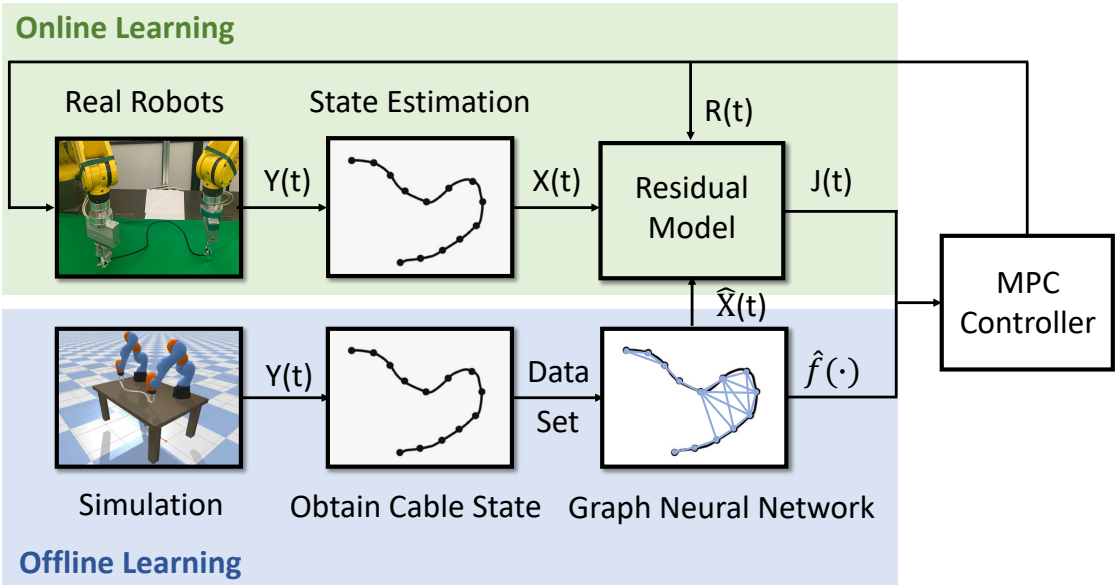


Figure 5.1: The proposed cable manipulation framework combines offline model learning with online residual learning. In the offline phase, we trained a GNN with simulated data as the global model of the cable. In the online phase, a local residual model that estimates the GNN prediction error is learned in real-time to reduce the sim-to-real gap. The learned models are then sent to an MPC controller to obtain the optimal robot motion.

5.2 Related Works

Global Deformation Model Learning

Learning-based approaches approximate the cable dynamics from the collected data set. Nair et al. [63] proposed a predictive model learned from step-by-step images of a rope collected during human demonstration. Yan et al. [109] proposed to utilize bi-directional LSTM to model chain-like objects. By recurrently applying the same LSTM block along the cable, we can enforce how the interaction will propagate inside the model. Similarly, the graph neural network arises as a novel structure to learn the deformation dynamics. By explicitly modeling the vertices and edges, authors of [45] demonstrated the effectiveness of the proposed dynamic particle interaction networks (DPI-Nets) on fluids and deformable foam manipulation tasks. The graph network structure proposed in [79] further improved the generalizability in terms of prediction steps and particle scalability. Moreover, Pfaff et al. [74] generalized the graph structure to mesh-based dynamics, which proved to be effective for more complex simulation tasks.

Sim-to-real Transfer for Model Learning

While neural networks can capture complex behaviors of deformable objects, those methods typically require sim-to-real methods as complements in case of parameter changes or uncertainties encountered in the real-world experiment. Matching the simulator parameters with real-world parameters and directly applying the learned model to the real world is the most common way to utilize the model. However, the learned model may suffer from uncertainties in the real world. Another approach is to randomize several aspects of the domain to provide enough variability in training [96, 11, 76]. As a result, the learned model can cover a large fraction of scenarios. Fine-tuning the learned model with real-world data is also demonstrated to be effective in closing the sim-to-real gap [109]. By collecting new data in the real world and re-training the network, we can adapt the model to the new scenario robustly.

Combining the offline model with a residual model is also beneficial to resolve the sim-to-real gap [2, 34]. Since the offline model does not need to match reality accurately, training in simulation is more efficient and more manageable.

Online Local Model Estimation

Instead of learning the global model and transferring it to the real world, online approximating a local model is another powerful way for cable manipulation. Visual servoing is a commonly used method for online cable manipulation. Visual servo approaches approximate the local linear deformation model by iteratively updating the deformation Jacobian [67, 121, 122], which represents the mapping between the robot end-effector's velocity and the object state. Compared with offline global model learning approaches, there are no sim-to-real gaps, and it can generalize to different objects and scenarios. Jin et al. [28] further improved the robustness of the online linear model under sensor noises and occlusions. However, as the linear model's expressiveness is limited, it is challenging for the online methods to manage large deformations.

Offline-Online Learning for Robust Manipulation

Though the online model estimation methods are useful in certain tasks, it is still desirable to take advantage of the more powerful offline global model. As demonstrated in [112, 111, 113], the offline learning provides the initial guess of the linear Jacobian matrix, and in the online phase, the Jacobian matrix is further updated with an adaptive control law. Distinct from the sim-to-real methods introduced in the previous subsection, this combination does not require additional online data collection. The sim-to-real gap can be instantaneously resolved as the robot executes. While the motivation of our proposed framework is similar to the motivation of [112], our algorithm is unique in several aspects: 1) we utilize GNN to capture the global model instead of providing the initial guess of the local Jacobian, 2) we

online learn a residual model for refinement, 3) we construct an optimization-based MPC controller to obtain optimal robot motions within a long horizon.

5.3 Proposed Framework

In this section, we introduce the proposed framework for cable manipulation as shown in Fig. 5.1. We mainly focus on the task of precisely shaping a cable to the desired curvature in a 2D and uncluttered environment. We utilized the proposed state tracking algorithm in Chapter 4 to get an estimation of the cable state. Then the cable dynamics are approximated by a combination of an offline graph neural network (GNN) and an online residual model. In the end, a model predictive controller (MPC) is utilized to control the robot to manipulate the cable to the desired states.

Offline Model Learning with Graph Neural Networks

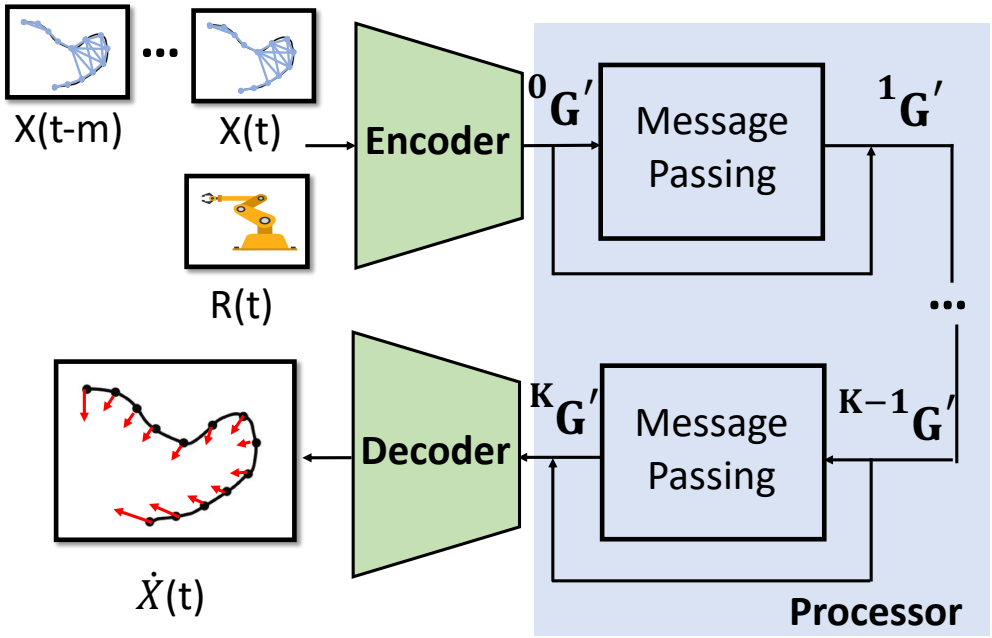


Figure 5.2: The structure of the Graph Neural Network (GNN). The offline GNN takes in a graph that represents the state of the cable and outputs the prediction of the cable movements.

Graph neural networks (GNN) are demonstrated to be effective in representing complex dynamics [79]. In this chapter, we adopt a similar idea from [79] to learn the graph dynamics of the cable.

We assume that the connection between the robot end-effector and the grasping point is fixed throughout task executions. The dynamics $f(\cdot)$ is then modeled by the interactions among the cable key points and the robot end-effectors as shown in (5.1). The dynamics takes in $m+1$ previous cable states from $X(t-m)$ to $X(t)$and the current robot end-effector velocities (translations in x, y and rotation around z) $R(t) = [r_1(t), r_2(t), \cdots, r_Q(t)]^T$, where $r_i(t) \in \mathbb{R}^{1\times 3}$ is the end-effector velocities of the i-th robot, and Q is the number of robots.

$$\dot{X}(t) = f(X(t-m:t), R(t)) \tag{5.1}$$

As studied in [79], the dynamics can be captured via a graph $G = (V, E)$, where the graph vertices $V = [v_1, v_2, \cdots, v_N]$ correspond to the key points, and the graph edges E correspond to the interactions between the key points pair. As shown in Fig. 5.2, the features of the i-th graph vertex v_i consist of a sequence of previous key point positions $[x_i(t), x_i(t-1), \cdots, x_i(t-m)]$, and the robot control input at that point. For the graph edge, it models the relative movement between vertices $e_{i,j} = \|x_i(t) - x_j(t)\|$. An edge is constructed if two vertices are within a 'connective radius'.

To learn such a graph, we follow the GNS network structure [79], which contains three steps – encoding, processing, and decoding:

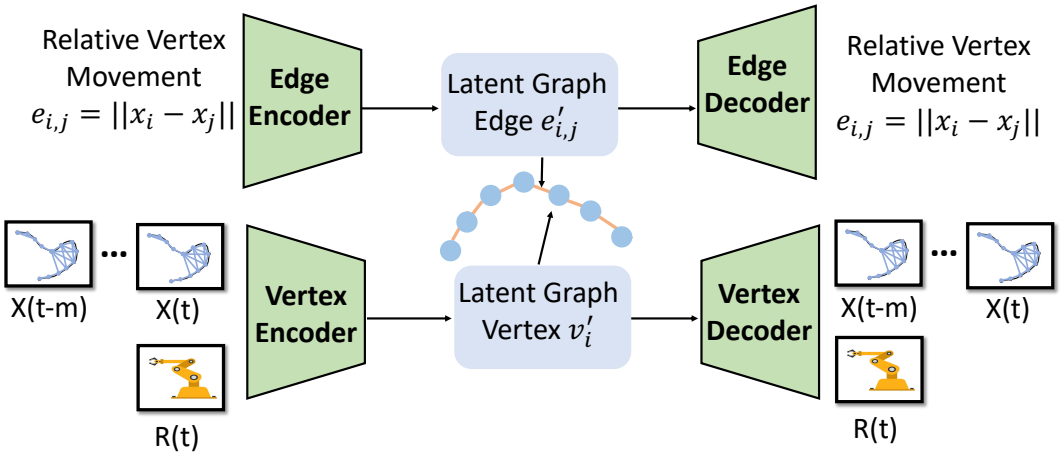


Figure 5.3: Illustration of the graph encoder and decoder. By minimizing the reconstruction loss in (5.2), the encoder can project the high dimensional graph representation to a latent space.

Encoding: Instead of operating in the high dimensional space, we project the graph G to a low dimensional representation $G' = (V', E')$ for efficiency. Graph vertices auto-encoder and graph edges auto-encoder are trained, respectively as shown in Fig. 5.3. The reconstruction error in (5.2) is minimized to make the network learn the optimal latent space that can capture all the original information. ϕ^v, ϕ^e denote the encoder for the graph vertices

and edges, and ψ^v, ψ^e are the corresponding decoders.

$$\begin{aligned} \phi^{v*}, \psi^{v*} &= arg\min_{\phi^v, \psi^v} \|v_i - (\phi^v \circ \psi^v)(v_i)\|^2 \\ \phi^{e*}, \psi^{e*} &= arg\min_{\phi^e, \psi^e} \|e_{i,j} - (\phi^e \circ \psi^e)(e_{i,j})\|^2 \end{aligned} \tag{5.2}$$

Processing: The processor mimics the dynamics of the cable by computing the interaction between vertices in the latent graph.The message passing block in Fig. 5.4 is to propagate the graph vertex, and edge features based on the current graph state and the robot control input as shown in (5.3).

$$\begin{aligned} {}^{k+1}v'_i &= f^v({}^kv'_i, \sum_j {}^ke'_{i,j}) \\ {}^{k+1}e'_{i,j} &= f^e({}^ke'_{i,j}, {}^kv'_i, {}^kv'_j) \end{aligned} \tag{5.3}$$

where $f^v(\cdot)$ and $f^e(\cdot)$ are the graph vertex network and graph edge network, and ${}^{k+1}v'_i$, ${}^{k+1}e'_{i,j}$ denote the latent vertex/edge features after the k-th message passing block.

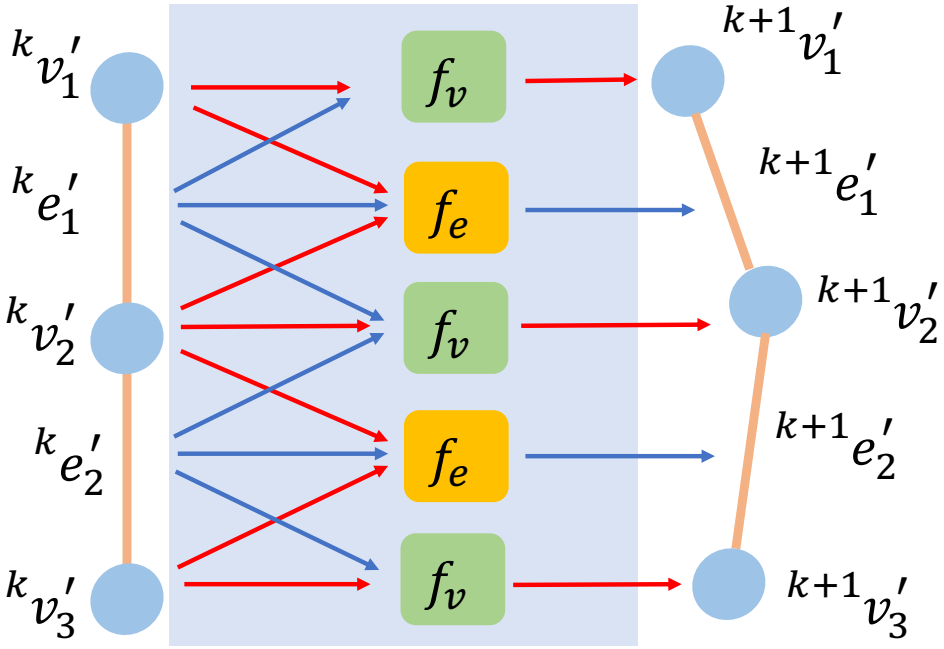


Figure 5.4: Illustration of the message passing block. Each message-passing block transmits the interaction from one graph vertex to others through the graph edge.

Decoding: As the inversion of the encoder, the decoder converts the latent graph representation back to get the prediction of the cable movement $\dot{X}(t)$. The weight is directly taken from the decoder part of the auto-encoder ψ^v and ψ^e.

Online Residual Model Learning

The offline GNN model is able to learn the global nonlinear dynamics and provide a rough prediction. However, due to the sim-to-real gap, the training data from the simulation may not accurately capture the dynamics of the physical object. To overcome the sim-to-real gap and the generalization issues, we propose to online learn a residual model to correct the local predictions. As shown in (5.4), $\delta\dot{X}(t)$ is the error between the actual cable state increment $\dot{X}(t)$ and the predicted cable state increment $\widehat{\dot{X}}(t)$ given by the offline learned dynamics denoted as $\widehat{f}(\cdot)$. For simplicity, we use X to represent the history key point trajectory $X(t-m:t)$. Given the history cable trajectory X, we assume that the ground truth residual model is linear with respect to $R(t)$ inside $\|R(t)\| \leq \epsilon$, where $J(t)$ denotes the local residual dynamics. For simplicity, we omit the bias term. It can be considered without much modification.

$$\begin{aligned} \delta\dot{X}(t) &= \dot{X}(t) - \widehat{\dot{X}}(t) \\ &= f(X, R(t)) - \widehat{f}(X, R(t)) \\ &= R(t)J(t), \quad \|R(t)\| < \epsilon \end{aligned} \tag{5.4}$$

To estimate the residual, we online collect $\delta\dot{\mathbf{X}}$ and the corresponding robot end-effector velocities $\mathbf{R}$ during the task execution as shown in (5.5).

$$\begin{aligned} \delta\dot{\mathbf{X}} &= \begin{bmatrix} \delta\dot{X}(t-m) & \cdots & \delta\dot{X}(t-1) & \delta\dot{X}(t) \end{bmatrix}^T \in \mathbb{R}^{(m+1)\times 2N} \\ \mathbf{R} &= \begin{bmatrix} R(t-m) & \cdots & R(t-1) & R(t) \end{bmatrix}^T \in \mathbb{R}^{(m+1)\times 3Q} \end{aligned} \tag{5.5}$$

Based on the collected data $\delta\dot{\mathbf{X}}$ and $\mathbf{R}$, we solve a regularized least squares (ridge regression). The problem can be decomposed to a $3Q$ independent ordinary ridge regression problem and solved in parallel. $\delta\dot{\mathbf{X}}_n$ denotes the n-th column of $\delta\dot{\mathbf{X}}$, and similarly, $J_n(t)$ represents the n-th column of $J(t)$.

$$\begin{aligned} \widehat{J}(t) &= \underset{J(t)}{arg\min} \quad \|\delta\dot{\mathbf{X}} - \mathbf{R}J(t)\|_F^2 + \lambda\|J(t)\|_F^2 \\ &= \sum_{n=1}^{6N} \underset{J_n(t)}{arg\min} \quad \|\delta\dot{\mathbf{X}}_n - \mathbf{R}J_n(t)\|_2^2 + \lambda\|J_n(t)\|_2^2 \end{aligned} \tag{5.6}$$

The solution is given below, where λ is the regularization weight, and I denotes the identity matrix.

$$\widehat{J}_n^*(t) = (\delta\dot{\mathbf{X}}_n{}^T\delta\dot{\mathbf{X}}_n + \lambda I)^{-1}\delta\dot{\mathbf{X}}_n{}^T\mathbf{R} \tag{5.7}$$

Model Predictive Control for Manipulation

To utilize the learned model for manipulation, we adopt the idea of model predictive control (MPC).

$$\begin{aligned} \min_{R(0:h)} \quad & \sum_{t=1}^{h+1} \|X(t) - X_d\|_2^2 \\ \text{s.t.} \quad & X(t+1) = X(t) + \dot{X}(t)\Delta t \\ & \dot{X}(t) = \widehat{f}(X, R(t)) + R(t)\widehat{J}(t) \\ & \|R(t)\| \leq \epsilon \\ & t = 0, 1, \cdots, h \end{aligned} \tag{5.8}$$

As shown in (5.8), the optimization variables are the robot's end-effector velocities within a horizon h, and the objective function is to minimize the shape position error. The constraints include the learned dynamics, where Δt denotes the sample time of the dynamics. More importantly, we consider movement limit constraints to ensure that the learned model (offline and online) is in effect in the optimized region. Borrowing the idea from the trust-region optimization, we update the trust-region size according to the ratio ρ between the actual shape error reduction Δe_{actual} and the predicted shape error reduction Δe_{pred} as shown in (5.9). The pseudo-code is summarized in Alg. 2, where the trust-region size will expand/shrink if ρ is greater/smaller than a threshold.

$$\rho = \frac{\Delta e_{actual}}{\Delta e_{pred}} \tag{5.9}$$

Algorithm 2: Trust Region Based Model Predictive Control

Require: Initialize R, ϵ
1: **while** $\|X(t) - X_d\| \geq e$ **do**
2: $R \leftarrow$ Solve the optimization in (5.8)
3: $X(t+1) \leftarrow$ Execute the robot and obtain the new cable state
4: $\rho \leftarrow$ Calculate the ratio according to (5.9)
5: **if** $\rho \geq \eta^+$ **then**
6: $\epsilon \leftarrow \tau^+ \epsilon$
7: **else if** $\rho \leq \eta^-$ **then**
8: $\epsilon \leftarrow \tau^- \epsilon$
9: **end if**
10: **end while**

During the task execution, the MPC controller iteratively solves for the optimal robot movements and sends the results to the robot for execution. Similar as our previous work [102], the optimization is solved by IPOPT [100], which implements a primal-dual interior-point linear search algorithm. The local convergence is guaranteed as proved in [99]. We empirically demonstrated the effectiveness through simulation and experiments.

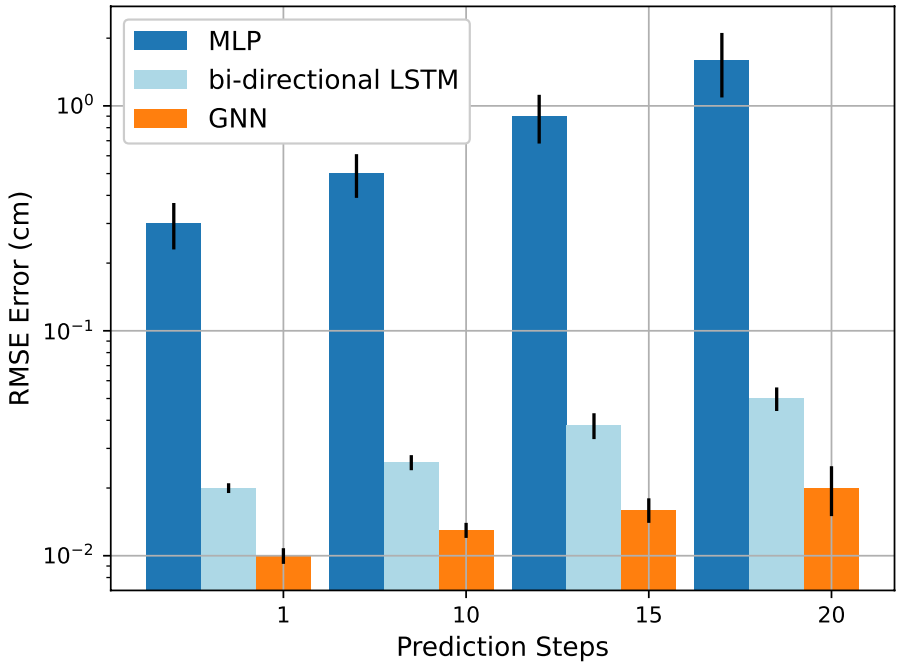


Figure 5.5: The relation between prediction error and rollout steps of various offline models. The models are forwarded 1 step to 20 steps with the same cable that is utilized for training. The y-axis is log-scaled.

5.4 Simulation and Experiments

Simulation and Experiment Setup

1) **Simulation Setup:** The simulation setup is shown in Fig. 5.7, two KUKA robots move the ends of a cable so that the cable matches the desired shape. The simulation environment is built upon the PyBullet Physics Engine. The cable model is generated by Gmsh and simulated by the built-in finite element method (FEM). Detailed cable simulation parameters are summarized in Table 6.2.

2) **Experiment Setup:** The experiment setup is shown in Fig. 5.1. Two FANUC LR-Mate 200iD robots collaboratively manipulate the cable to different desired shapes. An Intel Realsense RGB-D camera is utilized to obtain the cable point clouds. The online model learning and MPC calculation are achieved on a Ubuntu 18.04 desktop. The optimized robot velocities are sent to the robot for execution with a communication frequency of $100Hz$.

Table 5.1: Parameter Values of the Proposed Approach

Simulator Parameters	Value
Cable Length	$1m$
Cable Diameter	$0.01m$
Spring Elastic Stiffness	4×10^3
Spring Damping Stiffness	2×10^3
Spring Bending Stiffness	3×10^3
Global Scale	2
GNN Model Parameters	**Value**
Encoder	
Number of Hidden Layers	2
Size of Hidden Layers	128
Activation Function	ReLU
Processor	
Time Window Size (m)	5
Number of Message Passing (k)	10
Connective Radius	$0.2m$
Number of Hidden Layers	2
Size of Hidden Layers	128
Activation Function	ReLU
Decoder	
Number of Hidden Layers	2
Size of Hidden Layers	128
Activation Function	ReLU
GNN Training Parameters	**Value**
Batch Size	1
Learning Rate	10^{-4} to 10^{-6}
MPC Parameters	**Value**
MPC Horizon (h)	5
Dynamics Sample Time (Δt)	$1s$
Regularization Weight (λ)	10
Initial Trust Region Size	$0.05m$
η^+, η^-	0.8, 0.4
τ^+, τ^-	1.05, 0.95
optimizer	IPOPT

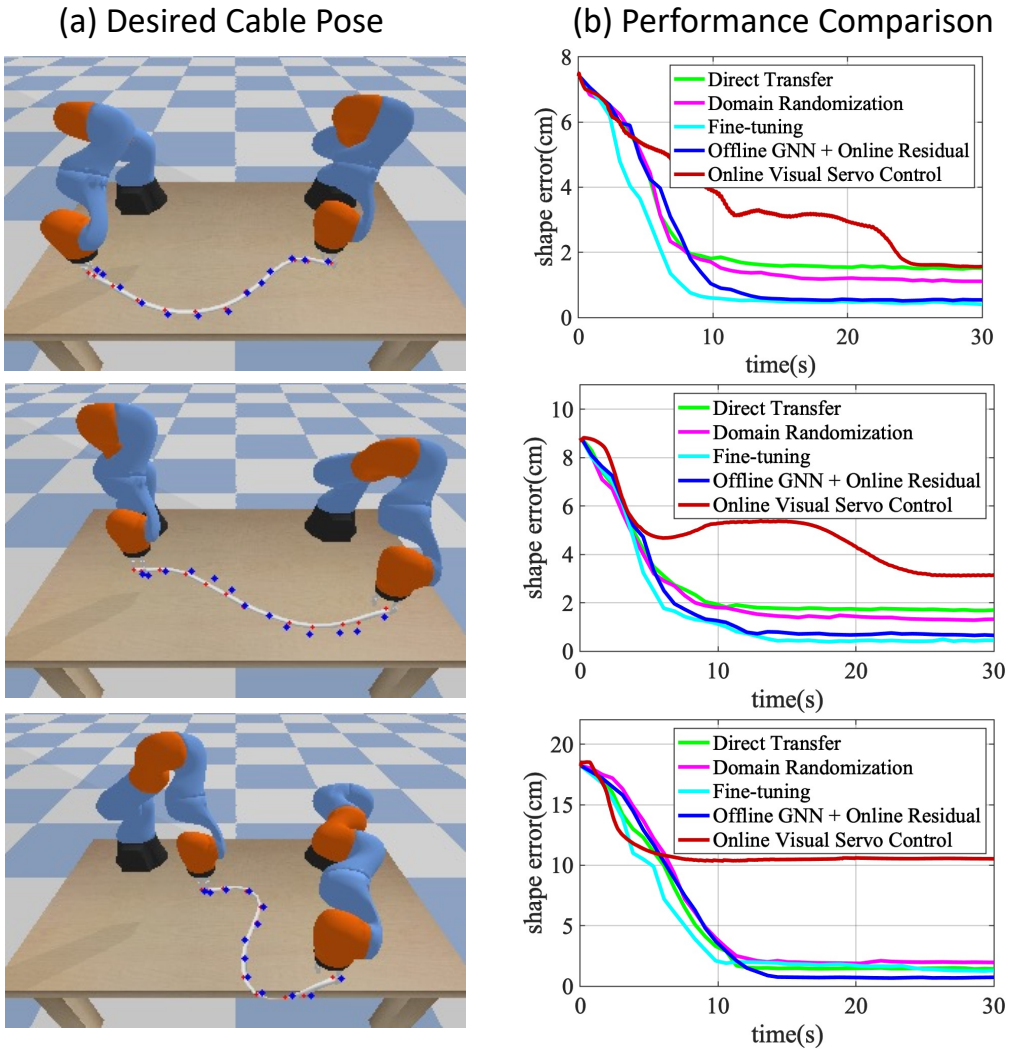
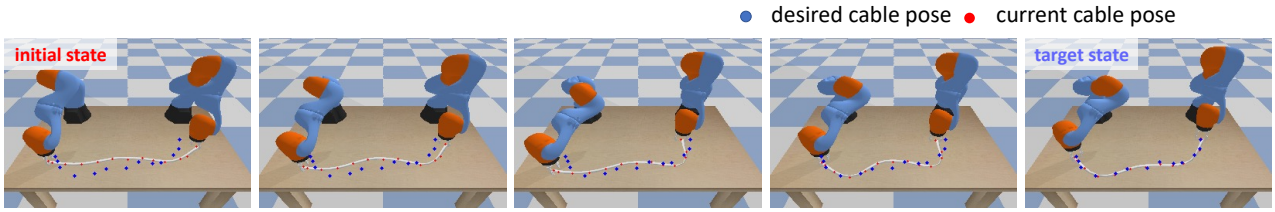


Figure 5.6: Benchmark comparison results. The robots will manipulate the cable from a straight line to three desired shapes. The curve shows the result in the environment that we double the cable stiffness.

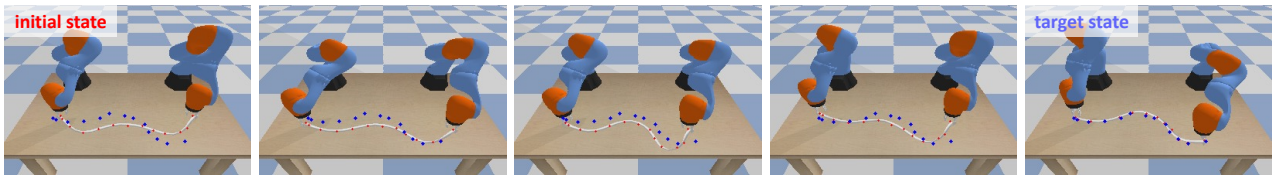
Table 5.2: Manipulation Performance Comparison of in Simulation

	Scenario 1: U shape		Scenario 2: S shape		Scenario 3: Z shape	
	Terminal Error(cm)	Settling Time(s)	Terminal Error(cm)	Settling Time(s)	Terminal Error(cm)	Settling Time(s)
Online Visual Servo	1.96 ± 0.29	27.47 ± 5.44	3.28 ± 0.82	24.76 ± 3.27	8.72 ± 1.15	19.43 ± 6.11
Direct Transfer GNN	1.54 ± 0.53	11.32 ± 1.46	1.61 ± 0.42	$\mathbf{13.98 \pm 1.31}$	1.87 ± 0.56	11.91 ± 1.75
Domain Randomization	1.11 ± 0.21	12.60 ± 1.32	1.30 ± 0.33	14.33 ± 1.47	1.76 ± 0.41	12.07 ± 1.66
GNN + Fine-tune	$\mathbf{0.42 \pm 0.10}$	$\mathbf{8.31 \pm 0.72}$	$\mathbf{0.57 \pm 0.08}$	13.58 ± 1.06	1.19 ± 0.23	$\mathbf{9.82 \pm 1.32}$
GNN + Online Residual	0.58 ± 0.13	12.35 ± 1.09	0.61 ± 0.14	13.79 ± 1.50	$\mathbf{0.78 \pm 0.17}$	16.32 ± 1.15

* Fine-tuning requires additional data collection before the robot execution, and its performance depends on the quality of the online dataset.



(a) Test 1: Random Initial and Target Cable States



(b) Test 2: Random Initial and Target Cable States

Figure 5.7: Snapshots of the proposed method. The proposed method is tested to manipulate the cable from random initial shapes to randomly generated targets. With the GNN model, the robot is able to find a rough global trajectory, while the online residual model is able to refine the local behaviors.

Table 5.3: Performance Comparison in Experiments with the Ethernet Cable

	Scenario 1: U shape		Scenario 2: S shape	
	Terminal Error(cm)	Settling Time(s)	Terminal Error(cm)	Settling Time(s)
Online Visual Servo	5.58 ± 1.11	52.04 ± 12.11	1.64 ± 0.28	44.18 ± 8.00
Direct Transfer GNN	2.54 ± 0.61	25.23 ± 5.19	1.57 ± 0.35	24.63 ± 5.16
GNN + Domain Randomization	1.91 ± 0.59	36.29 ± 5.45	1.36 ± 0.34	$\mathbf{16.73 \pm 4.37}$
GNN + Fine-tune	1.56 ± 0.43	$\mathbf{22.29 \pm 5.63}$	1.12 ± 0.33	23.03 ± 4.80
GNN + Online Residual	$\mathbf{1.03 \pm 0.24}$	24.29 ± 7.92	$\mathbf{0.99 \pm 0.16}$	19.29 ± 7.71

Table 5.4: Performance Comparison in Experiments with the USB Cable

	Scenario 1: U shape		Scenario 2: S shape	
	Terminal Error(cm)	Settling Time(s)	Terminal Error(cm)	Settling Time(s)
Online Visual Servo	3.41 ± 0.91	113.66 ± 20.93	2.04 ± 0.16	106.12 ± 17.09
Direct Transfer GNN	1.79 ± 0.45	21.16 ± 4.38	1.91 ± 0.67	23.71 ± 5.13
GNN + Domain Randomization	1.39 ± 0.40	$\mathbf{13.65 \pm 4.39}$	1.33 ± 0.43	31.19 ± 6.90
GNN + Fine-tune	0.86 ± 0.38	14.83 ± 3.87	1.08 ± 0.24	$\mathbf{21.44 \pm 4.06}$
GNN + Online Residual	$\mathbf{0.93 \pm 0.23}$	18.81 ± 7.64	$\mathbf{0.82 \pm 0.16}$	34.81 ± 8.61

Data Collection and Learning

Offline Data Collection and Training: The state of the cable is approximated by a series of key points. In simulation, we uniformly select 13 points on the cable. The cable is initialized with a straight line, and we randomly move the robot end-effector to obtain a trajectory $\{X(t), R(t)\}_{t=0,1,\cdots,200}$, where the trajectory contains 200 steps of transitions. With this mechanism, we form a data set with $10K$ trajectories. The network structure and

parameters are summarized in Table 6.2.

Online Data Collection and Learning: In the online phase, the proposed approach does not require pre-collect data before execution. The online linear residual model is obtained in real-time by solving (5.7) with 5 previous cable states and robot movements.

Simulation Results

We first evaluate the performance of the offline model. The cable is randomly initialized, and two robots will randomly move two cable tips for 1 to 20 steps. We tested the GNN model with two baselines: 1) a two-layer MLP with the hidden layer size of 128, 2) a bi-directional LSTM model proposed in [109]. All methods are trained with the same amount of data. The results are provided in Fig. 5.5. The GNN and bi-directional LSTM models outperform the MLP by a large margin (over 10 times). The result confirms our hypothesis that the structural designs of the GNN and bi-directional LSTM encode the prior knowledge of how the interaction will transmit inside the cable. This prior knowledge can make the training result better. While the LSTM model shares many similarities with the GNN on how the interaction is propagated through the vertex, the GNN model performs better in our scenarios. The average prediction error of GNN is less than 2×10^{-2}cm after 20 steps of predictions, which confirms our assumption that an offline GNN model is a good approximation of the ground truth cable model.

For manipulation, we change the stiffness of the cable (0.1×, 0.5×, 1×, 2×, and 3× of the original values) to test the performance of the proposed online residual model. We compare the performance of the proposed method with three sim-to-real baselines: 1) direct transfer, 2) domain randomization, and 3) fine-tuning the network. To be more specific, direct transfer refers to applying the offline learned model directly to the testing scenarios without modification. Domain randomization denotes randomly perturbs some parameters of the cable in the training dataset. We change the spring elastic stiffness, damping stiffness, and bending stiffness in the range of 0.1 to 3 times the original values. For fine-tuning, we collect $2k$ transitions in the testing scenario and re-train the network before executing the robot. In addition, we also compared it with an existing online visual servo control method proposed in [121, 28], where a linear deformation Jacobian is online estimated via least squares. The benchmark scenarios are illustrated in Fig. 5.6. The robots need to collaboratively manipulate the cable from straight lines to three desired shapes. The results are averaged over 5 trails with the cable stiffness changing from 0.1 to 3 times the original value.

The results are summarized in Table 5.2. For settling time, the methods with the offline model outperform the online visual servo method. Since the online visual servo method purely relies on the local information, it is easy to get stuck in a local optimal region and spend lots of time to get out. This phenomenon is also revealed in Fig. 5.6. In all three scenarios, the online visual servo controller is able to reduce the shape error quickly in the beginning but fails to continue this trend in the end. However, since the visual servo

control does not require offline training, the online visual servo method still has the inherent advantage of data efficiency.

For the terminal shape error, we notice that both domain randomization and fine-tuning can improve the test time performance than direct transfer. It is reasonable since the training data for both approaches covers more similar scenarios as in the test time, especially for fine-tuning, which utilizes the data collected in the test scenario to refine the network. However, it is worth noticing that the proposed online residual learning method does not require additional data, and it is robust across all the scenarios. Though fine-tuning slightly outperforms the proposed approach in the first two scenarios, it has a larger error in the third. This may indicate the performance of fine-tuning may greatly depend on the fine-tuned dataset.

We also applied the proposed framework in more challenging scenarios as shown in Fig. 5.7, where the initial state and the target state are randomly generated in simulation. Both tasks require large deformation and accurate local adjustment. Utilizing the proposed method, we are able to complete the tasks robustly.

Experiment Results

We further evaluate the performance of the proposed framework on two different cables, a blue Ethernet cable, and a black USB cable. The offline GNN dynamics are the same as the offline dynamics in the previous experiment. The online models are initialized by randomly moving around the initial configuration, and the fine-tuned dataset is similarly collected with $2k$ transitions. The desired shapes are selected as U shapes and S shapes as the purple curves in Fig. 5.9.

Table 5.3 and Table 5.4 summarize the performance of both cables. The online visual servo control method achieves good performance by gradually learning the local model and moving the cable. However, the online method may converge to a local optimal region and get stuck. As shown in Fig. 5.9(b), the online method fails to achieve the U shape and gets stuck in an S shape that is relatively far away from the desire. The offline GNN method has fair performance on both cables. The global model can efficiently guide the robot to avoid potential local optimal regions for better performance. However, due to the sim-to-real gap, the predictions from the offline model may become inaccurate, and the final shape errors are inevitable. Domain randomization can slightly perform better because the training data already includes the deformation of some higher stiffness cables. However, there is still a gap between simulation and reality. Fine-tuning performs better than domain randomization but is not comparable with the proposed approach. As we analyzed in the previous simulation benchmark, its performance may greatly depend on the fine-tuned dataset. Since it is expensive and time-consuming to obtain real-world data, it is challenging to obtain a dataset covering most of the scenarios. The proposed method achieves the best performance in terms of terminal error. The result corroborates that the proposed framework is effective and robust in manipulating cables.

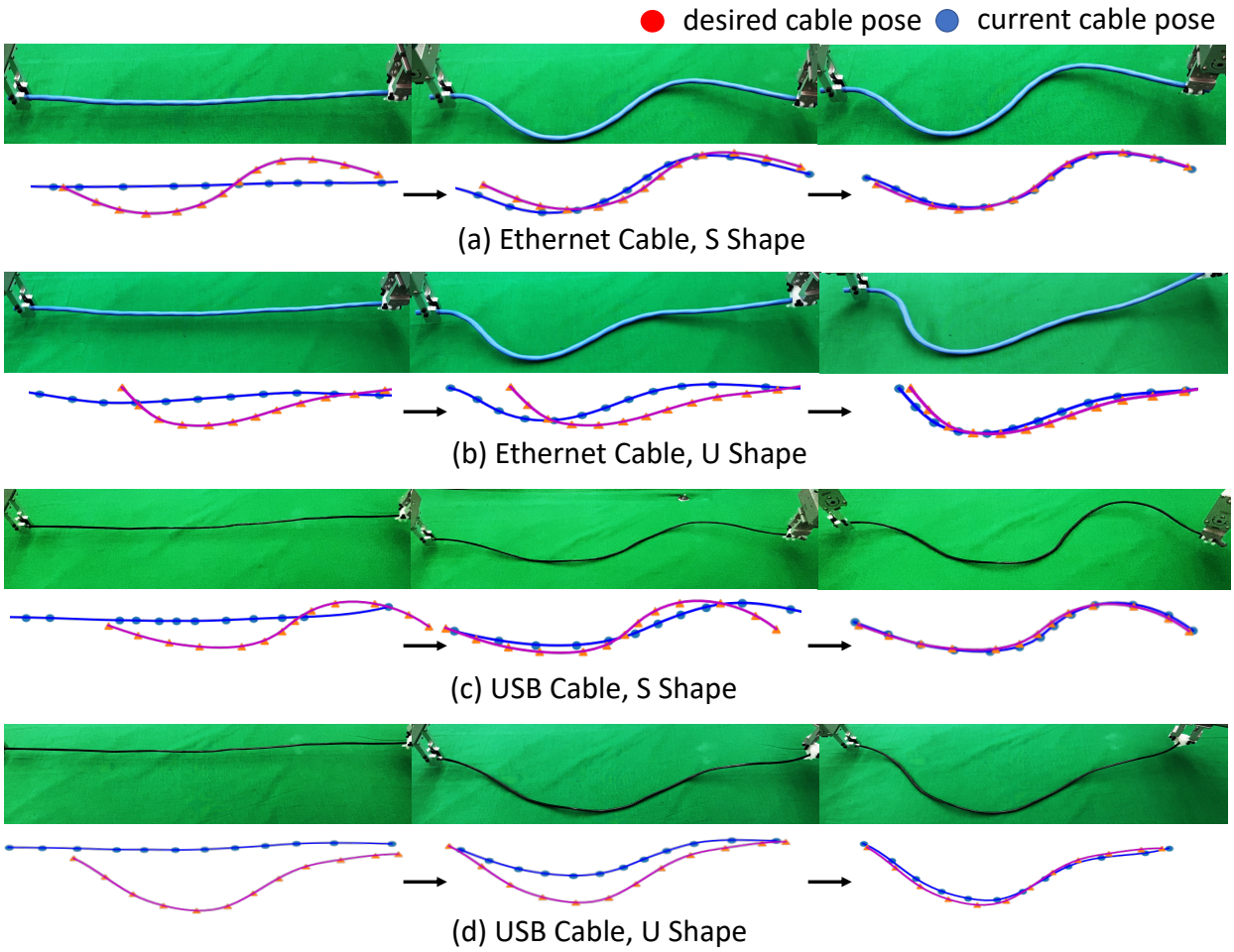


Figure 5.8: Snapshots of the proposed methods. We tested the performance of the proposed method on two cables. For each cable, we set two desired shapes: U shape and S shape. The proposed method is able to achieve high accuracy efficiently.

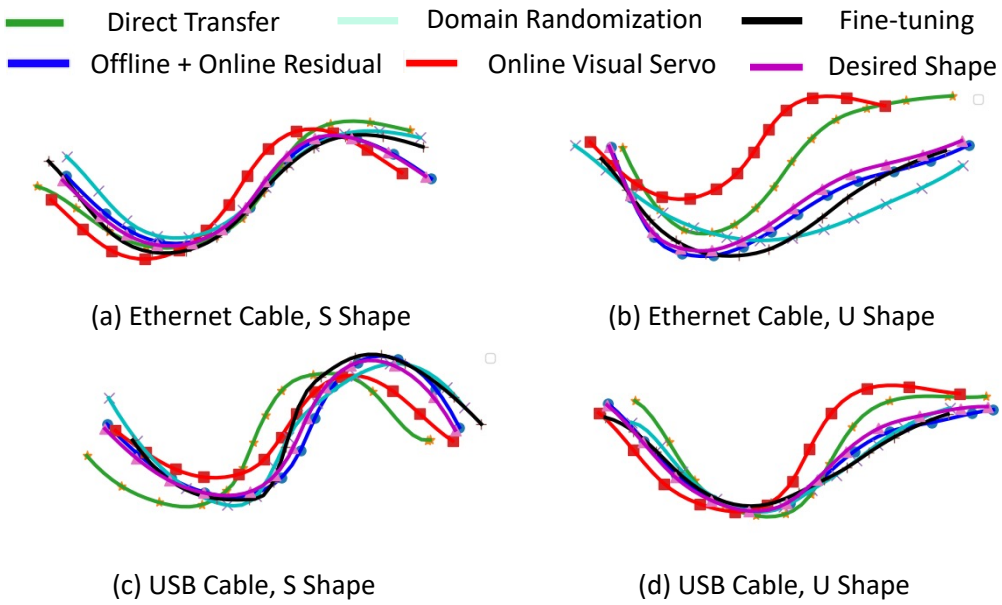


Figure 5.9: We show the final shapes that each method achieved for the benchmark. The benchmark is to manipulate the cable to a U shape and an S shape, as shown with the purple curve. The proposed method can achieve high accuracy across all four tasks.

5.5 Chapter Summary

This chapter has explored the robotic manipulation of deformable linear objects, a task with significant applications in sectors like manufacturing and medical surgery. A key challenge in these tasks is the ability to accurately model and predict the deformation of such objects for robust and precise control. Addressing this need, we introduced a novel hybrid offline-online methodology for learning the dynamics of cables. In the offline phase, we employed a Graph Neural Network (GNN) to understand the deformation dynamics from simulated data, laying a foundational model of the object's behavior. To align this model with real-world scenarios, we further developed a linear residual model in real-time, effectively bridging the gap between simulation and practical application. The integration of the learned model into a trust region-based Model Predictive Controller (MPC) forms a critical component of our approach. This crucial step enables the calculation of optimal movements for the robotic manipulator, which dynamically adapts to the model and is continually refined through real-time updates. We demonstrated the effectiveness of the proposed method through several simulations and real-world experiments.

Chapter 6

Robust Deformation Model Learning under Uncertainties

In Chapter 5, we explored a framework that synergizes offline and online information for the effective manipulation of deformable linear objects. Advancing further in this chapter, we take a step forward by demonstrating how our methods can robustly accomplish these tasks in environments characterized by substantial sensor noise and occlusions. We will showcase how the integration of Structure Preserved Registration (SPR), an enhanced version of the tracking algorithm discussed in Chapter 4, combined with the online learning methods outlined in Chapter 5, and a robust optimization formulation, enables reliable manipulation of a variety of deformable linear objects even under challenging conditions of noise and visual interference. This chapter aims to illustrate the effectiveness of this comprehensive approach, highlighting its capability to maintain high precision and control in manipulating deformable objects amidst significant sensory disruptions.

6.1 Introduction

Robotic cable manipulation has a wide range of applications, such as cable harnessing in factories, thread packing in production lines, and suturing in medical surgeries. However, these tasks are challenging for robots. Compared with rigid objects, models of cables are high dimensional and computationally expensive. Besides, such an object can easily deform to unexpected shapes during manipulation, which may make the manipulation process fail.

There are already some studies on robotic cable manipulation. Many of them are model-based methods. The deformation properties of the cable in terms of stiffness, Young's modules, and/or FEM coefficients are required to build models for trajectory planning. However, such deformation parameters are difficult to estimate accurately and may even change during the manipulation process, especially for objects made of nonlinear elastic or plastic materials.

In this chapter, we introduce a robust online deformation model approximation method for cable manipulation planning. A deformation model is constructed to describe the rela-

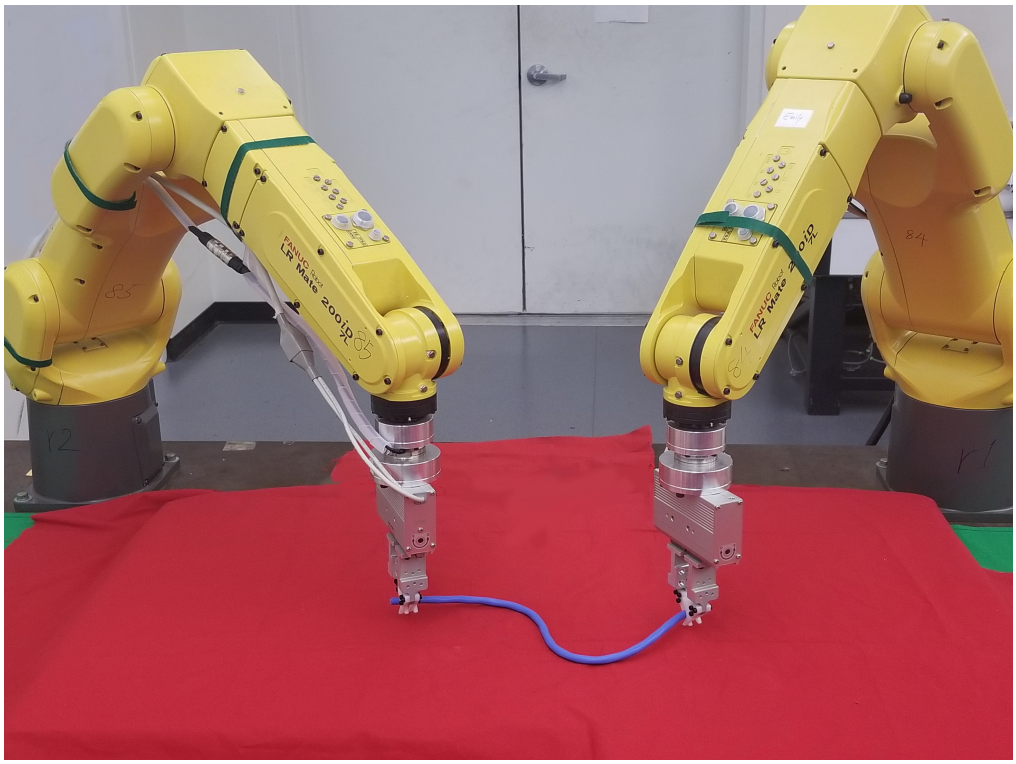


Figure 6.1: Two robots manipulating a cable to a desired shape

tion between the movement of the robot's end-effectors and the displacement of the cable. The idea of the online deformation model approximation was first proposed by Navarro-Alarcon [66]. Cables have infinite degrees of freedom and it is hard to find an explicit model. Instead of finding a global model, real-time data is utilized for local linear model approximation. After the model is obtained, we find the pseudo-inverse of the model, which takes the desired movement of the cable as the input and the velocity of the robot's end-effectors as the output. As the robots manipulate the cable, the deformation model is updated online using real-time data.

In order to approximate the local deformation model of the cable, the motion data of the robot's end-effectors and the cable are required. The motion of the end-effectors can be accurately obtained by solving forward kinematics. However, the motion of the cable is hard to obtain without the help of markers. Sensor noise and occlusions could introduce uncertainties when estimating the motion of the cable. Such uncertainties significantly affect the accuracy of the approximate deformation model.

To handle the above challenges, we propose a framework that is robust in both cable tracking and model approximation. For cable tracking, the core technique we use is called structure preserved registration (SPR) [93], which is a robust non-rigid registration method

for mapping one point set to another. Considering both global and local structures, SPR can robustly estimate the motion of the deformable object in real time even in the presence of sensor noise, outliers, and occlusions.

For model approximation, we take tracking uncertainties into account by solving a robust optimization problem. We formulate the problem as a 'Min-Max' problem, where the maximization takes the worst case of the measurement uncertainty into account, and the minimization penalizes the cost function to find an optimal deformation model. The deformation model is then utilized to plan a trajectory to manipulate the cable.

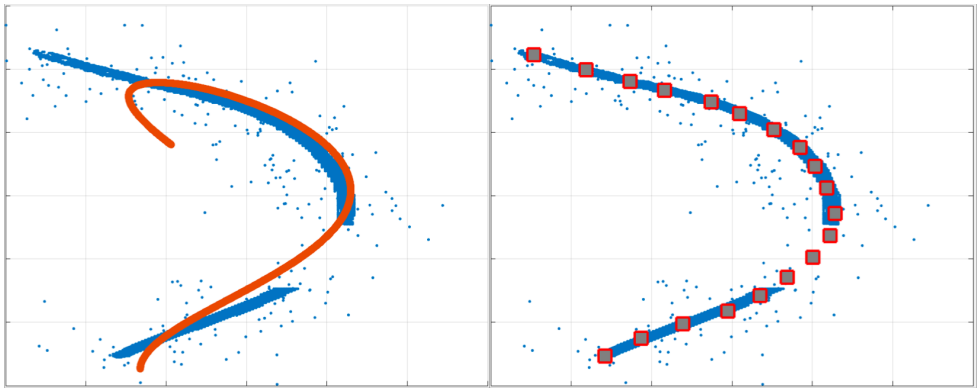
The remainder of this chapter is organized as follows. Section 6.2 introduces related works on cable tracking and manipulation. Section 6.3 describes the SPR method for cable representation and tracking. Section 6.4 explains the local model approximation method using robust optimization. Section 6.5 explains the design of the framework, which includes point tracking, local model approximation, and trajectory planning modules. Section 6.6 tests the performance of the proposed method by a series of experiments. Section 6.7 concludes the chapter.

6.2 Related works

Robotic cable manipulation has been gaining more attention recently for its broad applications. In order to accomplish this challenging task, a robust state estimator to track the configuration of the cable in real time is vital. Metaxas and Terzopoulos [58] constructed a second-order dynamic model for multi-body objects and recursively estimated the body motion from sequences of point clouds by an extended Kalman filter. Schulman et al. [84] proposed a modified expectation maximization (EM) algorithm for deformable object tracking,

Similarly, Petit et al. [73] introduced a finite element method for tracking elastic objects. Navarro-Alarcon et al. proposed a Fourier-based shape servoing method for deformable object representation [65]. Tang and Tomizuka used a non-rigid registration tracking method called structure preserved registration (SPR) [93]. SPR is able to estimate the positions of virtual tracking points on the deformable object in real-time robustly by considering both the local structure and the global topology of the deformable object (Fig. 6.2).

For manipulation planning, Morita et al. [59] developed a 'knot planning from observation' (KPO) system that estimated the states of ropes, especially the overlap orders by knot theory. Kudoh et al. [36] built a multi-finger hand and programmed skill motions by imitating human knotting procedures. They realized three-dimensional air knotting with diverse types of knots. However, many of these methods require empirical laws and are developed for a specific task, which is not easy to generalize for other tasks. To generalize the manipulation skills, Schulman et al. [82] proposed to teach robots to manipulate deformable objects from human demonstrations. They implemented the thin plate spline robust point matching (TPS-RPM) algorithm [15] to warp the original trajectory taught by human demonstration to get a new trajectory that was suitable for the test scene. Tang et al. [95] proposed a



(a) Fourier-Based Estimation (b) SPR Estimation

Figure 6.2: Comparison of Fourier-based method and SPR on cable tracking. Blue dots (or the blue line) are the point clouds with outliers and occlusions, and red circles (or the red line) are the estimated position. The Fourier-based method fails to estimate the state, while SPR still works well and can give the variance of estimation uncertainty.

tangent space mapping method that guaranteed not to overstretch the cable during manipulation. Several follow-up works further improved this demonstration-based method. Tang et al. [94] proposed a uniform framework based on coherent point drift for robustly manipulating deformable linear objects. However, these methods lack the ability to achieve accurate position and can hardly apply to new scenarios that have significant differences with the training scene. Navarro-Alarcon et al. [66] proposed the idea of local deformation model approximation and then utilized the local model to automatically servo-control the soft object to desired shapes. Hu et al. [25] improved the performance by Gaussian process regression. Zhu et al. [121] extended their work by setting up a framework, Fourier-LS, which combines the truncated Fourier series visual servoing method and an efficient continuous local model approximation method. The framework proposed in this chapter has a similar structure to the Fourier-LS.

Compared with other methods, the proposed method SPR-RWLS is the first to take visual tracking uncertainties into consideration for robotic cable manipulation. As shown in Fig. 6.2, for cable tracking, compared with the Fourier-based visual servoing method, SPR works robustly in the presence of outliers and occlusions. For deformation model approximation, a novel algorithm for solving robust weighted least squares is introduced in this chapter. The robust local deformation model for trajectory planning can be obtained efficiently by solving several second-order cone programs (SOCP) in parallel. We show that our method is able to obtain robust deformation models in different scenarios with modest computational costs by several experiments.

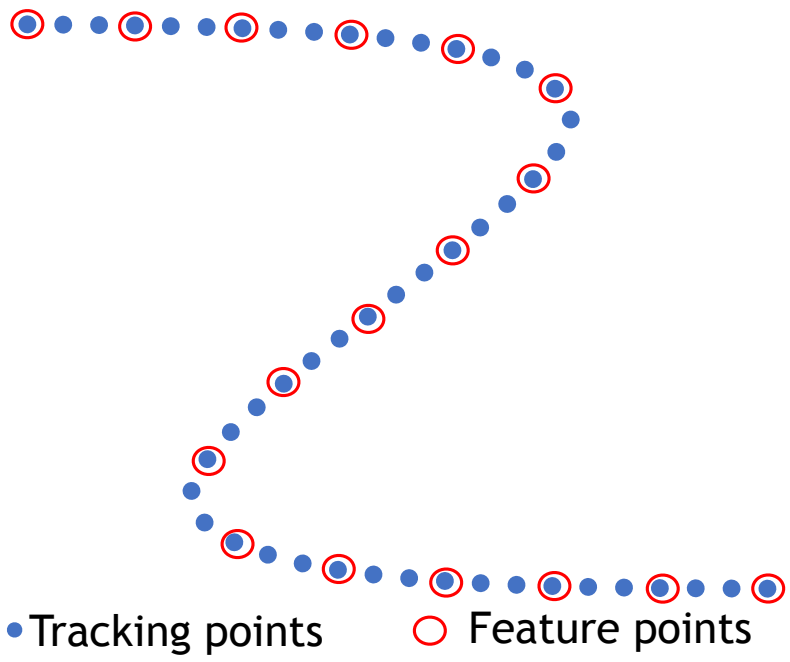


Figure 6.3: Illustration of tracking points and feature points.

6.3 Structure preserved registration

In order to track the shape of the cable, we select a finite number of virtual tracking points along the cable to represent the state of the cable. In Fig. 6.3, the blue dots represent the tracking points, which we select to track the shape of the cable, and the red circles are the feature points, which we use to solve the deformation model in the later section. Usually, feature points are a subset of tracking points. Because of the sparsity of feature points, we can assume that the position of every feature point is uncorrelated with other feature points.

The proposed CPD-based framework in Chapter 4 can reliably track and estimate the state of deformable objects, the estimation performance is poor especially when a part of the point cloud is missing, for example when occlusion happens during perception, which is very common in robot manipulation. The major problem is that there are no constraints on the positions of Gaussian centroids between different time steps. In reality, the cable cannot move arbitrarily and its deformation must follow some topological constraints. Globally those registered Gaussian centroids must form a smooth curvature, and locally those Gaussian centroids should maintain a certain distance from their neighborhood.

To deal with this problem, we introduce both global and local structure regularization to (4.9) in GMM registration.

$$\tilde{Q} = Q(x_n^t, \sigma^2) - \frac{\tau}{2} E_{Local} - \frac{\lambda}{2} E_{Global} \tag{6.1}$$

where $\tau \in \mathbb{R}^+$ and $\lambda \in \mathbb{R}^+$ are trade-off weights that balance the regularization on the local and global structure. E_{Local} and E_{Global} are regularization terms that will be explained in the following.

For local structure, any point at time step $t-1$ can be characterized as a weighted sum of its neighbor points. That is $x_n^{t-1} = \sum_{i \in I_n} S_{ni} \cdot x_i^{t-1}$, where S_{ni} is the weight matrix and I_n is the set for K nearest points to x_n^{t-1}. When the cable deforms to another shape at time step t, the position of tracking points might change, but their relative local structure is expected to be maintained, which means at time step t, $x_n^t \approx \sum_{i \in I_n} S_{ni} \cdot x_i^t$. The local structure weights S_{ni} can be obtained by solving a least squares problem. There could be many sub-optimal weights due to the singularity of the matrix when solving least squares, so we integrate all L sub-optimal weights to characterize the local structure. More details can be found in [93].

$$E_{Local} = \sum_{n=1}^{N} \sum_{l=1}^{L} || \sum_{i=1}^{N} S_{ni}^{(l)} x_i^t ||^2 \tag{6.2}$$

The global structure should also be preserved during registration. Since cable in the real world cannot move arbitrarily, the registered tracking points should also follow a smooth trajectory in neighboring time steps. In order to preserve the global structure, we regularize the coherent movement $x_n^t = v(x_n^{t-1})$. $v : \mathbb{R}^D \to \mathbb{R}^D$ is a transformation function which should be as smooth as possible. The smoothness can be evaluated by $\int_{R^D} \frac{|V(s)|^2}{G(s)}$, where $V(s)$ is the Fourier transform of v and $G(s)$ is a low-pass filter.

$$E_{Global} = \int_{R^D} \frac{|V(s)|^2}{G(s)} ds \tag{6.3}$$

Substituting (6.2) and (6.3) into (6.1) we obtain the modified likelihood function $\tilde{Q}$.

$$\tilde{Q} = Q(x_n^t, \sigma^2) - \frac{\tau}{2} \sum_{n=1}^{N} \sum_{l=1}^{L} || \sum_{i=1}^{N} S_{ni}^{(l)} x_i^t ||^2 - \frac{\lambda}{2} \int_{R^D} \frac{|V(s)|^2}{G(s)} ds \tag{6.4}$$

Though the global and local structure regularization can be formulated in other different ways, the reason for the above regularization is to obtain a closed-form solution for it. This is crucial if we want to solve the problem in real time. More details about SPR and the proof of the existence of closed-form solution can be found in [93].

6.4 Local Linear Deformation Model

Deformation Model

To approximate the deformation model, we uniformly select several feature points along the cable, which are a subset of the tracking points (Fig. 6.3).

By holding two tips of the cable, the end-effectors of the robots are assumed to be fixed with cable tips. We can build a deformation model for describing the interaction between the end-effectors and the cable. Assuming that there are N_f selected feature points on the cable and the degrees of freedom for each point is D. The state of the cable is denoted as $c = [c_1, c_2, ..., c_{N_f}]^T \in \mathbb{R}^{N_f \times D}$, where $c_i = [c_{i1}, c_{i2}, ..., c_{iD}] \in \mathbb{R}^{1 \times D}$, c_{ij} denotes the coordinate of the *i-th* point in the *j-th* direction, for example in 2D space, $c_{5,2}$ denotes the second direction of the 5-*th* selected point. Assume that there are L manipulators and each end-effector has K degrees of freedom. The motion of the robots end-effectors is denoted as $r = [r_{11}, r_{12}, ..., r_{1K}, ..., r_{LK}] \in \mathbb{R}^{L \times K}$. The local linear model we used is expressed in (6.5) [66]. As shown in (6.5) the desired local linear model is $\frac{\delta c}{\delta r}$, and each row of $\frac{\delta c}{\delta r}$ is decoupled with each other, therefore we can make use of parallel computation to find linear model $\frac{\delta c_i}{\delta r}$ for each direction simultaneously, which can greatly improve the efficiency.

$$\begin{aligned} \delta c(t) = \begin{bmatrix} \delta c_1(t) \\ \vdots \\ \delta c_{N_f}(t) \end{bmatrix} = \frac{\delta c}{\delta r}(t)\delta r(t) = \begin{bmatrix} \frac{\delta c_1}{\delta r}(t) \\ \vdots \\ \frac{\delta c_{N_f}}{\delta r}(t) \end{bmatrix} \delta r(t) \\ = A(t)\delta r(t) \end{aligned} \tag{6.5}$$

Local Linear Model

The time-varying deformation model is difficult to obtain due to its high dimensions, nonlinear behaviors, and configuration-dependent properties. Actually, it is unnecessary to find a global deformation model that is suitable for every possible cable configuration. If the displacement of the robot's end-effectors is small enough, the local deformation model can be approximated with linear functions.

The local linear deformation model is expressed in (6.5), where $A(t) \in \mathbb{R}^{N_f \times LK}$ is a time-varying Jacobian Matrix, which represents the relation between the movement of the robots and the movement of the feature points. Remember that our goal is to plan the trajectory for the robot's end-effectors to manipulate the deformable cable to a desired shape. To achieve this, we try to find the motion of the robots $\delta r(t)$ given the desired displacement of the cable $\delta c(t)$. For convenience, instead of calculating the Jacobian matrix $A(t)$, we directly find the pseudo-inverse of the Jacobian matrix $G(t) = A^{\dagger}(t)$. Since $A(t) \in \mathbb{R}^{N_f \times LK}$, in practice, the number of feature points on the cable is always larger than the DOF of robots end-effectors $N_f >> LK$. Therefore, we can guarantee that $G(t)$ exists.

To estimate $G(t)$, we denote the current time as t_m. Using a constant sampling period δt, within the time period $(m-1)\delta t$, we collect m consecutive data of δc_i and δr_i while the robot is moving:

$$\begin{aligned} \delta C(t_m) &= \begin{bmatrix} \delta c(t_1) & \delta c(t_2) & . & . & . & \delta c(t_m) \end{bmatrix} \in \mathbb{R}^{N_f \times m} \\ \delta R(t_m) &= \begin{bmatrix} \delta r(t_1) & \delta r(t_2) & . & . & . & \delta r(t_m) \end{bmatrix} \in \mathbb{R}^{LK \times m} \end{aligned}$$

The local linear model can be found by solving (6.6), which can be decomposed to a sum of several least squares. $G_n^T(t)$ represents the *n-th* column of the matrix $G^T(t)$, and similarly $\delta R_n^T(t)$ represents the *n-th* column of $\delta R_n^T(t)$.

$$\begin{aligned} G(t)^* &= \underset{G(t)}{arg\min} \quad \|\delta C^T(t) G^T(t) - \delta R^T(t)\|_F^2 \\ &= \sum_{n=1}^{LK} \underset{G_n(t)}{arg\min} \quad \|\delta C^T(t) G_n^T(t) - \delta R_n^T(t)\|_2^2 \end{aligned} \tag{6.6}$$

In the next subsection, we will introduce how to improve the robustness of the local model approximation by using robust optimization.

Robust weighted least squares

The displacement of the robot's end-effectors $\delta R(t)$ can be calculated accurately using the robotic forward kinematics. However, $\delta C(t)$ is an estimation with lots of uncertainties from visual tracking. If uncertainties are not considered, we may fail to recover a suitable local deformation model. Uncertainty in $\delta C(t)$ can be approximated by Gaussian distribution, so we are able to bound it inside a certain area given a confidence probability.

In SPR, we regard the tracking points in the last time step as Gaussian centroids and each point in the new point cloud as a sample from the Gaussian mixture model. The objective is to maximize the log-likelihood of the point cloud sampled from the GMM. Thus it is reasonable to regard the variance of each Gaussian as the uncertainty of this movement. Taking the *i-th* point as an example, $\mu(c_i(t))$ is the mean of the *i-th* point at time step t, and σ_t is the uncertainty from time step $t-1$ to t. Besides, we assume that each Gaussian has an equal membership probability $1/N$ and a consistent isotropic covariance $\sigma^2 I$ in SPR registration. So all the selected tracking points on the cable have the same variance σ_t at time step t, and all the feature points are uncorrelated.

$$\delta C^T(t) = \mu(\delta C^T(t)) + \Delta \tag{6.7}$$

(6.7) shows the uncertainty in the matrix $\delta C^T(t)$ for robust optimization, where Δ describes the uncertainty. From the above analysis, the $j-th$ column of the matrix Δ can be regarded as a sample from a Gaussian Distribution $N(0, \Sigma)$, where $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_m^2)$.

We rewrite (6.6) in the form of robust optimization in (6.8),

$$\sum_{n=1}^{LK} \min_{G_n(t)} \max_{\|\Delta\|_2 \le s} \quad \|W[(\mu(\delta C^T(t)) + \Delta) G_n^T(t) - \delta R_n^T(t)]\|_2^2 \tag{6.8}$$

where s is the upper bound or equivalently the largest singular value of Δ, and $W = diag(w_1, w_2, ..., w_m)$ is a weight matrix for the data from different time steps.

When solving the robust optimization problem (6.8), a tight bound s is preferred. Each column of Δ can be regarded as a random sample from the Gaussian distribution, and

there are some existing works in statistics to bound the largest singular value with a desired probability.

Theorem 2. *Let $\Delta \in \mathbb{R}^{m\times n}$ be drawn according to the Σ-Gaussian ensemble. Then for all $\delta > 0$, the maximum singular value $\sigma_{max}(\Delta)$ satisfies the upper deviation inequality,*

$$\mathbb{P}[\frac{\sigma_{max}(\Delta)}{\sqrt{n}} \leq \gamma_{max}(\sqrt{\Sigma})(1+\delta) + \sqrt{\frac{trace(\Sigma)}{n}}] \geq 1 - e^{-n\delta^2/2}$$

where $\gamma_{max}(\sqrt{\Sigma})$ denotes the largest eigenvalue of $\sqrt{\Sigma}$.

Theorem 2 provides a theoretically tight bound of the random matrix Δ which is proved in Chapter 6 of [101]. Using this theorem, we can find an upper bound of the largest singular value of uncertainty matrix Δ given a desired probability.

Theorem 3. *Any robust least squares in the form:*

$$\min_{x\in\mathbb{R}^n} \max_{\|\Delta\|_2 \leq s} \quad \|(A+\Delta)x - b\|_2$$

is equivalent to a SOCP problem:

$$\min_{x\in\mathbb{R}^n} \quad \|Ax - b\|_2 + s\|x\|_2$$

Proof. For fixed x, and using the fact that the Euclidean norm is convex, we have

$$\|(A+\Delta)x - b\|_2 \leq \|Ax - b\|_2 + \|\Delta x\|_2$$

By the definition of the largest singular value norm, and given our bound on the size of the uncertainty, we have

$$\|\Delta x\|_2 \leq \|\Delta\|_2\|x\|_2 \leq s\|x\|_2$$

Thus, we have a bound on the objective value of the robust least squares problem:

$$\max_{\|\Delta\|_2\leq s} \quad \|(A+\Delta)x - b\|_2 \leq \|Ax - b\|_2 + s\|x\|_2$$

The upper bound is actually attained by

$$\Delta = \frac{s}{\|Ax-b\|_2\|x\|_2}(Ax-b)x^T$$

□

Therefore, the robust weighted least squares (RWLS) in (6.8) can be written in the form of a summation over several SOCPs as shown in (6.9). For each SOCP, we find one column of the model matrix $G^T(t)$. The columns of $G^T(t)$ do not depend on each other, which means

that we can make use of parallel computation to solve each SOCP and greatly increase the efficiency of the solution process.

$$\sum_{n=1}^{LK} \min_{G_n(t)} \quad \|W\mu(\delta C^T(t))G_n^T(t) - W\delta R_n^T(t)\|_2 + s\|WG_n^T(t)\|_2 \tag{6.9}$$

If the matrix $G(t)$ is obtained, it means that the local model at time step t is approximated. Given the desired movement of the cable, we are able to obtain the trajectory of the robot's end-effectors.

6.5 Framework Details

Algorithm Overview

Combining the above SPR estimation with RWLS for solving the local deformation model, we propose our method 'SPR-RWLS' to manipulate soft cables to desired shapes. SPR is utilized to estimate the cable state in real time. Robust weighted least squares (RWLS) are used to obtain a robust local deformation model considering tracking uncertainties. The proposed method is summarized in Algorithm 3.

Algorithm 3: SPR-RWLS

```
Initialize cable, and record desired cable shape;
Using SPR to get initial and desired tracking points along the cable;
Downsample tracking points with a fixed index to get feature points;
Initialize data set D(δR, δC) by randomly executing robot δR and collecting
  corresponding movement of cable δC for m_0 times;
while diff(c_current, c_desired) > ϵ do
    Compute weight matrix W;
    Solve Robust Weighted Optimization for local deformation Jacobian Matrix G(t);
    Compute δr = λG(t)δC_desired;
    Execute δr, collect new δc by SPR;
    Append δr and δc to dataset D;
end
```

Cable Tracking

We select $N_{tracking}$ tracking points uniformly distributed along the initial point cloud. At time step t, the tracking points are denoted as $X^t = \{x_1^t, x_2^t, ..., x_{N_{tracking}}^t\}$, where $x_n^t \in \mathbb{R}^2$. At the next time step $t+1$, the cable deforms to a new shape, and its point cloud $Y^{t+1} = \{y_1^{t+1}, y_2^{t+1}, ..., y_M^{t+1}\} \in \mathbb{R}^2$ is captured by the camera. By applying SPR registration

as described in Section III, the node positions X^k can be smoothly registered towards the point cloud Y^{t+1}, and we can get the new estimation of tracking point positions X^{t+1}, and the variance σ_{t+1} of this step.

As shown in Fig. 4.4, running the above procedure iteratively, we can obtain the estimated position of each tracking point in real-time.

Deformation Approximation

We select $N_{feature}$ feature points from the tracking points just as Fig. 6.3 shows. Also, at time step t, the movement of each feature point $\delta c(t) = c(t) - c(t-1)$, and the tracking uncertainties of this step σ_k can be obtained from SPR registration. The movement of the robot's end-effectors δr can also be calculated by forward kinematics. Then we append these new data $\delta c(t)$, $\delta r(t)$, and σ_t to data set $\delta C(t)$, $\delta R(t)$, and $\Sigma(t)$ respectively. Before we calculate the deformation model, we assign weights to different $(\delta r, \delta c)$ pairs in the data set. We rank the data pairs in the data set based on their mean squared errors to the current cable shape. A discount factor γ is used to assign weights to different data pairs. The discount factor is a tuning parameter and we use 0.95 in our experiments. Besides, the bound of the uncertainties can be efficiently calculated by a given confidence probability. In practice, we set the confidence probability larger than 99%. Finally, the local deformation model $G(t)$ can be solved by the robust optimization problem (6.9).

By running this algorithm iteratively, we are able to get an approximation of the deformation model in real time.

Trajectory Planning

After the deformation model $G(t)$ is obtained at time step t, we first need to compute the desired movement of the cable $\delta c_{desired}(t)$ in order to get the motion of the robots. For the scenarios in which the desired shape is far away from the initial shape, several intermediate desired shapes $c_{intermediate}$ are preferred to be generated and reached in sequence. In order to achieve such δC, the desired movement of end-effectors is computed using $\delta r(t) = \lambda G(t)\delta c_{desired}(t)$, where λ is a gain we need to tune. In order to make the cable move at a low speed without vibration and make sure the deformation model is locally accurate, the gain λ is chosen to be small. In our experiment, λ is set to 0.1.

6.6 Experiments and Results

We conducted several experiments on two FANUC LR-Mate 200iD robots to show that SPR-RWLS is efficient and robust in the presence of outliers and occlusions for robotic cable manipulation.

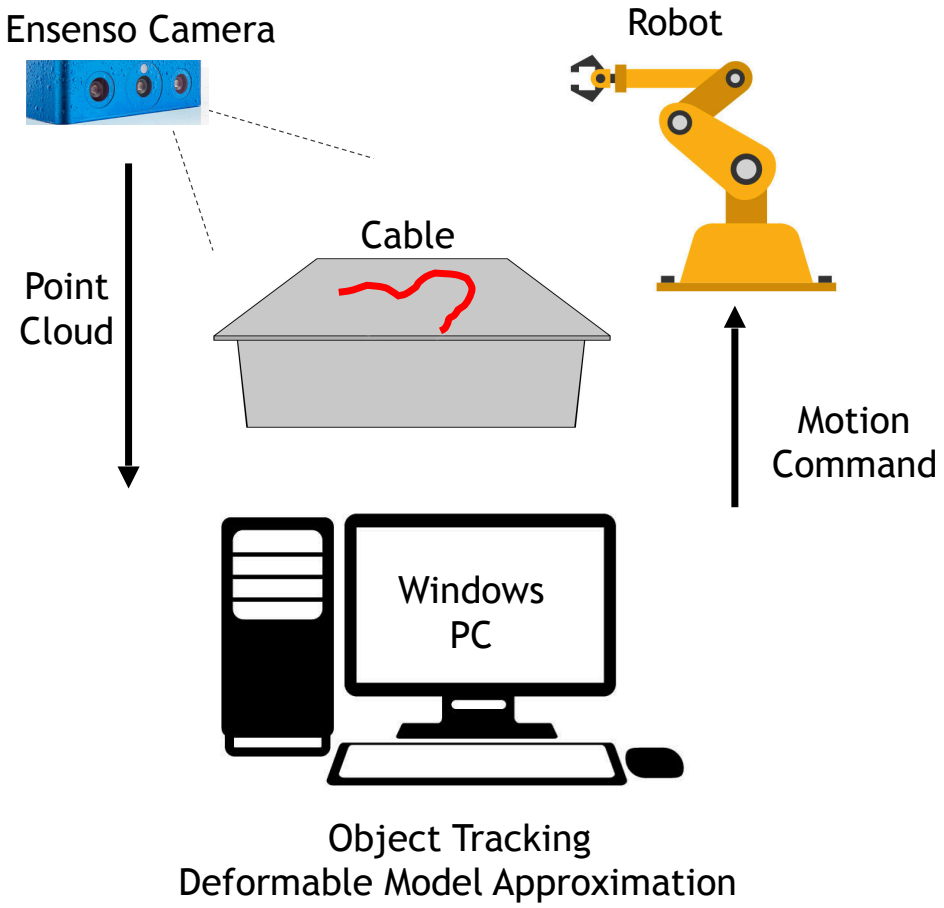
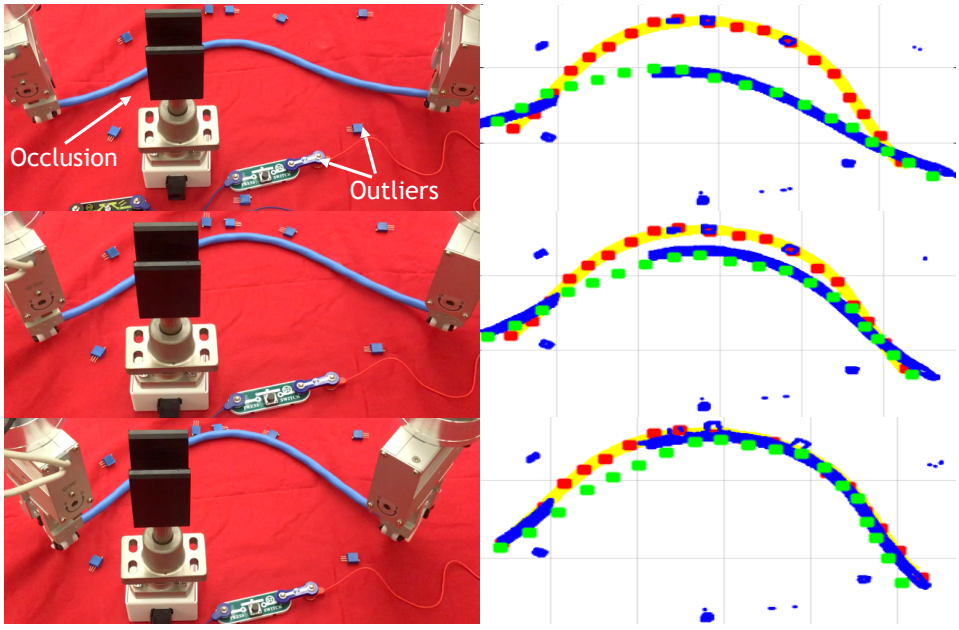


Figure 6.4: The testbed setup

Cable Tracking

As shown in Fig.6.4, an IDS Ensenso N35 stereo camera was utilized to monitor the environment. The captured point cloud was sent to a Windows 10 desktop (Intel i7@3.60 GHz + RAM 8GB), which ran SPR registration algorithms in real-time in MATLAB. Using a color filter, the point cloud of the cable was extracted from the red background. Since the cable was manipulated in a two-dimensional plane, the $3D$ point cloud was projected to the $2D$ plane. Given the initial point cloud of a straight cable, we manually selected 55 tracking points uniformly distributed along the cable. When the cable deformed to a different shape in the next step, we registered the newest point cloud to the point cloud from the last time step using SPR. Then the corresponding 55 tracking points which represent the current cable state were obtained.

Tracking results show that SPR cable tracking module is able to robustly track the



(a) RGB Image (b) Object Tracking

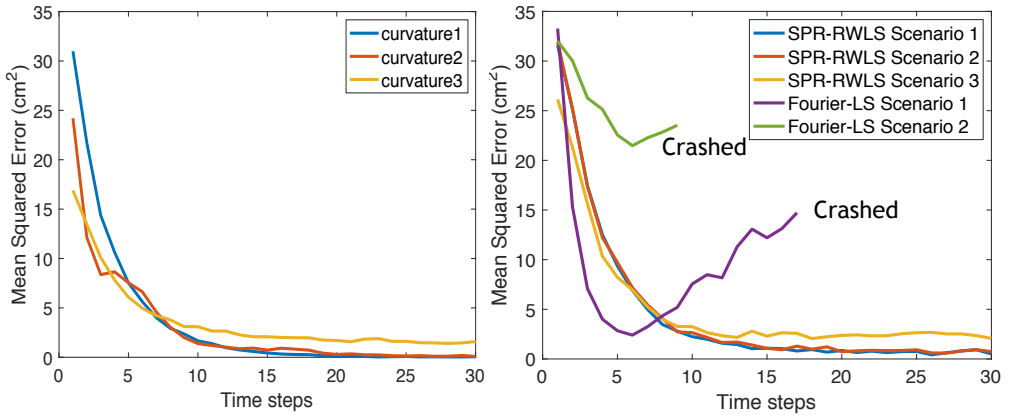
Figure 6.5: SPR cable tracking in the presence of outliers and occlusions. In (b), blue dots represent the perceived point cloud; yellow dots represent the target shape; red squares represent the desired feature points; and green squares represent current feature points.

movement of the cable in real-time. SPR outperforms Fourier-Based estimation when the point cloud is contaminated by outliers, noise, and occlusions. When manually adding white noise to the point cloud and removing 20% of the point cloud from the middle part to simulate occlusions (Fig. 6.2), Fourier-Based estimation fails tracking the cable, while SPR is still able to provide a robust estimation of the cable position as well as estimation of the uncertainties. Fig.6.5 shows SPR tracking performance in the presence of occlusions and outliers due to objects of similar colors to the cable appearing in the workspace.

Cable Manipulation

To evaluate the performance of the proposed framework, we conducted experiments to manipulate cables of different thicknesses under different scenarios. The goal is to manipulate the cable from straight lines to given desired shapes.

The robot's end-effectors are parallel grippers that can open and close. When closing, the gripper can clamp the cable firmly without any slipping. Since the experiment is conducted on a $2D$ plane, each end-effector has 3 degrees of freedom including two orthogonal displacements on the horizontal plane and one rotation along the axis that is perpendicular



(a) SPR-RWLS performance

(b) Comparison of Fourier-LS and SPR-RWLS under outliers and occlusions

Figure 6.6: Mean Squared Error vs Timesteps. (a) shows the results of manipulation of a cable to different curvatures, which are shown in Fig. 6.7, and (b) compares SPR-RWLS with Fourier-LS in different scenarios, which are introduced in Section VI.B

to the horizontal plane. For the model approximation, ECOS [18], an efficient SOCP solver, was utilized to solve the problem (6.9).

Given the positions and variances of 55 estimated tracking points from SPR, we select 19 feature points from the tracking points to estimate the deformation model. Each feature point has two degrees of freedom in the $2D$ plane. Using Algorithm 3 introduced in section V, several experiments are conducted to test the proposed framework. We use the mean squared distance errors of 55 tracking points between the cable and the desired shape to evaluate the manipulation performance.

A sequence of snapshots is shown in Fig.6.7. The cable was successfully manipulated from a straight line to given desired curvatures. For a simple desired shapes (Fig.6.7 (a), (b)), the manipulated cable overlaps with the desired shape perfectly with the mean squared error (MSE) smaller than 0.08 cm^2. For complicated desired shapes with more curvatures (Fig.6.7 (c)), the MSE is about 1.5 cm^2. Fig. 6.6 (a) shows that the MSE converges over time steps.

The performance of the algorithm with different cables is analyzed in Table 6.1. We conducted experiments with two cables of different diameters. One Ethernet cable has a diameter of $4.04mm$, and the other cable has a diameter of $8.10mm$. We manipulated both cables to the same simple desired shape (curvature 1 in Fig. 6.7) 10 times. Both experiments have very high success rates. The thinner cable has a slightly higher MSE. It is reasonable because the thinner cable is more likely to deform, which makes the deformation model not accurate.

We also performed several experiments to show that SPR-RWLS is able to perform robustly in the presence of outliers and occlusions. As shown in Table 6.2, we conducted

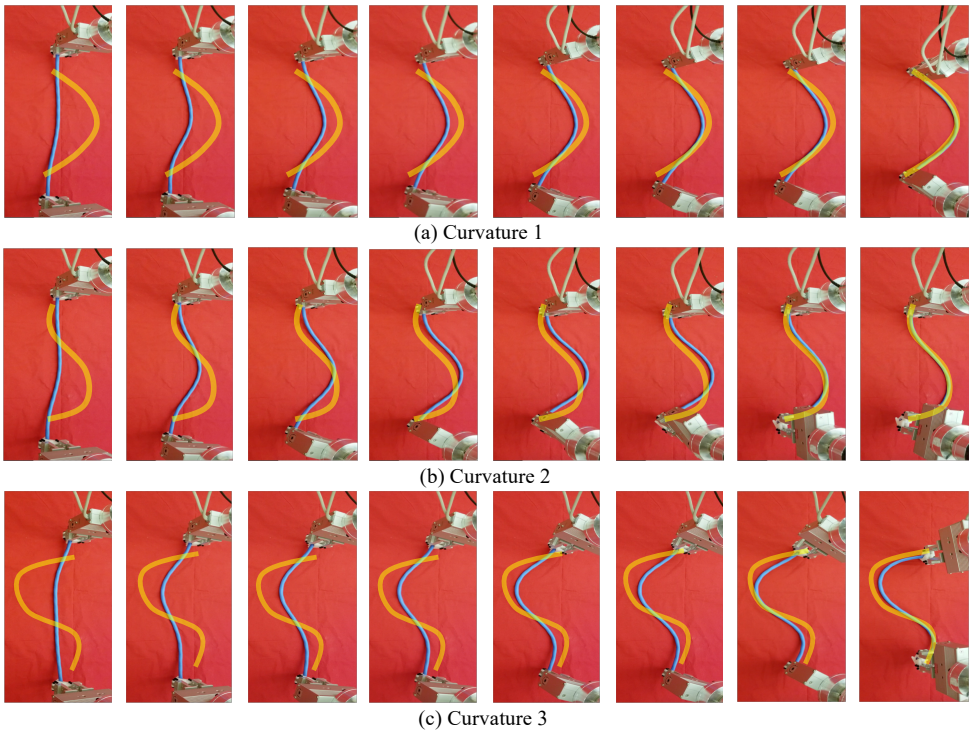


Figure 6.7: Snapshots of the cable manipulation experiments. Two robot arms were collaborating to manipulate a cable to desired shapes, which are shown by the yellow lines. (a), (b), and (c) show three different curvatures.

Table 6.1: Experiments with different cables

Cable diameters	Success rate	Mean squared error (cm^2)
$4.04mm$	9/10	0.242 ± 0.079
$8.10mm$	10/10	0.051 ± 0.014

experiments to manipulate the blue Ethernet cable to curvature 1 in 4 different scenarios. In the first scenario where there are no outliers and occlusions, both Fourier-LS and SPR-RWLS performed very well. Uncertainty scenarios 1,2 and 3 are the scenarios in the presence of noise, outliers, and occlusions. In uncertainty scenario 1, we manually occluded 20% of the point cloud and added 5% white noise with $\sigma = 10\%\delta c$. In uncertainty scenario 2, we occluded 25% point cloud and added 10% white noise with $\sigma = 15\%\delta c$. Uncertainty scenario 3 is shown in Fig.6.5, where objects of color similar to the Ethernet cable are placed on the table and part of the Ethernet is occluded by other objects. In Table 6.2, we can clearly see that SPR-RWLS outperforms Fourier-LS [121] in the presence of noise, outliers, and occlusions.

Table 6.2: Comparison of mean squared error of our robust method and Fourier-LS Method

Feature	Fouriers-LS	SPR-RWLS
No outliers and occlusions	$0.045 \pm 0.026 cm^2$	$0.051 \pm 0.014 cm^2$
Uncertainty scenario 1	$14.712 \pm 1.5832 cm^2$	$0.411 \pm 0.093 cm^2$
Uncertainty scenario 2	$23.45 \pm 1.259 cm^2$	$0.629 \pm 0.1623 cm^2$
Uncertainty scenario 3	fail	$2.503 \pm 0.438 cm^2$

6.7 Chapter Summary

This chapter introduces a novel framework, SPR-RWLS, specifically designed for cable manipulation under massive sensor noises and occlusions. For the real-time tracking aspect, we employ the Structure Preserved Registration (SPR) method that demonstrates robust performance in tracking deformable cables, even under challenges such as sensor noise, outliers, and occlusions. On the other hand, for the deformation model approximation, we utilize an approach where the local deformation model of the cable is dynamically approximated online. This is achieved through solving a robust optimization problem in parallel, effectively handling uncertainties that may arise during manipulation.

The experiments conducted under this framework have proven the efficacy of the SPR-RWLS method. The results highlight the framework's capability to accurately manipulate deformable cables into desired shapes and configurations. This successful manipulation is achieved consistently despite the presence of environmental challenges and uncertainties, showcasing the robustness and reliability of the proposed method in practical applications.

Part III

Real-time Control Policy Adaptation for Contact-Rich Manipulation Tasks

Chapter 7

Efficient Control Policy Adaptation with Online Admittance Residual Learning

The previous two parts of the dissertation focus on two important aspects of online manipulation: planning and model learning. This chapter turns its focus to another crucial aspect of reliable online manipulation: robotic control. The ability of robots to interact with a range of environments using appropriate force is important for successful task execution. This chapter is dedicated to presenting algorithms designed to enhance real-time control adaptability in robotic systems through online optimization techniques. In this chapter, we will introduce a hybrid offline-online framework to learn robust manipulation skills. We employ model-free reinforcement learning for the offline phase to obtain the robot motion and compliance control parameters in simulation. Subsequently, in the online phase, we learn the residual of the compliance control parameters to maximize robot performance-related criteria with force sensor measurements in real time. The effectiveness and robustness of our approach will be demonstrated through a series of real-world experiments. Videos are available at https://sites.google.com/view/admitlearn.

7.1 Introduction

Contact-rich manipulation is common in a wide range of robotic applications, including assembly [27, 116, 98, 52, 29, 108, 107, 85], object pivoting [119, 115, 86], grasping [53], and pushing [89, 24]. To accomplish these tasks, robots need to learn both the manipulation trajectory and the force control parameters. The manipulation trajectory guides the robot toward completing the task while physically engaging with the environment, whereas the force control parameters regulate the contact force. Incorrect control parameters can lead to oscillations and excessive contact forces that may damage the robot or the environment.

Past works have tackled the contact-rich skill-learning problem in different ways. First,

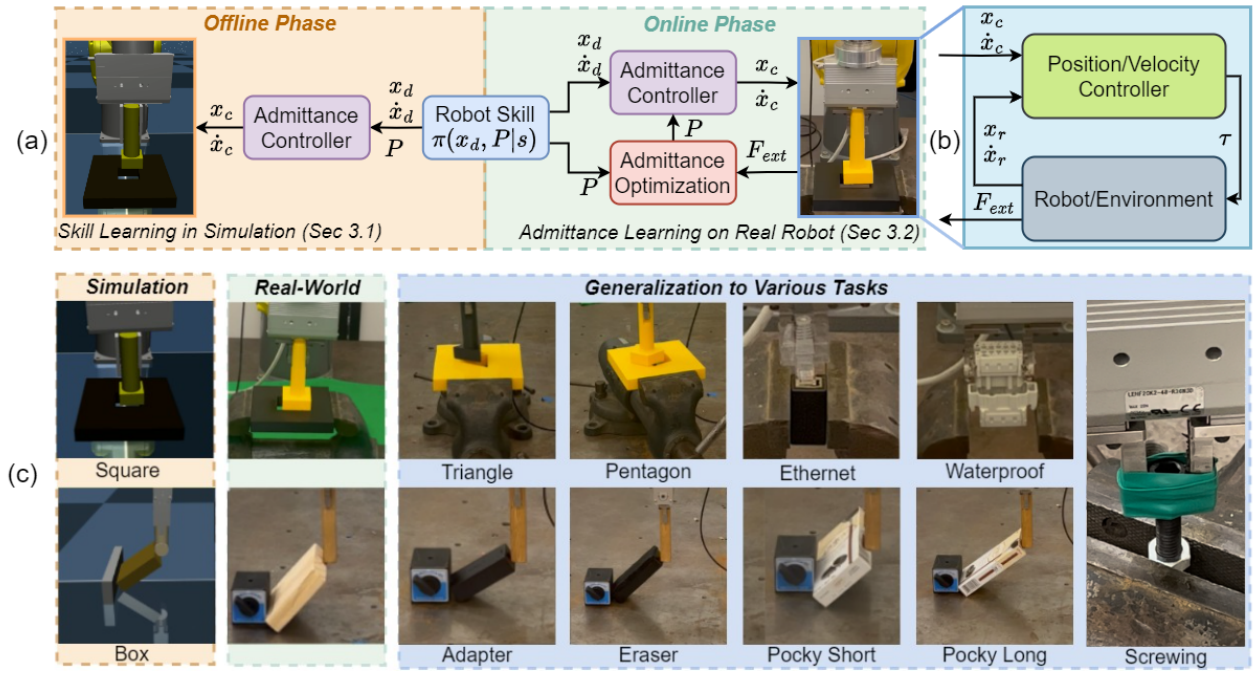


Figure 7.1: As shown in (a), we propose a robust contact-rich manipulation skill learning framework that offline learns the robot motion and compliance control parameters in the simulation and online adapts to the real world. The structure of the admittance controller is depicted in (b). Our framework demonstrates robustness in sim-to-real transfer and generalizability to diverse real-world tasks in (c)

the majority of previous works [116, 115, 108, 107, 98, 64, 119] focus on learning the manipulation trajectories and rely on human experts to manually tune force control parameters. While this simplification has demonstrated remarkable performance in many applications, letting human labor tune control parameters is still inconvenient. Furthermore, the tuned parameter for one task may not generalize well to other task settings with different kinematic or dynamic properties. For example, assembly tasks with different clearances will require different control parameters. Another line of work deals with this problem by jointly learning the robot's motion and force control parameters [72, 1, 117, 10, 78, 4, 52, 54, 85]. Such learning processes can be conducted in both real-world and simulation. However, learning such skills on real robots is time-consuming and may damage the robot or environment. Learning in simulation is efficient and safe, however, the learned control parameters may be difficult to transfer to real robots due to the sim-to-real gap, and directly deploying the learned control parameters may cause damage to the robot.

In this chapter, we focus on transferring robotic manipulation skills. We notice that the manipulation trajectory is more related to the kinematic properties, such as size and shape,

which have a smaller sim-to-real gap and can be transferred directly, as demonstrated by previous works [119, 116, 115, 98]. However, simulating the contact dynamics proves to be challenging, primarily due to its sensitivity to various parameters, including surface stiffness and friction [71]. This sensitivity will result in a large sim-to-real gap and affects the learned compliance control parameters.

Inspired by the above analysis, we propose a framework to learn robot manipulation skills that can transfer to the real world. As depicted in Fig. 7.1(a), the framework contains two phases: skill learning in simulation and admittance adaptation on the real robot. We use model-free RL [23, 83] to learn the robot's motion with domain randomization to enhance the robustness for direct transfer. The compliance control parameters are learned at the same time and serve as an initialization to online admittance learning. During online execution, we iteratively learn the residual of the admittance control parameters by optimizing the future robot trajectory smoothness and task completion criterion. We conduct real-world experiments on three typical contact-rich manipulation tasks: assembly, pivoting, and screwing. Our proposed framework achieves efficient transfer from simulation to the real world. Furthermore, it shows excellent generalization ability in tasks with different kinematic or dynamic properties, as shown in Fig. 7.1(c). Comparison and ablation studies are provided to demonstrate the effectiveness of the framework.

7.2 Related Works

Sim-to-real Transfer in Robot Contact-Rich Manipulation

Contact-rich manipulation tasks involve the interaction between robots and the environment through physical contact. In recent years, there has been a growing trend in utilizing simulation environments such as MuJoCo [97], Bullet [17], and IsaacGym [64] to learn and train robots for these tasks. These simulation environments offer advantages in terms of safety, scalability, and cost-effectiveness. Nevertheless, the sim-to-real gap remains a significant challenge.To address the gap, various approaches have been explored, including system identification, transfer learning, domain randomization, and online adaptation. System identification approaches [51, 47] involve the calibration of simulation parameters to improve accuracy and align the simulation with real-world dynamics. Transfer learning methods [14, 4] aim to fine-tune skills learned in simulation for application in real-world scenarios. Domain randomization techniques [115, 119, 4, 11] are employed to create diverse environments with varying properties, enabling the learning of robust skills for better generalization. Instead of collecting large datasets in the real world, online adaptation methods [104, 111, 28, 13, 91, 38] utilize real sensor measurements to optimize a residual policy/model or directly update the policy network in real-time. Tang et al. [92] further improve the sim-to-real transfer performance by combining the above techniques with a modified objective function design for insertion tasks.

Learning Variable Impedance/Admittance Control

Compliance control [69], such as impedance and admittance control, enables robots to behave as a mass-spring-damping system. Tuning the compliance control parameters is crucial for stabilizing the robot and accomplishing manipulation tasks. However, manual tuning can be time-consuming. To address this issue, learning-based approaches have been applied to automatically learn the control parameters. Previous methods have focused on learning compliance control parameters either from expert demonstrations [72, 1, 117] or through reinforcement learning (RL) [10, 78, 4, 52, 54, 35] to acquire gain-changing policies. [1, 52, 72, 10, 4] propose to directly collect data in the real world. However, it is time-consuming to collect the data. Authors in [117, 54] have demonstrated success in directly transferring the learned control parameters from the simulation to the real world. Nevertheless, their applications are limited to simple tasks, such as waypoint tracking and whiteboard wiping.

7.3 Proposed Approach

We focus on learning robust contact-rich manipulation skills that can achieve efficient sim-to-real transfer. We define the skill as $\pi(x_d, P|s)$, which generates both the robot's desired trajectory x_d and the compliance control parameters P given the current state s.

We use *Cartesian space admittance control* as the compliance controller. As shown in Fig. 7.1(b), the admittance control takes in the desired trajectory $[x_d, \dot{x}_d, \ddot{x}_d] \in \mathbb{R}^{18}$, and the external force/torque $F_{ext} \in \mathbb{R}^6$ measured on the robot end-effector and outputs the compliance trajectory $[x_c, \dot{x}_c, \ddot{x}_c] \in \mathbb{R}^{18}$ to the position/velocity controller according to the mass-spring-damping dynamics [69]:

$$M(\ddot{x}_c - \ddot{x}_d) + D(\dot{x}_c - \dot{x}_d) + K(x_c - x_d) = F_{ext} \tag{7.1}$$

where M, K, D are the robot inertia, stiffness, and damping matrices, respectively. We assume M, K, D is diagonal for simplicity and $P = \{M, K, D\}$ as the collection of all control parameters.

To achieve this goal, We propose an offline-online framework for learning contact-rich manipulation skills as depicted in Fig. 7.1(a). In the offline phase, we employ the model-free RL with domain randomization to learn the robot motion and the initial guess of compliance control parameters from the simulation (Section 7.3). In the online phase, we execute the offline-learned motions on the real robot and learn the residual compliance control parameters by optimizing the future robot trajectory smoothness and task completion criteria(Section 7.3).

Learning offline contact-rich manipulation skills

We utilize model-free RL to learn contact-rich manipulation skills in MuJoCo simulation [97]. The problem is modeled as a Markov decision process $\{S, A, R, P, \gamma\}$ where S is the state

space, A is the action space, R is a reward function, P denotes the state-transition probability, and γ is the discount factor. For each timestep t, the agent is at the state $s_t \in S$, executes an action $a_t \in A$, and receives a scalar reward r_t. The next state is computed by the transition probability $p(s_{t+1}|s_t, a_t)$. Our goal is to learn a policy $\pi(a|s)$ that maximizes the expected future return $\mathbb{E}\left[\sum_t \gamma^t r_t\right]$.

Specifically, we focus on learning robot skills for three contact-rich tasks: assembly, pivoting, and screwing. In these tasks, the robot needs to utilize the contact to either align the peg and hole or continuously push and pivot the object, which makes them suitable testbeds for our proposed framework. The detailed task setups can be found below:

Assembly Task: The goal is to align the peg with the hole and then insert it.

State space: The state space $s \in \mathbb{R}^{18}$ contains peg pose $s_p \in \mathbb{R}^6$ (position and Euler angles) relative to the hole, peg velocity $v_p \in \mathbb{R}^6$, and the external force measured on the robot wrist $F_{ext} \in \mathbb{R}^6$.

Action space: The action $a \in \mathbb{R}^{12}$ consists of the end-effector velocity command $v_d \in \mathbb{R}^6$ and the diagonal elements of the stiffness matrix $k \in \mathbb{R}^6$. To simplify the training, the robot inertia M is fixed to $diag(1, 1, 1, 0.1, 0.1, 0.1)$, and the damping matrix $D = diag(d_1, \cdots, d_6)$ is computed according to the critical damping condition $d_i = 2\sqrt{m_i k_i}, i = \{1, 2, 3, 4, 5, 6\}$.

Reward function: The reward function is defined as $r(s) = 10^{(1-\|s_{pos}-s^d_{pos}\|_2)}$, where $s_{pos} \in \mathbb{R}^3$ is the peg position and $s^d_{pos} \in \mathbb{R}^3$ is the nominal hole location. The exponential function encourages successful insertion by providing a high reward.

Pivoting Task: The goal is to gradually push the object to a stand-up pose against the wall.

State space: The state $s \in \mathbb{R}^{12}$ consists of the robot pose $s_p \in \mathbb{R}^6$ and the external force $F_{ext} \in \mathbb{R}^6$.

Action space: For simplicity, we consider a 2d pivoting problem: the robot can only move in the X, Z direction. The robot action $a \in \mathbb{R}^4$ contains the velocity command in X, Z direction and the corresponding stiffness parameter.

Reward function: We use the rotational distance between the goal orientation R^{goal} and current object orientation R as the cost and define the reward function as $r = \frac{\pi}{2} - d$, with $d = \arccos\left(0.5(\mathrm{Tr}(R^{goal}R^T) - 1)\right)$, which computes the distance of two rotation matrices between R and R^{goal}. The constant term $\frac{\pi}{2}$ simply shifts the initial reward to 0. This reward encourages the robot to push the object to the stand-up orientation.

In both the assembly and pivoting tasks, we introduced Gaussian noise with a mean of zero and a standard deviation of 0.2 N to the FT sensor readings as measurement noise. Additionally, we applied a clipping operation to the collected contact force, limiting it to the range of ± 10 N for regulation purposes. To enhance the robustness of the learned skills, we incorporated randomization into the robot's initial pose.

For the assembly task, the robot's initial pose was uniformly sampled from a range of $[\pm 30\ mm, \pm 30\ mm, 30 \pm 5\ mm]$ along the X, Y, and Z axes, respectively. As for the pivoting task, the range for the initial pose was set to $[150 \pm 30\ mm, 5 \pm 5\ mm]$ along the X and Z axes relative to the rigid wall.

RL Training Details

We use the Soft Actor-Critic [23] with implementation in RLkit [75] to learn robot manipulation skills in simulation. The hyperparameter selections are summarized in Table. 7.1.

Hyperparameters	Assembly	Pivoting
Learning rate - Policy	1e-3	1e-4
Learning rate - Q function	1e-4	3e-4
Networks	[128,128] MLP	[128,128] MLP
Batch size	4096	4096
Soft target update (τ)	5e-3	5e-3
Discount factor (γ)	0.95	0.9
Replay buffer size	1e6	1e6
max path length	20	40
eval steps per epoch	100	400
expl steps per epoch	500	2000

Table 7.1: Hyperparameters for RL training

Online Optimization-Based Admittance Learning

We have learned a policy that can perform contact-rich manipulation tasks in simulation. However, the sim-to-real gap may prevent us from directly transferring the learned skills to the real world. Our goal is to adapt the offline learned skills, especially the admittance control parameters, with online data in real time. Instead of retraining skills with real-world data, we propose locally updating the control parameters using the latest contact force measurements during online execution. We formulate online learning as an optimization problem that optimizes the residual control parameters to achieve smooth trajectory and task completion criteria while respecting the interaction dynamics between the robot and the environment. We will describe the optimization constraints, objective function, and overall online learning algorithm in Section 7.3, 7.3, and 7.3, respectively.

Optimization Constraints

Robot dynamics constraint: Admittance control enables robot to behave as a mass-spring-damping system as shown in Eq. 7.1. We consider the robot state $x = [e, \dot{e}]$, where $e = x_c - x_d$, and we can obtain the robot dynamics constraint in the state space form:

$$\dot{x} = \begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} = f(x, F_{ext}, u) = \begin{bmatrix} \dot{e} \\ -M^{-1}D\dot{e} - M^{-1}Ke + M^{-1}F_{ext} \end{bmatrix} \tag{7.2}$$

where the optimization variable $u = [m_1^{-1}, \ldots, m_6^{-1}, k'_1, \ldots, k'_6, d'_1, \ldots, d'_6]^T$ is the diagonal elements of M^{-1}, $K' = M^{-1}K$, $D' = M^{-1}D$. $e, \dot{e}$ are the robot states that can be directly accessed, and F_{ext} is the external force that should be modeled from the environment dynamics.

Contact force estimation: Modeling the contact force explicitly is difficult because the contact point and mode can change dramatically during manipulation. Therefore, we propose to estimate the contact force online using the force/torque sensor measurements. In our experiments, we utilize a simple but effective *record & replay* strategy, where we record a sequence of force information $\{F_{ext}^0, \ldots, F_{ext}^T\}$ within a time window $[0, T]$ and replay them during the optimization.

There are other approaches for force estimation, such as using analytical contact models [20, 118] or numerically learning the contact force by model fitting.

To estimate or approximate the contact force in real time, we compare four approaches:

- *record & replay*: We record the force/torque from the most recent measurements within a time window and directly use the pre-recorded data as F_{ext} in the optimization.
- *hybrid impulse dynamics*: We use Eq. 7.2 with $F_{ext} = 0$ when there is no contact. For the contact, we model it implicitly as $M\dot{x}^- = \gamma M\dot{x}^+$, where $\dot{x}^-$ and $\dot{x}^+$ are the robot end-effector velocities before and after the contact. By online fitting the γ, we can optimize these hybrid dynamics to calculate the optimal parameters.
- *analytical contact model with online parameter fitting*: We model the contact explicitly using analytical models and fit the necessary parameters using online data, following [20, 118].
- *contact force fitting*: We fit a contact force model using online force sensor measurements.

However, the *hybrid impulse dynamics* approach is not suitable for our requirements. As shown in Fig. 7.2, the contact force profile in contact-rich manipulation indicates that the robot maintains contact with the environment most of the time. Therefore, neglecting the entire contact process and modeling it implicitly is not appropriate for our applications.

Similarly, *analytical contact model with online parameter fitting* does not fit our scenarios either. Although it has been successful in some pivoting tasks, it relies on the quasi-static assumption that does not hold in our scenario. One of the main challenges of transferring the admittance parameters is to avoid the robot bouncing on the object. Moreover, the analytical model assumes point or sliding contact modes, which may be hard to generalize to different tasks, such as assembly.

Finally, for *contact force fitting*, we assume a linear (spring-damping) contact force model: $F_{ext} = a(t)x(t) + b(t)\dot{x}(t) + c(t)$ within a short time window. We use the least square to estimate the parameters a, b, and c in real-time. Fig. 7.2 shows an example of fitting results. It can fit the force profile well in a short time window. However, as we need to apply the

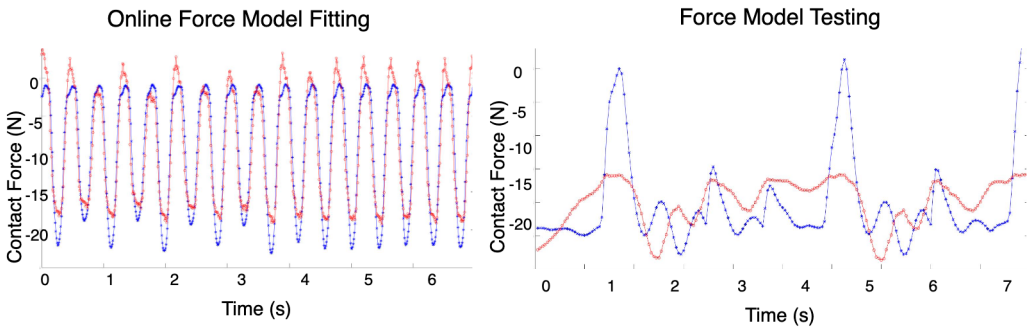


Figure 7.2: Performance of online force fitting (in z axis). In every time window, we collect the force/torque measurements and use the least square to fit the force model $F_{ext}(x, \dot{x}) = a(t)x(t) + b(t)\dot{x}(t) + c(t)$. On the left, it shows the linear model can fit the force profile locally. However, it can be extremely challenging to generalize to the next time window, as shown on the right.

model learned in the previous time window to the next step, the generalization ability is poor as it is hard to capture the peak of the force profile. Experiment videos comparing the performance of *contact force fitting* and *record & replay* are available on our website. We can observe that the contact force fitting method cannot stabilize the robot during contact.

Stability constraint: To ensure stability for admittance control, we need the admittance parameters to be positive-definite. Therefore, we constrain the optimization variable u to be positive.

Objective Function Design

We want to optimize the admittance parameters to establish stable contact and successfully achieve the task. Previous work [105] introduces the FITAVE objective $\int_0^T t|\dot{e}(t)|dt$ to effectively generate smooth and stable contact by regulating the robot's future velocity error. In addition, the ITAE objective $\int_0^{+\infty} t|e(t)|dt$ in [55] minimizes the position error to ensure the robot tracking the desired trajectory and finishes the task. We combine those two functions as our objective:

$$C(x) = \int_0^T t[w|e(t)| + (1-w)|\dot{e}(t)|]dt \tag{7.3}$$

where $w \in \mathbb{R}$ is a weight scalar to balance the trajectory smoothness and task completion criterion.

Online admittance learning

The optimization formulation is shown in Eq. 7.4. We optimize the residual admittance parameters δu, with u_{init} obtained from the offline learned skill.

$$\begin{aligned} \min_{\delta u} \quad & C(x) \\ \text{s.t.} \quad & \dot{x} = f(x, F_{ext}, u_{init} + \delta u) \\ & F_{ext} \leftarrow \textit{record \& replay} \\ & u_{init} + \delta u > 0 \end{aligned} \tag{7.4}$$

We illustrate the online admittance learning procedures in Alg. 4. In the online phase, we executed the skill learned offline on the real robot and recorded the contact force at each time step. Every T seconds, the online optimization uses the recorded force measurements, the current robot state and the admittance parameters learned offline to update the admittance parameter residual. The process runs in a closed-loop manner to complete the desired task robustly.

$$u = u_{init} + \delta u^*, M = diag\{m_1, \cdots, m_6\}, K = M \cdot diag\{k'_1, \cdots, k'_6\}, D = M \cdot diag\{d'_1, \cdots, d'_6\} \tag{7.5}$$

Algorithm 4: Online Admittance Residual Learning

Require: u_{init} from the offline policy $\pi(a|s)$, current robot state x

```
while task not terminated do
    if every T seconds then
        δu* ← admittance optimization in (7.4)
        M, K, D ← Recover admittance parameters from (7.5)
    end if
    {F_ext} ← record force sensor data
end while
```

7.4 Experiment Results

We conduct experiments on three contact-rich manipulation tasks, peg-in-hole assembly, pivoting, and screwing, to evaluate: 1) the robustness of sim-to-real transfer and 2) the generalizability of different task settings. We provide comparison results with two baselines in the assembly and pivoting tasks: 1) **Direct Transfer**: directly sim-to-real transfer both the learned robot trajectory and the control parameters [54], 2) **Manual Tune**: transfer learned trajectory with manually tuned control parameters [116]. We consider three metrics for evaluation: 1) **success rate** indicates the robustness of transfer, 2) **completion time** for successful trials denotes the efficiency of the skills, and 3) **max contact force** shows the safety. The screwing experiments further demonstrate the robustness of our method for solving complex manipulation tasks. In addition, the effect of weight selection of the proposed online optimization objective is discussed.

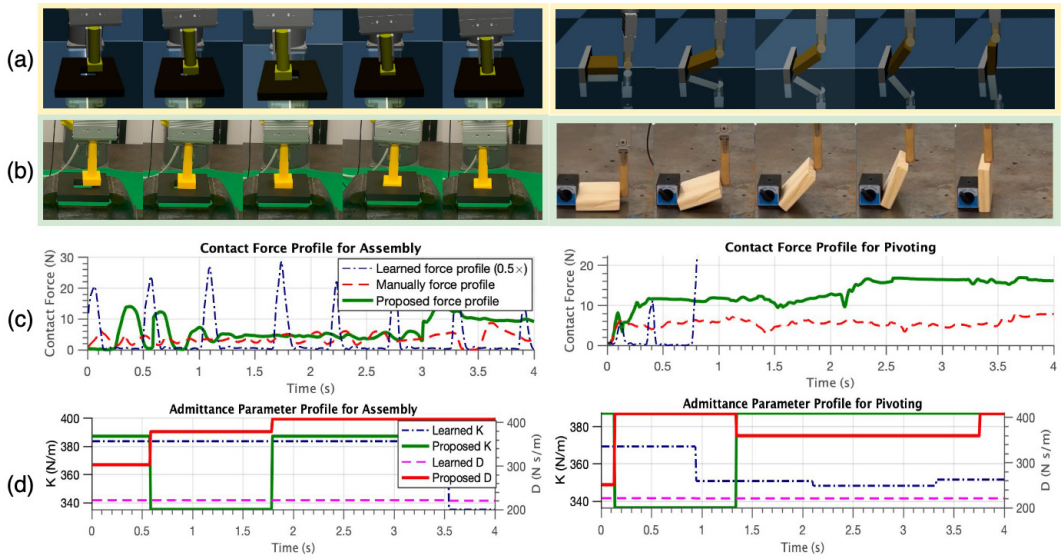


Figure 7.3: (a) shows the snapshots of the learned policy in simulation.(b) demonstrates the snapshots using the proposed approach for sim-to-real transfer. (c)(d) illustrate the forces and control parameters profiles for both the learned and proposed approach in the real world. The proposed approach can adjust the parameters to get the best performance in real time.

Skill Learning in Simulation

We use Soft Actor-Critic [23] to learn manipulation skills in simulation.[1] During the evaluation, the learned assembly and pivoting skills both achieved a 100% success rate. Fig. 7.3(a) shows the snapshots of the learned assembly skills. The robot learns to search for the exact hole location on the hole surface with a learned variable admittance policy and smoothly inserts the peg into the hole. For the learned pivoting skill, the robot pushes the object against the wall and gradually pivots it to the target pose with suitable frictional force.

Sim-to-Real Transfer

We evaluate the sim-to-real transfer performance on the same task. In the real world, the task setup, such as the object and robot geometry, is identical to the simulation. We mainly focus on evaluating the effect of the sim-to-real gap on robot/environment dynamics.

We first apply the offline learned skill to the real world. From the experiments, we notice that the *Direct Transfer* baseline fails to produce safe and stable interactions. As depicted in Fig 7.3(c), for peg-in-hole assembly tasks, the peg bounces on the hole surface and generates

[1]We also tested other RL algorithms like DDPG [46] and TD3 [23]. As a result, all the methods are able to learn a policy and have similar performance when transferring to the real robot. Details can be found on our website.

large contact forces, making the assembly task almost impossible to complete. Similarly, in the pivoting task, the robot cannot make stable contact with the object and provide enough frictional force for pivoting.

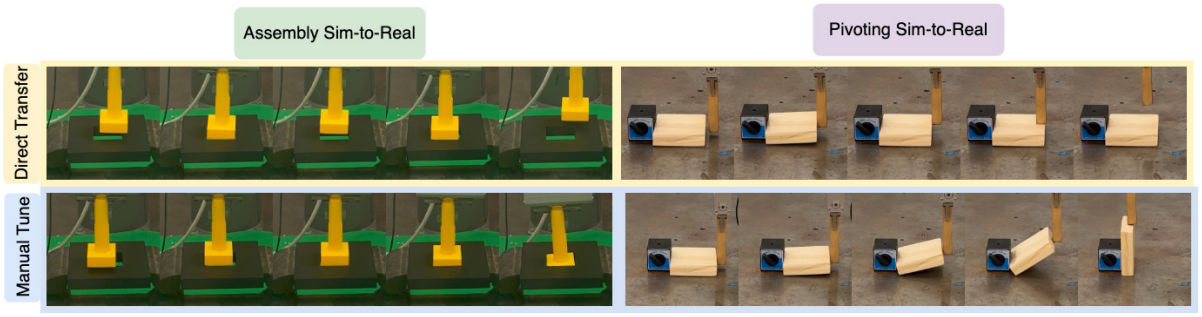


Figure 7.4: Snapshots of baseline approaches for the sim-to-real experiment. The control parameters learned in the simulation will result in a large contact force and make the robot bounce on the surface, which will, in turn, result in failures of the tasks.

Then we examine whether the learned robot motion is valid with manually tuned control parameters. As shown in Tab. 7.2, the *Manual Tune* baseline can achieve a 100% success rate for both tasks. This supports our hypothesis and previous works that the manipulation trajectory is directly transferable with suitable control parameters to address the sim-to-real gap.

However, manual tuning requires extensive human labor. We want to evaluate whether the proposed online admittance learning framework can perform similarly without any tuning. Table 7.2 presents an overview of the sim-to-real transfer results. For all experiments, the weight parameter w of the proposed approach was consistently set to 0.4. Notably, our proposed method achieves a 100% success rate in the assembly task, along with a 90% success rate in the pivoting task. Furthermore, it achieves these results while exhibiting shorter completion times than the other two baselines.

We also investigate the contact force and the adjusted admittance parameters during the manipulation, shown in Fig. 7.3(c)(d). Initially, the robot establishes contact with the environment using the offline learned parameters, resulting in a large applied force. In the subsequent update cycle, the proposed method effectively adjusts the parameters by decreasing K and increasing D, enabling the robot to interact smoothly with the environment and reduce the contact force. Later, it increases K and decreases D to suitable values to finish the task more efficiently.

Generalization to Different Task Settings

The aforementioned experiments highlight the ability of the proposed framework to achieve sim-to-real transfer within the same task setting. In this section, we aim to explore the

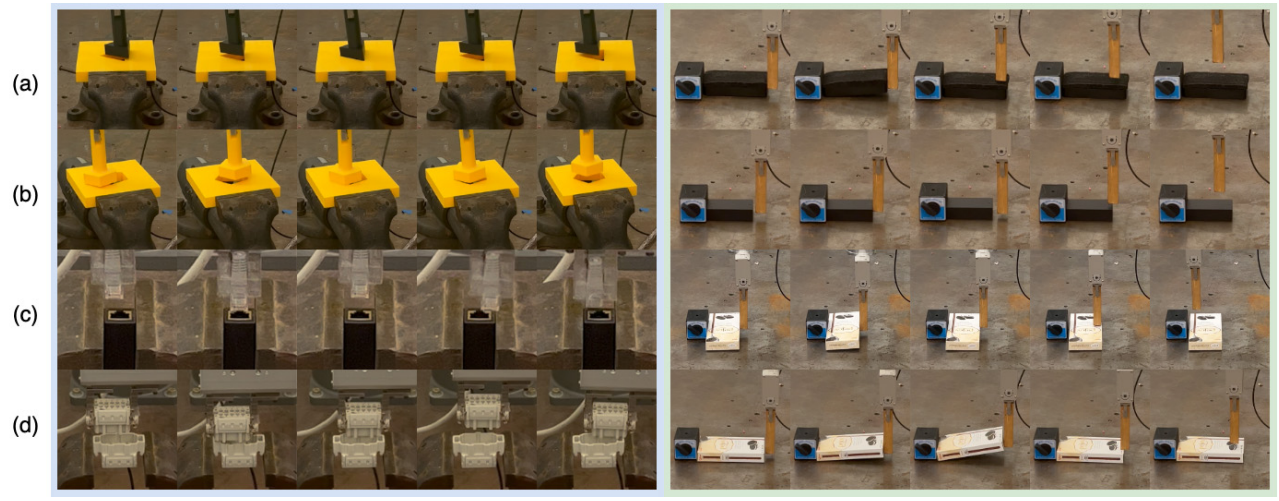


Figure 7.5: Snapshots of directly using the learned policy to generalize to various task settings. The snapshots and videos of the baseline methods are available on our website.

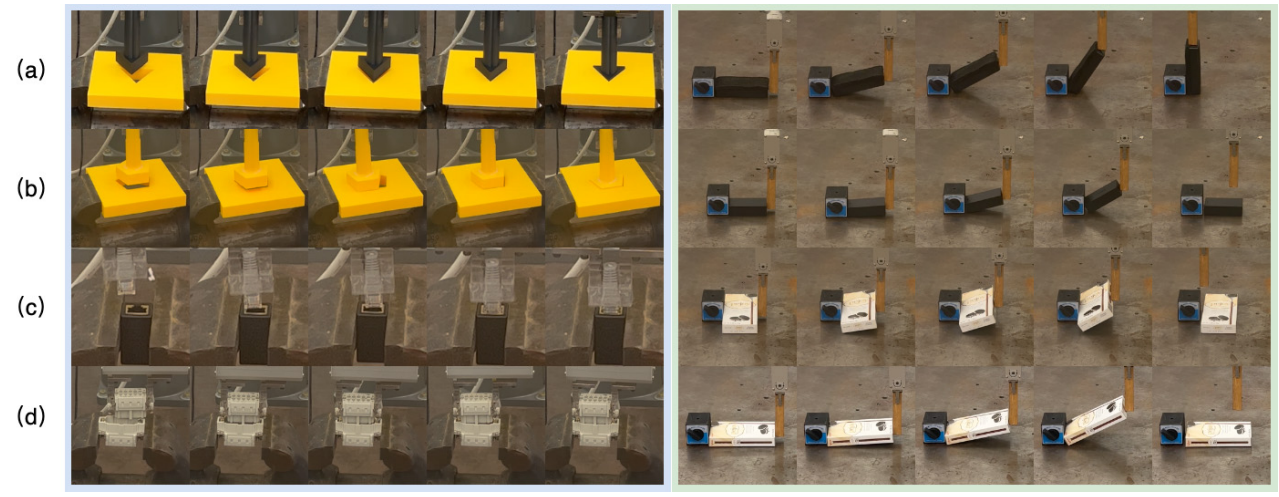


Figure 7.6: Snapshots of directly using learned trajectory and the manually tuned admittance control parameters to generalize to various task settings. The snapshots and videos of the baseline methods are available on our website.

	Assembly Task			Pivoting Task		
	Succ. Rate	Time (s)	Max F (N)	Succ. Rate	Time (s)	Max F (N)
Proposed	10/10	19.0 ± 11.2	23.6 ± 6.3	9/10	25.6 ± 2.1	20.1 ± 4.1
Manual	10/10	28.1 ± 8.6	10.3 ± 2.2	10/10	25.3 ± 3.6	9.2 ± 0.6
Direct	3/10	39.0 ± 12.8	63.7 ± 6.8	0/10	*N/A*	30.7 ± 4.6

Table 7.2: Success rate evaluation in real-world experiments.

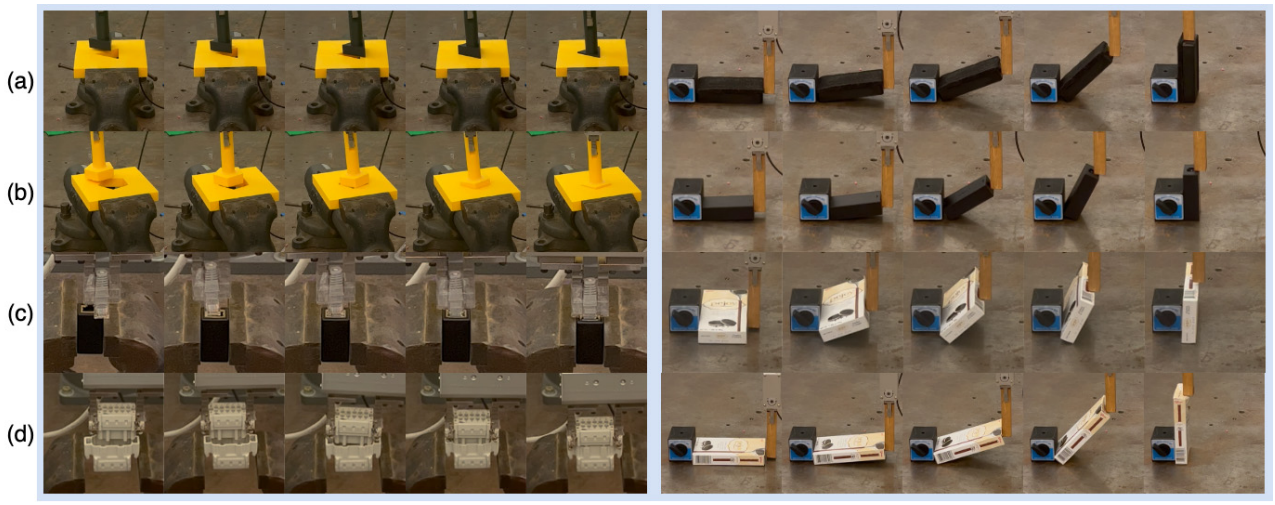


Figure 7.7: Snapshots of using the proposed approach to generalize to various task settings. The snapshots and videos of the baseline methods are available on our website.

generalization capabilities of the proposed approach across different task settings, which may involve distinct kinematic and dynamic properties. For two baselines, we directly use the manually tuned or learned control parameters of the training object for new tasks.

	Triangle (gap = 1mm)		Pentagon (gap = 1mm)		Ethernet (gap = 0.17mm)		Waterproof (gap = 0.21mm)	
	Succ. Rate	Time (s)	Succ. Rate	Time (s)	Succ. Rate	Time (s)	Succ. Rate	Time (s)
Proposed	10/10	15.9 ± 6.2	10/10	20.1 ± 8.9	9/10	42.1 ± 13.7	9/10	37.8 ± 17.7
Manual	8/10	$43. \pm 17.0$	9/10	38.0 ± 18.0	1/10	78.0 ± 0.0	0/10	*N/A*
Direct	0/10	*N/A*	1/10	7.0 ± 0.0	0/10	*N/A*	0/10	*N/A*

	Adapter [L=8.8 cm, w=69g]		Eraser [L=12.2 cm, w=36g]		Pocky Short [L=7.9 cm, w=76g]		Pocky Long [L=14.8 cm, w=76g]	
	Succ. Rate	Time (s)	Succ. Rate	Time (s)	Succ. Rate	Time (s)	Succ. Rate	Time (s)
Proposed	8/10	25.0 ± 4.8	9/10	28.4 ± 2.7	8/10	12.9 ± 1.7	7/10	31.8 ± 11.0
Manual	0/10	*N/A*	10/10	30.0 ± 1.0	1/10	19.0 ± 0.0	1/10	40.0 ± 0.0
Direct	0/10	*N/A*	0/10	*N/A*	0/10	*N/A*	0/10	*N/A*

Table 7.3: Generalization performance to different assembly tasks (**Top**) and pivoting tasks (**Below**).

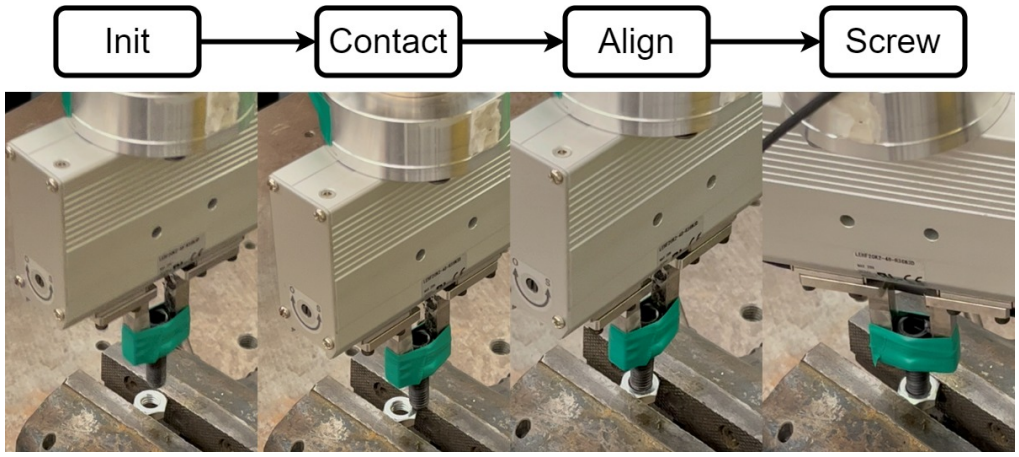


Figure 7.8: Snapshots of the screwing task

Peg-in-hole assembly: We test various assembly tasks, including polygon-shaped peg-holes such as triangles and pentagons, as well as real-world socket connectors like Ethernet and waterproof connectors. These tasks are visualized in Fig. 7.1(c). The outcomes of our experiments are outlined in Table 7.3. Our proposed method achieves 100% success rates on the polygon shapes and a commendable 90% success rate on Ethernet and waterproof connectors. Moreover, the completion time of the proposed method is much shorter than other baselines. The *Manual Tune* baseline also achieves decent success rates on the polygon shapes as it is similar to the scenario in that we tune the parameters. However, for the socket connectors, due their tighter fit and irregular shapes, substantial force is required for insertion (approximately 15N for Ethernet and 40N for waterproof connectors) and *Manual Tune* baseline cannot accomplish these two tasks.

Pivoting: Similarly, we conduct a series of pivoting experiments on various objects, encompassing diverse geometries and weights as shown in Table 7.3. Remarkably, our proposed approach exhibits robust generalization capabilities across all tasks, achieving a success rate exceeding 70%. However, when relying solely on manually tuned parameters, the ability to pivot an object is limited to the eraser that has a similar length to the trained object and is the lightest object in the test set. As the object geometry and weight diverge significantly, the manually tuned parameters often fail to establish stable contact with the object and exert sufficient force to initiate successful pivoting.

Screwing:

We conducted experiments on a more challenging robot screwing task to further validate our method as shown in Fig. 7.8. Its primary challenge is to precisely align the bolt with a nut and then smoothly secure them together. To address this, we employed the assembly skills previously learned for aligning the bolt and nut and then used a manually-designed rotation primitive to complete the screwing. Throughout the process, online admittance learning

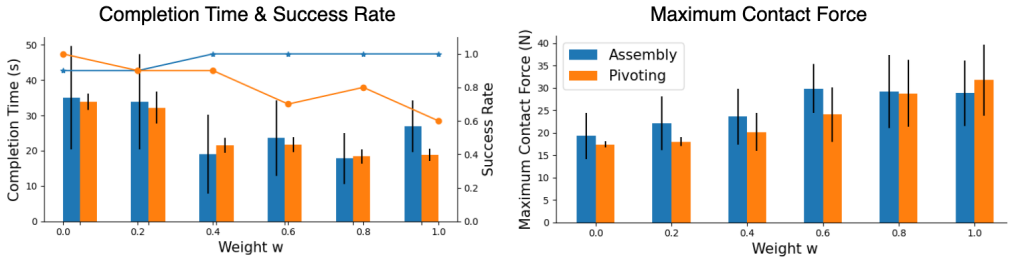


Figure 7.9: Ablation on the weight parameter. The left figure shows the completion time and success rate with respect to different w, and the right figure shows the contact force.

continually optimizes the admittance controller. Impressively, our approach allowed the robot to consistently and reliably align and secure the nut and bolt. We executed this task five times, achieving a 100% success rate.

Ablation: Objective Weight Selection

In this subsection, we would like to study the effect of weight selection. We evaluated different weight parameters on both the assembly and pivoting tasks. The results are depicted in Figure 7.9. For the assembly task, all proposed method variations achieve a 100% success rate except for $w <= 0.2$. Smaller weight parameters tend to prioritize trajectory smoothness, which may not provide sufficient contact force for successful insertion. On the other hand, in pivoting tasks, larger weight values led to a decrease in the success rate. This is because larger weight values prioritize task completion, potentially leading to a failure in establishing a stable initial contact for pivoting. These observations align with the objective design motivation. Based on our findings, selecting the parameter 0.4 strikes a good balance between both objectives and yields the best overall performance.

7.5 Chapter Summary

This chapter proposes a contact-rich skill-learning framework for sim-to-real transfer. It consists of two main components: skill learning in simulation during the offline phase and admittance learning on the real robot during online execution. These components work together to enable the robot to acquire the necessary skills in simulation and optimize admittance control parameters for safe and stable interactions with the real-world environment. We evaluate the performance of our framework in three contact-rich manipulation tasks: assembly, pivoting, and screwing. Our approach achieves promising success rates in both tasks and demonstrates great generalizability across various tasks.

Chapter 8

Conclusions and Future Work

This dissertation has presented methodologies aimed at enabling robots to interact reliably with their environment and objects in real time. Our focus was distributed across three key aspects: robotic motion planning (Chapters 2 and 3), adaptive model learning (Chapters 4, 5, and 6), and online control policy adaptation (Chapter 7).

In Chapter 2, we introduced a novel trajectory optimization algorithm that effectively breaks down complex, long-range trajectory planning problems into manageable segments. These segments are processed in parallel, significantly boosting computational efficiency. The resultant trajectories from each segment are then cohesively integrated into a single path, maintained by a consensus constraint that ensures segment continuity. This innovative approach allows for the distribution of computational complexity and the creation of high-quality trajectories more efficiently. Future efforts could focus on optimizing the convergence speed of the consensus optimization process to further enhance efficiency.

Chapter 3 proposed the Bilevel Path Optimization Formulation for Motion Planning (BPOMP), a novel formulation that efficiently addresses potential collisions using sparse waypoints. An innovative collision constraint, focusing on the closest position to obstacles along the continuous path, was integrated. The problem was then formulated as a bilevel optimization, and subsequently relaxed to canonical nonlinear programming (NLP). Comparative analyses with leading path optimization/sampling algorithms demonstrated that BPOMP enhances the efficiency and success rates of optimization-based planners. Future research might integrate sampling planners within this framework to generate initial paths that circumvent local optima.

Chapter 4 detailed a uniform framework for manipulating deformable linear objects, incorporating state estimation, task planning, and trajectory planning. Utilizing coherent point drift (CPD), a robust real-time observer was developed to accurately estimate the position of each node on a rope, even in noisy and occluded environments. A task planner was then introduced to guide robotic actions during manipulation processes. Experimentation on rope knotting tasks validated the effectiveness of these methods. The current work limits to simple scenarios with minor sensor noise and occlusions. Future studies will aim to extend these methods to more complex and challenging environments.

In Chapter 5, we explored the robotic manipulation of deformable linear objects. We tackled the challenge of accurately modeling object deformation for precise control by developing a hybrid offline-online learning methodology. The offline phase utilized a Graph Neural Network (GNN) for foundational model building, which was then aligned with real-world scenarios through a real-time linear residual model. This model was integrated into a trust region-based Model Predictive Controller (MPC), crucial for calculating optimal robotic movements. The method's effectiveness was proven in simulations and real-world tests, but only in 2D environments. Expanding this framework to three-dimensional contexts and more complex objects is a promising direction for future work.

Chapter 6 introduced SPR-RWLS, a novel framework designed for cable manipulation in environments with significant sensor noise and occlusions. The Structure Preserved Registration (SPR) method was employed for robust real-time tracking, complemented by an online dynamic deformation model approximation through robust parallel optimization. Experimental results underscored the framework's ability to manipulate cables precisely, even under challenging conditions. While experimental results were promising, the current methodology has limitations in real-world applications like cable assembly or medical scenarios. Future work will look to adapt this framework to more realistic industrial settings.

Chapter 7 presented a framework for skill learning in contact-rich manipulation tasks, suitable for sim-to-real transfer. This framework combined skill learning in simulation (offline phase) with admittance learning on the actual robot (online phase), enabling safe and stable real-world interactions. Tested on tasks such as assembly, pivoting, and screwing, the framework demonstrated high success rates and significant generalizability across various tasks. However, its application has been limited to these specific tasks. The next steps involve expanding these methodologies to more complex scenarios such as cutting and polishing.

Bibliography

[1] Fares J Abu-Dakka, Leonel Rozo, and Darwin G Caldwell. "Force-based Learning of Variable Impedance Skills for Robotic Manipulation". In: *2018 IEEE-RAS 18th Int. Conf. on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 1–9.

[2] Anurag Ajay et al. "Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3066–3073.

[3] Jonathan F Bard. *Practical bilevel optimization: algorithms and applications*. Vol. 30. Springer Science & Business Media, 2013.

[4] Cristian C Beltran-Hernandez et al. "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach". In: *Applied Sciences* 10.19 (2020), p. 6923.

[5] Gino van den Bergen. "A fast and robust GJK implementation for collision detection of convex objects". In: *Journal of graphics tools* 4.2 (1999), pp. 7–25.

[6] Dimitri P Bertsekas. "Nonlinear programming". In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334.

[7] Paul T Boggs and Jon W Tolle. "Sequential quadratic programming". In: *Acta numerica* 4 (1995), pp. 1–51.

[8] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[9] Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. *The mathematical theory of finite element methods*. Vol. 3. Springer, 2008.

[10] Jonas Buchli et al. "Learning variable impedance control". In: *The Int. J. of Robotics Research* 30.7 (2011), pp. 820–833.

[11] Yevgen Chebotar et al. "Closing the sim-to-real loop: Adapting simulation randomization with real world experience". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.

[12] Yao-Chon Chen. "Solving robot trajectory planning problems with uniform cubic B-splines". In: *Optimal Control Applications and Methods* 12.4 (1991), pp. 247–262.

[13] Cheng Chi et al. "Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects". In: *arXiv preprint arXiv:2203.00663* (2022).

[14] Rohan Chitnis et al. "Efficient bimanual manipulation using learned task schemas". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1149–1155.

[15] Haili Chui and Anand Rangarajan. "A new point matching algorithm for non-rigid registration". In: *Computer Vision and Image Understanding* 89.2 (2003), pp. 114–141.

[16] Andrea Colesanti and Daniel Hug. "Hessian measures of semi-convex functions and applications to support measures of convex bodies". In: *manuscripta mathematica* 101.2 (2000), pp. 209–238.

[17] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. `http://pybullet.org`. 2016–2019.

[18] Alexander Domahidi, Eric Chu, and Stephen Boyd. "ECOS: An SOCP solver for embedded systems". In: *2013 European control conference (ECC)*. IEEE. 2013, pp. 3071–3076.

[19] Jing Dong et al. "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs." In: *Robotics: Science and Systems*. Vol. 12. 2016, p. 4.

[20] Neel Doshi, Orion Taylor, and Alberto Rodriguez. "Manipulation of unknown objects via contact configuration regulation". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2693–2699.

[21] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. "A fast procedure for computing the distance between complex objects in three-dimensional space". In: *IEEE Journal on Robotics and Automation* 4.2 (1988), pp. 193–203.

[22] Federico Girosi, Michael Jones, and Tomaso Poggio. "Regularization theory and neural networks architectures". In: *Neural computation* 7.2 (1995), pp. 219–269.

[23] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).

[24] Karol Hausman et al. "Learning an embedding space for transferable robot skills". In: *International Conference on Learning Representations*. 2018.

[25] Zhe Hu, Peigen Sun, and Jia Pan. "Three-dimensional deformable object manipulation using fast online gaussian process regression". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 979–986.

[26] Zhe Hu et al. "3-D deformable object manipulation using deep neural networks". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4255–4261.

[27] Tadanobu Inoue et al. "Deep reinforcement learning for high precision assembly tasks". In: *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Syst. (IROS)*. IEEE. 2017, pp. 819–825.

[28] Shiyu Jin, Changhao Wang, and Masayoshi Tomizuka. "Robust deformation model approximation for robotic cable manipulation". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6586–6593.

[29] Shiyu Jin et al. "Contact pose identification for peg-in-hole assembly under uncertainties". In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 48–53.

[30] Shiyu Jin et al. "Real-time state estimation of deformable objects with dynamical simulation". In: *Workshop on Robotic Manipulation of Deformable Objects 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[31] Shiyu Jin et al. "Robotic cable routing with spatial representation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5687–5694.

[32] Mrinal Kalakrishnan et al. "STOMP: Stochastic trajectory optimization for motion planning". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 4569–4574.

[33] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[34] Alina Kloss, Stefan Schaal, and Jeannette Bohg. "Combining learned and analytical models for predicting action effects". In: *arXiv preprint arXiv:1710.04102* 11 (2017).

[35] Shir Kozlovsky, Elad Newman, and Miriam Zacksenhouse. "Reinforcement learning of impedance policies for peg-in-hole tasks: Role of asymmetric matrices". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10898–10905.

[36] Shunsuke Kudoh et al. "In-air Knotting of Rope by a Dual-arm Multi-finger Robot". In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 6202–6207.

[37] James J Kuffner and Steven M LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[38] Ashish Kumar et al. "Rma: Rapid motor adaptation for legged robots". In: *arXiv preprint arXiv:2107.04034* (2021).

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

[40] Steven M LaValle et al. "Rapidly-exploring random trees: A new tool for path planning". In: (1998).

[41] A.X. Lee et al. "A non-rigid point and normal registration algorithm with applications to learning from demonstrations". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. May 2015, pp. 935–942. DOI: 10.1109/ICRA.2015.7139289.

[42] Alex X Lee et al. "Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 4402–4407.

[43] Jessica Leu et al. "Efficient Robot Motion Planning via Sampling and Optimization". In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 4196–4202.

[44] Mushu Li et al. "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization". In: *IEEE Transactions on Vehicular Technology* 69.3 (2020), pp. 3424–3438.

[45] Yunzhu Li et al. "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids". In: *arXiv preprint arXiv:1810.01566* (2018).

[46] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[47] Vincent Lim et al. "Real2Sim2Real: Self-Supervised Learning of Physical Single-Step Dynamic Actions for Planar Robot Casting". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 8282–8289. DOI: 10.1109/ICRA46639.2022.9811651.

[48] Hsien-Chung Lin et al. "Human guidance programming on a 6-dof robot with collision avoidance". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 2676–2681.

[49] Changliu Liu, Chung-Yen Lin, and Masayoshi Tomizuka. "The convex feasible set algorithm for real time optimization in motion planning". In: *SIAM Journal on Control and optimization* 56.4 (2018), pp. 2712–2733.

[50] Qinghua Liu, Xinyue Shen, and Yuantao Gu. "Linearized admm for nonconvex nonsmooth optimization with convergence analysis". In: *IEEE Access* 7 (2019), pp. 76131–76144.

[51] Lennart Ljung. *System identification*. Springer, 1998.

[52] Jianlan Luo et al. "Reinforcement learning on variable impedance controller for high-precision robotic assembly". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3080–3087.

[53] Jeffrey Mahler et al. "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics". In: *arXiv preprint arXiv:1703.09312* (2017).

[54] Roberto Martín-Martín et al. "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks". In: *arXiv preprint arXiv:1906.08880* (2019).

[55] Fernando G Martins. "Tuning PID controllers using the ITAE criterion". In: *International Journal of Engineering Education* 21.5 (2005), p. 867.

[56] Jonathan Meijer, Qujiang Lei, and Martijn Wisse. "Performance study of single-query motion planning for grasp execution using various manipulators". In: *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE. 2017, pp. 450–457.

[57] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. "Spline-based motion planning for autonomous guided vehicles in a dynamic environment". In: *IEEE Transactions on Control Systems Technology* 26.6 (2017), pp. 2182–2189.

[58] Dimitris Metaxas and Demetri Terzopoulos. "Shape and nonrigid motion estimation through physics-based synthesis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.6 (1993), pp. 580–591.

[59] Takuma Morita et al. "Knot planning from observation". In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 3. IEEE. 2003, pp. 3887–3892.

[60] Mustafa Mukadam, Xinyan Yan, and Byron Boots. "Gaussian process motion planning". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 9–15.

[61] Mustafa Mukadam et al. "Continuous-time Gaussian process motion planning via probabilistic inference". In: *The International Journal of Robotics Research* 37.11 (2018), pp. 1319–1340.

[62] Andriy Myronenko and Xubo Song. "Point set registration: Coherent point drift". In: *IEEE transactions on pattern analysis and machine intelligence* 32.12 (2010), pp. 2262–2275.

[63] Ashvin Nair et al. "Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation". In: *arXiv preprint arXiv:1703.02018* (2017).

[64] Yashraj Narang et al. "Factory: Fast contact for robotic assembly". In: *arXiv preprint arXiv:2205.03532* (2022).

[65] David Navarro-Alarcon and Yun-Hui Liu. "Fourier-based shape servoing: A new feedback method to actively deform soft objects into desired 2-D image contours". In: *IEEE Transactions on Robotics* 34.1 (2017), pp. 272–279.

[66] David Navarro-Alarcon et al. "Model-free visually servoed deformation control of elastic objects by robot manipulators". In: *IEEE Transactions on Robotics* 29.6 (2013), pp. 1457–1468.

[67] David Navarro-Alarcon et al. "Visually servoed deformation control by robot manipulators". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 5259–5264.

[68] Brendan O'Donoghue, Giorgos Stathopoulos, and Stephen Boyd. "A splitting method for optimal control". In: *IEEE Transactions on Control Systems Technology* 21.6 (2013), pp. 2432–2442.

[69] Christian Ott, Ranjan Mukherjee, and Yoshihiko Nakamura. "Unified impedance and admittance control". In: *2010 IEEE international conference on robotics and automation*. IEEE. 2010, pp. 554–561.

[70] Jia Pan, Sachin Chitta, and Dinesh Manocha. "FCL: A general purpose library for collision and proximity queries". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3859–3866.

[71] Mihir Parmar, Mathew Halm, and Michael Posa. "Fundamental challenges in deep learning for stiff contact dynamics". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 5181–5188.

[72] Luka Peternel, Tadej Petrič, and Jan Babič. "Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface". In: *2015 IEEE int. conf. on robotics and automation (ICRA)*. IEEE. 2015, pp. 1497–1502.

[73] Antoine Petit, Vincenzo Lippiello, and Bruno Siciliano. "Real-time tracking of 3D elastic objects with an RGB-D sensor". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 3914–3921.

[74] Tobias Pfaff et al. "Learning mesh-based simulation with graph networks". In: *arXiv preprint arXiv:2010.03409* (2020).

[75] Rail-Berkeley. *Rail-Berkeley/Rlkit: Collection of reinforcement learning algorithms*. URL: `https://github.com/rail-berkeley/rlkit`.

[76] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. "Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators". In: *arXiv preprint arXiv:1906.01728* (2019).

[77] Felix Rey et al. "Fully decentralized ADMM for coordination and collision avoidance". In: *2018 European Control Conference (ECC)*. IEEE. 2018, pp. 825–830.

[78] Joel Rey et al. "Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies". In: *Autonomous Robots* 42.1 (2018), pp. 45–64.

[79] Alvaro Sanchez-Gonzalez et al. "Learning to simulate complex physics with graph networks". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8459–8468.

[80] John Schulman et al. "Finding locally optimal, collision-free trajectories with sequential convex optimization." In: *Robotics: science and systems*. Vol. 9. 1. Citeseer. 2013, pp. 1–10.

[81] John Schulman et al. "Learning from Demonstrations through the Use of Non-Rigid Registration". In: *in Proceedings of the 16th International Symposium on Robotics Research (ISRR)*. 2013.

[82] John Schulman et al. "Learning from demonstrations through the use of non-rigid registration". In: *Robotics Research: The 16th International Symposium ISRR*. Springer. 2016, pp. 339–354.

[83] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[84] John Schulman et al. "Tracking deformable objects with point clouds". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 1130–1137.

[85] Joohwan Seo et al. "Robot Manipulation Task Learning by Leveraging SE (3) Group Invariance and Equivariance". In: *arXiv preprint arXiv:2308.14984* (2023).

[86] Yuki Shirai et al. "Chance-Constrained Optimization in Contact-Rich Systems for Robust Manipulation". In: *arXiv preprint arXiv:2203.02616* (2022).

[87] Vikas Sindhwani, Rebecca Roelofs, and Mrinal Kalakrishnan. "Sequential operator splitting for constrained nonlinear optimal control". In: *2017 American Control Conference (ACC)*. IEEE. 2017, pp. 4864–4871.

[88] Bothina El-Sobky and Y Abo-Elnaga. "A penalty method with trust-region mechanism for nonlinear bilevel optimization problem". In: *Journal of Computational and Applied Mathematics* 340 (2018), pp. 360–374.

[89] Jochen Stüber, Claudio Zito, and Rustam Stolkin. "Let's push things forward: A survey on robot pushing". In: *Frontiers in Robotics and AI* (2020), p. 8.

[90] Lingfeng Sun et al. "Distributed Multi-agent Interaction Generation with Imagined Potential Games". In: *arXiv preprint arXiv:2310.01614* (2023).

[91] Yu Sun et al. "Online learning of unknown dynamics for model-based controllers in legged locomotion". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 8442–8449.

[92] Bingjie Tang et al. "IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality". In: *arXiv preprint arXiv:2305.17110* (2023).

[93] Te Tang and Masayoshi Tomizuka. "Track deformable objects from point clouds with structure preserved registration". In: *The International Journal of Robotics Research* (2018), p. 0278364919841431.

[94] Te Tang, Changhao Wang, and Masayoshi Tomizuka. "A framework for manipulating deformable linear objects by coherent point drift". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3426–3433.

[95] Te Tang et al. "Robotic manipulation of deformable objects by tangent space mapping and non-rigid registration". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 2689–2696.

[96] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[97] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.* IEEE. 2012, pp. 5026–5033.

[98] Nghia Vuong, Hung Pham, and Quang-Cuong Pham. "Learning Sequences of Manipulation Primitives for Robotic Assembly". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4086–4092.

[99] Andreas Wächter and Lorenz T Biegler. "Line search filter methods for nonlinear programming: Motivation and global convergence". In: *SIAM Journal on Optimization* 16.1 (2005), pp. 1–31.

[100] Andreas Wächter and Lorenz T Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106.1 (2006), pp. 25–57.

[101] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge university press, 2019.

[102] Changhao Wang, Jeffrey Bingham, and Masayoshi Tomizuka. "Trajectory splitting: A distributed formulation for collision avoiding trajectory optimization". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 8113–8120.

[103] Changhao Wang et al. "Bpomp: A bilevel path optimization formulation for motion planning". In: *2022 American Control Conference (ACC)*. IEEE. 2022, pp. 1891–1897.

[104] Changhao Wang et al. "Offline-online learning of deformation model for cable manipulation with graph neural networks". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5544–5551.

[105] Changhao Wang et al. "Safe online gain optimization for Cartesian space variable impedance control". In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2022, pp. 751–757.

[106] Yu Wang, Wotao Yin, and Jinshan Zeng. "Global convergence of ADMM in nonconvex nonsmooth optimization". In: *Journal of Scientific Computing* 78.1 (2019), pp. 29–63.

[107] Zheng Wu et al. "Prim-LAfD: A Framework to Learn and Adapt Primitive-Based Skills from Demonstrations for Insertion Tasks". In: *arXiv preprint arXiv:2212.00955* (2022).

[108] Zheng Wu et al. "Zero-Shot Policy Transfer with Disentangled Task Representation of Meta-Reinforcement Learning". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 7169–7175.

[109] Mengyuan Yan et al. "Self-supervised learning of state estimation for manipulating deformable linear objects". In: *IEEE robotics and automation letters* 5.2 (2020), pp. 2372–2379.

[110] Wilson Yan et al. "Learning predictive representations for deformable objects using contrastive estimation". In: *arXiv preprint arXiv:2003.05436* (2020).

[111] Mingrui Yu et al. "A coarse-to-fine framework for dual-arm manipulation of deformable linear objects with whole-body obstacle avoidance". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 10153–10159.

[112] Mingrui Yu et al. "Adaptive Control for Robotic Manipulation of Deformable Linear Objects with Offline and Online Learning of Unknown Models". In: *arXiv preprint arXiv:2107.00194* (2021).

[113] Mingrui Yu et al. "Generalizable whole-body global manipulation of deformable linear objects by dual-arm robot in 3-D constrained environments". In: *arXiv preprint arXiv:2310.09899* (2023).

[114] Xiang Zhang et al. "Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning". In: *arXiv preprint arXiv:2310.10509* (2023).

[115] Xiang Zhang et al. "Learning Generalizable Pivoting Skills". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5865–5871.

[116] Xiang Zhang et al. "Learning insertion primitives with discrete-continuous hybrid action space for robotic assembly tasks". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 9881–9887.

[117] Xiang Zhang et al. "Learning variable impedance control via inverse reinforcement learning for force-related tasks". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2225–2232.

[118] Jiaji Zhou et al. "A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation". In: *The International Journal of Robotics Research* 37.2-3 (2018), pp. 249–265.

[119] Wenxuan Zhou and David Held. "Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity". In: *ICRA 2022 Workshop: Reinforcement Learning for Contact-Rich Manipulation*. 2022. URL: `https://openreview.net/forum?id=Zrp4wpa9lqh`.

[120] Ziyi Zhou and Ye Zhao. "Accelerated admm based trajectory optimization for legged locomotion with coupled rigid body dynamics". In: *2020 American Control Conference (ACC)*. IEEE. 2020, pp. 5082–5089.

[121] Jihong Zhu et al. "Dual-arm robotic manipulation of flexible cables". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 479–484.

[122] Jihong Zhu et al. "Vision-based manipulation of deformable and rigid objects using subspace projections of 2d contours". In: *Robotics and Autonomous Systems* 142 (2021), p. 103798.

[123] Matt Zucker et al. "Chomp: Covariant hamiltonian optimization for motion planning". In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 1164–1193.