Balancing Probabilistic and Optimization-Based Planning Strategies for Task-Specific Robotic Manipulation

Liu Ruishuang

October, 2023

Balancing Probabilistic and Optimization-Based Planning Strategies for Task-Specific Robotic Manipulation

A dissertation submitted to
THE GRADUATE SCHOOL OF ENGINEERING SCIENCE
OSAKA UNIVERSITY
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY IN ENGINEERING

BY

Liu Ruishuang

October, 2023

Abstract

In the execution of complex, skillful tasks, humans leverage a combination of strategic planning, adept tool usage, and coordinated hand-eye movements. Replicating these human capabilities in robotic systems forms the central objective of this thesis. A critical aspect of this endeavor involves the careful selection of planning algorithms, which requires a balance between probabilistic and optimization-based planning strategies.

My research is primarily based on the development and evaluation of two robotic manipulation systems. These systems utilize general-purpose robot arms and grippers, representing substantial progress toward autonomous robotic manipulation. Each system provides unique insights into their respective tasks and lays the foundation for improvements in robotic performance and efficiency. A central challenge I tackled pertains to task and motion planning, including the formulation of tasks and optimization of robot motion, considering the environmental constraints and efficiency. Given the goal of robust system development, I also examine several pertinent while non-trivial aspects, including mechanism design, 3D shape modeling, and error recovery. These elements contribute significantly to the overall robustness and resilience of the systems.

The key contribution of this thesis lies in a comprehensive analysis of these novel robotic manipulation tasks. It highlights the importance of selecting suitable planning algorithms to serve specific tasks. By combining these strategies, I equip robot arms with the ability to perform complex tasks autonomously, effectively, and precisely.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Challenges in Robot Manipulation

The advent of robotics has catalyzed a profound transformation in diverse areas, ranging from industrial operations to the facets of everyday life. Their ability to perform complex tasks with precision, adapt dynamically to varying conditions, and function in diverse environments lends to their growing effectiveness. The application of general-purpose robot manipulators further improves the flexibility and adaptability of robotic manipulation. Their application has now broadened, reaching into our daily lives [1], and proving integral to sectors such as the high-mix, low-volume production industry [2].

In modern manipulation, robots are applied to not only mechanical and repetitive tasks but also flexible tasks to work autonomously in various situations [3][4]. Essentially, robotic manipulation encapsulates the activities performed by a human hand. It involves not just the physical movement of the robot, but also requires sensory inputs for environmental comprehension, planning algorithms to strategize movements, and control algorithms for actual motion execution. However, the integration of these elements into a cohesive system to ensure the safe and efficient accomplishment of tasks is non-trivial.

The complexity of robotic manipulation encompasses many difficult problems include: mechanism, perception, modeling and control, planning, and uncertainty [5][6].

Each facet presents unique challenges and together they shape the intricacy of robotic manipulation:

- Mechanism refers to the design of robotic grippers or tools. Researchers pursue designing and organizing mechanical components to realize versatile and robust functions in an elegant way [7][8].
- Perception bridges the gap between robots and the real world. This includes two groups of sensors that work complementarily. It comprises two complementary groups of sensors: one functioning as the robot's eyes, responsible for scanning and locating objects (visual perception) [9], and the other endowing the robot with the ability to perceive its own motion and interaction with the environment (haptic perception) [10][11][12].
- Modeling and control provide an environment for the manipulation process where we can analyze, simulate, plan, and control our manipulation systems. Many of the underlying phenomena are challenging, including contacts [13], impact [14], and deformation [15].
- Planning generates a sequence of actions to accomplish a specific task or goal, considering the physical and operational constraints of the system [16].
- Uncertainty [17] presents challenges that go beyond the combinatorics issue mentioned above. Within a robotic manipulation system, uncertainties can originate from various sources such as partial or noisy observations of the environment, detection errors, and more.

For a specific manipulation task, the above challenges should be thoroughly addressed to ensure optimal performance and efficacy. This involves a detailed understanding of the task-specific requirements and a careful selection of suitable modeling and planning methodologies.

In this thesis, I focus on skillful robotic manipulation systems, using general-purpose robot arms and grippers. Given a specific manipulation task, I structure it within a general framework, as depicted in Fig.1.1. The first stage involves the

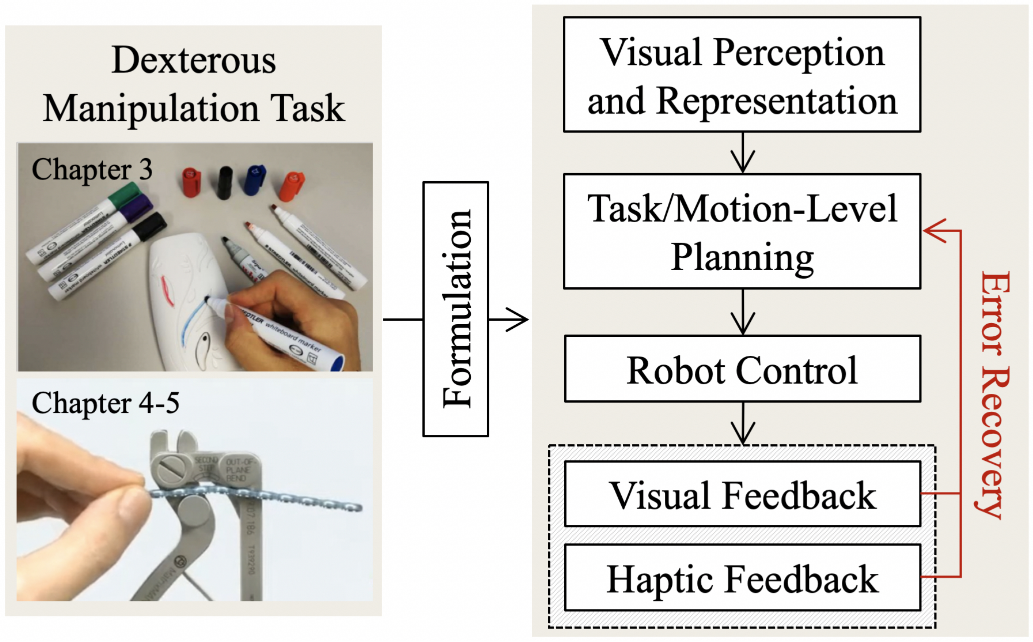

Dexterous Manipulation Task
Chapter 3
Chapter 4-5
Formulation
Visual Perception and Representation
Task/Motion-Level Planning
Robot Control
Visual Feedback
Haptic Feedback
Error Recovery

Figure 1.1: General robot manipulation system.

use of visual perception to extract relevant data about the task and its environment, which is subsequently represented in an appropriate format. Following this, task and motion planning is initiated, which involves transposing the human task into a robotic task executable by the system while considering various constraints. This stage also involves optimizing the robot's motion to enhance efficiency. Subsequently, the planned robotic task is carried out, guided by a suitable control policy. In order to ensure the system performs successfully under uncertainties, multi-sensory feedback mechanisms are utilized for error recovery. Upon receiving this information, the planner processes it and makes necessary adjustments accordingly, thus enhancing system adaptability and resilience.

In short, the systems proposed in this thesis leverage a general mechanism, focusing on perception, modeling and control, planning, and uncertainty management, with the aim of developing a reliable and robust robotic manipulation system.

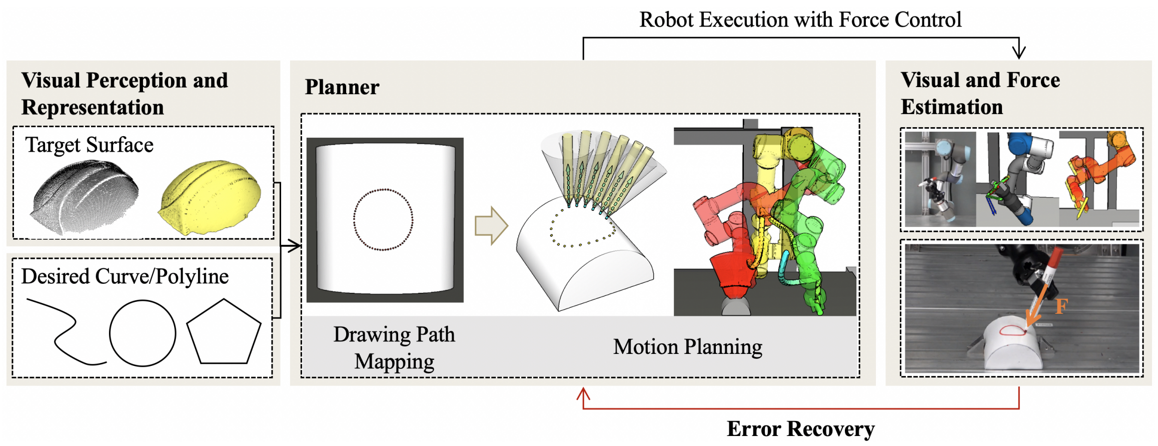

Robot Execution with Force Control
Visual Perception and Representation
Target Surface
Desired Curve/Polyline
Planner
Drawing Path Mapping
Motion Planning
Visual and Force Estimation
F
Error Recovery

Figure 1.2: Robotic drawing system. There are three essential components in the system: (1) "Drawing Path Mapping" accepts mesh models and vectorized 2D strokes from the first input, and maps the vectorized 2D strokes to the mesh model. (2) "Motion Planning" determine grasp poses and plan manipulation motion. (3) "Error Recovery" component using visual and force feedback to ensure the robustness of the system.

1.2 Robotic Manipulation Planning

For each system proposed in this thesis, there are several sub-tasks to be formulated. The choice of planning algorithm is contingent on the specific conditions and the state of the solution space. The term "planning" in this context is a layered process ranging from high-level task planning, low-level motion planning, and down to grasp planning. While maintaining flexibility for re-planning in response to environmental uncertainty. In this sub-section, I will analyze two primary methods of planning: the probabilistic-based method and the optimization-based method. Through this investigation, I strive to find insights that may inform the choice of method under various circumstances.

Robotic manipulation tasks can often be classified as either discrete or continuous, based on the type of actions they involve [18]. Discrete manipulation tasks involve actions that have clear, distinct stages or steps. An example might be a robot picking up an object, which is a single action that can be completed or not completed. Continuous manipulation tasks involve actions that have a range of possible outcomes

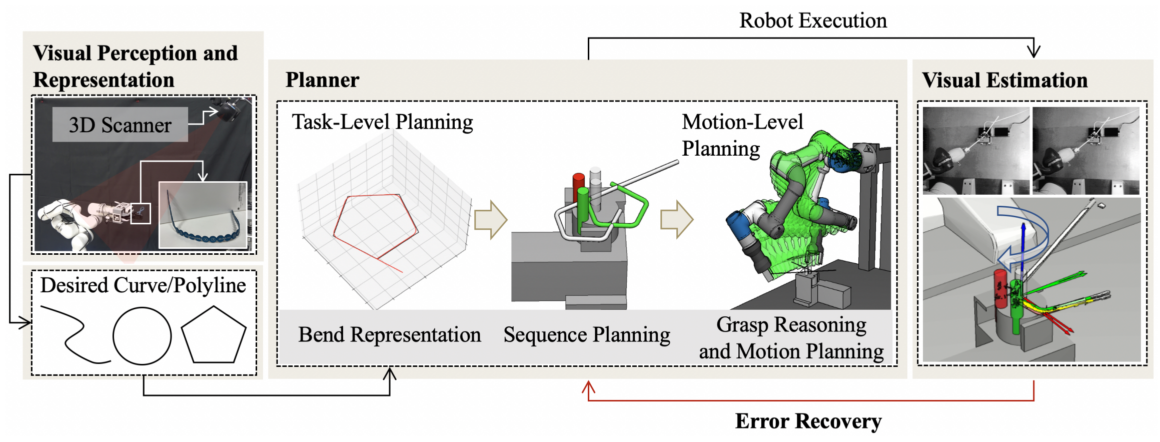


Figure 1.3: Robotic bending system. There are four essential components in the system: (1)“Visual Perception and Representation” capture the goal shape effectively using a robot arm and a static 3D scanner. (2) “Bending Representation” approximate the 3D shape with discrete bending action and schedules the sequence to fulfill geometric and kinematic constraints. (3) “Motion-Level Planning” determine grasp poses and plan manipulation motion. (4)“Error Recovery” component using visual feedback to compensate for the springback error.

and aren’t necessarily binary. For example a welding task or painting a surface, where the exact path of the robot’s tool matters a great deal.

These different types of tasks inherently impose diverse constraints [19] and result in solution spaces. Consequently, different planning methods are applied to handle these variations.

- **Probabilistic-based Method:** Probabilistic-based methods typically involve some level of randomness or stochasticity, and they excel at exploring complex, high-dimensional state spaces. The solution space is significantly large and encompasses not just one path. Probabilistic Roadmaps (PRM) [20] and Rapidly-exploring Random Trees (RRT) [21][22] are typical probabilistic methods used for motion planning. However, the trajectories they generate can be very jerky and include unnecessary motions. Therefore, after probabilistic-based planners return their solutions, trajectory smoothing and shortening are often needed, which can be achieved by trajectory optimization-based motion planners.

- **Optimization-based Method:** Optimization-based methods aim to find the best possible solution according to a defined objective function, and they are often deterministic, though stochastic optimization methods do exist. These methods are ideal to solve problems with continuous solutions since the time axis is directly integrated into the objective function. They are particularly effective when the problem requires finding an optimal solution and the state space is not overly large or can be effectively approximated. For example, Ichnowski et al. [23] present grasp-optimized motion planning, which speeds up the execution of a bin-picking robot's operations by incorporating robot dynamics and a set of candidate grasps into an optimizing motion planner. Hess et al. [24] achieved coverage path planning of 3D surfaces by modeling the problem as a generalized traveling salesman problem (GTSP).

However, real-world robotic manipulation tasks present a mix of discrete and continuous constraints. A task might involve a discrete action (picking up a tool) followed by a continuous action (using the tool to perform some task). Thus it is crucial to balance and integrate these two planning strategies effectively [25] considering different task constrain.

Generally, the selection of these two kinds of planning methods can be based on criteria as follows: (1) Solution Space Complexity: Probabilistic methods are particularly effective when the solution space is high-dimensional or non-convex; (2) Solution Continuity: The optimization method is ideal to solve problems with continuous solutions since the time axis is directly integrated into the objective function; (3) Precision Requirement: Optimization-based methods excel when the task necessitates finding the most optimal solution within a well-defined and relatively simple solution space; (4) Uncertainty: Probabilistic methods can handle uncertainty, as they can represent the probability distributions over the solution space; (5) Computation Time: Probabilistic methods can often find a feasible solution faster than optimization-based methods, especially when the number of parameters is large.

Note that the above criteria are not exhaustive. The choice between probabilistic and optimization-based methods also depends on the specific context, task constraints, and additional factors beyond this analysis.

1.3 Organization of the Thesis

This thesis is structured as follows:

Chapter 2 delves into existing literature and studies related to three aspects: 3D geometry representation; Robotic perception; Robotic manipulation planning. This lays the theoretical groundwork for the subsequent developments in this thesis.

Chapter 3 presents the design and functionality of a 3D robotic drawing system as shown in Fig. 1.2. There are three essential components in the system: (1) “Drawing Path Mapping” accepts mesh models and vectorized 2D strokes from the first input, and maps the vectorized 2D strokes to the mesh model. (2) “Motion Planning” determine grasp poses and plan manipulation motion. (3) “Error Recovery” component using visual and force feedback to ensure the robustness of the system.

Chapters 4 and 5 present the development of a robotic bending system using a peripheral bending machine as shown in Fig. 1.3. There are four essential components in the system: (1) “Visual Perception and Representation” capture the goal shape effectively using a robot arm and a stationary 3D scanner. (2) “Bending Representation” approximate the 3D shape with discrete bending action and schedules the sequence to fulfill geometric and kinematic constraints. (3) “Motion-Level Planning” determine grasp poses and plan manipulation motion. (4) “Error Recovery” component using visual feedback to compensate for the springback error. Chapter 4 primarily focus on the perception of the desired shape (component 1), and Chapter 5 dedicated to the task and motion planning process (component 2-4).

Chapter 6 concludes my study and suggests future research directions.

Chapter 2

Related Work

Related studies in three aspects are reviewed in this section: 3D geometry representation; Robotic perception; Robotic manipulation planning.

2.1 3D Geometry Representation

Representation of real-world geometry in the context of robotics and computer vision pertains to the methods and algorithms used to digitally depict the physical environment in a format that can be understood and manipulated by computers [26].

2.1.1 Representation Taxonomy

Fig. 2.1 shows a taxonomy of 3D representations, which is classified based on the representation itself. These representations can be broadly categorized into discrete and continuous types. Discrete representations involve distinct, separate values. An example of a discrete representation in a 3D model is a set of points in space, where each point is a distinct entity. Such representations are often used in scenarios where systems can be accurately modeled with a finite number of states or variables. Continuous representations, on the other hand, involve values that can vary continuously and have an infinite number of possible states. These representations are often used in scenarios where systems have an infinite number of possible states, or where more

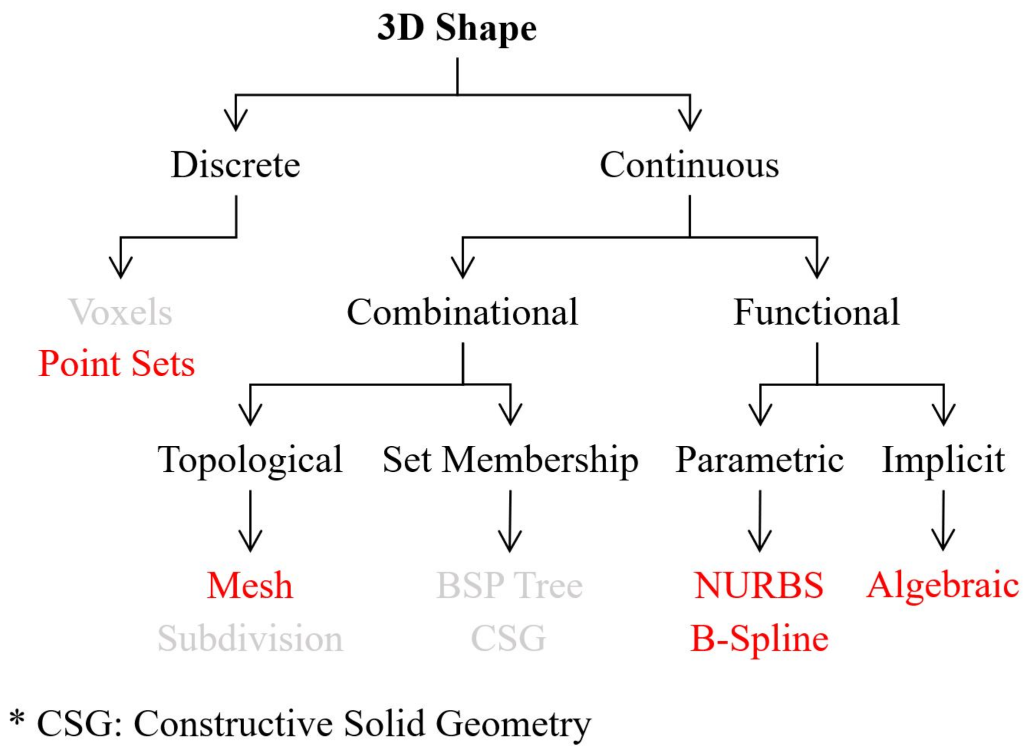


Figure 2.1: Taxonomy of 3D representations. The representations applied in this thesis are highlighted in red.

accurate or smooth models are required.

Continuous representations can be subdivided into two classifications: combinational and functional. Combinational representations merge different elements to create a complete 3D shape. Functional representations, define the shape as a function of one or more variables. This form of representation is used when an object can be accurately modeled or approximated by a mathematical function.

In the context of the tasks addressed in this thesis, various representations are employed to serve different purposes, as highlighted in red in Fig. 2.1.

2.1.2 Geometry-based Classification

These 3D representations can also be sorted according to their geometry into curves, surfaces, and solids. Here I introduce some common used example methods to model each type of shape.

Curve

A curve is a one-dimensional object. Curves in 3D space can be represented in multiple ways.

- Parametric Curve: A curve represented by three functions $x(t)$, $y(t)$, $z(t)$ where each function describes the coordinates of points on the curve as a function of parameter t. For example, parametric polynomial curves, Bezier curves[27] and B-splines [28].
- Vector: A position vector can be used to represent a curve in 3D space. This approach is useful when dealing with physical problems, where the position, velocity, or force at different points along a trajectory can be described as vectors.

Surfaces

Surface data refers to representations that only model the boundary or exterior of an object. Some common types of surface representations are as follows.

- Polygonal Meshes (Explicit) [29]: These are 3D models made up of vertices, edges, and faces, typically used in computer graphics. Meshes can be created from point clouds or other data and well preserve the details on the surface of 3D objects.
- Implicit Surface [30]: Surfaces can be represented by an equation of the form $f(x, y, z) = 0$. Any point (x, y, z) that satisfies this equation lies on the surface. They are highly suitable for representing complex, organic shapes and for performing operations like blending and deformation. For example, Algebraic surface.
- Parametric Surface [31]: A surface in 3D space can be represented using two parameters (u, v) where $x(u, v)$, $y(u, v)$, and $z(u, v)$ give the coordinates of points on the surface. They offer a compact representation of complex shapes and are widely used in computer-aided design (CAD) and graphics. For example, NURBS surface [32].

Solid

Solid data refers to the representation that includes information about the interior of a 3D object, not just the surface. Below are some common techniques used to create or represent solid models [33][34].

- Spatial occupancy enumeration: This method utilizes occupancy grids, applying either a uniform grid such as a voxel [35], or a non-uniform grid like an octree [36] to discretize the environment. It can be useful for tasks like occupancy mapping, where the robot needs to understand which parts of the environment are empty and which are occupied.
- Cell decomposition: This method represents a solid by decomposing it into several cells. Spatial occupancy enumeration is a particular case of cell decomposition where all cells are cubical and arranged in a uniform grid. Cell decomposition provides convenient ways for computing certain topological properties of solids such as their connectedness (number of pieces) and genus (number of holes).
- Constructive Solid Geometry (CSG) [37]: This method creates complex objects by combining primitive objects (such as spheres, cylinders, etc.) using Boolean operations like union, intersection, and difference.
- Sweeping [38]: This method involves extending a shape or profile along a path to create a 3D object. It is particularly useful for objects that have a consistent cross-section along a path, such as a bottle, a tube, or a wire.

2.2 Robotic Perception

Robotic perception refers to a robot's ability to sense and interpret its environment to make decisions or perform tasks. It involves the use of sensors [39][40][41] and algorithms [42][43][44] to create a representation of the world that the robot can understand and interact with. As for humans, robot perception for manipulation is multimodal. It can be classified as visual perception and haptic perception.

2.2.1 Visual Perception

Visual perception refers to enabling machines to "see" and understand their environment. This is typically achieved through computer vision, a field that includes methods for acquiring, processing, analyzing, and understanding digital images to produce numerical or symbolic information [9].

Visual perception is applied to achieve an estimation or a manipulation goal. For example, object segmentation allows for the division of a digital image into multiple segments to simplify the image or to identify objects of interest [45][46]. Object detection deals with detecting instances of visual objects of a certain class (such as humans, animals, or cars) within an image [47]. And object pose estimation provides valuable information about the spatial orientation of detected objects, proving crucial in tasks requiring precise manipulation [48][49]. These applications of visual perception help robotic systems effectively interact with and navigate within their environments.

2.2.2 Haptic Perception

Haptic perception plays a significant role in robotics by simulating the sense of touch through the use of forces, vibrations, or motions. It has a complementary nature to visual perception, and often fills the gaps where vision may not provide the complete picture. The importance of this technology becomes particularly evident when a robotic hand has made contact with an object directly or indirectly. The haptic data not only guide the manipulation but can also provide insights into uncertainties. For example, it can help in identifying inaccuracies in the estimated physical properties of the object, thereby enhancing the overall efficacy of the manipulation task.

Haptic perception includes kinesthetic perception [10], force perception [11], and tactile perception [12]. These subcategories respectively enable robots to know their own body position and movement, measure the force/torque exerted during interactions with the environment, and sense the texture or shape of an object. For instance, force perception was employed in early research to achieve compliant motion control [50]. Sato et al. [51] achieved the task of cleaning a vertical flat surface with the help of a trajectory/force tracking controller. Song et al. [52] realized robotic drawing on an arbitrary surface using force feedback. There are also a plenty of works on tactile perception. Li et al. [53] trained a deep neural network (DNN) for slip detection using visual and tactile information captured from the camera and Gelsight sensor [54]. Similar research was also done by [55][56] using different tactile sensors.

2.3 Robotic Manipulation Planning

Robotics researchers frequently draw inspiration from human behavior, particularly given our unique ability to utilize tools to tackle various manipulation challenges. The flexibility of the human form, combined with our cognitive abilities, enables us to handle a myriad of tasks with adeptness, ranging from intricate in-hand manipulation to the fabrication of advanced tools.

2.3.1 Combined Task and Motion Planning (TAMP)

The concept of Combined Task and Motion Planning (TAMP) integrates high-level task planning, often referred to as decision-making, with low-level motion planning. Task planning involves generating a sequence of actions, elementary or abstract, to guide an agent toward achieving a complex task in a given environment [57][58].

TAMP extends beyond this by concurrently developing high-level actions and planning the agent's motion in alignment with these actions [59] [60]. While combined task and motion planning has been studied for decades, recent literature extends it with uncertainty, force constraints, and deeply learned heuristics [16]. For example, Holladay et al. [61] extended an existing task and motion planner with controllers that exert wrenches while considering torque and frictional limits to achieve forceful manipulation. Toussaint et al. [62] leveraged optimization methods for physical reasoning and sequential manipulation planning by formulated force constraints. Silver et al. [63] proposed a bottom-up relational learning method for operator learning and demonstrated how the learned operators could be used for planning in a TAMP system. Wang et al. [64] learned constraints from several training examples at the task level to enable faster TAMP-based planning.

Previously in my group, there are research focused on reasoning and constrained planning methods for robots to manipulate tools. The studies were under the TAMP framework. For example, Raessa et al. [65] presented a method to teach a dual-arm robot to use common electric tools. Chen et al. [66] designed a motion planner to manipulate a suction cup tool. Sanchez et al. [67] solved the problem of robot-cable entanglements using four-arm collaborative TAMP. Chen et al. [68] developed a planner that automatically finds an optimal assembly sequence for a dual-arm robot to build a woodblock structure while considering supporting grasps from a second hand. Wan et al. [69] presented a planner to solve a test tube arrangement problem.

2.3.2 Manipulation for Tool Use

Using tools is an extensively studied robotic manipulation problem. Three levels of planning are needed to enable a robot with a general gripper to manipulate tools. The first level is grasp pose reasoning, where the goal is to find common grasp poses for starting and goal object poses. The reasoning problem has been studied a lot in the robotic manipulation community previously. For example, Saut et al. [70] developed a planner to plan grasp poses for multi-finger hands and then used the planned poses to reason pick-and-place sequences. Wan et al. [71] developed similar approaches and analyzed their performance for assembly tasks. After finding common grasps poses, the next steps include selecting a proper one from them and planning the motion. The related problems are grasp optimization [72] and constrained motion planning and control [73]. Grasp optimization relies on the chosen quality metrics. Previously, several different metrics have been proposed and compared [74][75][76]. The third level is task-constrained motion planning. Traditional motion planning algorithms used probabilistic roadmaps approaches to search collision-free motion in the joint space [77][20][78]. Incorporating task constraints into the planning process represents significant challenges [79] and the following works fight the difficulties. Toussaint et al. studied planning using tools while considering physical interactions [80]. Holloday proposed a method to formulate force constraints in tool manipulation tasks [81]. Beschi et al. [82] studied planning optimal trajectories for redundant industrial robots in machining applications.

Tool use can also be learned from the demonstration. Brown [83] presented a robot agent which learns to exploit objects in its environment as tools by watching a single demonstration.

Chapter 3

Robotic 3D Drawing

This section develops a flexible and robust robotic system for autonomously drawing on 3D surfaces. The system takes 2D drawing strokes and a 3D target surface (mesh or point clouds) as input. It maps the 2D strokes onto the 3D surface and generates a robot motion to draw the mapped strokes using visual recognition, grasp pose reasoning, and motion planning. The system is flexible compared to conventional robotic drawing systems as we do not fix drawing tools to the end of a robot arm. Instead, a robot recognizes and picks up pens online and holds the pens to draw 3D strokes. Meanwhile, the system has high robustness thanks to the following crafts: First, a high-quality mapping method is developed to minimize deformation in the strokes. Second, visual detection is used to re-estimate the drawing tool's pose before executing each drawing motion. Third, force control is employed to compensate for noisy visual detection and calibration and ensure a firm touch between the pen tip and the surface. Fourth, error detection and recovery are implemented to deal with slippage and other anomalies. The planning and executions are performed in a closed-loop manner until the strokes are successfully drawn. We evaluate the system and analyze the necessity of the various crafts using different real-world tasks. The results show that the proposed system is flexible and robust to generate robotic motion that picks up the pens and successfully draws 3D strokes on given surfaces.

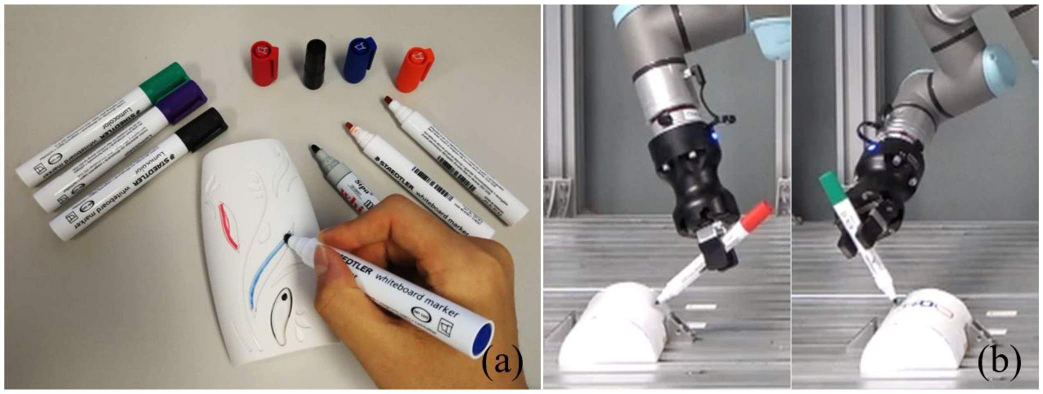


Figure 3.1: (a) A human draws on a 3D surface by picking multiple pens. (b) This section aims at developing a robotic system that performs similar online pen picking and 3D surface drawing using closed-loop planning and vision-force feedback.

3.1 Introduction

This section develops a flexible and robust robotic system for drawing on 3D surfaces using visual detection, planning, and control. We especially pay our attention to autonomously planning the drawing motion considering the following three aspects: 3D surface, flexibility, and robustness. First, the system accepts 2D drawing strokes and a 3D mesh model or 3D point cloud of the target surface as input. It maps the 2D strokes to the 3D surface for robotic motion trajectory planning. Second, the drawing pens are not fixed to the flange of a robot arm. The robot plans grasp poses to pick up a drawing pen and holds the pen to draw the mapped 3D strokes. The robot is able to change to other pens online and draw multi-color graphs. Third, since detecting, grasping, and manipulating pens may lead to errors, the system leverages in-hand pen pose estimation, force control, and error detection and recovery to ensure robust executions. It works in a closed-loop manner until the strokes are successfully drawn.

Robotic drawing can be viewed as moving a pen along a trajectory in a certain pattern concerning the target surface. Previously at manufacturing sites, the trajectories were taught in a point-to-point way by professional system integration engineers. The process was time-consuming and required expert knowledge. Modern systems solve the problem using motion planning. For example, [84][85][86][87] developed algorithms for painting path planning and optimization on a free-form surface to ensure the even distribution of the paint. Painting is a simplified robotic drawing case as there is no contact between a robotic end-effector and a painting surface. In contrast, robotic drawing, especially drawing with hand-held pens, is a contact-rich manipulation problem. Closing up the planning using vision and force feedback is an important solution to maintain the contact between a pen and a surface and is widely studied [88]. Especially, force control methods play an important role in helping to achieve a stable drawing motion [52]. Although we can find lots of state-of-the-art studies that explore closing up the loop to realize robust and practical robotic systems, they assume that drawing tools are fixed to a robot end flange, which significantly influences the flexibility of switching among diverse stroke colors and types.

In the human world, artists leverage rich tactile and force feelings to control a drawing pen and frequently change their pens during the drawing to switch to different stroke colors and types, as is shown in Fig. 3.1(a). Inspired by human artists, we develop a robotic system that performs similar 3D surface drawing using closed-loop planning and online picked pens. We especially focus on using visual and force feedback to make the system robust and flexible like human actions. The system's planning component is based on our previous work, which enabled a robot to use tools [66]. We improve the planner to realize a flexible and robust robotic 3D drawing system by including 3D stroke mapping and vision-force feedback. Multiple pens and grippers can be directly used without any modification. There is no need for human intervention – A robot autonomously reasons grasp poses, regrasp poses, and plans joint motion following the mapped 3D goal strokes while considering vision and force feedback.

We highlight our contributions as follows. (1) We do not fix pens to a robot end flange. Instead, we allow picking up pens using a vision system and holding the pen to draw graphs. (2) We develop a high-quality metrological method to map 2D strokes to 3D surfaces. (3) We use in-hand pose estimation to correct accumulated errors and trigger re-planning if no solution was found. (4) We employ force control to compensate for noisy visual detection and calibration and ensure a firm touch between the pen tip and the surface. (5) We implement error detection and recovery to deal with slippage and other unexpected problems. These contributions may look fragmented, but each of them plays an important role in solving a particular problem. They interact with each other to accredit robust multi-pen 3D drawing. In the experimental section, we evaluate our system using various tasks. The results show that the proposed system is flexible and robust to generate robot motion from grasping pens to finishing drawing.

The paper's organization is as follows: First, we review the related work in Section II. Then, we outline an overview of the system architecture in Section III. Detailed descriptions of the mapping and feedback planning are presented respectively in Section IV and V. Experiments, comparisons, and analysis are carried out in Section V. Conclusions and future work are presented in Section VI. Besides, we present optimization-based planning and error recovery methods in the Appendix for readers looking for flexible alternatives.

3.2 Related Work

We review the related studies in two aspects: Robotic drawing systems; Mapping 2D strokes to 3D object surfaces.

3.2.1 Robotic Drawing Systems

Developing automated systems to draw figures is an important and popular topic in the robotics and automation communities. One of the historical work that used machines to draw pictures is [89]. The book presented a computer program named AARON, which was able to draw figures and shapes on paper using a pen driven by a mechanical plotting machine. With the rapid development of robotic hardware and computer vision, many researchers studied drawing using multi-joint robots. The diversity and flexibility of automated drawing systems increased significantly with the increased number of joints. For example, Calinon et al. developed a portrait drawing system using a 4-DoFs robotic arm [90]. Tresset et al. [91] presented an advanced portrait drawing robot named "Paul", which was able to draw portraits using the equivalent of an artist's stylistic signature based on several processes that mimic drawing primitives or skills. Jun et al. [92] used a humanoid robot to draw a large picture on a wall. Sasaki et al. [93] developed a deep learning method to learn drawing motion with a given target image. More recently, Gulzow et al. presented the e-David (electronic Drawing Apparatus for Vivid Image Display) system, which allowed controlling a variety of painting robots and end-effectors to create artistic drawings [94]. Igno-Rosario et al. developed a novel interactive system for detecting regions and then creating robotic artworks by painting at the region level [95]. Guo et al. developed an architecture for human-in-the-loop robotic painting. The system was able to learn and interact with a human artist, imitate the artist's painting techniques, and reproduce the painting drawn by the human [96]. Scalera et al. used the Non-Photorealistic Rendering (NPR) technique to convert an image to artistic painting strokes and planned robotic manipulator motion to draw them [97]. The above studies focused on drawing on a plane. Besides them, drawing on non-planar surfaces is also a popular topic and received much research interest. For example, Lam et al. [98] developed an automated 3-DoFs sketching system to conduct pen drawing on a 2.5D surface. Song et al. [52] implemented a robot that used a pen to draw on an arbitrary surface with force feedback.

Compared to the previous robotic drawing systems, our difference is that we focus on 3D surface drawing using online picked pens. We develop computational geometric algorithms to map 2D strokes to 3D surfaces or 3D point clouds, develop reasoning and motion planning algorithms to select and change pens, and close the planning loop using feedback control.

3.2.2 Mapping 2D Strokes to 3D Object Surfaces

An intuitive solution to 2D-to-3D mapping used in previous robotic drawing systems[52, 98] was to project 2D strokes onto xy-plane and estimate the z values according to the surface's shape. Although the method could successfully create 3D strokes, they suffer from deformation problems. The Euclidean distances of the mapped 3D strokes get distorted from its 2D origin. Thus, researchers started to explore deformation-free mapping. One effective method is Least Squares Conformable Mapping (LSCM) [99, 100]. The method is widely used in computer graphics for creating a UV map from mesh models to textures. Song et al. [101] leveraged the LSCM method to implement distortion-free stroke mapping. They spread 3D meshes to 2D planes using LSCM, mapped 2D strokes on the spread meshes, and wrapped them back to create 3D ones.

Some other deformation-free mapping methods do not spread 3D surfaces explicitly. Instead, they control the deformation in distances by using mesh metrology [102]. These methods are widely used in applications like the surface following. For example, Carmelo et al. [103] proposed a method for computing the mesh-following trajectories of a surface by measuring distances along the intersection lines between the surface and a bunch of scanning planes. A dual-arm inspection system was developed based on the proposed method. Can et al. [104] presented an algorithm to project 2D patterns onto B-spline surfaces using a similar metrological method.

Table 3.1: Summary of 2D-to-3D Mapping Methods

Method	Projective	Conformal Mapping	Metrological
Deformation	Large	Medium	Small
Efficiency	Low	Low	High
Implementation	Simple	Difficult	Simple
Robustness	High	Low	High
Related Studies	[52][98]	[99][100][101]	[102][103][104]

Table 3.1 shows a summary of the various 2D-to-3D mapping methods mentioned above. The projective method is the most intuitive one. It is robust but suffers from large deformation. Compared to the projective method, the conformal mapping-based methods have smaller deformation. However, they depend a lot on initial conditions and are more difficult to implement. The metrological methods have the least deformation, but are time-consuming since the metrics need to be iteratively examined. All these methods require the 3D surfaces as input. The surfaces could

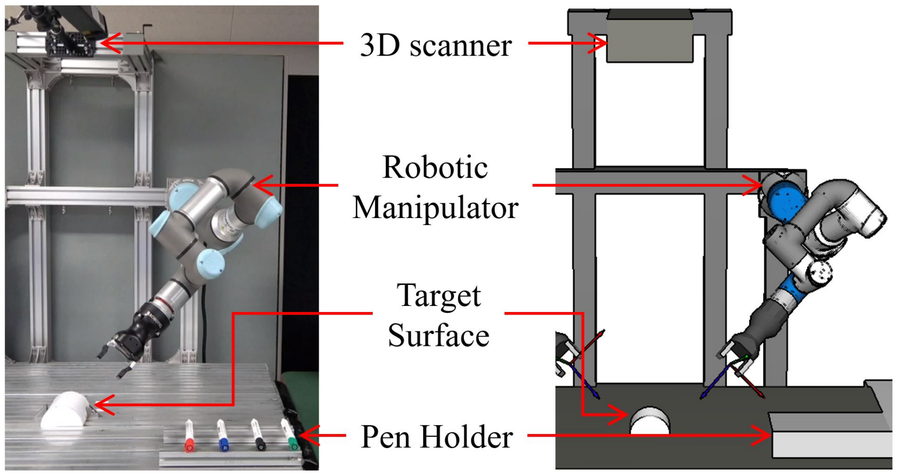


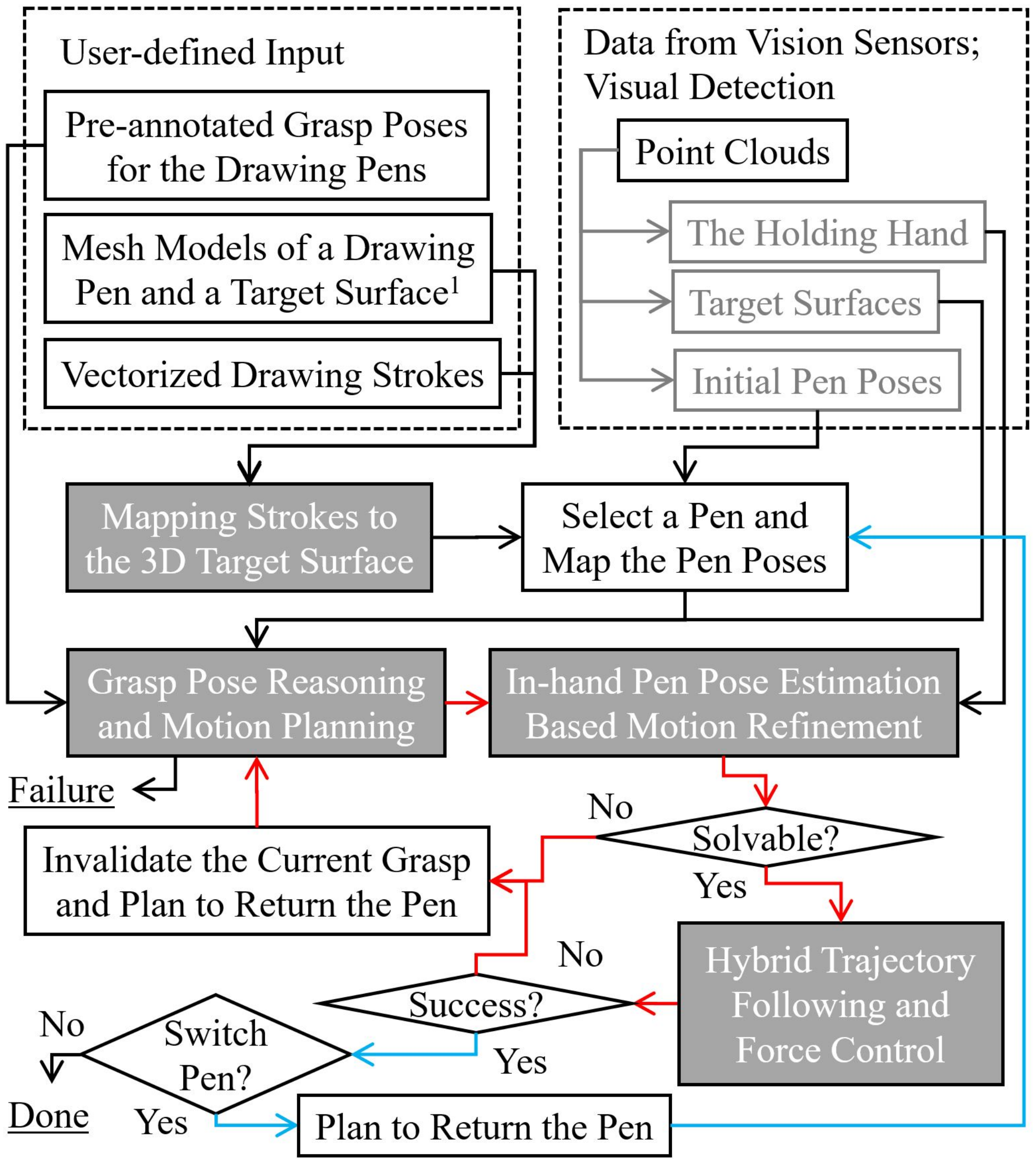
Figure 3.2: One exemplary hardware setup for robotic 3D drawing.

be meshes or meshes rebuilt from point clouds [105][106]. We will implement and compare all these methods. The details will be in Section VI.A.

3.3 Overview of Workflow

This section presents an overview of the methods and workflow to give readers an intuitive conception. First, we show one hardware setup and its corresponding planning interface in Fig. 3.2. Although the proposed method is not limited to the setup, we foreground it to provide a solid impression. In the setup, a robotic manipulator with a general 2-finger gripper is prepared to manipulate pens and draw graphs. A 3D depth sensor is installed on top of the workspace to detect pens and target surfaces that are placed randomly (with adequate clearance) on the table. A diagram of our workflow concerning the shown setup is presented in Fig. 3.3. The workflow starts from two inputs denoted by the dashed boxes on the top. The first input is from users. It includes the pre-annotated or pre-planned grasp poses[1] for the drawing pens, mesh models of the pens, and vectorized drawing strokes on a 2D surface. The second input is the point clouds obtained using a 3D vision sensor. The point clouds

[1] A set of grasp poses (including hand positions, rotations, and jaw-opening distances) defined or planned in advance [107].


User-defined Input
Pre-annotated Grasp Poses for the Drawing Pens
Mesh Models of a Drawing Pen and a Target Surface[1]
Vectorized Drawing Strokes
Data from Vision Sensors; Visual Detection
Point Clouds
The Holding Hand
Target Surfaces
Initial Pen Poses
Mapping Strokes to the 3D Target Surface
Select a Pen and Map the Pen Poses
Grasp Pose Reasoning and Motion Planning
In-hand Pen Pose Estimation Based Motion Refinement
Failure
No
Solvable?
Yes
Invalidate the Current Grasp and Plan to Return the Pen
No
Hybrid Trajectory Following and Force Control
Success?
Yes
No
Switch Pen?
Done
Yes
Plan to Return the Pen

[1] Point clouds will be used directly when mesh models are not available.

Figure 3.3: Workflow of the robotic 3D drawing system. The two dashed boxes are the input. The four gray boxes are the essential algorithms. The red arrows highlight the closed loop. The blue arrows highlight the switches to a different pen.

are divided into two areas where the system finds the initial pen poses in the first area and finds the target surface to draw the mapped 3D strokes in the second.

The four gray boxes in the lower part of the workflow indicate the essential algorithms. The “Mapping Strokes to the 3D Target Surface” box accepts mesh models and vectorized 2D strokes from the first input, and maps the vectorized 2D strokes to the mesh model[2]. Together with the initial pen poses detected from the second input, the workflow will produce a sequence of kinematic pen poses and send them to the “Grasp Pose Reasoning and Motion Planning” gray box to determine grasp poses and plan manipulation motion. The box will iterate through the pre-annotated grasp poses to find the candidates that can finish all the motion, including the motion to pick up a pen, move a pen to a visible position for in-hand pose estimation, and draw the mapped strokes. The drawing motion will be regulated by the “In-hand Pen Pose Estimation and Re-planning” box to determine if there is an in-hand error and will be refined or re-planned according to the errors. The “Hybrid Trajectory Following and Force Control” box ensures a firm contact between the pen tip and the target surface. It will check the forces and trigger recovery when anomalies happen.

The planning process has a closed loop, as illustrated by the red arrows in the diagram. If a refined pose or re-planning is not solvable, the system will invalidate the current grasp pose, iterate to the next grasp pose in the reasoned candidate set, and repeat the reasoning and planning. The iteration will be repeated until a solution is found or a failure is reported. Also, during execution, the robot leverages force control to make sure a stroke is firmly drawn on the target surface. It uses F/T sensors installed at a robot wrist to monitor slippage and anomaly. If slippage is detected, the system will trigger a recovery process. In case of an unrecoverable anomaly, the workflow will go back to the “In-hand Pen Pose Estimation” box to re-estimate the pen’s pose and re-generate the motion by continuing from the failed point. The details of the four essential algorithms and the closed-loop planning will be presented and analyzed in the remaining sections.

[2]Mesh models will be rebuilt from point clouds if they were not available.

3.4 Mapping Strokes to 3D Surfaces

The goal of "Mapping Strokes to the 3D Target Surface" is to map a sequence of vectorized drawing strokes (can be considered as points) in $\mathbb{R}^2$ onto the target surface in $\mathbb{R}^3$ while preserving the geometric relation between the points. We expect an ideal mapping method to comprise the following properties: (1) The method is applicable to both raw point clouds and Computer-Aided Design (CAD) software-generated mesh models; (2) The method has minimal deformation; (3) The method returns smooth drawing strokes that are easy to follow by robotic manipulators. Following this consideration, we propose both metrological methods and conformal mapping-based methods in the following two subsections.

3.4.1 Metrological Methods

Local geometry estimation

To better explain the metrological methods, we first present an intuitive projective mapping method as shown in Fig. 3.4(a). In this case, a point $\boldsymbol{c}_{i+1}$ in the 2D stroke is directly mapped onto a target surface by projecting along the given projection direction $\overrightarrow{\boldsymbol{c}_i\boldsymbol{s}_i}$. The projection does not consider surface normal and its deformation changes with the target surface's curvature. For example, $||\widehat{\boldsymbol{s}_0\boldsymbol{s}_1}||$ is much larger than $||\overrightarrow{\boldsymbol{c}_0\boldsymbol{c}_1}||$ in the figure and the difference changes along the curved surface. The projection will fail when the angle between the surface normal and the projection direction is larger than 90°.

We revise the intuitive projective mapping method by considering the cross product of the target surface normals and the vectors (moving directions) of the strokes and propose the metrological methods. Fig. 3.4(b) illustrates our idea. The mapping of the point $\boldsymbol{c}_1$ is considered as a projection of $\overrightarrow{\boldsymbol{c}_0\boldsymbol{c}_1}$ to a local geometry at $\boldsymbol{s}_0$. We use a tangential plane as an exemplary local geometry for explanation in the figure and also the following context. The blue dash line in the figure illustrates the tangential plane. The dashed red arrows at $\boldsymbol{c}_0$ and $\boldsymbol{s}_0$ in the figure show the vertical direction

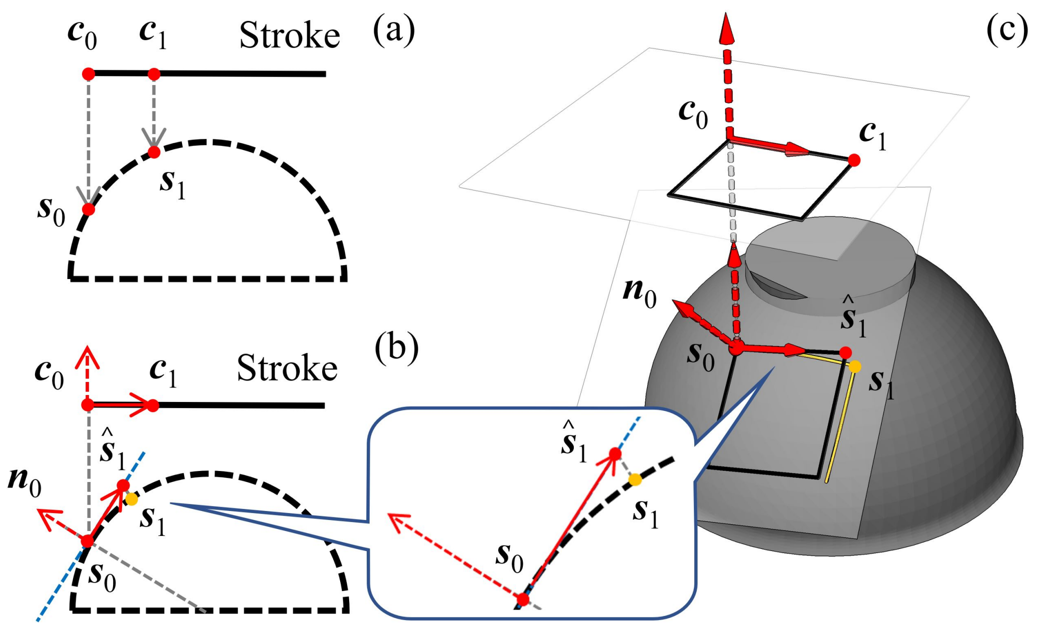


Figure 3.4: Projecting a stroke to a curved surface. (a) Intuitive mapping method. (b) Proposed method. (c) 3D illustration.

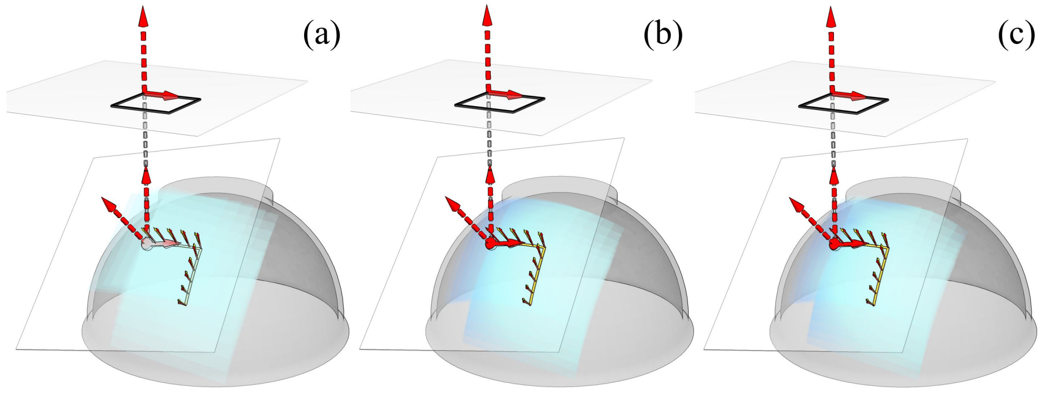


Figure 3.5: Various local geometries for the EI method. (a) Planes. (b) Quadratic surfaces. (c) B-spline surfaces.

```
Algorithm 1: Projecting a 2D stroke to a 3D surface
  Input: $\mathcal{M}$, the target surface
         $\mathcal{C} = \{\boldsymbol{c}_i\}$, a set of points in a stroke in $\mathbb{R}^2$
         $\boldsymbol{s}_0 \in \mathcal{M}$, a start point and its normal on $\mathcal{M}$
  Output: $\mathcal{S} = \{\boldsymbol{s}_i\}$, a set of points on $\mathcal{M}$
1 begin
2     Scale $\mathcal{C}$ to fit the 2D work-space of $\mathcal{M}$
3     Convert $\mathcal{C}$ to $\mathbb{R}^3$ by adding a $z$ value with 0
4     $\mathcal{S} \leftarrow \{\boldsymbol{s}_0\}$
5     for $i \in$ (1,2,...,n) do
6         $\mathbf{S} \leftarrow \texttt{sample\_nearby\_points}(\boldsymbol{s}_i)$
7         $\boldsymbol{n}_i \leftarrow \texttt{local\_plane}(\mathbf{S})$
8         $\boldsymbol{R} \leftarrow \texttt{rot}(\overrightarrow{\boldsymbol{c}_i\boldsymbol{s}_i} \times \boldsymbol{n}_i, \texttt{acos}(\frac{\overrightarrow{\boldsymbol{c}_i\boldsymbol{s}_i} \cdot \boldsymbol{n}_i}{||\overrightarrow{\boldsymbol{c}_i\boldsymbol{s}_i}|| \cdot ||\boldsymbol{n}_i||}))$
9         $\hat{\boldsymbol{s}}_{i+1} \leftarrow \boldsymbol{s}_i + \boldsymbol{R} \cdot \overrightarrow{\boldsymbol{c}_i\boldsymbol{c}_{i+1}}$
10        $\boldsymbol{s}_{i+1}, \boldsymbol{n}_{i+1} \leftarrow$ Nearest point to $\hat{\boldsymbol{s}}_{i+1}$ on $\mathcal{M}$
11        $\mathcal{S} \leftarrow \mathcal{S} \cup \boldsymbol{s}_i$
12    return $\mathcal{S}$
```

perpendicular to a 2D stroke and the normal $\boldsymbol{n}_0$ of the tangential surface at $\boldsymbol{s}_0$, respectively. The solid red arrows between $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, and $\boldsymbol{s}_0$ and $\hat{\boldsymbol{s}}_1$, show the vectors before and after projection. The final projected point is $\boldsymbol{s}_1$. It is a snapped point of $\hat{\boldsymbol{s}}_1$ to the mesh surface, as shown by the zoom box in Fig. 3.4(b). The projection and snapping together maintain the metrology – The length of $\overrightarrow{\boldsymbol{c}_0\boldsymbol{c}_1}$ is the same as $\overrightarrow{\boldsymbol{s}_0\hat{\boldsymbol{s}}_1}$. It is approximated by $\widehat{\boldsymbol{s}_0\boldsymbol{s}_1}$ to minimize distortion. The relations between $\boldsymbol{c}_0$, $\boldsymbol{c}_1$, $\boldsymbol{s}_0$, $\hat{\boldsymbol{s}}_1$, and $\boldsymbol{s}_1$ in the 3D space are illustrated in Fig. 3.4(c). We compute a rotation matrix that implies a minimum geodesic distance between the given projection direction and the normal direction $\boldsymbol{n}_0$. The value of $\hat{\boldsymbol{s}}_1$ is obtained by transforming $\overrightarrow{\boldsymbol{c}_0\boldsymbol{c}_1}$ onto the tangential surface using the rotation matrix and extending $\boldsymbol{s}_0$ along the transformed direction.

The above process is formulated as the pseudo-code shown in Algorithm 1. The algorithm's input includes the target surface, a set of points in a stroke, and a starting point on the target surface. Line 8 of the algorithm carries out the estimation for a local geometry. It finds a set of nearby points to $\boldsymbol{s}_i$ from $\mathcal{M}$, and estimates a local plane for extending $\boldsymbol{s}_i$ to $\hat{\boldsymbol{s}}_{i+1}$. The extended $\hat{\boldsymbol{s}}_{i+1}$ is not necessarily on $\mathcal{M}$. Line 11 of the algorithm resolves the disparity by snapping $\hat{\boldsymbol{s}}_{i+1}$ back to a nearest point on $\mathcal{M}$. Since the algorithm iteratively estimates a local geometry, we call it the Estimation Implementation of the metrological method (EI). Depending on the local geometry types, we further divide the implementations into EI-P (local plane), EI-Q (local quadratic surface), and EI-B (local b-spline surface). Fig. 3.5 illustrates some working examples of them. The performance of these varied implementations is examined and compared later in the experimental section. Note that algorithm 1 is the routine for a single stroke. When there are multiple strokes, a linear interpolation may be applied between the endpoint of a previous stroke to the start point of its next one. The same algorithm finds the projected position of the next stroke's start point. The points on the interpolated segment will be overlooked during drawing.

Global surfaces

For the methods mentioned above, we have to iteratively estimate local geometries to find the next $\boldsymbol{s}_{i+1}$. The iteration is time-consuming and may lead to accumulated errors. Compared to it, a global parametric surface could help to avoid local iterations and be more efficient. Thus, we develop and study global methods in this part. The methods find global surfaces and allow us to carry out differential computations on the surface. After obtaining the global surfaces, our mapping algorithm will find the next drawing points by directly extending along them.

Specifically, we propose two kinds of global surfaces. The first one follows Carr's work [108] and is a parametric one rebuilt from a point cloud using Radial Basis Functions (RBF). Fig. 3.23(b) exemplifies this surface. The second one is a numerical one rebuilt using ball pivoting [109]. It does not have an explicit parametric form. Fig. 3.23(c) exemplifies the second surface. Like the local methods, the two surfaces and the mapping methods based on them (G-RBF and G-BP) will also be examined and compared in the experimental section.

3.4.2 Conformal Mapping-Based Methods

The conformal mapping-based methods consider the projection problem inversely. Instead of projecting 2D strokes onto a 3D surface, the conformal mapping-based methods unfold the 3D surface onto a 2D plane, find the correspondence between 2D strokes and the unfolded surface, and pack the correspondence back into 3D space. The conformal mapping transforms a 3D graph into a 2D one while minimizing angular deformation (not necessarily lengths). In this work, we use LSCM [100] to map 3D vertices of a surface to 2D positions.

The workflow of our conformal mapping-based methods is as follows. First, we unfold mesh vertices or the point clouds to a 2D plane using LSCM. Then, we use a given starting position to find the correspondence between a 2D stroke and the unfolded 2D positions. Third, we pack the found correspondences back to 3D to obtain a 3D stroke. The "find the correspondence between a 2D stroke and the unfolded 2D positions" step is critical throughout the process. We propose two implementations to find the correspondence. In the first implementation, we find the nearest unfolded 2D position to a point in a 2D stroke and only pack back this single point. We call it the Single-point Implementation of the conformal mapping-based method (SI). The implementation could be problematic as the uneven unfolding or the point clouds' noisy points may induce errors and lead to non-smooth strokes. In the second implementation, we sample the target surface to find the k-nearest points of a point in a 2D stroke from the unfolded 2D positions, pack them back to 3D samples on the 3D surface, and perform interpolation to find the target point. This implementation is similar to the one introduced in [101]. We call this implementation the Interpolation Implementation of the conformal mapping-based method (II). The performance of these two implementations will also be examined later.

After examining all the aforementioned methods, including the metrological ones like EI-P, EI-Q, EI-B, G-RBF, and G-BP, and the conformal mapping-based ones like SI and II, we will use the most satisfying one to generate 3D strokes for real-world robotic executions and other evaluations.

3.5 Reasoning and Closed-Loop Planning

This section presents the remaining three essential algorithms in the workflow – the "Grasp Pose Reasoning and Motion Planning", the "In-hand Pose Estimation", and the "Hybrid Trajectory Following and Force Control". It comprises three subsections. The first subsection presents the methods used to detect the object poses. The remaining two subsections explain the details of the three essential algorithms.

3.5.1 Detecting the Object Poses

A depth sensor is installed on top of the workspace to capture a point cloud. The algorithm will segment the point cloud and estimate the poses of the surfaces and pens from it by matching the segments with their mesh models (if available). The segmentation and estimation could be performed easily using conventional algorithms like Density-Based Spatial Clustering of Applications with Noise (DBSCAN) based segmentation [110], RANdom SAmple Consensus (RANSAC) based global search [111], and Iterative Closest Point (ICP) based local refinement [112].

An important improvement we made to these conventional algorithms is that we sample the models unevenly considering the depth sensor's viewpoint. The old routine to perform the matching was: (1) Generating a partial view of the model; (2) Sampling the partial view evenly to generate a template and compute the features; (3) RANSAC; (4) ICP. We revise step (2) to improve the precision of matching - Instead of sampling evenly, we leverage a changing sampling density considering the angles between the normals of the mesh surface and the viewing vector of the depth sensor. The sampling density is formulated as a function of the angle as follows:

$$\rho = \frac{1}{1+\exp(\frac{\pi}{2}-\theta)} - 0.5 \tag{3.1}$$

where ρ is the symbol used to denote the density of surface sampling, and θ is the angle between the normal of a mesh triangle and the depth sensor's viewing vector. Influenced by this equation, the triangles that face the camera will be sampled with a higher density. The side triangles will be less sampled. The back triangles will be ignored. The uneven samples increase the fitness of RANSAC and ICP, thus improves the precision of pose estimation. Fig. 3.6 exemplifies a captured point cloud, the template created using uneven sampling, the segmented clusters, and the estimated surface pose and pen poses using the methods mentioned above.

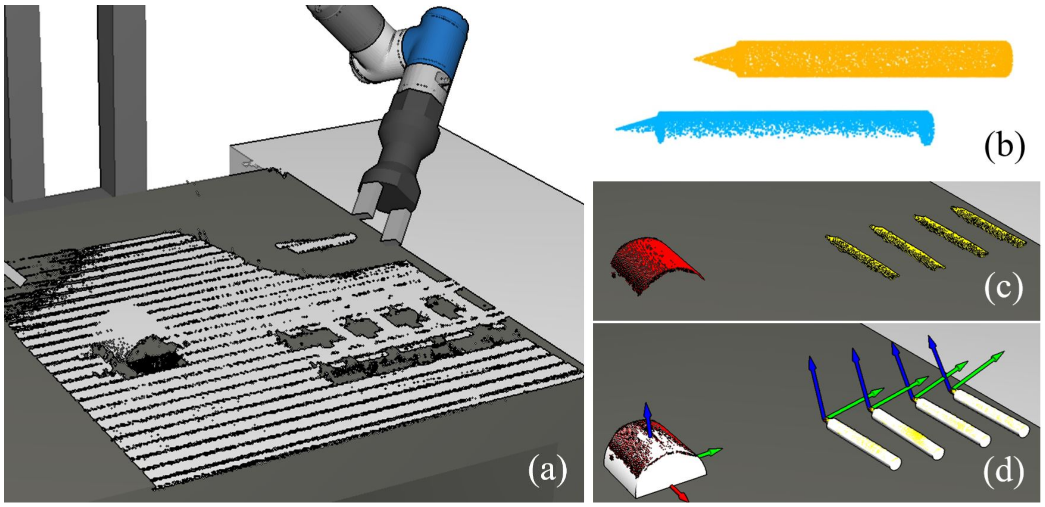


Figure 3.6: (a) A captured point cloud. (b) Generating a partial view template for visual recognition. The surface of the partial view is sampled unevenly following the viewpoint direction to improve recognition performance. Orange: A full pen model. Blue: An unevenly sampled partial view. (c) Background subtraction and clustering. (d) Detected initial target surface pose and pen poses. The cylinder mesh is known and matched.

3.5.2 Grasp Pose Reasoning and Motion Planning

The detailed diagram of the "Grasp Pose Reasoning and Motion Planning" algorithm is shown in Fig. 3.7. It is a direct expansion of the counterpart in Fig. 3.3. The diagram accepts (1) pre-annotated grasp poses for the pens, (2) estimated initial pen poses, (3) a sequence of pen poses generated by attaching the pen tips to the mapped 3D strokes along inversed surface normal directions, as the input. The input will be used in the "IK-Feasible and Collision Free Common Grasps" box to reason the IK-feasible and collision-free common grasps. Here, by "common grasps", we mean the commonly available grasps at all pen poses. Fig. 3.8 shows an example of common

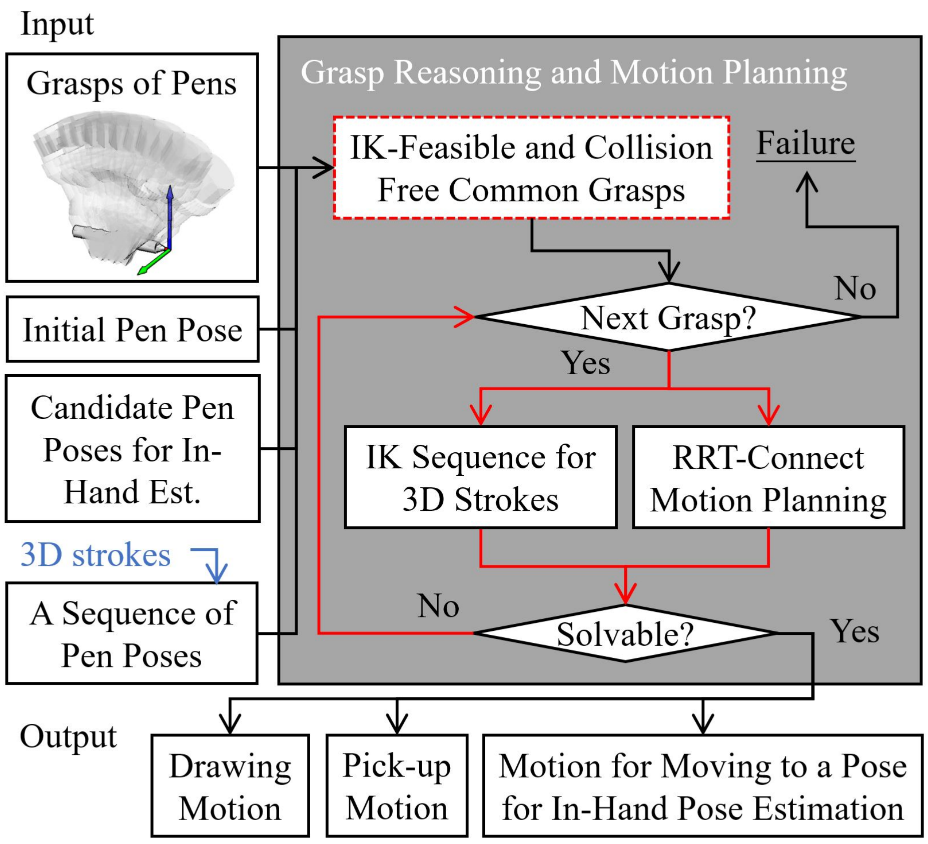

Input
Grasps of Pens
Initial Pen Pose
Candidate Pen Poses for In-Hand Est.
3D strokes
A Sequence of Pen Poses
Grasp Reasoning and Motion Planning
IK-Feasible and Collision Free Common Grasps
Failure
No
Next Grasp?
Yes
IK Sequence for 3D Strokes
RRT-Connect Motion Planning
No
Solvable?
Yes
Output
Drawing Motion
Pick-up Motion
Motion for Moving to a Pose for In-Hand Pose Estimation

Figure 3.7: Detailed diagram of the "Grasp Pose Reasoning and Motion Planning" gray box in Fig. 3.3.

grasps. The gray hand poses in Fig. 3.8 are the pre-annotated grasp candidates. They are transformed to new positions and orientations following the pen poses in Fig. 3.8(b.1) and (b.2). The IK and collisions of the robot at the transformed hand poses are examined, with the IK-feasible and collision-free ones marked in green and the remaining ones marked in red. The green hand poses in Fig. 3.8(c) illustrate the "common grasps". They are identical in the pen's local frame, and are the intersection of the green hand poses in Fig. 3.8(b.1) and (b.2). A robot can hold the pen using a "common grasp" at one pen pose and move it to another without regrasp.

After obtaining the common grasps, the "RRT-Connect Motion Planning" and "IK Sequence for 3D Strokes" boxes will iterate through them to plan the "Pick-up Motion", the "Motion for Moving to a Pose for In-Hand Pose Estimation", and

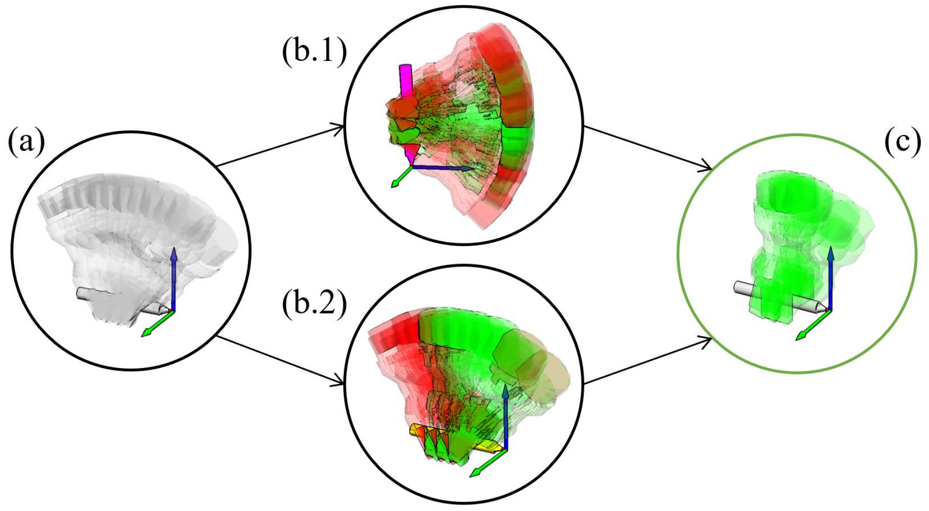


Figure 3.8: Grasp reasoning. (a) Pre-annotated grasps. (b.1,2) Transforming the pre-annotated grasps to two different pen poses. The IK-feasible and collision-free grasps are shown in green. The others are shown in red; (c) IK-feasible and collision-free grasps at all pen poses (common grasps).

the "Drawing Motion". The "RRT-Connect Motion Planning" box plans joint space motion for picking up and moving a pen to a pose for in-hand estimation. The "IK Sequence for 3D Strokes" box generates the drawing motion for the mapped 3D strokes. It examines the IK and collisions again[3] and uses workspace interpolation to connect the IK configurations of two adjacent pen poses and produce a drawing trajectory.

Especially for the "3D Strokes" input, we attach pen tips to each 3D stroke point while requiring that the pen's main axis[4] are aligned with the surface normals and get a sequence of pen poses for drawing each point of a stroke. The "IK Sequence for 3D Strokes" box examines the IKs and collisions of each pen pose. The smoothness or optimization of adjacent IK configurations is considered implicitly in the IK solver by using the result of a previous pen pose as the seed value for numerically solving the following one. The solved configurations may be discontinuous due to the sudden change of surface normals and, consequently, lead to a fragile drawing motion. The

[3]The IK and collisions have been solved wahen finding common grasps.
[4]A pen's main axis is defined as a vector that points from its tip to end.

sudden changes get worse when there is no exact CAD model, and the target surface is a point cloud captured by depth sensors. To avoid discontinuity, we perform quaternion Spherical Linear Interpolation (SLERP) [113] in the "IK Sequence for 3D Strokes" box to interpolate pen poses in-between the directly attached pen pose sequence.[5]

3.5.3 Close the Loop to Improve Robustness

At the beginning of this section, we proposed improving visual estimation performance using uneven sampling. The method exhibits higher precision. However, the errors cannot be completely removed even if we are perfect in detection. There remain errors caused by (1) pen displacements during manipulation, (2) sensor calibration, and (3) numerically modeled CAD meshes. In this subsection, we look into these errors and present our solutions – We use in-hand pen pose estimation to avoid the errors caused by (1) and use force control to eliminate the errors caused by (2) and (3). The in-hand pose estimation and force control will trigger a recovery mechanism to correct the errors. In case of a failure, they will invalidate the current actions and restart the planning.

In-hand pose estimation

We use the same methods as detecting the initial pen poses to perform the in-hand estimation. The workflow is shown in Fig. 3.9. The algorithm extracts the point cloud near the robot gripper[6], estimates the pose of a grasped pen from the point cloud, and compares it to the expected value in the simulation to measure errors. In case of a large error, the algorithm will refine or re-plan the drawing motion to avoid failures.

[5]The method is fast but finds fewer solutions since constraining the pen's main axis to the normal directions was a very strong condition. The FK-based optimization planner introduced in the appendix of this manuscript could be a better choice if you are interested in a more flexible alternative.

[6]We compute the pose of the gripper using joint encoder values and forward kinematics. The point cloud near the gripper can be obtained by cropping the data around the computed gripper pose.

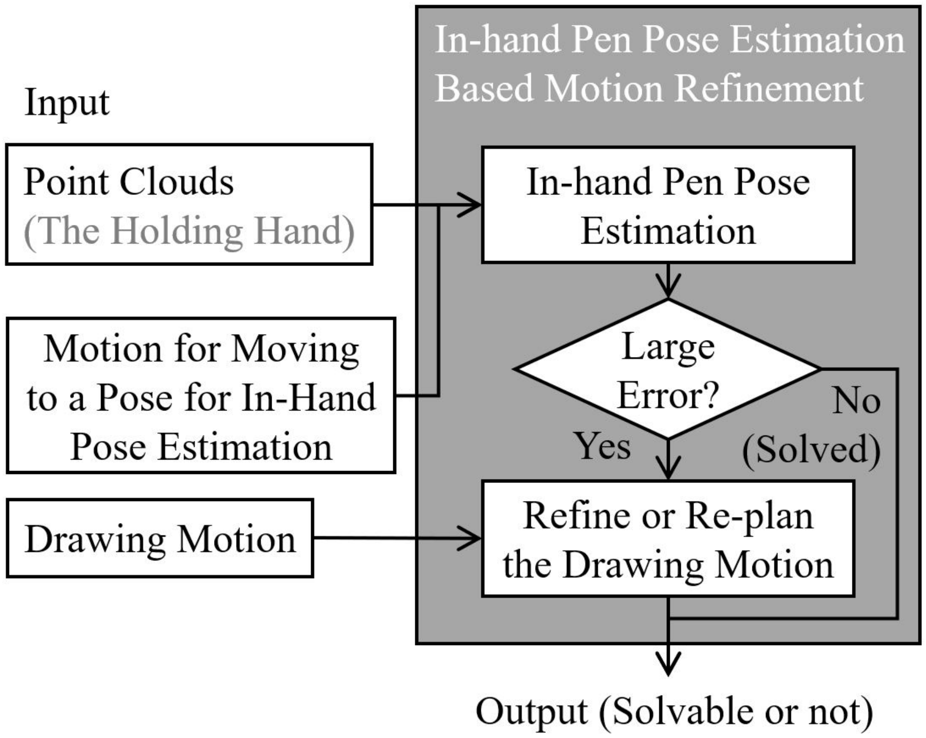

In-hand Pen Pose Estimation Based Motion Refinement
Input
Point Clouds (The Holding Hand)
Motion for Moving to a Pose for In-Hand Pose Estimation
Drawing Motion
In-hand Pen Pose Estimation
Large Error?
Yes
No (Solved)
Refine or Re-plan the Drawing Motion
Output (Solvable or not)

Figure 3.9: Detailed diagram of the "In-hand Pose Estimation Based Motion Refinement" gray box in Fig. 3.3.

Fig. 3.10 shows an error recovery example. Fig. 3.10(a) and (b) are the robot configurations for in-hand estimation in real execution and simulation respectively. Fig. 3.10(c.1) shows the error between the captured point cloud of the pen and the pen pose in the simulation. The point cloud is rendered in green. The pen pose in the simulation is rendered in yellow. Fig. 3.10(c.2) shows the matched pen pose (rendered in red. The in-hand estimation algorithm corrects the pen pose from the yellow one to the red one following

$$^{Real}_{Hand}\mathrm{T} = (^{World}_{Real}\mathrm{T})^{-1} \times {}^{World}_{Sim}\mathrm{T} \times {}^{Sim}_{Hand}\mathrm{T}, \tag{3.2}$$

where $^{Real}_{Hand}\mathrm{T}$ is the refined in-hand pen pose and $^{Sim}_{Hand}\mathrm{T}$ is the expected value in simulation.

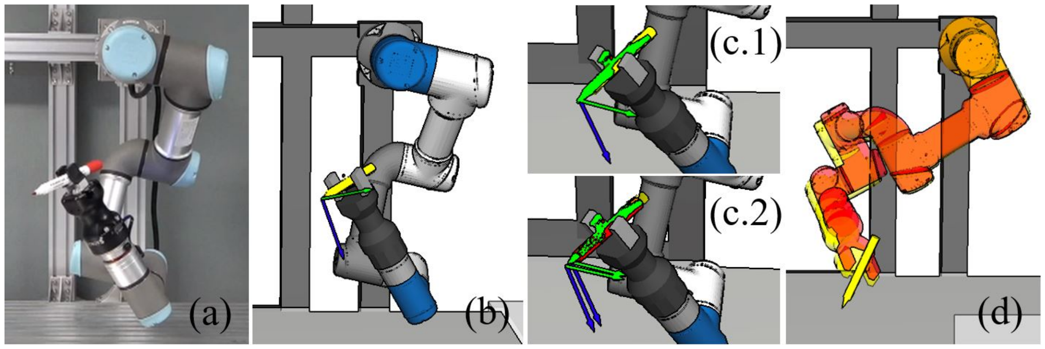


Figure 3.10: (a) A grasp pose in real execution. (b) The correspondent ideal grasp pose in simulation. (c.1) The error between the captured point cloud of the pen and the ideal pen pose. The point cloud is shown in green. The ideal pen pose is shown in red. (c.2) Comparison of the matched in-hand pose (red) to the ideal pose (yellow). (d) An example of a corrected drawing pose. The yellow robot configuration is corrected to the red configuration by the estimated in-hand pose.

Fig. 3.10(d) compares the robot drawing configurations before and after correction. The yellow configuration is the one planned in the simulation, while the red configuration is the corrected result. Note that the red configuration in Fig. 3.10(d) is not necessarily solvable. When there is no solution, the planner will invalidate the current grasp pose and trigger a new "Grasp Pose Reasoning and Motion Planning" routine, as shown by the red arrows in Fig. 3.3. The correction, invalidation, re-reasoning, and re-planning close up the planning loop, thus improve robustness and planning success rate.

Hybrid Trajectory Following and Force Control

Force Control We consider the force feedback measured from an F/T sensor installed at the wrist of a manipulator and use force control along the pen's main axis to compensate for sensor calibration errors and uncertainty in CAD models. Essentially, the force control method builds a connection between the force and motion of the pen so that the robot can attach the pen to the target surface with firm contact. The force control is performed by considering a decomposed force along the pen's main axis and thus provides compliance with surface curvature. The pen will follow the planned motion path while maintaining a steady contact at the tip. Fig. 3.11 shows the workflow of the force control method.

Algorithm 2: Error Detection and Recovery

```
Input: ${\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_n}$, a joint motion trajectory found by our planner;
begin
    for each $\boldsymbol{q}_i \in \{\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_n\}$ do
        $\boldsymbol{P}_i, \boldsymbol{R}_i \leftarrow \texttt{fk}(\boldsymbol{q}_i)$
        Move the robot to $\boldsymbol{q}_i$
        Collect force and torque in the pen frame $\boldsymbol{F}_i$
        if lost($\boldsymbol{F}_i$) then
            for each $\boldsymbol{q}_j \in \{\boldsymbol{q}_{i+1}, \boldsymbol{q}_{i+2}, ..., \boldsymbol{q}_n\}$ do
                $\boldsymbol{P}_j, \boldsymbol{R}_j \leftarrow \texttt{fk}(\boldsymbol{q}_j)$
                $\sigma \leftarrow \texttt{acos}(\frac{\boldsymbol{R}_i(x) \cdot \boldsymbol{R}_j(x)}{||\boldsymbol{R}_i(x)|| \cdot ||\boldsymbol{R}_j(x)||})$
                if $\sigma > threshold$ then
                    update($\{\boldsymbol{q}_{i+1}, \boldsymbol{q}_{i+2}, ...\}, \boldsymbol{P}_j, \boldsymbol{R}_j$)
                    break
        else if inhnd_slip($\boldsymbol{F}_i$) or blocked($\boldsymbol{F}_i$) then
            $forward\_search \leftarrow$ True
            for each $\boldsymbol{q}_j \in \{\boldsymbol{q}_{i-1}, \boldsymbol{q}_{i-2}, ...\boldsymbol{q}_{i-m}\}$ do
                $\boldsymbol{P}_j, \boldsymbol{R}_j \leftarrow \texttt{fk}(\boldsymbol{q}_j)$
                $\sigma \leftarrow \texttt{acos}(\frac{\boldsymbol{F}_i(x) \cdot \boldsymbol{R}_j(x)}{||\boldsymbol{F}_i(x)|| \cdot ||\boldsymbol{R}_j(x)||})$
                if $\sigma < threshold$ then
                    update($\{\boldsymbol{q}_{i+1}, \boldsymbol{q}_{i+2}, ...\}, \boldsymbol{P}_j, \boldsymbol{R}_j$)
                    $forward\_search \leftarrow$ False
                    break
            if $forward\_search$ is True then
                for each $\boldsymbol{q}_j \in \{\boldsymbol{q}_{i+1}, \boldsymbol{q}_{i+2}, ...\boldsymbol{q}_n\}$ do
                    $\boldsymbol{P}_j, \boldsymbol{R}_j \leftarrow \texttt{fk}(\boldsymbol{q}_j)$
                    $\sigma \leftarrow \texttt{acos}(\frac{\boldsymbol{F}_i(x) \cdot \boldsymbol{R}_j(x)}{||\boldsymbol{F}_i(x)|| \cdot ||\boldsymbol{R}_j(x)||})$
                    if $\sigma < threshold$ then
                        update($\{\boldsymbol{q}_{i+1}, \boldsymbol{q}_{i+2}, ...\}, \boldsymbol{P}_j, \boldsymbol{R}_j$)
                        break
    return
```

Error Detection and Recovery Considering Force Deviation Pen-surface slippage and pen-hand contact slippage may happen during drawing. The slippage has two causes. The first one is the material properties of the target surface. The slippage caused by it is complicated (a mixed pen-surface slippage and pen-hand slippage) and is difficult to compensate. We have no clever tricks to overcome these slippage and suggest preventing them by adding a strong constraint to the planners. The second cause of slippage is positioning errors. In this part, we focus on the slippage caused by it and develop an algorithm to predict and recover from these slippage.

Specifically, we analyze and categorize the slippage caused by positioning errors by monitoring the force errors measured by the F/T sensor during force control. The slippage includes: (1) An in-hand slippage. When there is a large torque around the hand opening direction, the pen will slip inside the hand and exhibit an in-hand slippage. The slippage could be caused by an uncertain surface that blocks the drawing motion and pushes the pen to slide in the hand. (2) A lost surface. When the force along the pen's main axis has a sudden drop, the surface is lost. (3) A blocked surface. When the torque or force in the directions other than the hand opening direction and the pen's main axis has a significant increase, the pen is blocked by a surface. In the latter two cases, the surface might be in a wrong position and the pen either loses contact with or runs into it. Following these analysis and categorizations, we detect slippage by examining the changes of forces and torques measured by the F/T sensor installed at the wrist of a robot manipulator and propose Algorithm 2 to correct or avoid slippage. The algorithm includes two main sections. The magenta section deals with a lost surface slippage. In this case, the algorithm searches forward to the following drawing points until the pen orientation is significantly changed (larger than the *threshold* in line 10). The blue section deals with an in-hand slippage or a blocked surface. The algorithm will first look backward to the drawn points a bit (the number of points is specified by the m in line 15) to see if the drawing configuration at a previous drawing point could compensate for the force along the pen's main axis. The wheat-color block of the blue section shows the backward search. The algorithm will resume the previous point and repeat the drawn stroke if a previous drawing configuration provides satisfying force compensation (smaller than the *threshold* in line 18). It will search forward like the magenta section if the previous drawing configurations cannot afford satisfying compensation. The part after the wheat-color block in the blue section shows the forward search. The algorithm will jump to a future point when it judges that a future drawing configuration can provide reasonable compensation (lines 26-27).

3.6 Experiments and Analysis

This section presents the experiments performed to analyze the robustness and flexibility of the developed robotic 3D drawing system. It compares both simulations and real-world results to verify the proposed methods. The simulation platform used in our experiments is a PC with an Intel Core i5-8250U CPU and 32 GB memory. The programming language is Python 3, and the software environment is WRS[7]. A Photoneo PhoXi 3D Scanner M with 1032×772 resolution is used for visual perception. The robot used is the one shown in Fig. 3.2, which includes a UR3e robotic arm equipped with a Robotiq Hand-E two-finger parallel gripper.

3.6.1 Robustness of 2D-to-3D Stroke Mapping

First, we evaluate the performance of the various stroke mapping algorithms presented in section IV. The target surface includes both mesh models made by CAD software and raw point clouds obtained from the depth sensor. To provide a direct view for observation and comparison, we use a lattice stroke made of a series of grids as the 2D strokes in the experiments. Readers may easily assess the performance of the various mapping algorithms by observing the shapes of mapped results. Besides the visual observation, we define one local indicator and two global indicators to quantitatively measure in-stroke deformation, inter-stroke displacement, and inter-stroke distortion,

[7]WRS is an open-source environment developed in our laboratory. It is available on github at https://github.com/wanweiwei07/wrs

respectively. The local indicator is

$$e_l = \frac{||\widehat{\boldsymbol{s}_i\boldsymbol{s}_{i+1}}|| - ||\overrightarrow{\boldsymbol{c}_i\boldsymbol{c}_{i+1}}||}{||\overrightarrow{\boldsymbol{c}_i\boldsymbol{c}_{i+1}}||}, \tag{3.3}$$

where $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$ are two adjacent points in a stroke in $\mathbb{R}^2$, $\boldsymbol{s}_i$ and $\boldsymbol{s}_{i+1}$ are their corresponding points in a mapped stroke in $\mathbb{R}^3$. The length $||\widehat{\boldsymbol{s}_i\boldsymbol{s}_{i+1}}||$ is the geodesic distance between $\boldsymbol{s}_i$ and $\boldsymbol{s}_{i+1}$ on the target surface. The length $||\overrightarrow{\boldsymbol{c}_i\boldsymbol{c}_{i+1}}||$ is the Euclidean distance between $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$ in $\mathbb{R}^2$. The e_l indicator shows the deformation of a single stroke after mapping while ignoring displacement and distortion between the strokes. The global indicators are

$$e_g = ||\widehat{\boldsymbol{s}_m\boldsymbol{s}_n}|| - ||\overrightarrow{\boldsymbol{c}_m\boldsymbol{c}_n}||, \tag{3.4}$$

and

$$e_\alpha = |\mathtt{acos}(\frac{\boldsymbol{s}'_m \cdot \boldsymbol{s}'_n}{||\boldsymbol{s}'_m|| \cdot ||\boldsymbol{s}'_n||}) - \mathtt{acos}(\frac{\boldsymbol{c}'_m \cdot \boldsymbol{c}'_n}{||\boldsymbol{c}'_m|| \cdot ||\boldsymbol{c}'_n||})|, \tag{3.5}$$

where $\boldsymbol{c}_m$ and $\boldsymbol{c}_n$ are two closest points in two different strokes in $\mathbb{R}^2$, $\boldsymbol{s}_m$ and $\boldsymbol{s}_n$ are their corresponding points in the mapped strokes in $\mathbb{R}^3$. The arc and arrow on top of the symbols in equation (3.4) indicate geodesic and Euclidean distances like equation (3.3). For the lattice strokes, $||\overrightarrow{\boldsymbol{c}_m\boldsymbol{c}_n}||$ equals to 0. The $'$ superscripts on the symbols in equation (3.5) indicate the derivative (tangential direction) at the points. The `acos`() function computes the angle between the tangential directions. The e_g indicator shows the translational displacement between strokes after mapping. The e_α indicator shows the angular distortion between strokes after mappings.

Table 3.2 shows detailed information about the used 3D surfaces and lattice strokes. The table has two sections where the upper section shows the surfaces with known mesh models. The lower section shows the ones without mesh models. In each section, there are seven rows where the first four are the size and mesh information about the surfaces, the fifth and sixth rows are information about the lattice strokes, and the last row host pointers to the mapping results in Fig. 3.12-3.14.

Table 3.2: Information of the 3D Surfaces and Lattice Strokes

	Known Mesh Model		
	Box	Cylinder	Sphere
Volume (mm)	100×100×100	100×100×50	100× 100 ×50
Surface (mm)	80×80	80×80	60×60
# Vertices	6	360	32401
# Faces	4	362	64440
# Strokes	18	18	14
# Points/Stroke	81	81	61
Results	Fig. 3.12-3.14(a)	Fig. 3.12-3.14(b)	Fig. 3.12-3.14(c)
	Raw Point Clouds		
	Cylinder	Helmet	-
Volume (mm)	89×101×31	166×199×72	-
Surface (mm)	80×80	80×80	-
# Vertices	12010	43617	-
# Faces	21407	80159	-
# Strokes	18	18	-
# Points/Stroke	81	81	-
Results	Fig. 3.12-3.14(d)	Fig. 3.12-3.14(e)	

* Meanings of items: Volume - The bounding box dimensions of a mesh model or point clouds; Surface - The 2D dimensions of the target surface for drawing; # Vertices, # Faces - The number of vertices and faces (original or reconstructed) on a target surface; # Stroke - The number of strokes in a lattice to be drawn; # Points/Strokes - The number of points in each stroke.

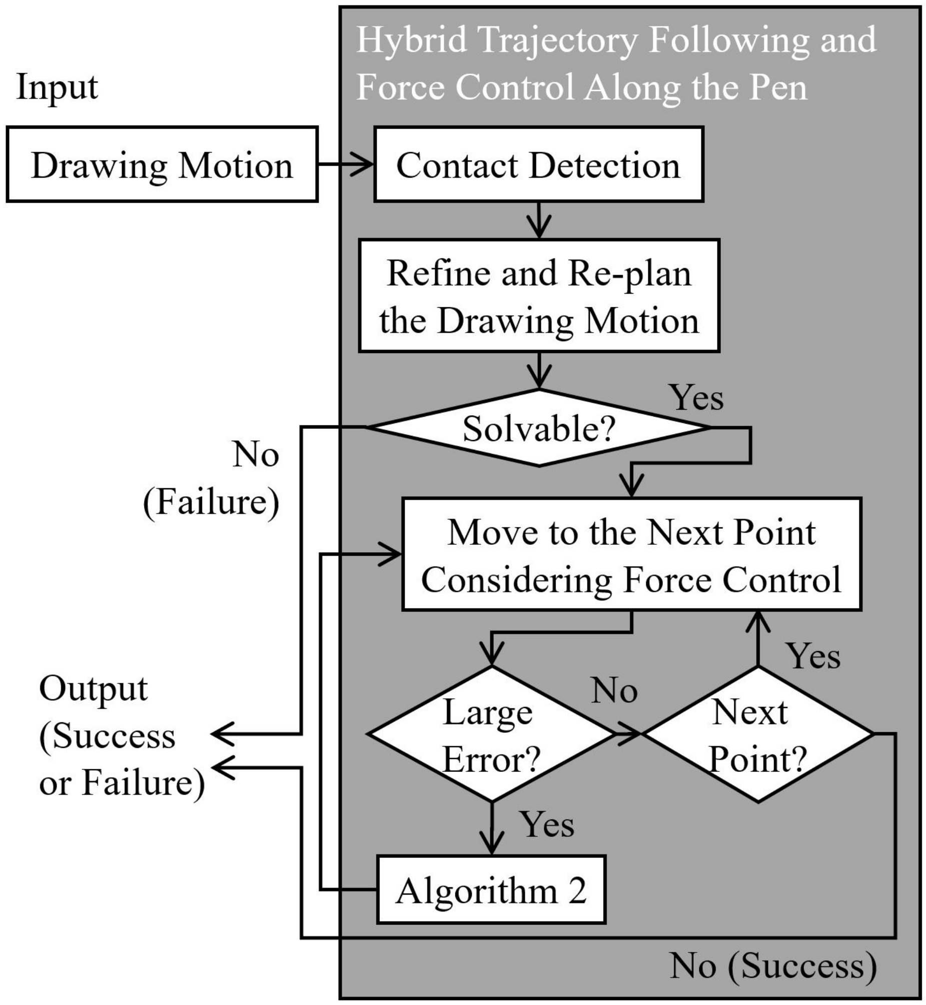

Hybrid Trajectory Following and Force Control Along the Pen
Input
Drawing Motion
Contact Detection
Refine and Re-plan the Drawing Motion
Solvable?
Yes
No (Failure)
Move to the Next Point Considering Force Control
Output (Success or Failure)
Large Error?
No
Next Point?
Yes
Yes
Algorithm 2
No (Success)

Figure 3.11: Detailed diagram of the "Hybrid Trajectory Following and Force Control" gray box in Fig. 3.3. The "Refine and Re-plan the Drawing Motion" box is identical to that in Fig. 3.9.

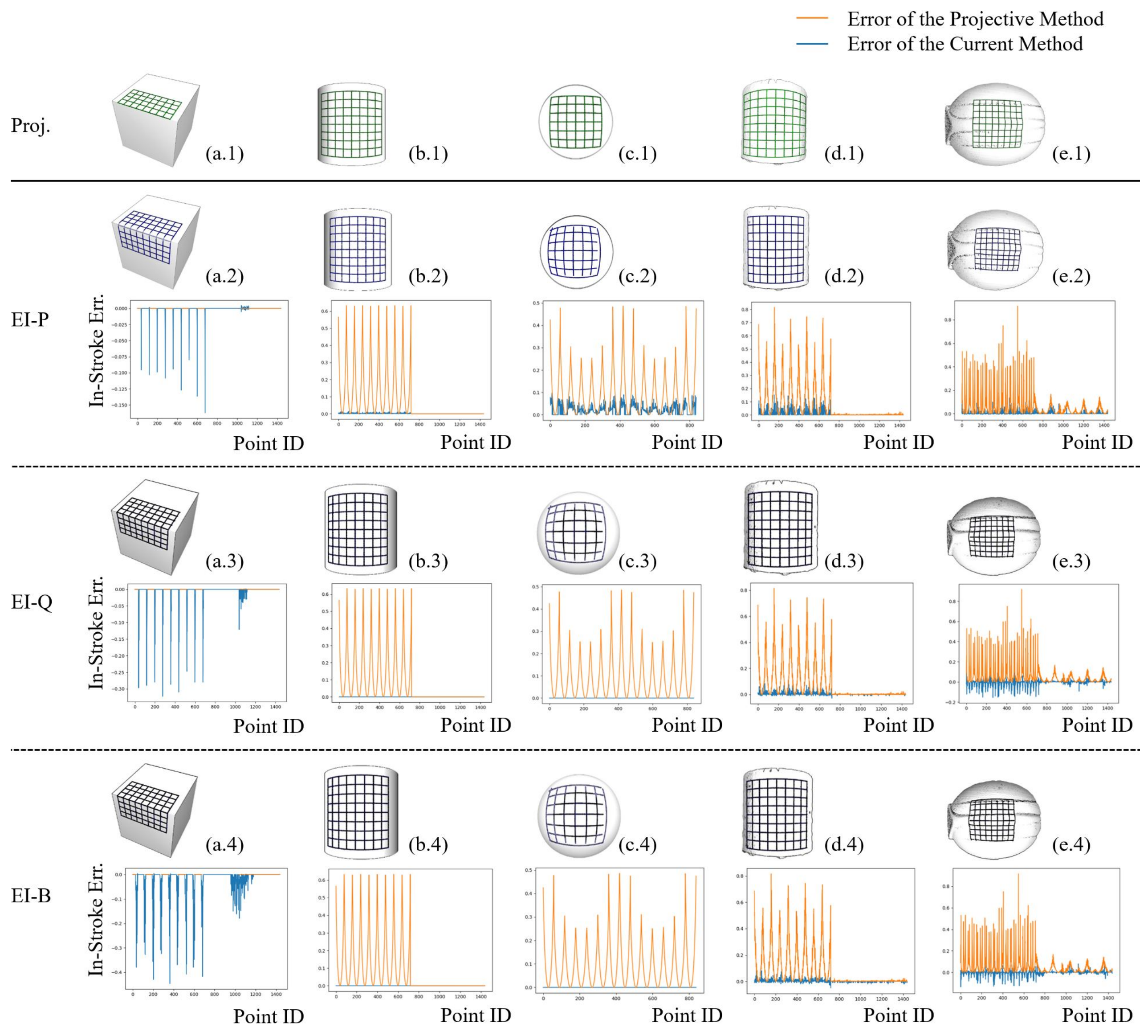


Figure 3.12: Comparison of 2D-to-3D stroke mapping methods (I): Metrological – EI. (a.1-e.1) Results of the projective method (baseline). (a.2-e.2), (a.3-e.3) are EI-P method, EI-Q and EI-B method respectively.

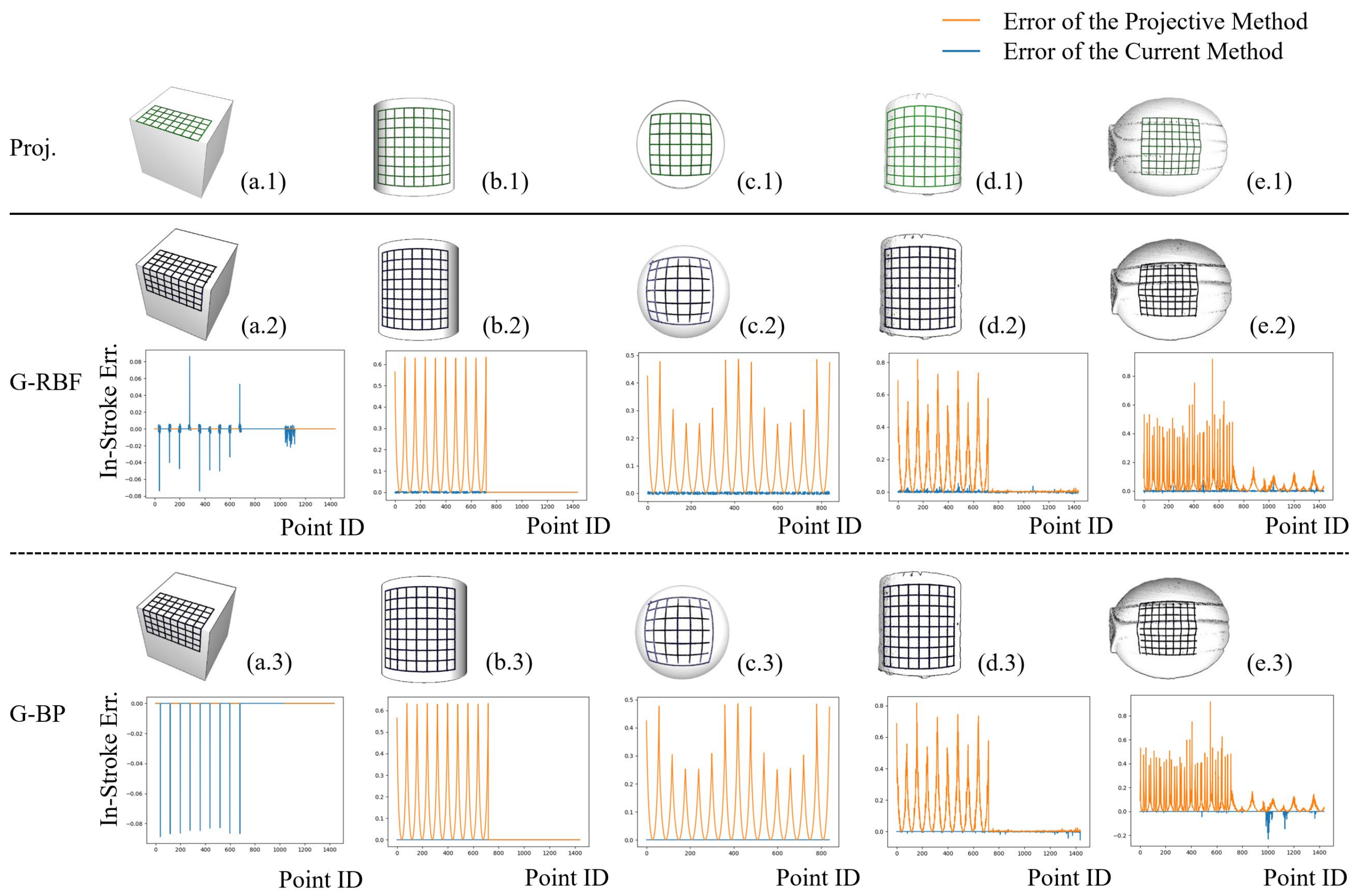


Figure 3.13: Comparison of 2D-to-3D stroke mapping methods (II): Metrological – Global surfaces. The arrangements and various indicators are the same as Fig. 3.12, except that the methods in (a.2-e.2) and (a.3-e.3) are G-RBF and G-BP respectively.

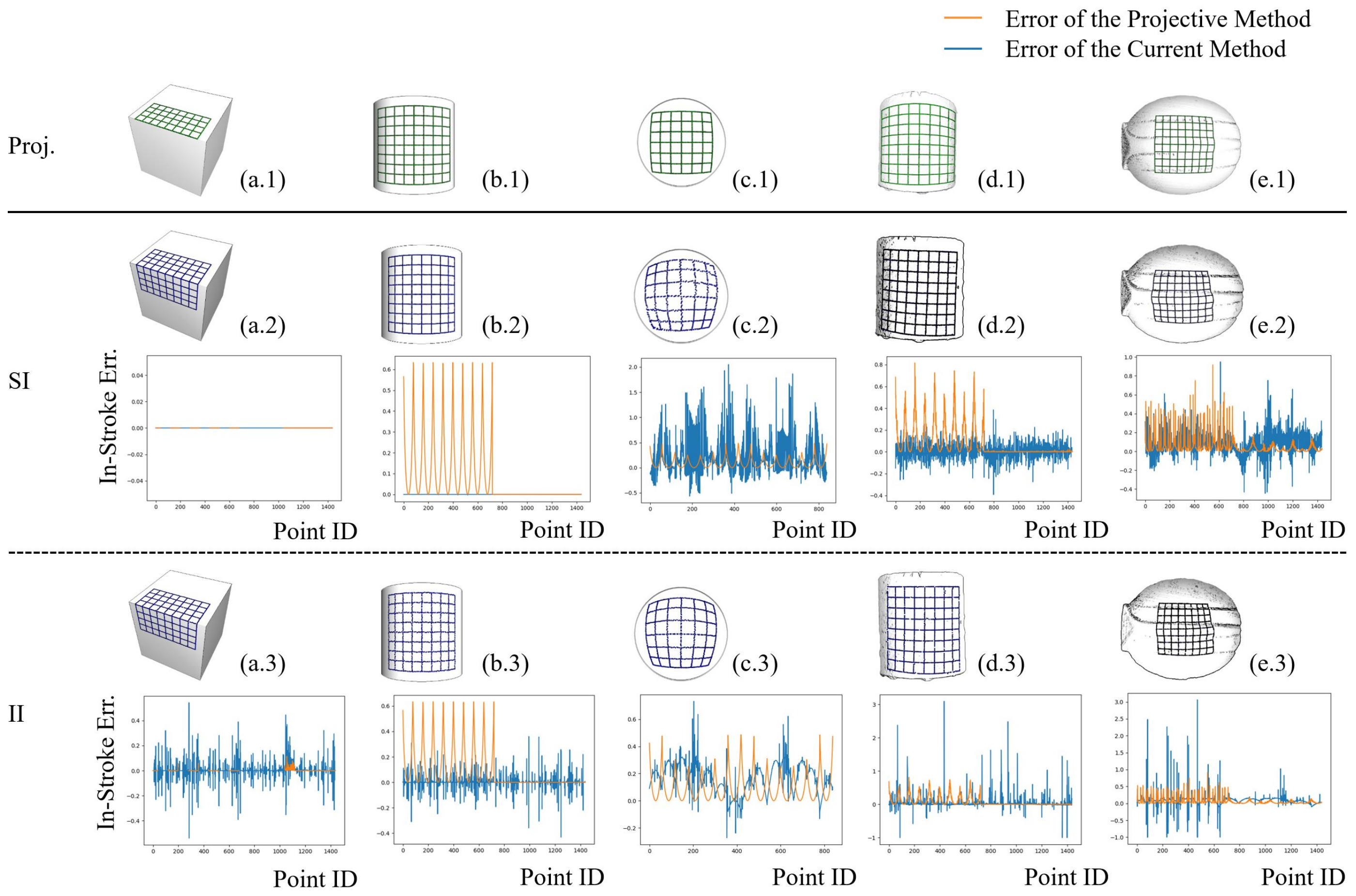


Figure 3.14: Comparison of 2D-to-3D stroke mapping methods (III): Conformal mapping. The arrangements and various indicators are the same as Fig. 3.12 and 3.13, except that the methods in (a.2-e.2) and (a.3-e.3) are SI and II respectively.

Fig. 3.12-3.14 visualize the mapping results and plot the e_l curves using the surfaces and strokes shown in Table 3.2. The first rows of them are the mapping results of the projective method mentioned in Section IV.A. They are considered as the baseline for comparison and are repeated in each figure for readers' convenience. The lower three rows of Fig. 3.12 show the results of the EI-P, EI-Q, and EI-B methods. The lower two rows of Fig. 3.13 show the results of the G-RBF and G-BP methods. The lower two rows of Fig. 3.14 show the results of the SI and II methods. Each of the small subfigures in the lower rows identified by an (alphabet.number) includes two parts where the upper part visualizes the mapped result. The lower part plots the error chart. The blue curves in the error charts show the in-stroke errors of the current method and model. The orange curve shows the in-stroke errors of the baseline method. In all figures, the columns (a-c) are mapping results on surfaces with known mesh models. The columns (d,e) are results with point clouds. One thing to note is that a closed mesh model should be decomposed into segments with shapes homographic to discs to realize the conformal mapping-based methods. We performed manual intervention to simplify the routine and reduce the error caused by the non-uniform parameterization scale between the segments. The mapping is performed on the main segment that we selected. Table 3.3 further shows the detailed time costs and average e_l, e_g, and e_α values of the various mapping methods using the surfaces and strokes shown in Table 3.2. The values with a lime background show the minimum element of each row. The values with a pink background are the maximum element of each row. Since the mesh models of the raw point clouds are not available, we cannot directly compute the geodesic lengths and distances, and the meshes must be reconstructed first. To make the comparison fair, we reconstruct meshes using two methods – One uses the RBF global surface estimation method and the other one uses the ball-pivoting estimation method. Then, we obtain the errors by computing the difference between the geodesic stroke lengths or geodesic stroke inter-distances on both the two reconstructed meshes and the original 2D ones. Depending on the reconstruction method, a prefix is added to the end of the error labels for the point clouds. We do not know which is a better reference as the ground truth, but leave it to readers for evaluation.

Table 3.3: Time Costs and Average Errors of the Mappings in Fig. 3.12, 3.13, and 3.14

			Baseline	Metrological					Conformal	
				EI-P	EI-Q	EI-B	G-RBF	G-BP	SI	II
Known Mesh Models	Box	Time Cost (s)	2.120	28.702	66.019	11.025	42.211	5.543	1.959	48.396
		Avg. e_l	-	0.0013	0.0063	0.0199	0.0006	0.0005	0.0	0.0023
		Avg. e_g (mm)	-	0.1016	0.6168	0.9390	0.7947	0.2224	0.0	0.0
		Avg. e_α (°)	-	0.0480	0.0665	0.1750	0.2862	0.0453	0.0	1.0178
	Cylinder	Time Cost (s)	6.298	28.892	65.364	11.378	46.909	10.533	2.249	186.513
		Avg. e_l	0.0795	0.0010	4.4e-5	4.8e-5	0.0001	0.0	0.0	0.0018
		Avg. e_g (mm)	0.0	0.0026	0.0084	0.0201	0.0547	0.0168	0.0	0.0
		Avg. e_α (°)	0.0	0.0074	0.0087	0.0141	0.0747	0.0150	0.0013	1.3480
	Sphere	Time Cost (s)	532.324	6.846	16.606	14.476	300.777	31.849	144.122	365.971
		Avg. e_l	0.0952	0.0223	8.4e-5	8.4e-5	0.0003	0.0	0.2281	0.1696
		Avg. e_g (mm)	0.0	2.4419	2.3820	2.3829	0.6052	2.3833	0.0	0.0
		Avg. e_α (°)	9.0984	0.5026	0.5039	0.4935	0.4354	0.4941	16.2798	0.5050
Raw Point Clouds	Cylinder	Time Cost (s)	236.751	3.276	14.806	11.854	49.315	20.054	24.282	17.713
		(RBF) Avg. e_l	0.0894	0.0866	0.0027	0.0258	0.0005	0.0036	0.0110	0.1072
		(BP) Avg. e_l	0.0782	0.0085	0.0033	0.0032	0.0011	0.0007	0.0088	0.0557
		Avg. e_g (mm)	0.0	0.2994	0.3626	0.2561	0.3563	0.3057	0.0	0.0
		Avg. e_α (°)	0.7791	0.0286	0.0235	0.0221	1.0725	0.0565	3.7820	1.6169
	Helmet	Time Cost (s)	972.322	10.619	33.763	31.211	379.583	54.186	264.103	59.204
		(RBF) Avg. e_l	0.0977	0.2132	0.0597	0.0507	0.0002	0.0193	0.1341	0.1148
		(BP) Avg. e_l	0.0651	0.0051	0.0100	0.0050	0.0007	.00017	0.0960	0.0975
		Avg. e_g (mm)	0.0	1.9197	2.7618	1.8226	2.5763	3.4344	0.0	0.0
		Avg. e_α (°)	4.1355	0.2966	0.3482	0.2769	1.2382	0.4316	3.8458	1.0038

By observing the results in Fig. 3.12-3.14, we get the following conclusions. First, the metrological mapping methods produce smooth strokes for both known mesh models and point clouds. Readers may compare Fig. 3.12(a.2-4, b.2-4, c.2-4) to (d.2-4, e.2-4), and also Fig. 3.13(a.2-3, b.2-3, c.2-3) to (d.2-3, e.2-3) for better inspection. Second, the metrological methods tend to preserve in-stroke distances between adjacent drawing points for the known mesh models, while the conformal mapping-based methods tend to preserve inter-stroke relations. This point can be drawn by comparing Fig. 3.12(c.2-4) and Fig. 3.13(c.2-3) to Fig. 3.14(c.2-3). The strokes in Fig. 3.12(c.2-4) and Fig. 3.13(c.2-3) are less globally connected compared to those in Fig. 3.14(c.2-3). For the raw point clouds shown in the last two columns of the three figures, all methods have more distortion compared to mapping using known mesh models. Third, the results also show that the performance of conformal mapping-based methods is unstable. For example, both the SI and II methods exhibit better performance on known mesh surfaces, as seen in Fig. 3.14(a.2-3, b.2-3). When the surface gets complicated, the deformation increases, as seen in Fig. 3.14(c.2-3). The reason is two-fold: (1) The parameterization scale of conformal mapping is non-uniform. (2) More errors get accumulated while solving the conformal transformation. Also, the conformal mapping-based methods will not work on more complicated models like models with concavity. Segmentation must be performed to make the methods applicable. Another observation about the conformal mapping-based methods is that they have low performance on the point clouds, as shown in Fig. 3.14(d.2-3, e.2-3). The reason is also two-fold: (1) Controlling the parameterized results' rotation is difficult. (2) The data is noisy. Based on the above observations, we decide not to use the conformal mapping-based methods. By observing the results in Table 3.3, we get the following additional conclusions for the metrological methods. First, we that find the EI-P, EI-Q, EI-B, and G-BP methods are faster than the G-RBF method. The reason is that G-RBF rebuilds a global parametric surface, which solves optimization equations and is time-consuming. In the worst case, the optimization may not converge and the method will fail. Second, the G-BP method is a fast one. However, it numerically approximates a mesh using near points in a ball and is sensitive to ball radius. Third, the EI-Q and EI-B methods are satisfying for curved objects, but they are less competitive on the box. Assuming a curved local surface is inherently questionable. Fourth, the EI-P method exhibits bad performance in e_g. The strokes may have a large displacement for multi-stroke drawing. However, it is a very efficient algorithm and is potentially good for fast prototype examination. Fifth, similar to our observations in Fig. 3.12-3.14, the conformal mapping-based methods are unstable. They sometimes show bad e_α for complicated meshes and point clouds. After balancing the time costs and errors of each method, we decide to use EI-P in our other experiments[8].

3.6.2 Costs of Reasoning and Motion Planning

We use the four drawing tasks shown in Fig. 3.15 to study our reasoner and motion planner's performance and costs. The strokes for the four tasks are the same circle consisting of 72 points. The first two target surfaces shown in Fig. 3.15(a, b) are human-designed CAD models. They have known meshes. The target surfaces in Fig. 3.15(c, d) are point clouds.

The grasp reasoning and motion planning performance for the four tasks are shown in Table 3.4. The labels (a-d) in the first row indicate the correspondence of each column to the tasks shown in Fig. 3.15. Below the first row, the table has three sections where the first section shows the size of the surfaces and the number of SLERPed pen poses along the mapped stroke. The following two sections show information about trials and failures in reasoning and time costs. We pre-annotated 62 candidate grasp poses for the pen. The reasoner iterated through the pen poses to find the common grasps for all key pen poses. The "Grasps" section of Table 3.4 includes four items where the first one shows the index of the 1st common grasp in the 62 pre-annotated

[8]Note that all metrological methods have a drawback – The errors of the following drawing points inherit the previous ones and a closed stroke may be broken after mapping. The error accumulation and broken position depend on both the starting point and the shape of a target surface. If a stroke is symmetric and the surface has symmetric normal, we may start from a central point of the symmetric stroke, map it to a central point on the symmetric surface, and begin drawing by moving towards a symmetric direction. On the other hand, if the stroke is non-symmetric or the surface is arbitrary, we do not have a good solution to avoid error accumulation and broken loops, unless we allow stretching the strokes. If ensuring a closed loop is the most important condition to meet, we recommend conformal mapping-based methods.

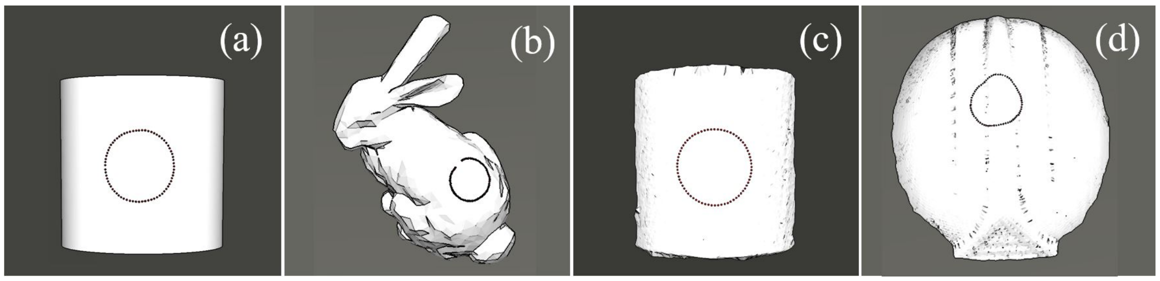


Figure 3.15: Four tasks used to evaluate the costs of the proposed reasoner and planner. The stroke is a circle. (a, b) Mesh models. (a): A cylinder. (b): A Stanford bunny. (c, d) Point clouds. (c): A point cloud captured the cylinder shown in (a). (d): A point cloud captured using a helmet.

grasps. The second item shows the number of all available common grasps. A motion planner will use a common grasp to generate the robot motion that moves the pen between adjacent key poses. The third item of the "Grasps" section shows the index of the 1st common grasp that has a solvable motion. For easy comparison with the first item, the index is also counted concerning the 62 pre-annotated grasps. Their values in (a), (b), and (d) columns are the same as the first item, which means the first common grasp led to successful motion. Contrarily, the value in (c) is different from the first item, indicating that the planner encountered unsolvable motion and switched to different common grasps for replanning. The value in (c) is highlighted with a gray background to signify the difference. The fourth item of the "Grasps" section shows the number of all common grasps with solvable motion. Note that the number of total common grasps and common grasps with solvable motion are counted for better evaluation. They do not need to be fully scanned in practice. The reasoner and planner stop and the robot begins execution as long as the 1st successful motion is found. The "Time Costs" section shows the time needed to find the 1st common grasp, all common grasps, the 1st solvable motion, and all solvable motion. On average, the planning takes 30 s to generate a drawing motion. It is more costly than reasoning.

Table 3.4: Costs of Grasp Reasoning and Motion Planning by Sequentially Scanning Pre-Annotated Grasps

		(a)	(b)	(c)	(d)
Information of the Surface and Strokes	Surface (mm)	40×40	25×25	40×40	40×40
	# SLERPed Pen Poses Along the Mapped Stroke	110	157	115	356
Grasps (62 Pre-annotated Grasp Poses in Total)	Index of 1st Common Grasp	0	0	1	0
	# Total Common Grasps	27	29	40	42
	Index of 1st Common Grasp with Solvable Motion	0	0	13	0
	# Total Common Grasp with Solvable Motion	8	1	4	7
Time Costs (s)	Find the 1st Common Grasp	0.05	0.07	0.05	0.05
	Find All Common Grasps	4.05	3.98	3.95	3.72
	Find 1st Solvable Motion	13.42	43.47	31.25	16.47
	Find All Solvable Motion	151.40	142.49	132.97	250.23

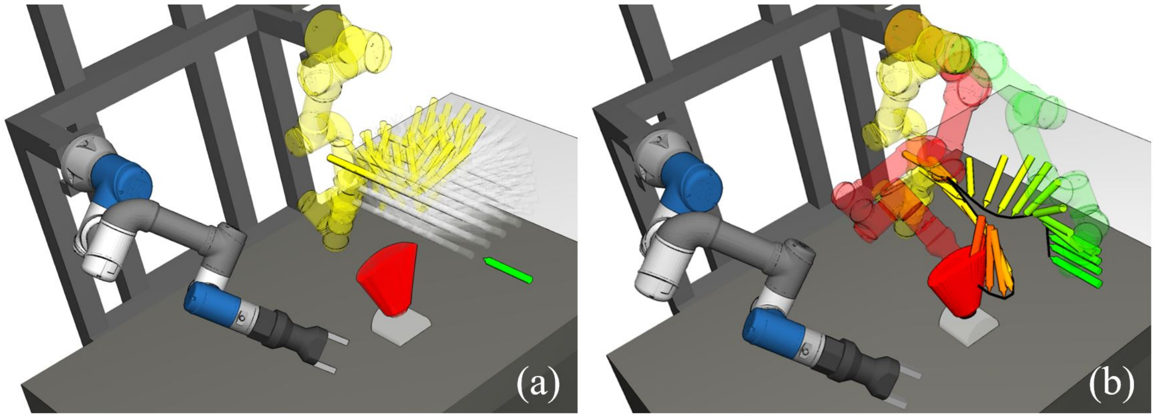


Figure 3.16: (a) Candidate pen poses. Red: Pen poses for drawing the mapped 3D strokes; Green: Initial pen pose; Gray and Yellow: Intermediate pen poses for in-hand estimation. Especially, the yellow ones indicate the intermediate poses that share common grasps with the green and red poses. (b) The successfully planned motion based on one common grasp.

The results in Table 3.4 are subject to the order of scanning and may not be enough to estimate a general performance. Thus, we further present Table 3.5 to show the minimum, maximum, average time costs, and standard time cost deviations of reasoning and planning of all common grasps with solvable motion. Each of the common grasps is examined independently instead of in a sequential scan. In the table, # G indicates the number of total common grasps with solvable motion. It is the same as the counterpart in Table 3.4.

Fig. 3.16 exemplifies one of the reasoned and planned results. The reasoner finds common grasps among the pen poses shown in Fig. 3.16(a), where the red ones indicate the pen poses for drawing the mapped 3D strokes, the yellow and gray ones show the candidate poses for in-hand estimation, and the green pose is the detected initial pose. The yellow-highlighted gray poses are the IK-feasible and collision-free in-hand estimation poses. The yellow robot in Fig. 3.16(a) shows a possible robot configuration to reach a common grasp at the in-hand estimation pose. Fig. 3.16(b) shows the successfully planned motion based on the common grasp.

Table 3.5: Statistic Costs of All Solvable Grasps

		(a)	(b)	(c)	(d)
	# G	8	1	4	7
Grasp Reasoning (s)	Min	0.03	0.07	0.04	0.04
	Max	0.09	0.07	0.05	0.05
	Mean	0.05	0.07	0.05	0.05
	Std.	0.02	0.00	0.01	0.01
Pick-up and Move-to Motion (s)	Min	11.04	40.73	12.71	10.77
	Max	19.08	40.73	17.90	20.62
	Mean	14.38	40.73	15.01	15.48
	Std.	2.67	0.00	1.86	3.71
Drawing Motion (s)	Min	0.89	2.75	0.92	2.84
	Max	0.91	2.75	0.94	8.45
	Mean	0.90	2.75	0.93	4.19
	Std.	0.00	0.00	0.01	2.14
Total (s)	**Min**	**11.98**	**43.55**	**13.68**	**13.64**
	Max	**20.02**	**43.55**	**18.88**	**27.05**
	Mean	**15.33**	**43.55**	**15.98**	**19.72**
	Std.	**2.68**	**0.00**	**1.87**	**4.81**

3.6.3 Performance of Real-world Executions

We carry out robotic executions to study real-world performance. We specially concentrate on analyzing (1) the influence of in-hand pose estimation on the final trajectories, (2) the influence of force control on the drawing results, (3) the difference between model-based and point cloud-based drawing, (4) the flexibility for multiple pens and complicated strokes, (5) error detection and recovery, and (6) precision of the final resultant drawing.

The necessity of in-hand pose estimation

We evaluated the necessity of the in-hand pose estimation by using the task of drawing a circle on a cylinder, like the ones seen in Fig. 3.15(a, c). Both the mesh models and point clouds are used in the evaluation. Fig. 3.17 shows the results. The figure has two columns (a) and (b) where each column includes four subfigures and an error chart. In the subfigures, the yellow robot configurations indicate the planned results. The red robot configurations indicate the refined motion after in-hand pose estimation. The green robot configuration is the real execution data read online from a working robot. The four subfigures are arranged in a 2×2 grid. The lower row of the grid shows the close-up view of the pen poses and pen tip trajectories in the upper row. The left and right columns of the grids are enclosed by frames with different colors. The left columns (the ones with orange frames) show the results of the originally planned trajectories. The right column (the ones with blue frames) shows the results using the refined trajectories after in-hand estimation. Note that although the orange-frame subfigures do not have in-hand pose estimation, we still repeated the red configurations in them as a reference for comparison.

Fig. 3.17(a) are the results using a mesh model. The orange-frame subfigures are the results of a direct execution. It does not include an in-hand pose estimation process. The green configurations and trajectories in the subfigures are the originally planned motion's direct execution results. Compared to the theoretical values (the red trajectories), a large offset is observable. It indicates that direct executions suffer a lot from noises. The blue-frame subfigures are the results with in-hand estimation and trajectory refinement. The green trajectories in the subfigures are the execution result of the refined motion. Compared to the theoretical values, the offset is moderate. The results indicate that the in-hand pose estimation and refinement significantly improved precision. Fig. 3.17((b) are the results using point clouds. Like their orange-frame and blue-frame counterparts in (a), the green trajectories in differently framed subfigures are the execution results of originally planned motion and refined motion respectively. The offset in these cases is smaller since the strokes are directly mapped to the measured surfaces, but there remains a discernible difference between the red and green trajectories.

The error charts in Fig. 3.17 visualize the distance between pen tip trajectory in simulation and real execution. The horizontal axis of the figure is the point ids. The vertical axis is the difference between correspondent points. The blue curve shows the value before applying in-hand pose estimation. The orange curve shows the value after in-hand pose estimation. The curves further confirm that the offset decreased after in-hand pose estimation and motion refinement.

The necessity of force control

The results in 1) partially show that force control is important to remove detection errors and ensure a solid drawing. In this part, we further examine the importance of force control by comparing the task shown in Fig. 3.18. Here, the goal is to draw a circle on a cylinder. The subfigures with orange frames show the results without force control. The subfigures with blue frames show the results with force control. Both methods can perform the drawing, but without force control, it is not easy to ensure the pen to be always in contact with the target surface. One section of the circle is lost when force control is not applied, as seen from the binarized boundary view in the lower part of the subfigures. The chart on the rightmost side of Fig. 3.18 compares the changes of force curves during drawing. The horizontal axis is the point id. The vertical axis is the magnitude of the force applied to the pen tip. The orange curve shows the force changes without force control, while the blue curve shows the changes with force control. The curves show that the force at the pen tip has large values between the 1st and 15th points and after the 65th point without force control. The force disappeared between the 20th and 40th points. We examined the real-world motion and found that since the surface position used to generate the motion in the simulation environment was different from that of the real world, the pen tip sometimes got lower than the surface and a large force appeared, leading to a heavy stroke. It also sometimes got higher than the surface and the force disappears, resulting in a lost stroke. With the help of force control, the positional error could be successfully suppressed. The controller adjusted the robot motion to ensure the forces fall inside the blue shadow range.

Influence of mesh models

We also compared the difference between the drawing results planned using mesh models and point clouds. The details are shown in Fig. 3.19(a, b). The task is the same as the one shown in 2) - draw a star on the cylinder object. Fig. 3.19(a.1) is the result based on a mesh model. Fig. 3.19(a.2) shows a close-up view of the strokes. Fig. 3.19(b.1) is the result based on point clouds. Fig. 3.19(b.2) shows a similar close-up view. Our system can finish the drawing in both cases. However, the result planned with a mesh model is smoother than the result planned with point clouds, for the face normals estimated using point clouds are noisier than the inborn values of a mesh model. The pen poses of the model-based drawing are thus more continuous than those of the point cloud-based drawing.

Fig. 3.19(c, d) further show the results using two more difficult objects. Fig. 3.19(c.1-2) are the results of drawing a circle and a star on the back of a Stanford bunny. The robot motion and pen trajectories are generated based on the bunny's mesh model. Fig. 3.19(d.1-2) are the results of drawing the same circle and star on an engineering helmet. There is no mesh model for the helmet. The robot motion and pen trajectories were planned based on point clouds. Although the results based on point clouds are less smooth in Fig. 3.19(d.1-2), they are only visible in a close-up view. From the viewpoints of Fig. 3.19(c, d), the roughness is less noticeable and satisfying. Thus, we conclude that the developed robot system can draw on complicated surfaces both with and without mesh models.

Multiple pens, complicated strokes

We examined the ability of our system to switch pens and draw complicated strokes using the two tasks shown in the upper part of Fig. 3.20(a.6) and (b.7). The first task is to draw the word "DRAW" on the cylindrical surface. Each letter in the word is required to have a specific color. The system successfully planned the motion for each pen as well as the motion for switching them. Fig. 3.20(a.1-5) show snapshots of the execution sequence and the final drawing results. The second task is to draw a Peppa Pig face on the same cylindrical surface. The Peppa Pig face strokes include several small primitive elements. The developed system was still able to successfully and robustly perform the task. Fig. 3.20(b.1-6) show snapshots of the execution and the final drawn Peppa Pig face.

Error detection and recovery

We carried out experiments to study the error detection and recovery ability of the developed system using a challenging task shown in Fig. 3.21. The goal was to draw a circle across the edge of a box in the presence of the following errors: (1) Backward displacement error (translation in global -y axis); (2) Forward displacement error (translation in global y axis); (3) Rotation error around global z axis. The results are as follows.

When there was no error, the motion was well-performed as shown in Fig. 3.21(a). When translational error existed and the real edge position was before the edge in simulation, a lost surface slippage was detected at the red point shown in Fig. 3.21(b.1). The force exhibited an abrupt drop in the pen's main axis, as shown by the force curves in the (b) column. The robot took an action to skip to a future point and continued the drawing motion, as shown by the TCP rotation curves in the (b) column and also the green dots in Fig. 3.21(b.1). By comparing the TCP rotation curves in (b) and (a), we can see that a later section was shifted forward to skip the failed points. When translational error existed and the real edge position was behind the edge in simulation. An in-hand slippage was detected at the red point shown in Fig. 3.21(c.1). The forces exhibited a large torque around the robot hand opening direction, as shown by the force curves in the (c) column. The robot resumed to a previous point and repeated the drawing motion, as shown by the TCP rotation curves in the (c) column and also the green dots in Fig. 3.21(c.1). By comparing the TCP rotation curves in (c) and (a), we can see that an early section was repeated to delay the following drawing points. When rotational error existed, the robot may either detect a lost surface slippage or an in-hand slippage, depending on the rotation directions. Fig. 3.21(d.1) shows an example of an in-hand slippage. Like the TCP rotation curves in the (c) column, an early section of the TCP rotation curves in the (d) column was repeated to delay the following drawing points. The robot did not detect a slippage when moving the pen across the edge for a second time, and the later sections of the TCP rotation curves were the same as that of (a). The experimental results show that the proposed algorithm can successfully draw the circle while avoiding errors. The final drawn results are shown in the last figure row.

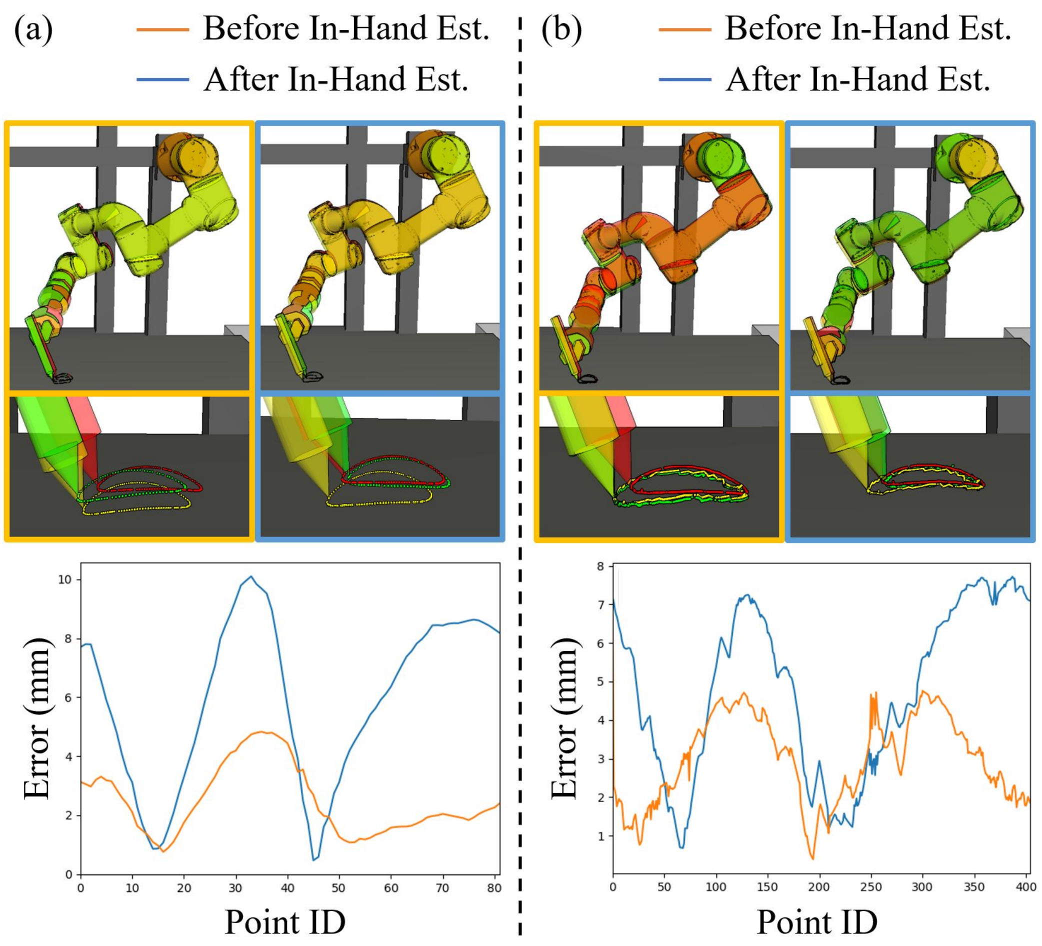


Figure 3.17: Comparison of the originally planned robot configurations (yellow robot configurations and drawing points in the upper subfigures), the refined trajectories based on in-hand pose estimation (red robot configurations and drawing points), and the real execution results using force control (green robot configurations and drawing points). The subfigures with orange frames show the results of the originally planned trajectories. The green robot configurations and drawing points in it are the execution results of the yellow ones. The red robot configurations and drawing points are illustrated as a reference. The subfigures with blue frames show the results using the refined trajectories after in-hand estimation. The green robot configurations and drawing points in it are the execution results of the red ones. (a) uses the mesh model shown in Fig. 3.15(a) as the input. (b) uses the point clouds shown in Fig. 3.15(c) as the input. The bottom charts show the error curves. The blue curves are the original results. The orange curves are the results after in-hand pose estimation. The horizontal axes of the charts are the point IDs. The vertical axes are the errors (difference between correspondent points).

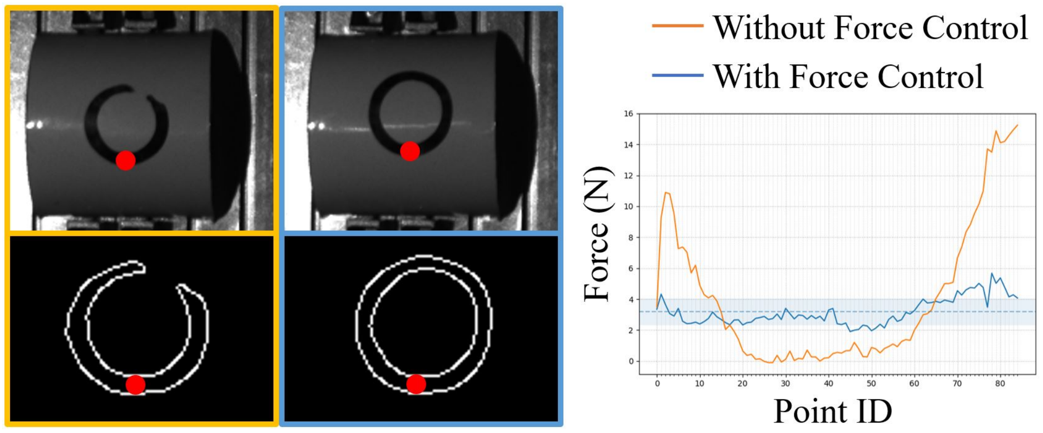


Figure 3.18: Drawing a circle without (orange) and with (blue) force control. The chart shows the force along the pen's main axis concerning the point ids of the mapped strokes.

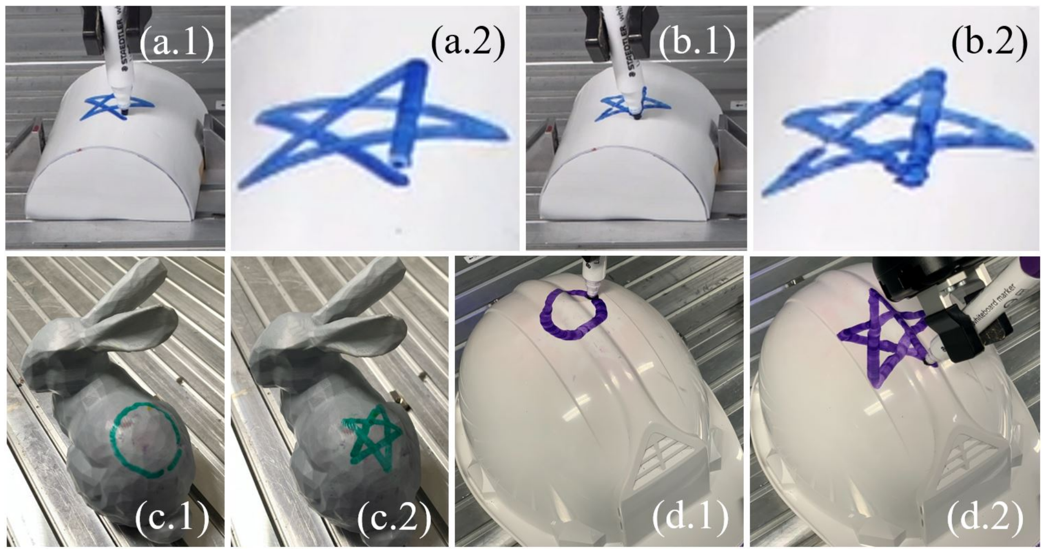


Figure 3.19: Comparison of the drawing results planned using mesh models and point clouds. (a, b) Drawing a star on the cylinder object. (a) Results using a mesh model. (b) Results using a point cloud. (c, d) Other examples. (c) Drawing on a Stanford bunny. The mesh model is used. (d) Drawing on a helmet. The captured point cloud is used.

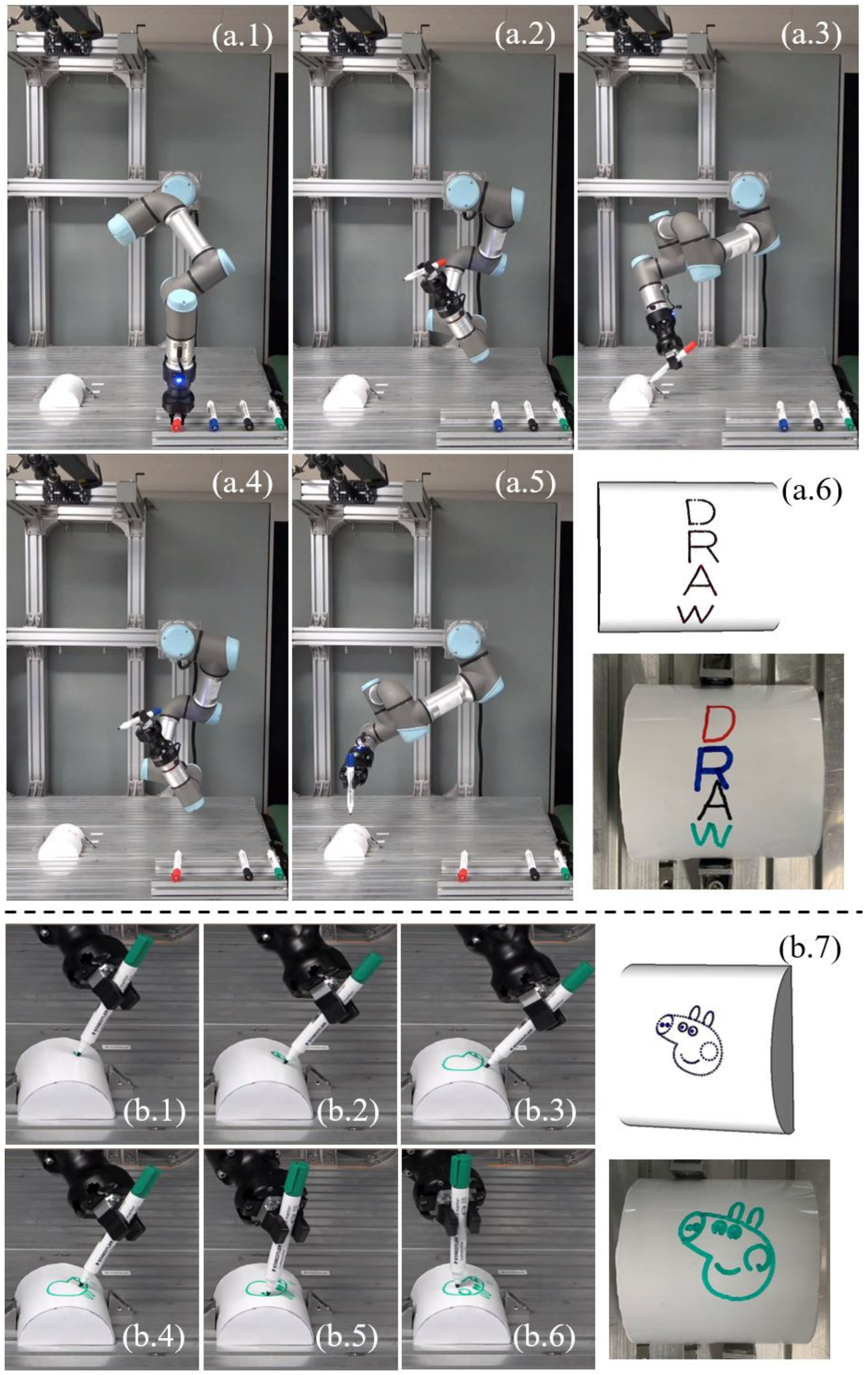


Figure 3.20: (a) Drawing a word "DRAW" on a cylinder surface with each letter in a different color. (b) Drawing a Peppa Pig face on the same cylinder surface.

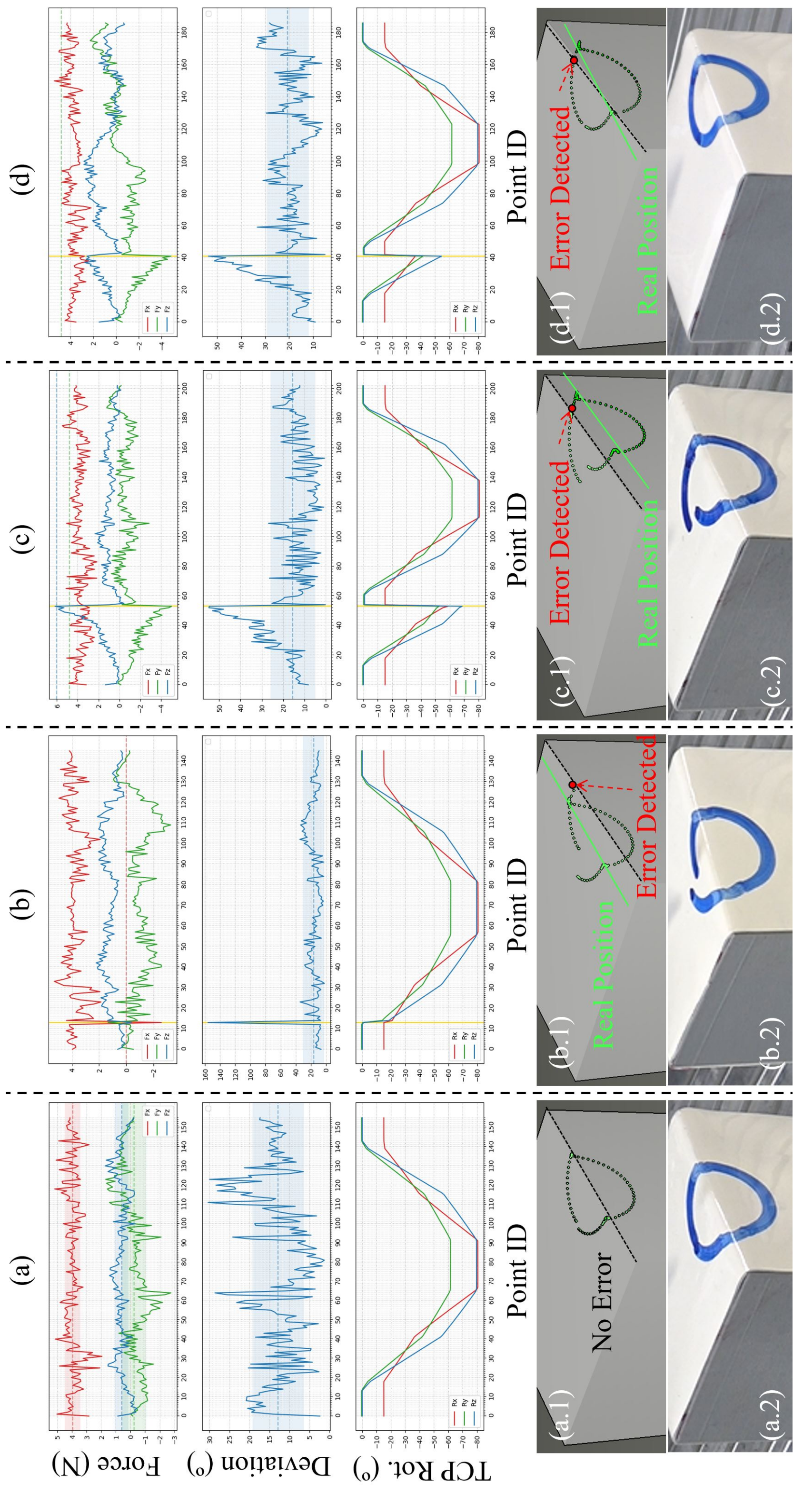


Figure 3.21: Error detection and recovery using drawing a circle across a box edge as an example. (a) Results without errors. (b) Results in the presence of a backward displacement error (translation in the global -y axis). (c) Results in the presence of a forward displacement error (translation in the global y axis). (d) Results in the presence of a rotational error (rotation around the global z axis). The charts in the first row of each column show the force along the pen's local x (red), y (green), z (blue) axes during the motion. The x axis is the pen's main axis. The second row shows the angle between the resultant forces and the surface normal at each point. The third row shows the TCP rotation angles (roll, pitch, yaw). (a.1) There is no error. (b.1) The real edge position is before the edge in simulation. A pen-surface slippage is detected at the red point. (c.1) The real edge position is behind the edge in simulation. A pen-hand slippage is detected at the red point. (d.1) The real edge position is rotated. A pen-hand slippage is detected at the red point. (a.2-d.2) Final drawn results.

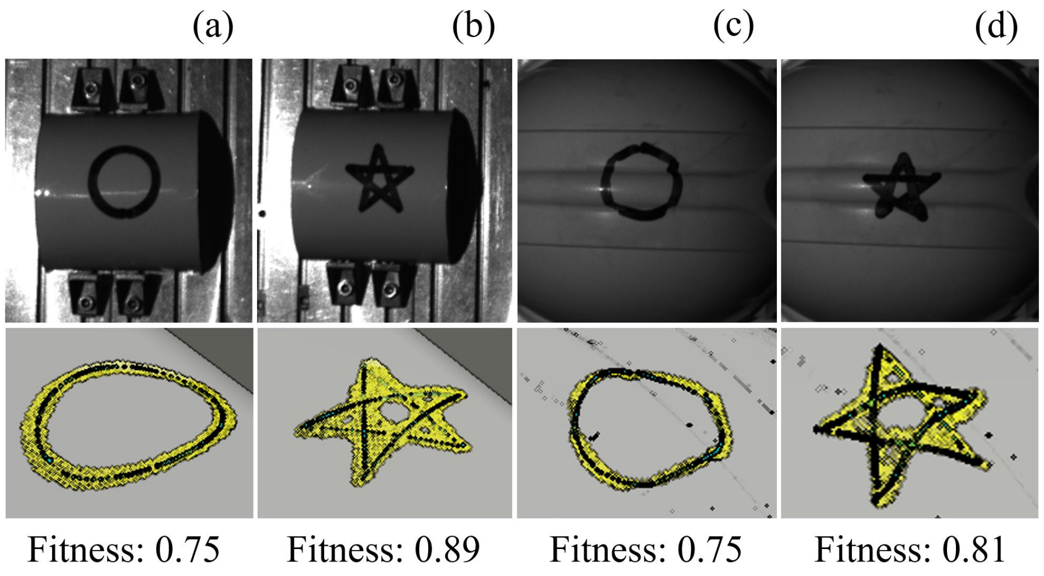


Figure 3.22: Evaluation of the final resultant drawings. The first row shows pictures of the final results. The second row shows the 3D point clouds of the drawn strokes (yellow) and the expected goals (blue dots). The fitness values of the ICP matching between them are shown below.

Evaluation of the final resultant drawing

We also carried out experiments to globally evaluate the final resultant drawing. The goal of our robotic drawing system is to ensure the mapped strokes (drawing goals) and the results drawn by the robot are highly alike. To judge if this goal is effectively reached, we captured images of the drawing results as shown in the first row in Fig. 3.22, extracted their point cloud, and compared the extracted point cloud with the drawing goals. The second row of Fig. 3.22 illustrates the extracted point clouds (yellow area) and drawing goals (blue dots). We used the ICP algorithm to match the drawing goals to the point clouds and used the fitness of the ICP matching to evaluate their similarity. The matched fitness values are shown below the second row. The values are small. They confirm that the resultant drawings are very similar to the drawing goals.

3.7 Conclusions

This section presented a robotic drawing system that maps 2D strokes to 3D surfaces, plans pen picking and manipulation motion, and performs drawing on 3D surfaces considering vision and force feedback. The system is flexible as it treats pens as tools that can be picked up and manipulated online. It is robust as it runs in a closed-loop manner and leverages high quality-stroke mapping, in-hand pose estimation and motion refinement, force control, and error detection and recovery to suppress failures. The system works on a wide range of surfaces as long as they are continuous and do not trap a pen. Experimental results demonstrated the excellent expected performance. We encourage our readers to watch the supplementary video to better view the mentioned features. The video includes drawing demonstrations using different surface shapes (convex and concave shapes), input types (meshes and point clouds), and pens (marker pens and ballpoint pens).

One limitation of the current system is that the input must be vectorized 2D strokes, which is difficult for non-professional users. In the future, we are interested in developing an interface that extracts strokes by watching a human demonstration. Also, the size of the 2D strokes must fit the workspace of the robot, Or else a nearby point cannot be found, and the system will report a failure. We are interested in solving this problem using dual-arm robots [114] and regrasp [115] in the future.

Appendix

Comparsion of projection step

We have to iteratively estimate local geometries to interpolate the point cloud and find the next $\boldsymbol{s}_{i+1}$. The iteration is time-consuming and may lead to accumulated errors. Compared to it, a global parametric surface could help to avoid local iterations and be more efficient. Thus, we studied the performance of global estimation-based mapping methods. The global surface allows us to carry out differential computations and measure the deformation using geodesic distances.

Specifically, we implemented two kinds of global geometries. The first one followed

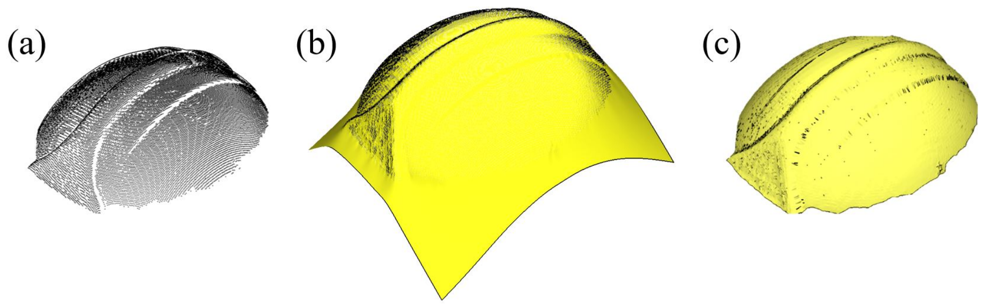


Figure 3.23: Global surfaces. (a) A point cloud. (b) A global parametric surface rebuilt using the RBF method. (b) A global numerical surface rebuilt using the ball-pivoting method.

Carr's work [108] and modeled a point cloud using Radial Basis Functions (RBF). Fig. 3.23(b) shows the modeling result of this method. The second one is a numerical one that rebuilds the mesh model using ball pivoting [109] but does not return an explicit parametric surface. Fig. 3.23(c) shows the modeling result of the second method.

Based on the surfaces, we carried out more experiments to compare its performance with other methods. The results are shown in Fig. 3.24. The figure uses the point cloud of a helmet to compare both local estimation-based methods and global estimation-based methods. The input to these methods is the helmet point cloud shown in Fig. 3.23(a). The first three columns are the results of EI-based methods. Each specific column uses a different local geometry for estimation. The last two columns are the results of global surface-based methods. The two specific columns use global surfaces estimated using RBF and ball-pivoting respectively. The different rows of the figure show the results under different stroke step length (distance between two adjacent $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$, as was shown in Fig. 3.24). The stroke step length $\overrightarrow{\boldsymbol{c}_i\boldsymbol{c}_{i+1}}$ is set to 1, 2, 5, 10, 20 mm from top to bottom in each column to compare the influence of the different algorithms to stoke deformation under the same mapping method. The dashed box in the top-right corner illustrates the 2D strokes under the different stroke lengths for readers' convenience. By observing the graphical results, we can find the methods do not have a large difference when the stroke step length is small, as shown by the first row in the chart. However, the results of the EI methods

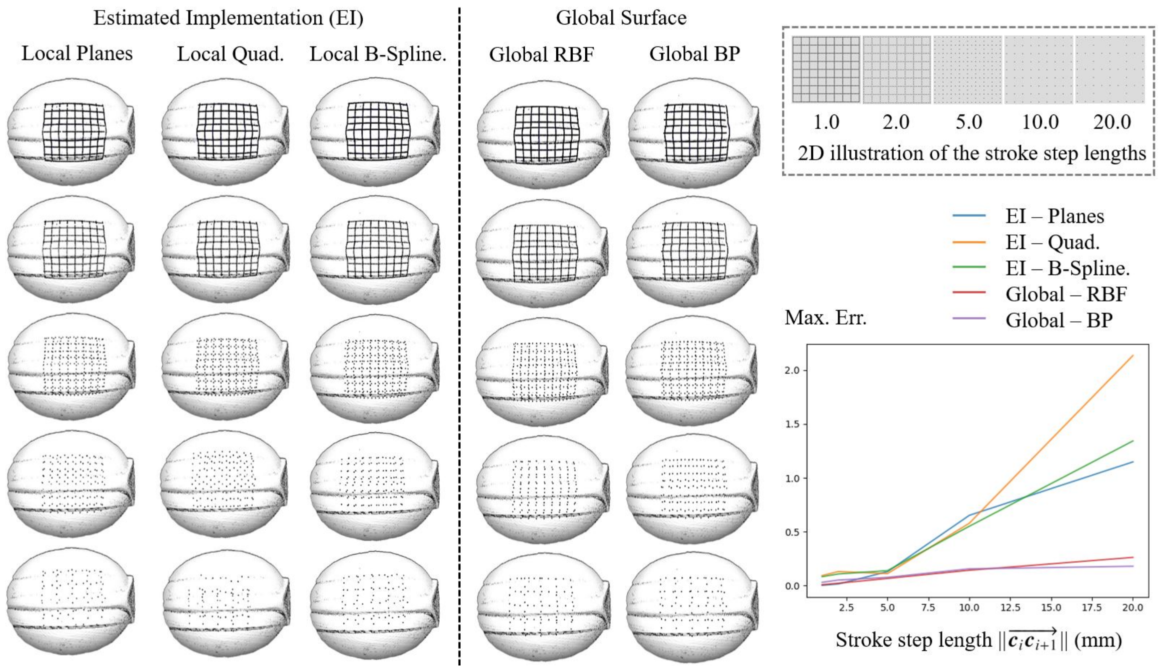


Figure 3.24: Mapping a grid on a helmet (point cloud) using the metrological methods. Errors of the results change with the length between adjacent points in strokes (stroke step length). From top to bottom: The stroke step lengths are set to 1, 2, 5, 10, 20 mm respectively. From left to right: Different mapping methods. The chart on the right shows the maximum error curves of the various methods under the changing stroke length.

degenerate significantly as the stroke step length increases. At 20 mm, the mapped grids almost lose their original shapes.

The maximum error curves of the various methods under the changing stroke length are shown in the right chart of the same figure. Same to our conclusion from the graphical observation, the projection error increases together with the increasing stroke step length. The increasing tendency is more significant in the EI methods. On the other hand, all methods have satisfying precision when the stroke step length is small, i.e. 1-2 mm. Especially, the EI-P and G-RBF methods have even better performance. Considering that we will always use a small step length (2 mm) in our implementation and the EI-P method is much faster compared to the others, we used EI-P in our real-world experiments.

An Optimization-based Planner

The reasoning and planning method presented in the main text requires that the pen poses are aligned with surface normals. The method is fast but finds fewer solutions since constraining the pen pose to be along the normal direction is a very strong condition that highly reduces the solution space. In this appendix, we present an FK-based optimization solver as an alternative for it. The solver has explicitly defined constraints. It allows us to relax constraints and produce a drawing path that is smooth, has less motor action, but meanwhile does not restrict pen poses exactly to surface normal.

The optimization solver treats the 3D drawing as an FK-based optimization problem considering the following goals and constraints.

$$\begin{aligned}
\min_{\boldsymbol{q}_{i+1}} \quad & (\boldsymbol{q}_{i+1}-\boldsymbol{q}_i)^T(\boldsymbol{q}_{i+1}-\boldsymbol{q}_i) && (3.6a)\\
\text{s.t.} \quad & \boldsymbol{q}_{i+1}=[q^0_{i+1},q^1_{i+1},...,q^d_{i+1}]^T \\
& q^k_{i+1}\in[\theta^-_k,\theta^+_k], k\in\{0,1,...,d\} && (3.6b)\\
& \boldsymbol{p}_{i+1},\boldsymbol{R}_{i+1},\{\boldsymbol{l}^r_j\}_{i+1},\boldsymbol{l}^p_{i+1}=\texttt{fk}(\boldsymbol{q}_{i+1}) && (3.6c)\\
& (\boldsymbol{p}_{i+1}-\boldsymbol{s}_{i+1})^T(\boldsymbol{p}_{i+1}-\boldsymbol{s}_{i+1})\leq\epsilon && (3.6d)\\
& \texttt{acos}(\frac{\boldsymbol{R}_{i+1}(x)\cdot\boldsymbol{n}_{i+1}}{||\boldsymbol{R}_{i+1}(x)||\cdot||\boldsymbol{n}_{i+1}||})\leq\phi && (3.6e)\\
& \texttt{min}(\texttt{dist}(\boldsymbol{l}^p_{i+1},\mathcal{O}))\geq\xi_p \\
& \forall\boldsymbol{l}^r_j\in\{\boldsymbol{l}^r_j\}_{i+1},\texttt{min}(\texttt{dist}(\boldsymbol{l}^r_j,\mathcal{O}))\geq\xi_r && (3.6f)
\end{aligned}$$

Here, we set the optimization goal to be minimizing the changes of joint angles between every two adjacent arm configurations. The notation $\boldsymbol{q}_{i+1}$ denotes an arm configuration at the i+1th drawing point. It includes d elements, which respectively indicate the joint angle of each degree of freedom. The joint angles must meet the work range limits, as shown by equation (3.6b). The forward kinematics function (fk() in equation (3.6c)) finds the pen tip position $\boldsymbol{p}_{i+1}$, pen's rotation matrix $\boldsymbol{R}_{i+1}$, and the vectors of the robot links ($\{\boldsymbol{l}^r_j\}_{i+1}$, a set of vectors) and the grasped pen ($\boldsymbol{l}^p_{i+1}$, a single vector). The distance between the pen tip position and a mapped drawing

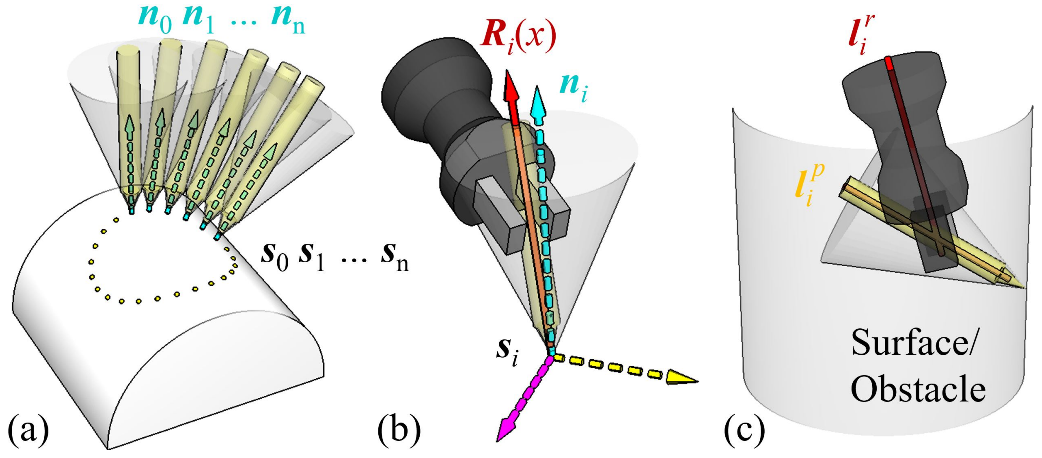


Figure 3.25: Some of the constraints for optimization-based planning. (a) Pen tip position constraint. (b) Pen pose constraint. (c) Collision constraint.

point must be smaller than a threshold offset ϵ, as expressed by equation (3.6d) and also the illustration in Fig. 3.25(a). The angle between the held pen and the drawing point's surface normal must be smaller than a threshold angle ϕ, as expressed by equation (3.6e) and the illustration in Fig. 3.25(b). Here, we define the local x direction of a pen to be its main axis (the axis points from a pen tip to a pen end). Thus $\boldsymbol{R}_{i+1}(x)$ (the x component of the pen's rotation matrix) is used to carry out the computation. The obstacle avoidance constraint is taken into account by limiting the distances between $\boldsymbol{l}^p_{i+1}$ and obstacles $\mathcal{O}$, and also all $\boldsymbol{l}^r_j$ in $\{\boldsymbol{l}^r_j\}_{i+1}$ and obstacles $\mathcal{O}$. Fig. 3.25(c) illustrates the constraint. Especially for $\boldsymbol{l}^p_{i+1}$, the pen tip is ignored since it must contact the target surface.

The following tasks and results compare the IK-based method introduced in the main text and this optimization-based alternative. The task settings and results are shown in Fig. 3.26[9]. The first task requires drawing a circle and a star on the inner surface of a plastic cylindrical bucket. The second task requires drawing a circle on the inner corner of a rectangular glass tank. The detailed geometric information of the objects and the success rate, average time costs, and average joint displacements of the planned motion are shown in black text in Table 3.6. For readers'

[9]We use transparent objects for better visualization. The poses of these objects, together with their mesh models, are pre-specified in the program.

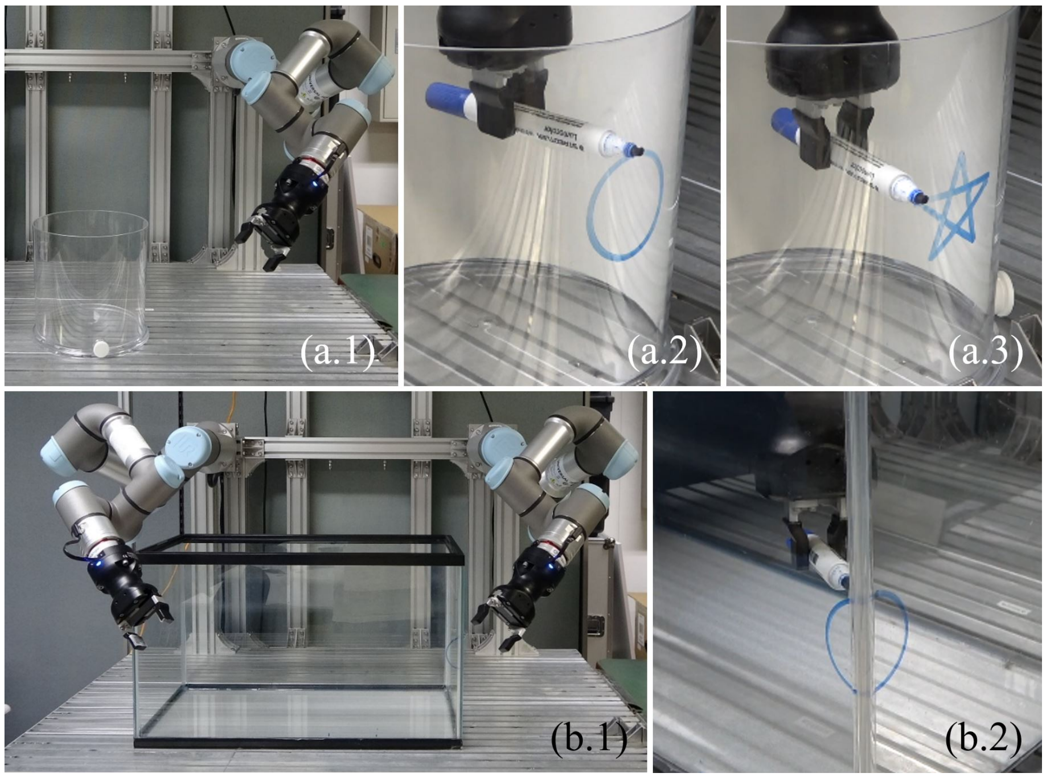


Figure 3.26: Task settings and results of drawing on concave surfaces. (a.1-3) Draw a circle and a star inside a cylinder. (b.1-2) Draw a circle on the inner corner of a tank.

convenience, we also show the comparison with convex objects (cylinder and helmet) in the second part of the table. For the 62 pre-annotated grasps, the planner finds fewer successful candidates for the concave objects, indicating the concave problems are more constrained. Meanwhile, the average time costs become smaller since the constraints reduce the search space. The execution results of these tasks can be found in the supplementary video[10].

[10]The optimization-based method has two drawbacks. The first one is speed. Table 3.6 shows that an optimization-based method takes more than 200s to find a solution. The second drawback is the collision or the measurement between the pen/robot and the surrounding obstacles. We have to simplify the pen/robot models into vectors and measure the distance between an obstacle surface and the vectors to ensure the metric functions are differentiable. On the other hand, the optimization-based method has advantages in that it is more complete and can find the solution for tasks that are not solvable by the IK-based method (i.e. the “Tank” task in the table). We thus leave it to a practitioner to decide which method to use.

Table 3.6: Concave Tasks and Comparison

	Convex Object			
	Cylinder		Helmet	
	Opt-based	IK-based	Opt-based	IK-based
Volume (mm)	100 × 100 × 50		166 × 199 × 72	
Size (mm)	40 × 40		40 × 40	
# Pen Poses	106		370	
Success Rate	40/62	20/62	47/62	34/62
Avg. Time (s)	208.20	0.85	704.61	3.09
Avg. Jnt. Mov. (°)	250.65	279.57	790.48	1036.21

	Concave Object			
	Bucket		Tank	
	Opt-based	IK-based	Opt-based	IK-based
Volume (mm)	230 × 230 × 230		600 × 300 × 360	
Size (mm)	60 × 60		60 × 60	
# Pen Poses	84		129	
Success Rate	30/62	27/62	17/62	0/62
Avg. Time (s)	208.40	0.80	293.57	-
Avg. Jnt. Mov. (°)	238.67	341.46	129.37	-

* Meanings of items: # Pen Poses - Number of Pen Poses; Avg. Jnt. Mov. - Average Joint Movement.

Experiments using Ballpen

The proposed system can also be adapted to other types of pens. Some supplementary experiments are provided in this subsection.

We are using the F/T sensor embedded at the wrist of a Universal Robots e-series manipulator for the experiments. The precision of the F/T sensor is 3.5 N in force and 0.10 Nm in torque. The resolution is around 1.0 N in force and 0.02 Nm in torque. The precision, we would say, is a big large for drawing tasks. For this reason, we chose a strong pen to carry out the experiments. Also, since we have to perform experiments repeatedly, we chose to use a non-permanent marker pen with a soft tip in our study.

In practice, our system is not limited to the selected pen type. However, due to the limitation of the F/T sensor's precision, it remains advisable to use a strong one. Fig. 3.27 shows an example where our robot uses a ballpoint pen to draw on a piece of paper wrapped on a wooden handle. The ballpoint pen is still strong enough for the F/T sensor to control. A video clip about using the ballpoint pen has been included in the newly submitted video supplementary. On the other hand, if a robot has a high-performance F/T sensor, a weaker pen should definitely work. Right now, we do not have a high-precision F/T sensor to demonstrate this point, but we hope we can show some interesting results in the future.

The planning needs to be tuned a bit to make it applicable to pens with particular mechanisms. For example, the ballpoint pen must be moved with a considerable speed and smaller pressure force to make sure the ball at the pen tip rolls around the contact point and the ink is firmly transferred to the surface.

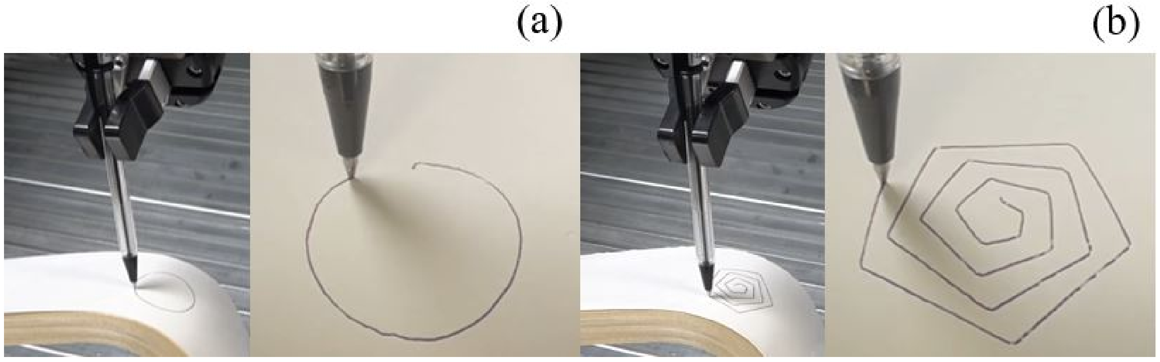


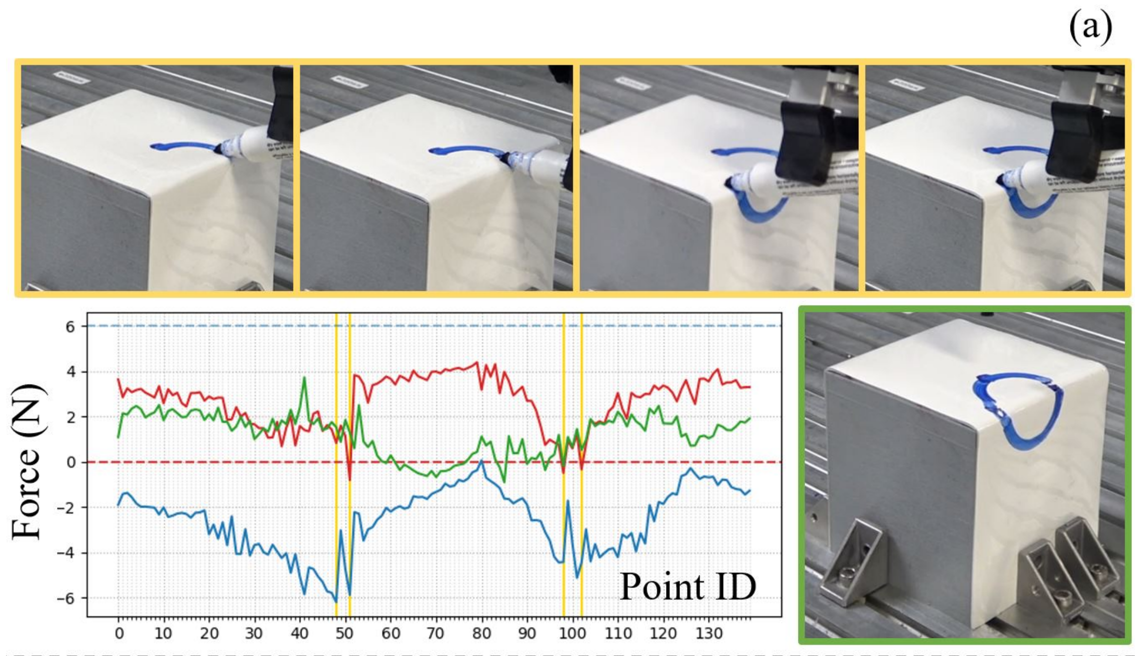
Figure 3.27: Drawing with a ballpoint pen on a piece of paper wrapped on a wooden handle. (a) A circle. (b) A rose spiral.

Error Recovery by Skipping Failed Positions

The error recovery method presented in the main text can help to maintain continuity and also help to correct drawing poses. However, the lengths of forward and backward searches matter a lot. For a surface with a repeated pattern, e.g. the jagged surface shown in our supplementary video, a large search length may lead the stroke to jump to the next repetition, and the system will get trapped in a repeated failure. In this part of the appendix, we present an alternative error recovery method. In this method, we close the loop by skipping the failed positions. In case of slippage, the method will pause the current motion, pull the pen up, and move to the next drawing point to start a new iteration.

Fig. 3.28 shows the results of drawing a circle across a box edge using this method. It includes two cases of errors. The first one has the same error as that of Fig. 22(c). The second one is a side displacement error (translation in the global x axis) where the pen encounters a lost section during drawing. The charts show the forces along pen's local x (red), y (green), z (blue) axes. At the points marked by the yellow vertical lines, the error detector finds large differences. The robot arm moves the pen away from the target surface, gives up a current drawing point, and switches to a next point to recover from the errors. The recovery is triggered four times in the task shown in Fig. 3.28(a) and five times in the task shown in Fig. 3.28(b).

A shortage of this alternative method is that it gives up the current drawing point in case of slippage and moves on to the next point for recovery. This policy may result in broken drawings. On the other hand, the shortage can also be an advantage. It provides a chance to skip a surface section and has the merit of dealing with bad surface sections (e.g. scratched, dirty, lost sections, narrow gaps, etc.). For example in the second case shown in Fig. 3.28, the method presented in the main text will get trapped at the lost point to avoid breaking lines. Contrarily, the alternative method skips to the next reachable surface to continue drawing. Practitioners may use the method in the main text if continuity is most important to them. They may also use the alternative method if their surfaces have defects or discontinuity and would like to overlook the bad surface sections.



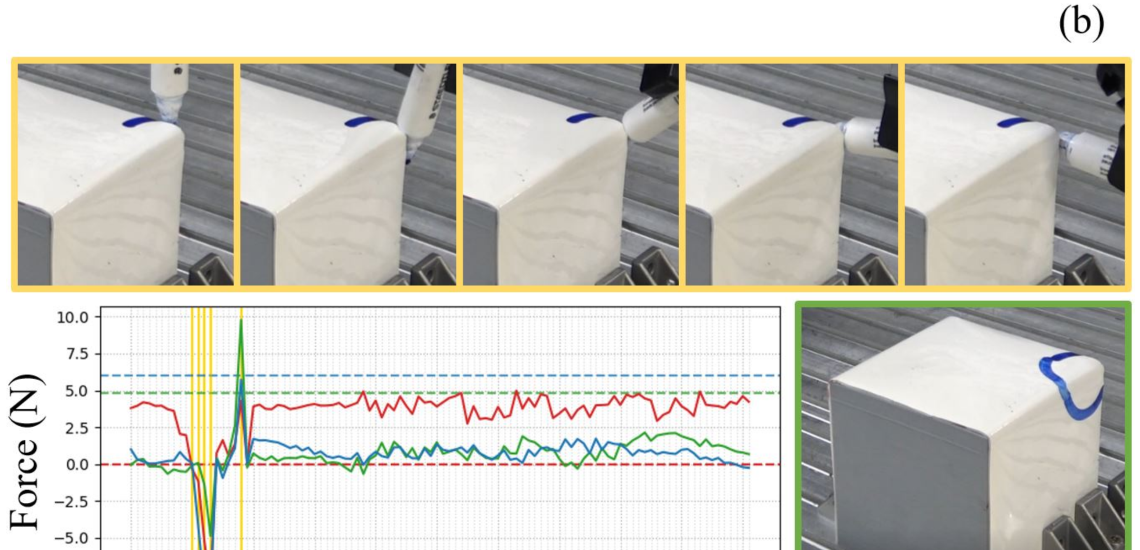


Figure 3.28: Drawing a circle across the edge of a box with the alternative error recovery method. The robots detected errors (as shown labeled by the yellow vertical lines in the force charts) and recovered from the errors by moving ahead to the next drawing point. Like previous figures, the charts show the forces along the pen's local x (red), y (green), z (blue) axes. (a) Results in the presence of a forward displacement error (the same as Fig. 3.21(c)). (b) Result in the presence of a side displacement error (translation in the global x axis).

Chapter 4

Next Best View Planning for Metal Plate Reconstruction

In this section, we present a novel approach for planning the Next Best Views (NBV) of an object so that a depth camera can collect the object's surface point cloud and reconstruct its 3D model with a small number of consequent views. Our focus is especially on thin and curved metal plates, and we use a robot manipulator and an externally installed stationary depth sensor as the experimental system. The targeted objects have shiny and flat surfaces, which leads to noisy point cloud data and low guidance in the surface normal for completion. To overcome these challenges, we propose using a Point cloud Completion Network (PCN) to find heuristics for NBV or Next Best robot Configuration (NBC) optimization. Unlike previous methods, our approach predicts NBV by considering a holistic view of the object predicted by neural networks, which is not limited by the local information captured by the sensors and is, therefore, robust to deficiencies in known point cloud data and normal. We conducted simulation and real-world experiments to evaluate the proposed method's performance. Results show that the proposed method efficiently solves the NBV problems and can satisfactorily model thin and curved metal plates.

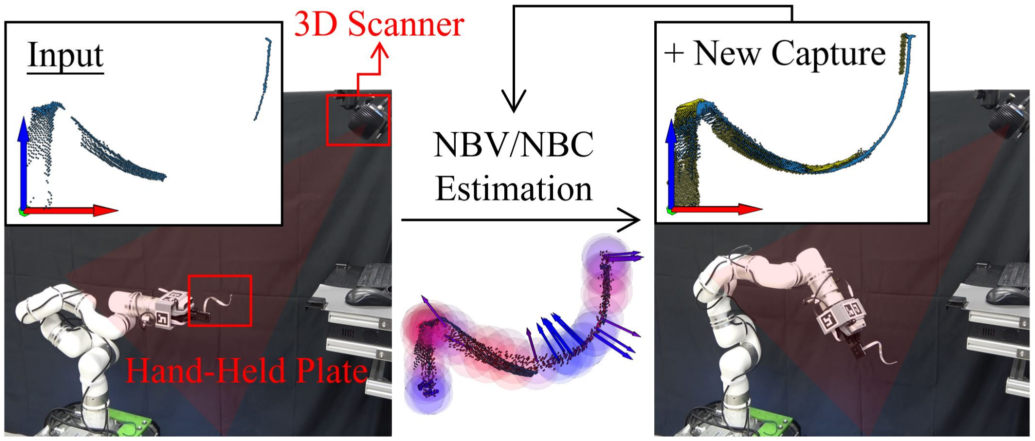


Figure 4.1: A robot that obtains the shape of a hand-held thin metal plate by observing it from NBV/NBCs obtained based on shape completion learning. The thin metal plate has a shiny and flat surface, making it difficult to infer optimal NBVs. This work solves the problem by using a point cloud completion network as a heuristic for NBV/NBC optimization.

4.1 Introduction

This section studies using a robot manipulator and a stationary depth sensor to obtain a metal plate's 3D mesh model with a minimal number of views. The challenge in obtaining the model of such objects lies in their shiny and flat surface, which leads to noisy point cloud data and low surface normal guidance for determining the next viewpoints. We propose using a point cloud completion network to provide heuristics for NBV/NBC optimization, thus overcoming the challenge. Compared with existing methods, our approach predicts NBV/NBC based on a holistic view of the object predicted using PCN. It is not limited by the local information captured by the sensors and is, therefore, suitable for thin plates with either insignificant[1] or complex features[2]

[1]Insignificant features cannot provide enough information for surface normal estimation. For example, when only the edge of a plate is captured, the point cloud forms a curved or line-like shape with inadequate 2D variance. Using Principal Component Analysis (PCA) to estimate the normal of such point clouds may lead to incorrect results.

[2]Complex features have intricate local geometry and will lead to an overabundance of point clouds and thus noisy normal estimation.

Previously, researchers proposed using a robot to scan an object's 3D model automatically. Robots could be configured as actuators to grasp a target object and show it to external cameras for scanning. They could also be equipped with hand-eye systems to observe objects on a table. In both cases, the robot must decide how to move itself to obtain the necessary views for 3D scanning [116][117], which is known as the NBV/NBC problem. Remarkably, researchers proposed using NBV/NBC planning to determine the object or camera poses that lead to an optimal amount of information about the unknown object. However, state-of-the-art NBV/NBC planners were based on local information. If the local information is insignificant, noisy, or noncontinuous, it may lead to a failure in NBV/NBC planning. In this work, we propose using a point cloud completion-based approach to learn the global geometry and estimate the confidence and NBV/NBC considering the learned global information. The proposed method provides more accurate and efficient view-planning results than previous methods. The NBV/NBCs can be formulated as the same problem and optimized in a continuous multi-DoF space without a significant change.

Fig. 4.1 illustrates the workflow of the proposed method. We especially assumed a thin, curved metal plate as the target object and a robot manipulator with an external stationary depth sensor as the collection system. The robot manipulator grasps the metal plate and reposes it for the camera to collect data from different viewpoints. The proposed NBV/NBC planning method finds the optimal robot configuration for observation, considering the confidence of the learned complete point cloud. The difficulties in the NBV/NBC estimation of such thin metal plates are deficiencies caused by: (1) Reflection and noisy point cloud data; (2) Erroneous normal estimation; (3) Self-occlusion. The approach proposed in this work utilizes a PCN trained using synthesized data as the heuristic for NBV/NBC optimization. It is robust to the deficiencies mentioned above and is thus more advantageous than state-of-the-art methods when applied to metal plates.

We evaluate the system using simulated and real-world tasks in the experimental section. The results show that the proposed method can find NBV/NBC solutions more accurately and efficiently compared to baseline methods. The trained PCN had satisfying generalizability. The optimization algorithms were thus flexible and applicable to various metal plates without a significant change.

This section's organization is as follows. Section 4.2 reviews related work. Section 4.3 presents a schematic overview of the proposed method. Section 4.4∼4.5 shows algorithmic details of each essential component of the proposed system. Section 5.5 presents experiments and analysis. Conclusions and future work are drawn and discussed in Section 4.7.

4.2 Related Work

4.2.1 Next Best View

Early research on NBV can be traced back to [118], which determines good "next best views" to cover a given model. Wong et al. [119] studied a volumetric representation of the model after a 2D view was obtained to complete the NBV reconstruction task. Mendoza et al. [120] proposed a supervised learning-based scheme for NBV planning in an end-to-end manner, which predicts the sensor pose from predefined candidates. The above approaches represent searching space by sampling viewpoints over a cylinder or sphere model. They are in a low-dimensional space and have difficulties collecting every corner of an object with complex geometry. To overcome the difficulties, researchers have proposed using sensors mounted on robot arms [116][121], or objects grasped by robot arms [117][122] to capture the surfaces not visible in low dimensional searching space. Researchers have explored using prior knowledge to improve the estimation efficiency of NBVs. For example, Kay et al. [123] considered the importance of objects in the environment using semantic segmentation to inform UAV navigation. Wu et al. [124] leveraged the prior knowledge of plant structure to improve the effectiveness of the NBV algorithm for plant phenotyping.

The existed works are generally based on surface boundary [116][117] or volumetric analysis [123][124][125]. However, these approaches have their respective limitations when applied to curved, thin metal plates. Surface boundary-based approaches can be misled by the plates' surface-like behavior and inherent boundaries. They also struggle with noisy or sparse point cloud data, where reliable surface normal estimation is difficult. On the other hand, volumetric analysis discretizes the 3D space into a grid and thus can not precisely represent fine details. This limitation is significant in our context, where the task is highly sensitive to the surface's normal direction.

The system in this work comprises a robot manipulator and an external stationary depth sensor. We develop a complete shape learning-based NBV/NBC planning method to help determine the next camera view point or robot configuration for observation. The method is more advantageous than previous ones as it is not limited to local information captured by the sensors and is suitable for objects with uncertain features.

4.2.2 3D Shape Completion

The goal of 3D shape completion is to generate a complete 3D model from a partial observation. It plays an important role in robotic tasks [124][126]. Previously, Choy et al. [127] proposed an amortized maximum likelihood approach for 3D shape completion. Stutz et al. [128] learned a mapping from partial observations to shape primitives using extensive training data collection. Rock et al. [129] proposed an exemplar-based approach that retrieves a similar model from a training set based on an input depth image and deforms the retrieved model to approximate the input better. Litany et al. [130] used a variational autoencoder with graph convolutional operations to learn a latent space for complete shapes. They perform optimization in the latent space to iteratively deform the shape and align it with a partial input. Yuan et al. [131] presented a point cloud completion model that used a PointNet-based encoder to extract global features and folding operations to up-sample coarse predictions. Like PCN, Tchapmi [132] proposed the TopNet, which essentially employed a tree-structured decoder to predict complete shapes.

The problem in this work requires iteratively collecting and predicting partial point clouds. Consequently, we use the point cloud-based 3D shape completion to predict complete point cloud data and then use the predicted data to estimate NBV/NBCs parameters heuristically. We primarily use the PCN model [131] for the iterative shape completion and prediction. The model allows our proposed method to effectively avoid issues related to local features and identify the optimal viewpoint from a global perspective. It is robust to deficiencies caused by reflection and noisy point cloud data, erroneous normal estimation, self-occlusion, etc., and is thus more advantageous than state-of-the-art methods.

4.3 Schematic View of the Proposed Method

Fig. 4.2 presents the schematic diagram of the proposed method. The workflow starts with a deformed metal plate held in a robot's hand. An external stationary 3D scanner will capture a partial view of the point cloud and send it to the "Point Completion Network" for prediction. The partial view is highlighted as the "Partial Point Cloud" blue box in the diagram. The "Point Completion Network" will predict a complete point cloud, as denoted by the "Complete Point Cloud" green box in the diagram. It will be combined with the "Partial Point Cloud" and fed to the "NBV/NBC Optimization" component to estimate the confidence of the point cloud and the next robot configuration for holding the target object. Note that we used "NBV/NBC Optimization" as the title since the developed approach applies to the NBV problem and can also be used for solving the NBC of a robot to move an object for the next best view. The NBV and NBC problems share essentially the same formulation, except that additional robotic constraints must be included, and a few optimization parameters must be replaced.

If the combined point cloud is confident enough, the "NBV/NBC Optimization" component will produce a signal to stop the routine, switch to the "Mesh Reconstruction" module, and build a mesh model using the received point cloud as the final output. Vice verse, the workflow will be directed to the "Robot Execution" block to move the object to the next best view pose and then restart the capturing and prediction routine.

Details of the "Point Completion Network" and "NBV/NBC Optimization" will be respectively presented in the following sections. The "Mesh Reconstruction" is not the primary concern. It will only be briefly mentioned in the experimental section.

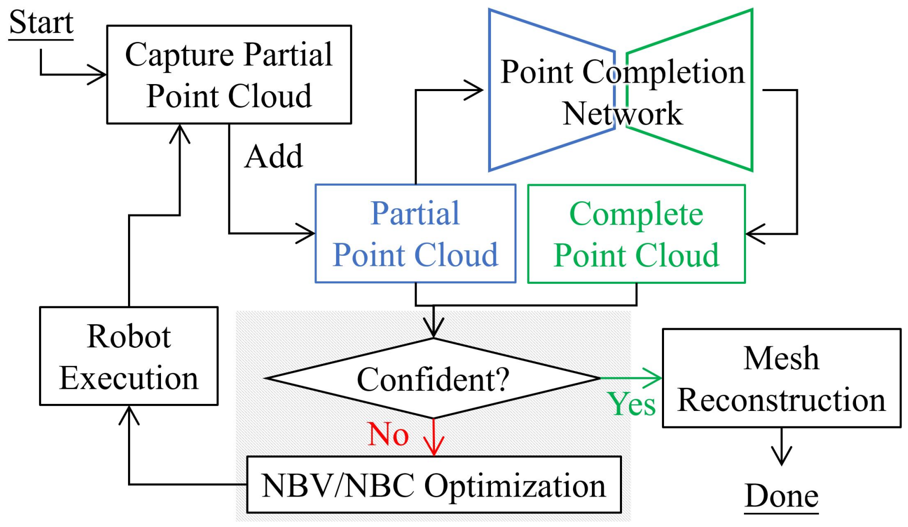


Figure 4.2: Schematic diagram of the proposed method. The blue “Partial Point Cloud” represents a single-view point cloud in the beginning and a multi-view point cloud starting from the second loop. We train a point completion neural network using both synthesized single-view and multi-view point clouds to predict the NBV/NBCs.

4.4 Training the Point Cloud Completion Network

We use the Point Completion Network (PCN) proposed in [131] to learn the complete geometry. PCN is an encoder-decoder network. It encodes the input point cloud as a single global feature vector and reconstructs the completed point cloud by a FoldingNet [133] decoder. The algorithmic details can be referred to [131][134]. In this section, we focus on the training process and present how we prepare the data needed to train a PCN for metal plate objects.

We use synthetic shapes to create a large-scale dataset. The dataset contains single and multi-view partial point clouds and their corresponding complete ground-truth. It is used to train the PCN model. Particularly, the synthetic shapes are created using flat plates, as shown in Fig. 4.3. We randomize the w, h, and l values of the plates and use B-spline functions to transform the randomized plates into curved shapes. Then, we use the Poisson disk sampling method [135] to sample the surfaces and create ground-truth point clouds[3]. We capture partial views of the synthesized

[3]The thickness h is kept when creating partial-view point clouds but ignored when creating the

curved shapes and include occlusions, noises, and loss to the views to create the partial point clouds. The partial point clouds approximate the data collected by sensors. We also align several partial point clouds of a single object with random transitional and rotational errors to generate multi-view partial point cloud data. The multi-view partial point cloud data approximate the point clouds collected after a sequence of views.

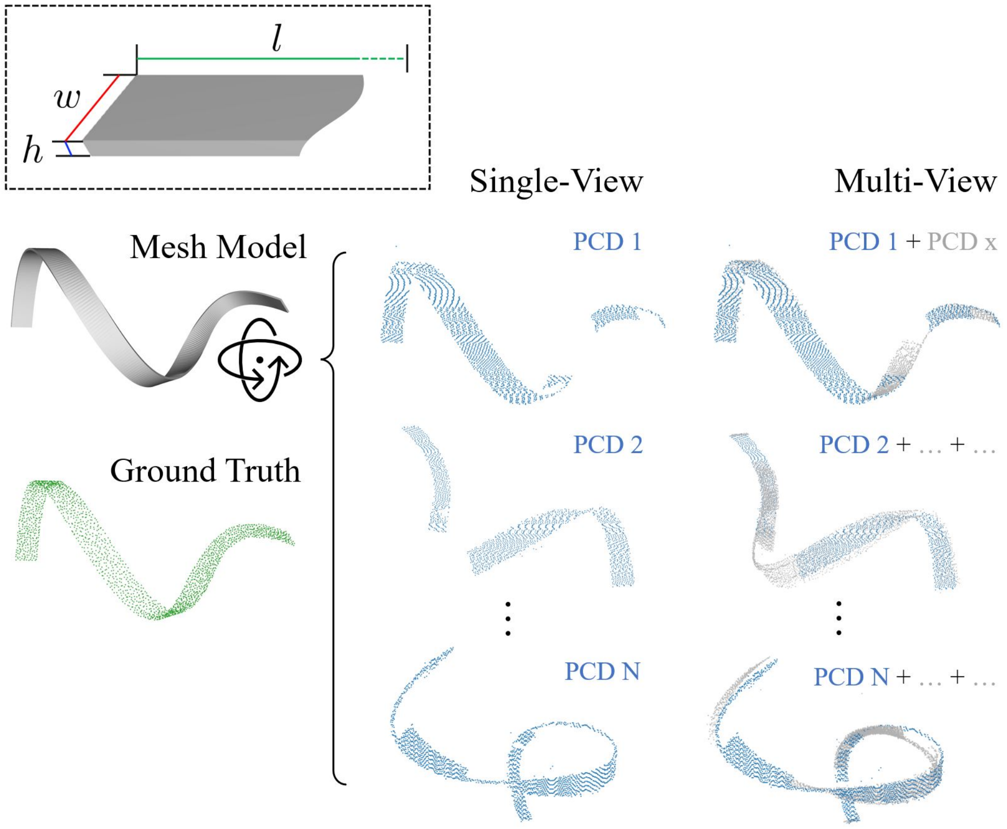


Figure 4.3: Synthetic datasets created based on flat plates. We randomize w, h, and l values to generate a variety of flat bases and then transform the bases into curved shapes using B-splines. The ground truth is obtained by sampling the surface of the curved shapes. Algorithmic details for creating single and multi-view partial point clouds are shown in Fig. 4.4.

ground-truth data. The goal is to let the PCN model predict a complete point cloud that is thin and irrelevant to thickness. Each region has a distinct normal direction, which helps the planner find an optimal solution while staying unaffected by complex local geometry.

Fig. 4.4 illustrates the routine for creating partial-view point clouds. First, we use a hyperparameter $\bar{\psi}$ to determine visibility. Points are considered visible if the angle between the surface normal of the points and the line of sight is smaller than $\bar{\psi}$. Fig. 4.4(b.iii) exemplifies two partial point clouds generated using different $\bar{\psi}$. Higher $\bar{\psi}$ values will result in more visible points. Also, since real point clouds usually have non-uniform sampling density, noises, outliers, missing data, and misalignment [136], we use the process shown in Fig. 4.4(c.i-c.iii) to include these disturbances and reduce the sim-to-real gap. The processes include (1) removing points randomly to approximate missing data, (2) adding random noisy points, and (3) applying random Gaussian noises to existing points.

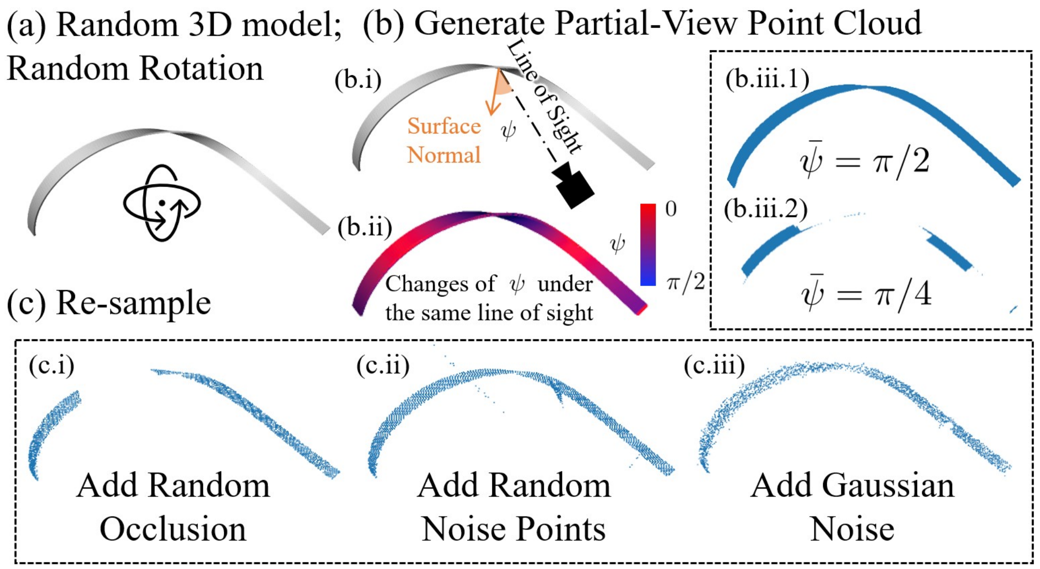


Figure 4.4: (a) Randomly select a curved shape with a random pose. (b) Generate raw partial-view point clouds considering the camera's line of sight, surface normal, and a threshold angle $\bar{\psi}$. (b.i) Definition of ψ (Angle between the line of sight and surface normal) (b.ii) Changes of ψ under the same line of sight. (b.iii.1,2) Results under different $\bar{\psi}$ values. (c) Re-sample the raw clouds by including occlusions and noises.

We use both the single and multi-view partial point clouds and their corresponding ground-truth point clouds as the input and output for training the PCN. The Earth Mover's Distance (EMD) [137] is used as the loss during the training. The definition of EMD is as follows:

$$\mathrm{EMD}(\mathrm{S}, \mathrm{S}^*) = \min_{\phi:\mathrm{S}\to\mathrm{S}^*} \frac{1}{|\mathrm{S}|} \sum_{x\in\mathrm{S}} \|x - \phi(x)\|_2 \,. \tag{4.1}$$

Here, S is the input point cloud. S^* is the predicted complete point cloud. The definition finds a bijection $\phi : \mathrm{S} \to \mathrm{S}^*$ that minimizes the average distance between corresponding points of the two point clouds.

4.5 NBV And NBC Optimization

This section presents the algorithmic details of the "NBV/NBC Optimization" component. The algorithms comprise two steps introduced in the following subsections. The first step compares the partial-view point cloud with the PCN output and prepares a list of candidate regions for further observation. The second step determines the NBV/NBC using optimization.

4.5.1 Confidence Estimation

We estimate the confidence regions using the distance between the input partial-view point cloud and the predicted complete result. We use S to represent a partial-view point cloud and S^* to represent a completed result. S^* is downsampled to n regions. They are denoted by the center points: $\mathcal{M} = \{\boldsymbol{p}_{\mu_0}, \boldsymbol{p}_{\mu_1}, \ldots, \boldsymbol{p}_{\mu_n}\}$. For each center point $\boldsymbol{p}_{\mu_i}$, we search for its neighbors in a denoised S set (where points that are far from S^* are removed) within a specified radius r and count the number of neighbors. The number of neighbours is denoted as $u(\boldsymbol{p}_{\mu_i})$. The number of neighbours for all points is denoted as $\mathcal{U} = \{u(\boldsymbol{p}_{\mu_0}), u(\boldsymbol{p}_{\mu_1}), \ldots, u(\boldsymbol{p}_{\mu_n})\}$. The confidence of a region (represented by its center $\boldsymbol{p}_{\mu_i}$) is computed by normalizing against $\mathcal{U}$ using:

$$c(\boldsymbol{p}_{\mu_i}) = \frac{u(\boldsymbol{p}_{\mu_i}) - \min\mathcal{U}}{\max\mathcal{U} - \min\mathcal{U} + 1}, 0 \le i \le n. \tag{4.2}$$

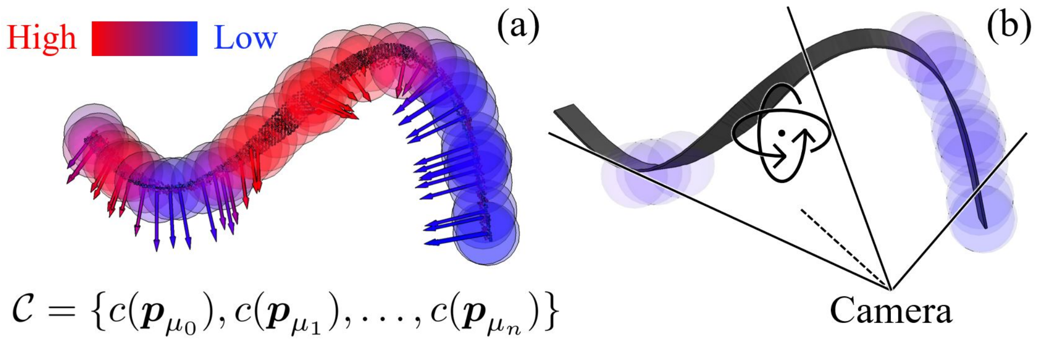

High
Low
(a)
(b)
$\mathcal{C} = \{c(\boldsymbol{p}_{\mu_0}), c(\boldsymbol{p}_{\mu_1}), \ldots, c(\boldsymbol{p}_{\mu_n})\}$
Camera

Figure 4.5: NBV example. (a) Confidence in color. (b) Optimize the rotation and transition of the object to capture more low-confidence regions on the reconstructed mesh model Ω.

The $c(\boldsymbol{p}_{\mu_i})$ on the left hand side denotes the confidence of $\boldsymbol{p}_{\mu_i}$. It is computed by normalizing the number of each center point's neighbors. A greater number of neighbors correlates to higher confidence values. $\mathcal{C} = \{c(\boldsymbol{p}_{\mu_0}), c(\boldsymbol{p}_{\mu_1}), \ldots, c(\boldsymbol{p}_{\mu_n})\}$ is a set formed by all confidences.

Fig. 4.5(a) visualizes the confidence using a random plate as an example. A redder color in the illustration indicates a region that has a larger confidence score. Inversely, a bluer color indicates a region with a smaller confidence score. The normals $\{\boldsymbol{n}(\boldsymbol{p}_{\mu_0}), \boldsymbol{n}(\boldsymbol{p}_{\mu_1}), \ldots, \boldsymbol{n}(\boldsymbol{p}_{\mu_n})\}$ of these regions are denoted by arrows. They are computed by applying Principal Component Analysis (PCA) to the queried neighbors.

4.5.2 Formulation of the Optimization Problem

Note that, the NBV and NBC optimization are treated as two distinct processes. Both of them are introduced to show that the proposed method is adept in both scenarios: where the robot holds the camera or when the robot holds the object.

NBV

Based on the confidence acquired in the previous subsection, we formulate the NBV estimation as the following optimization problem.

$$\begin{aligned}
\max_{\boldsymbol{x}} \quad & Gain(\mathcal{M}, \mathcal{C}, \tilde{\mathrm{S}}) && (4.3a)\\
\text{s.t.} \quad & \boldsymbol{x} = (x, y, z, \varphi_x, \varphi_y, \varphi_z) && (4.3b)\\
& \mathrm{S}^* \cap \mathtt{fov}(\boldsymbol{x}) \neq \emptyset && (4.3c)\\
& \varphi_x, \varphi_y, \varphi_z \in [-\pi, \pi] && (4.3d)\\
& \tilde{\mathrm{S}} = f(\boldsymbol{x}, \Omega) && (4.3e)
\end{aligned}$$

The goal of the optimization is to find a viewpoint that leads to a larger gain in confidence. The following constraints are considered in the formulation:

1. Camera Field of View (FOV): The object should stay in the camera's FOV.
2. Occlusion: Both self-occlusion and robot occlusion should be considered.
3. Visibility: The surface is deemed well captured only when the angle between the sensor's line of sight and the surface normal is below a certain threshold value.

The above equation (4.3c) corresponds to item 1). Equation (4.3e) corresponds to item 2). $\boldsymbol{x}$ is a vector that represents the NBV, with its six elements denoting translation and rotation, respectively. The function $\mathtt{fov}(\cdot)$ represents the camera's viewing frustum for a given pose. $f(\cdot)$ is a function that captures the partial-view point cloud of a given mesh model using the ray-casting method. With a given viewpoint, a set of rays are projected towards the mesh models visible within the viewing frustum. Each of these rays undergoes an intersection test with the mesh. If an intersection occurs, the initial point of intersection is recorded. Its result is assigned to $\tilde{\mathrm{S}}$. Ω is a mesh model reconstructed using S^*.

The optimization goal $Gain(\cdot)$ is defined by the pseudo-code shown in Algorithm 3. The algorithm's input includes the set of region centers, the set of confidence scores for the regions, a new partial-view point cloud captured using equation (4.3e), and a user-defined threshold confidence value for filtering regions. The algorithm checks the visibility of each region (corresponds to item 3)). The smaller angle between the surface normal and the line of sight means that the surface is largely facing towards the viewpoint. Recognizing the inherent ambiguity in the directionality of a surface's normal–owing to it having two potential normal directions that are, for our purpose, functionally equivalent–we conceptualize the normal more as a line rather than a vector with a specific direction. Thus if the acute angle between $\boldsymbol{n}(\boldsymbol{p}_{\text{query}})$ and the line of sight ($\overrightarrow{\boldsymbol{p}_{\text{cam}}\boldsymbol{p}_{\text{query}}}$) is within a range of $[0, \bar{\psi}]$, the point $\boldsymbol{p}_{\text{query}}$ is considered to be successfully captured (lines 9-10). The output value g sums up the gained confidence when a new $\boldsymbol{p}_{\text{query}}$ is captured. Visible regions with lower confidence score $c(\boldsymbol{p}_{\mu i})$ contribute to higher gain. It thus helps find an $\boldsymbol{x}$ that brings more unseen object regions into the camera view.

Algorithm 3: Computing the Gain in Coverage

```
  Input: $\mathcal{M}$, a set of region centers;
         $\mathcal{C}$, a set of confidence scores for the regions;
         $\tilde{\text{S}}$, a newly captured partial-view point cloud;
  Output: $g$, total gain in $\tilde{\text{S}}$'s coverage
1 begin
2     $g \leftarrow 0$
3     for $i \in$ (1,2,...,n) do
4         if $c(\boldsymbol{p}_{\mu_i}) > \epsilon$ then
5             continue // $\epsilon$ is a given threshold
6         $\mathcal{Q} \leftarrow$ Query for the neighbors of $\boldsymbol{p}_{\mu_i}$ from $\tilde{\text{S}}$ within a given radius $r$
7         for $\boldsymbol{p}_{\text{query}} \in \mathcal{Q}$ do
8             $\sigma \leftarrow \texttt{acos}(\frac{|\overrightarrow{\boldsymbol{p}_{\text{cam}}\boldsymbol{p}_{\text{query}}} \cdot \boldsymbol{n}(\boldsymbol{p}_{\text{query}})|}{||\overrightarrow{\boldsymbol{p}_{\text{cam}}\boldsymbol{p}_{\text{query}}}|| \cdot ||\boldsymbol{n}(\boldsymbol{p}_{\text{query}})||})$
9             if $\sigma \in [0, \bar{\psi}]$ then
10                $g \leftarrow g + (1 - c(\boldsymbol{p}_{\mu_i}))$
11    return $g$
```

NBC

Next, by replacing the optimization parameters with joint angles $\boldsymbol{q}$, the proposed optimization method can also help solve the next best robot configuration problem:

$$\max_{\boldsymbol{q}} \quad Gain(\mathcal{M}, \mathcal{C}, \tilde{\text{S}}) \tag{4.4a}$$

$$\text{s.t.} \quad \boldsymbol{q} = [q_0, q_1, ..., q_d]^T \tag{4.4b}$$

$$q_k \in [\theta_k^-, \theta_k^+], k \in \{0, 1, ..., d\} \tag{4.4c}$$

$$\boldsymbol{p}_h, \boldsymbol{R}_h = \texttt{fk}(\boldsymbol{q}) \tag{4.4d}$$

$$\mathrm{S}^* \cap \texttt{fov}([\boldsymbol{p}_{\text{cam}}, \boldsymbol{R}_{\text{cam}}]) \neq \emptyset \tag{4.4e}$$

$$\tilde{\mathrm{S}} = f([\boldsymbol{p}_{\text{cam}}, \boldsymbol{R}_{\text{cam}}], \texttt{rot}([\boldsymbol{p}_h, \boldsymbol{R}_h], \Omega) \cap \Omega_{\text{r}}) \setminus \Omega_{\text{r}} \tag{4.4f}$$

Here, d is the number of arm joints; θ_k^- and θ_k^+ are joint angle limits. $\boldsymbol{p}_h$ and $\boldsymbol{R}_h$ are the position and rotation of the robot hand. $\boldsymbol{p}_{\text{cam}}$ and $\boldsymbol{R}_{\text{cam}}$ are constant and represent the camera pose. $\texttt{rot}(\boldsymbol{a}, \boldsymbol{b})$ is a function that transforms $\boldsymbol{b}$ using $\boldsymbol{a}$. Ω_{r} is the mesh model of the robot. Thus equation (4.4f) additionally considers the constraint for robot occlusions.

4.5.3 Optimization Solver and Termination Criteria

We use the Constrained Optimization BY Linear Approximation (COBYLA) solver to solve the optimization problems. The solver is typically used when a problem is non-smooth or the gradient of the objective function is not known. The targeted tasks exactly meet the conditions of the COBYLA solver. The optimization problems will be solved repeatedly each time a newly captured partial view is included until a termination criterion is met. Defining termination is difficult since there is no ground truth. We thus used a threshold: If the minimum value of set $\mathcal{U}$ is larger than a threshold value, the input point cloud is considered complete. Otherwise, the algorithm proceeds to the next iteration.

4.6 Experiments and Analysis

In this section, we present the experiments used to examine the performance of the proposed method. The experimental platform comprised a robot arm (Ufactory xArm 7) with a two-finger parallel gripper and a stationary depth sensor (Photoneo PhoXi 3D Scanner M with 1032×772 resolution) fixed above it. We used a PC with an Intel Core i7-10700 CPU, RTX 3090 GPU, and 16 GB memory to carry out the learning

Table 4.1: Information of the Dataset

Type	# Curve Shapes	# Point Clouds	Coverage			
			max.	min.	mean	std.
Single-View	5k	50k	0.96	0.05	0.58	0.15
Multi-View	5k	50k	1.0	0.15	0.87	0.10

* Meanings of abbreviations: # Curved Shapes - Number of randomly generated curved shapes; # Point Clouds - Number of point clouds for respective single or multiple partial views.

and optimization processes. We performed quantitative studies and analysis in both simulation and real-world using the robot and PC platform. Two types of metal plates are evaluated: Curve Flat Plates (CFP) and Curved Linear Array Plates (CLAP). Their respective single or multi-view partial point clouds are denoted as P-CFP and P-CLAP.

4.6.1 Point Cloud Completion Network

First, we evaluated the performance of the PCN trained using point cloud data created by the proposed method. We first generated 5000 flat plates with $w \times l \times h$ randomized from a range of $(200\pm50)\times(10\pm5)\times(2\pm1)$ [mm], and then transformed them into curved shapes using B-spline with degrees from 3 to 5. The point cloud data used for training was sampled from the 5000 synthesized curved shapes (10 partial point clouds for each shape). Detailed information about the point cloud data is shown in Table 4.1.

We used the P-CFP created using the methods shown in Section 4.4 and partial point clouds created based on a new type of linear array plates to test the performance of the trained PCN. The new linear array plates were essentially a sequence of repeated geometric elements. Notably, we used the two geometric elements shown in Fig. 4.6 and respectively morphed them into curved shapes using RBF morphing [138]. The control points for the RBF morphing of the repeated geometric elements are randomly moved in a range of 20 [mm]. Although the linear array plates remain thin and long

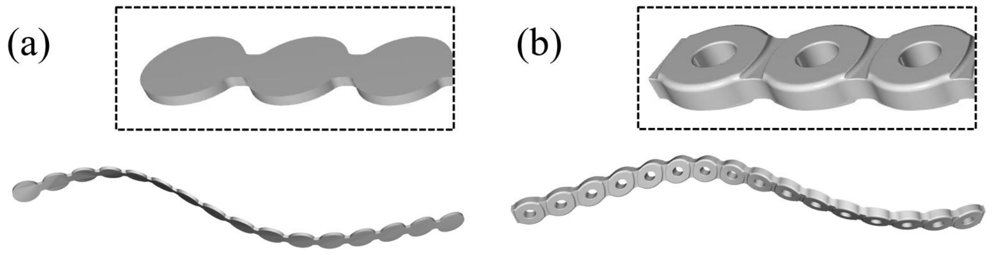


Figure 4.6: Two Curved Linear Array Plates (CLAPs). (a) CLAP created using circular elements. (b) CLAP created using ring elements.

like the flat ones, they had complicated local features, and the point clouds created based on them were significantly different. Such point clouds help understand if the trained model was generalizable to point clouds with complicated variations.

Table 4.2 presents the testing results. We used the P-CFP data for training and tested the trained model using both single-view and multi-view P-CFP and P-CLAP data. In order to better compare the performance difference, we used the Chamfer Distance (CD) [139] and Earth Mover's Distance (EMD) [137] to quantify the testing results. The CD value represents the level of coincidence between the points in the predicted cloud and those in the ground truth cloud. A lower CD value indicates a higher degree of coincidence. On the other hand, the EMD value reflects the completeness of the predicted point clouds when compared to the ground truth. A lower EMD value indicates greater coverage of the ground truth clouds. The results in the table show that the trained model had similar performance on either the P-CFP or P-CLAP data. It had satisfying generalizability and was applicable to point clouds created using curved linear array plates. The results also show that the accuracy of the prediction results for multi-view point clouds was higher than single-view ones, which implies that higher surface coverage leads to better estimation results, and the negative influence of registration errors was insignificant in the presence of high surface coverage.

Some examples of the results are visualized in Fig. 4.7. Interested readers may examine it for a detailed comparison.

Table 4.2: Performance of the Trained PCN Model

Training Data	Type	Testing Data P-CFP CD	Testing Data P-CFP EMD	Testing Data P-CLAP CD	Testing Data P-CLAP EMD
P-CFP (35k+35k)	Single-View	2.62	6.78	2.12	7.53
	Multi-View	1.75	4.02	1.81	6.69

* The digits in the parenthesis of the training data column indicate the number of single-view and multi-view point clouds, respectively. A lower CD value indicates a higher degree of coincidence between the points in the predicted point clouds and those in the ground truth. A lower EMD value indicates a more excellent coverage on the ground truth clouds.

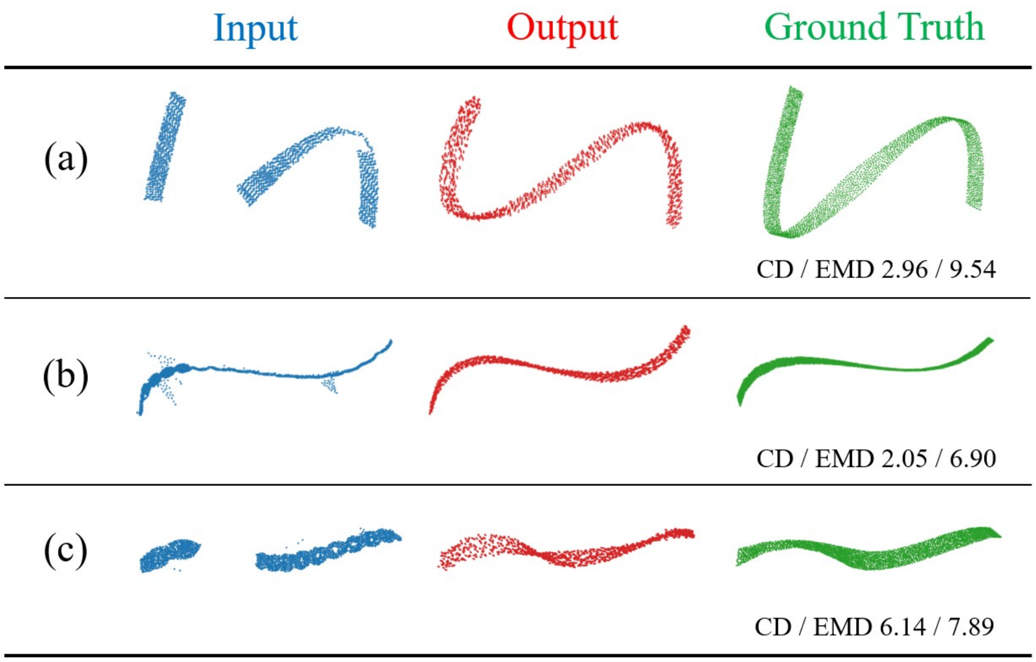


Figure 4.7: Visualization of some PCN outputs. (a) A P-CFP example. (b, c) Two P-CLAP examples.

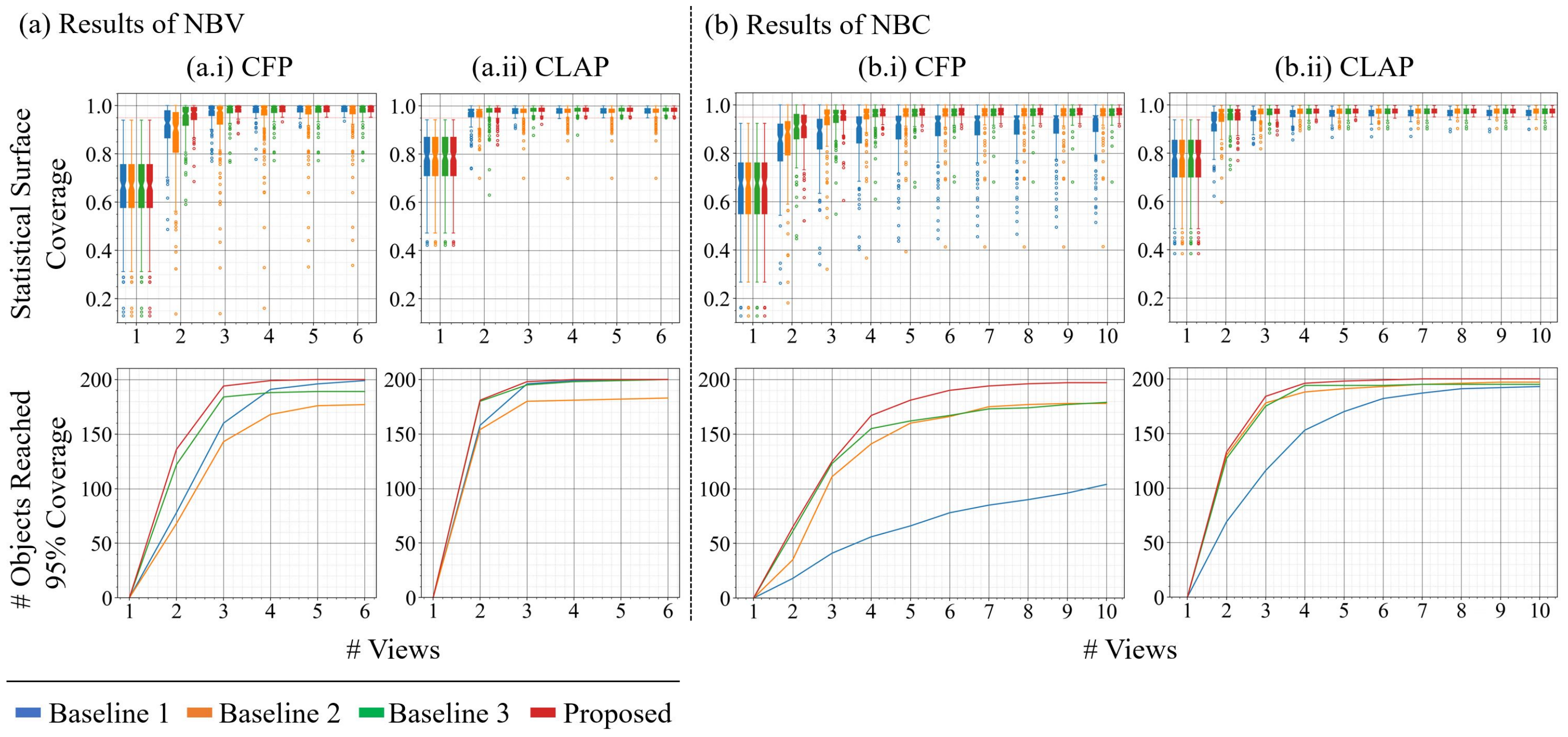


Figure 4.8: Comparisons between our proposed method and baseline methods. (a) Results of NBV. (b) Results of NBC. The first row of diagrams: Changes in statistical surface coverage (Vertical Axis) as the number of views (Horizontal Axis) increased. The second row of diagrams: Number of objects reached 95% coverage (Vertical Axis) as the number of views (Horizontal Axis).

4.6.2 Simulation Results

Second, we evaluated the effectiveness of the proposed NBV/NBC optimization method and compared it with several baseline methods in simulation. The target objects used in the evaluation and comparison included 200 random CFP shapes and 200 random CLAP shapes, respectively. The PCN model trained in the previous subsection was used in the evaluation. Two indicators, namely the average number of captures and success rates, were used to quantify the effectiveness values. The average number of captures indicates the number of views needed to achieve surface coverage above a particular threshold value. It is computed by summing up the number of captures across all trials and then dividing the sum by the total number of trials conducted. The success rates indicate how well an object can be adequately observed (with surface coverage larger than a particular threshold value) within a pre-defined number of views (6 for NBV and 10 for NBC).

NBV

We compared the following methods to understand the advantages and disadvantages of the proposed one:

- Baseline Method 1: Randomize a matrix by sampling rotations around the vector of a level-1 icosphere.
- Baseline Method 2: Local dimensional information-based method developed by Kobayashi et al. [117].
- Baseline Method 3: Use the normal of the point $\boldsymbol{p}_{\mu_i}$ with the minimum confidence score as the viewing direction, and calculate the rotation matrix by aligning the NBV and camera's line of sight.
- Proposed Method: The proposed direct optimization-based method.

Fig. 4.8(a.i-ii) illustrate the results of these methods. The diagrams in the first row of the figure show the statistical change in surface coverage as the views increased. Each box represents a summary of all 200 trials. When employing prior

Table 4.3: Average Number of Captures and Success Rate of NBV Estimation Algorithms.

	(a) CFP		(b-c) CLAP	
	# Captures	Success Rate	# Captures	Success Rate
Baseline 1	2.88	199/200	2.24	**200/200**
Baseline 2	3.23	177/200	2.52	183/200
Baseline 3	2.59	189/200	2.14	**200/200**
Proposed	**2.36**	**200/200**	**2.11**	**200/200**

* Meanings of abbreviations: # Captures - Average number of captures; Success Rate - Number of trials that reach surface coverage of more than 0.95 within 6 times of captures / Total number of trials.

knowledge (as indicated by the green and red boxes), the average increase in surface coverage shows the most substantial improvement in the second view. Notably, the optimization-based method demonstrates the most significant increase among the approaches considered. The diagrams in the second row show the number of objects that were successfully observed (more than 95% coverage) as the view increased. A large portion was successfully observed within two views using the proposed method. It outperformed other methods for both CFP and CLAP. The previous method (Baseline 2) had wrong normal estimations and, thus, a minor increase in surface coverage. Baseline 3 considered the region with the lowest confidence but did not regard global optimization. It is improved compared with the previous method but remains worse than the optimization-based method. Detailed comparisons of these two methods can be found in a supplementary file published together with this manuscript. On average, CLAP tends to have fewer views than CFP. This could be attributed to the fact that CLAP shapes have more complex local geometry, and their surface normal can be more readily inferred. Table 4.3 shows the average number of views for all successfully observed objects and the success rates.

Table 4.4: Average Number of Captures and Success Rate of NBC Estimation Algorithms.

	(a) CFP		(b-c) CLAP	
	# Captures	Success Rate	# Captures	Success Rate
Baseline 1	7.34	104/200	3.60	193/200
Baseline 2	4.20	178/200	2.66	197/200
Baseline 3	3.89	179/200	2.61	195/200
Proposed	**3.43**	**197/200**	**2.45**	**200/200**

* Meanings of abbreviations: # Captures - Average number of captures; Success Rate - Number of trials that reach surface coverage of more than 0.95 within 10 times of captures / Total number of trials.

NBC

We also evaluated the four methods by additionally considering robot configurations. For the first three, we computed the NBC based on the predicted NBV using the method presented in [117]. In contrast, we did not have an intermediate NBV estimation process for the proposed method and optimized the robot joint values directly. The diagrams in Fig. 4.8(b.i-ii) show the results of the NBC methods. Table 4.4 shows the average number of views for all successfully observed objects and the success rates. We can see from the results that the proposed method outperformed the other methods. They confirmed a presupposition that more viewpoints were required to obtain NBCs as the participation of the robot caused occlusions. Baseline 1 which randomizes the viewpoint does not work due to these occlusions. The success rates are also lower than the NBV results, as robotic kinematic constraints caused additional failures.

4.6.3 Real-World Results

Third, we carried out experiments in the real world using the xArm 7 robot manipulator and five curved metal plates. Fig. 4.9 shows pictures of the five curved plates. They were randomly handed over to the robot by humans. Fig. 4.10 shows the results. The (a) to (e) rows in the figure correspond to the results for objects

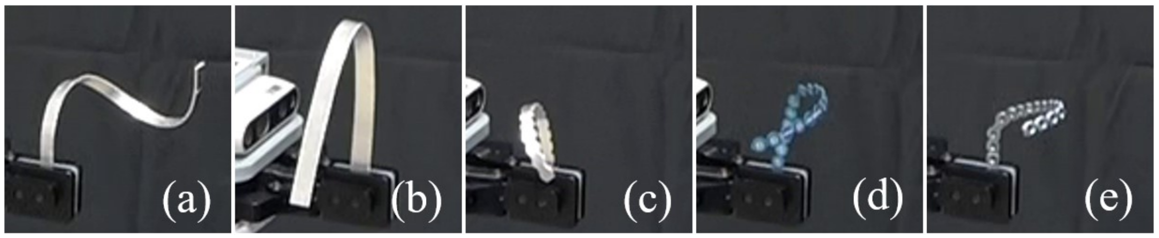


Figure 4.9: Five curved plates used in real-world experiments. (a) Aluminum CFP 1. (b) Aluminum CFP 2. (c) Aluminum CLAP with circular pattern. (d) Aluminum CLAP with circular pattern and blue coating. (e) Titanium CLAP with ring pattern.

Fig. 4.9(a)-(e). We can see from the results that the robots moved the plates to different poses and captured and stitched the partial views to finally reconstruct the 3D models. It took three views to reconstruct the 3D models of the objects in Fig. 4.9(a) and (c), and two views to reconstruct the 3D models of objects (b), (d), and (e). The registration of partial-view point clouds is done by the AR markers fixed on the robot hand. The results of the real-world experiments are coherent with the simulation. They demonstrate that the proposed method can quickly capture enough point cloud data and thus help efficiently obtain the object models. The method is robust to noisy point cloud data and low guidance in surface normal and is suitable for thin metal plates with either insignificant or complex features.

4.7 Conclusions

We proposed a point completion network-based NBV/NBC estimation method in the section. Experimental results show that the proposed method is applicable to various metal plates and can reach a high coverage with a small number of viewpoints. The method is robust to objects with noisy surface properties and uncertain surface normal. It can flexibly and robustly solve the NBV/NBC problems of thin metal plates.

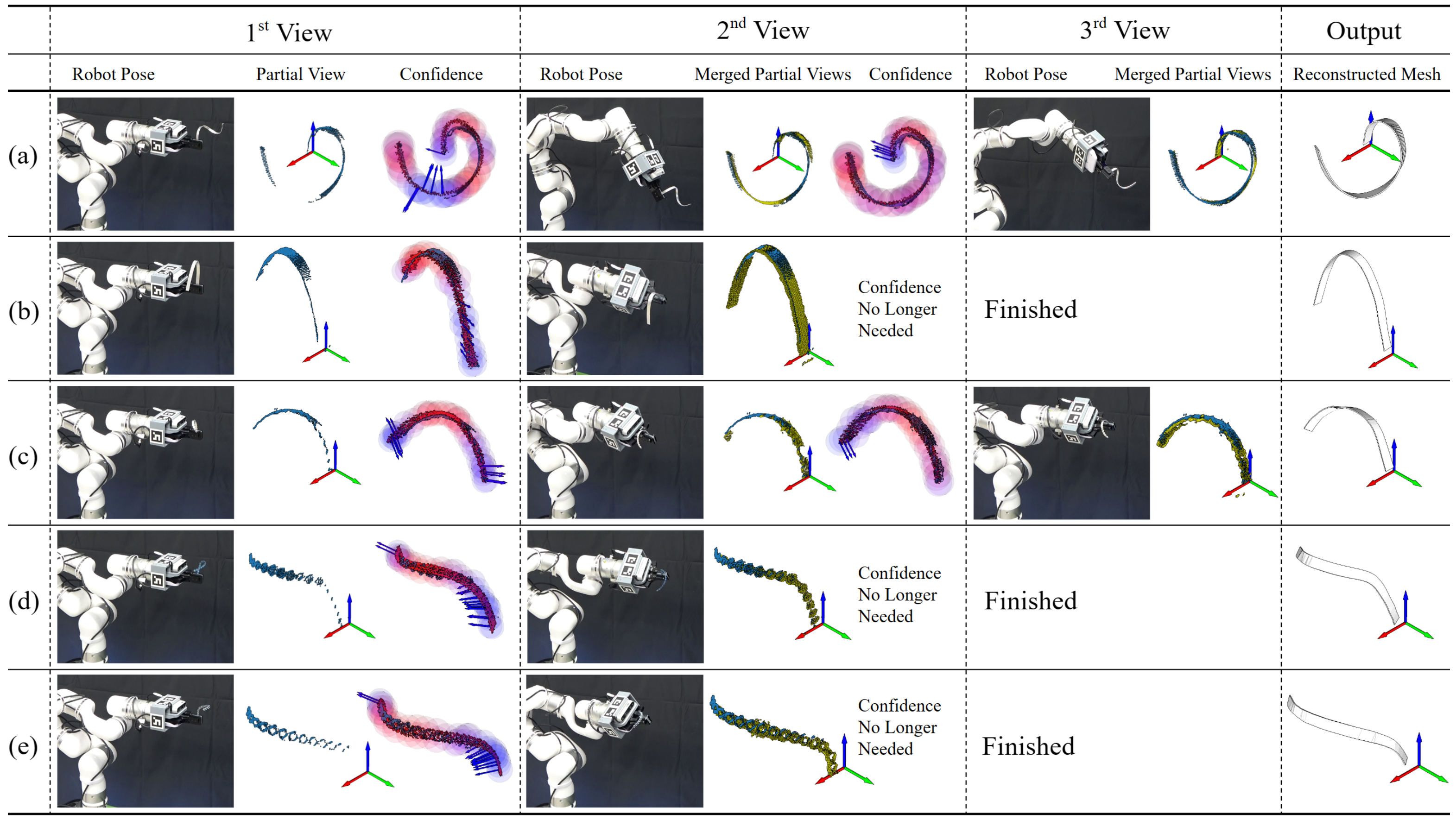


Figure 4.10: Results of real-world experiments. The (a) to (e) rows in the figure correspond to the results for objects Fig. 4.9(a)-(e). The robot captured three views of the objects in (a) and (c), and captured two views of the objects in (b), (d), and (e) to reconstruct their models. The reconstruction was carried out by assuming flat plates. Detailed local features were ignored in (c), (d), and (e).

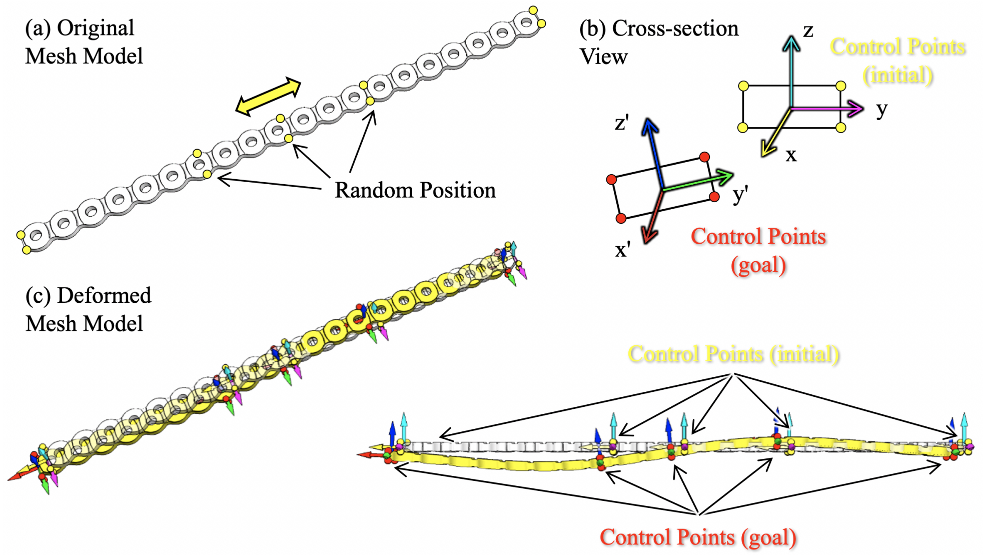


Figure 4.11: (a) Original mesh model. The control points (shown in yellow) are sampled on the original mesh model with random intersections on cross-sections that are perpendicular to the xy-surface and paralleled to the yz-surface. (b) Cross-section view of the change of the control points. (c) Deformed mesh model using RBF morphing.

Appendix

Curved Linear Array Plate Generation

Mesh morphing techniques aim to change the shape, size, and orientation of the mesh while preserving its properties and integrity as much as possible. Additionally, the update of node positions usually requires less computational effort compared with a full mesh regeneration. We adopt it to generate a large amount of curved linear array plates (CLAP). Specifically, Radial Basis Functions (RBF) morphing [138] is used. It comes with two distinctive advantages that make it very flexible: it is mesh independent and it allows node-wise precision [140].

Here we introduce the implementation details. With a given mesh model, we first define control points on it. These points are sampled on cross-sections perpendicular to the xy-plane and parallel to the yz-plane, with four points taken from each cross-section. Eight control points are sampled on cross-sections of the two ends of the plates. The position of control points on the cross-sections in-between are randomized. Fig. 4.11 (a) presents these control points, marked in yellow. Next, we establish the goal positions of these control points. Each group of four points on a cross-section is treated as a group, where their translation is restricted to a maximum of 20 mm. In addition, the rotations about the x, y, and z-axis are constrained to be less than 30°, 30°, and 5° respectively. These values were carefully chosen to seek a balance between allowing significant deformation and preventing unrealistic outcomes. Overly large values could distort the shape beyond practical limits and make the model unrealistic. Fig. 4.11 (b) shows the goal position of the control points (red). An exemplary comparison of the mesh model before and after morphing is illustrated in (c).

Comparison with Previous Method

This work is an extension of our previous work [117]. It was inspired by the following limitations of the previous method:

- **Surface Normal Estimation:** The previous method was unsuitable for thin objects due to incorrect estimations of surface normals on thin metal plates. These incorrect estimations were often caused by noisy input or insignificant local information. In this context, "insignificant" refers to scenarios where only the edge of the plate is captured. In that case, the obtained point clouds form a 2D curve instead of a 3D surface. Estimating the normal of the 2D curve is challenging, as it provides insufficient 3D information.

- **Noise Reduction:** Our previous work lacked an efficient method to remove outliers from the point cloud. Lacking such a method is particularly problematic for metal plates since the reflective metal materials will cause the time-of-flight (ToF) sensor to output a very noisy point cloud. Furthermore, the point cloud of metal plates is relatively discrete compared to general objects, particularly in the case of Curved Linear Array Plates (CLAPs). As a result, the outliers of these point clouds are difficult to remove using conventional clustering-based methods.

- **Termination Condition:** The previous method defined the object as well-captured using a confidence value computed by a dimension feature [141]. In this approach, continuous surfaces will have higher confidence, while non-continuous sections will have lower confidence. It is unsuitable for thin plate tasks as the objects have dominant non-continuous sections (edges).

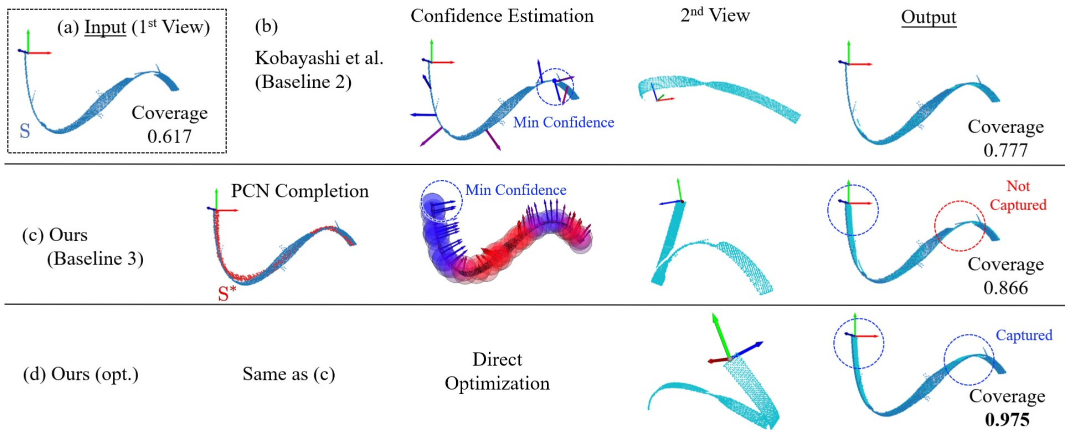


Figure 4.12: Performance of various NBV methods. (a) Input partial-view point cloud (first capture). (b) Baseline 2 (previous method) has wrong normal estimation. (c) Baseline 3 uses the region with the lowest confidence for NBV. The newly captured point cloud can cover the specific region but fails to cover other low-confidence regions. (d) The proposed optimization-based method can cover more low-confidence regions. It achieves the highest improvement of surface coverage after the second scan.

Quantitative comparisons of the new method and the previous one are presented in Section 4.6.2 of the main manuscript. Here we showcase several qualitative examples in Fig. 4.12. We hope readers can get an intuitive view of performance through the results.

Additionally, the real-world results can be observed in the supplementary video. Fig. 4.13 shows the details of the results. Both algorithms use the same point cloud shown in Fig. 4.13(a) as input. Fig. 4.13(b) shows the NBV/NBC estimation results of the previous method and visualizes the point clouds after 3 time of scanning. The result remains incomplete. Fig. 4.13(c) is the result of the proposed method.

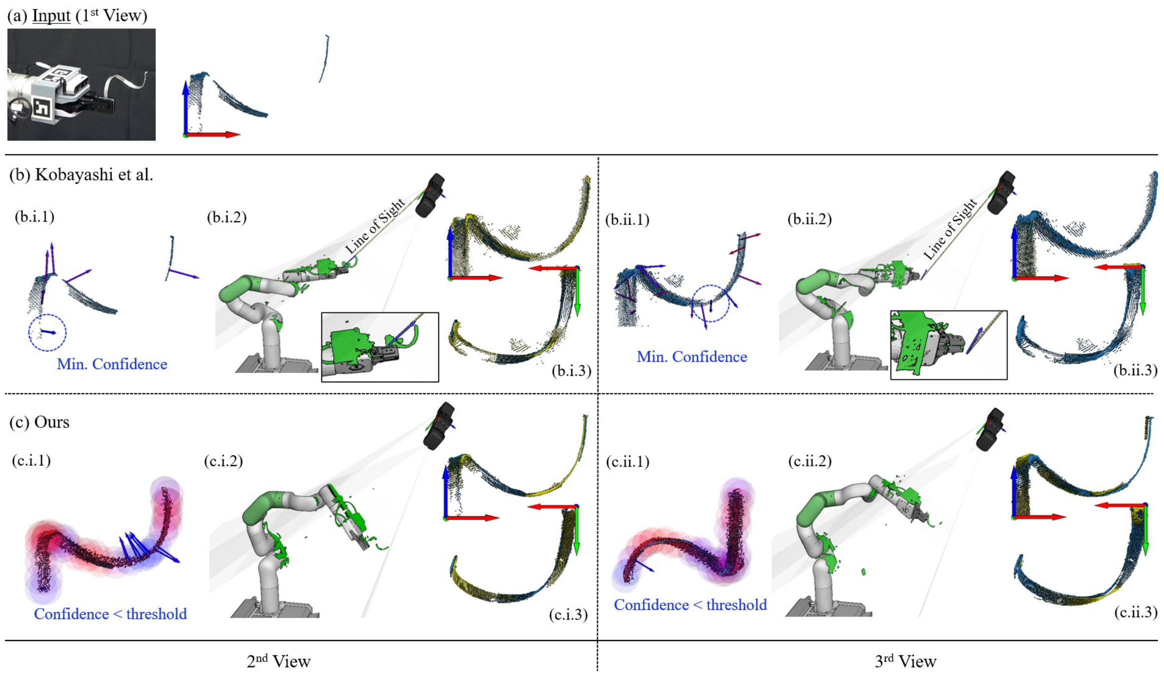


Figure 4.13: Comparison of real-world NBC estimations. (a) Input partial-view point cloud (first capture). (b) Previous method. The optimization goal is to make the angle between the normal direction of the point with minimum confidence and the camera line of sight smaller than a tolerance. (c) The new method. Directly optimize the joint angles by maximizing the gain. (*.*.1) Confidence estimation. (*.*.2) NBC optimization result. (*.*.3) Updated point cloud.

Chapter 5

TAMP for 3D Curving

Elasto-plastic metal wire curving task is commonly seen in manufacturing and medical fields. This section presents a combined task and motion planner (TAMP) for a robot arm to work aside a bending machine and carry out 3D metal wire curving tasks. We assume a collaborative robot that is safe for humans but has a weak payload and develop the combined planner for the robot to use the bending machine. The contributions of the study are three-fold. First, we propose a coarse-to-fine optimization-based method to convert a 3D curve to a structured bending set. Second, we build a planner to generate the feasible bending sequence, machine operation, robotic grasp poses, and arm motion while considering constraints from the bending machine and the robot. Third, we use visual feedback to build and dynamically update the springback model of a metal wire and use the model to predict and compensate for bending errors caused by springback. Compared with previous work, the proposed planner does not require the robot arm to have a large payload, making it suitable for lightweight collaborative robots. We evaluate the system using both simulated and real-world 3D curving tasks. The results show that the proposed planner can solve robotic 3D curving problems with satisfying time efficiency and precision. It is flexible and applicable to different robots and metal wire materials without a significant change. The method is expected to accelerate the high-variation low-volume manufacture of 3D metal wire curves.

Note to Practitioners: Using robots to bend metal wires has been an old topic in robotics and automation. In previous robotic metal wire bending systems, the robot motion was usually pre-programmed to feed parts to bending machines. It was not easy to be extended to multiple goal shapes. Also, the bending was limited to a few discrete action points instead of an arbitrary curve. The method we developed solved the problem by adding up optimized goal shape parameterization, combined task and motion planning, and springback compensation. It helps to auto-program robot actions and ensures satisfying precision by correcting bending results online with visual feedback. Practitioners are encouraged to use the planner for either offline programming or online motion generation. The springback estimation and compensation are independent of motion generation and can be connected to both motions pre-programmed offline or generated online. However, it is advisable to employ robots with higher DoFs (Degree of Freedom) and avoid the online planning of curves with many bending actions.

5.1 Introduction

This section presents a combined task and motion planner (TAMP) [16] for a robot to curve metal wires into 3D shapes by working aside a bending machine. We assume a collaborative robot that is safe for humans. Such a robot does not have enough payload to directly bend objects with high stiffness. We designed a bending machine made of a stepper motor and a gear set to solve the low-payload problem. The designed bending machine can provide a much more significant bending force than the robot arm to bend high-stiffness objects. The developed planner enables a collaborative robot to work aside the bending machine like a professional human worker and curve stiff and various-shaped metal wires. It plans the bending sequences at the task level and robotic joint trajectories at the motion level while considering constraints from the bending machine, the robot, and the wire's physical properties like springback. The planner helps solve robotic 3D curving problems with satisfying time efficiency and precision. It does not require manual definition or programming and is flexible and applicable to different robots and metal wire materials without a significant change.

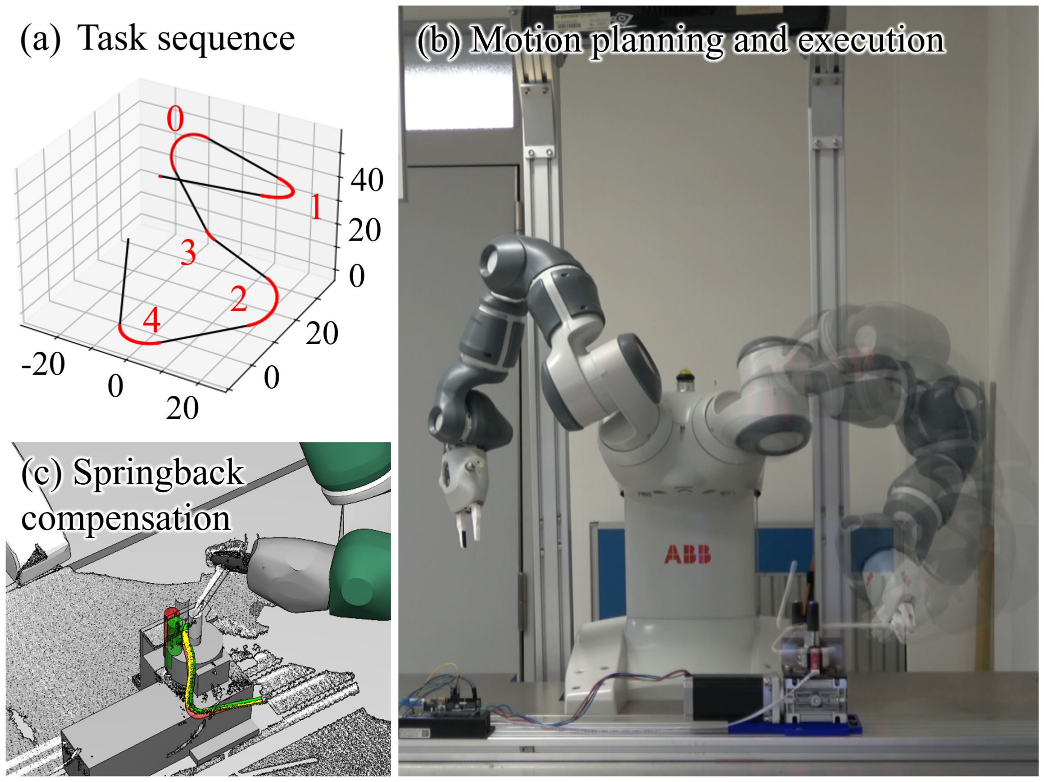

(a) Task sequence
0
1
2
3
4
40
20
0
-20
0
20
20
0
(b) Motion planning and execution
(c) Springback compensation
ABB

Figure 5.1: (a) Curve parameterization and closed-loop planning considering combined task-level and motion-level constraints. (b) A collaborative robot (low payload) bends a metal wire with a bending machine following the planned result. (c) Springback is compensated by visual estimation during the robotic execution.

Previously, using robots to curve metal wires was widely studied, as robots provided a promising solution to flexible or personalized manufacturing of metal-wire products. The previous methods treated robots as automation machines. They programmed a robot to hold a metal wire and bend it by pressing against a fixture or an external robotic gripper. The payload of the robot limited the maximally affordable metal stiffness. Unlike the previous methods, we develop a planner for a robot to curve metal wires by working aside a bending machine. The planner helps determine the bending sequence, the grasping poses, and the robot joint trajectories to conduct 3D curving with the machine. Fig. 5.1 illustrates the robot-machine system and workflow. In the first step, we use a coarse-to-fine parameterization to represent a desired 3D curve as a set of bending action points. Then, we perform a combined task and motion planning to generate the bending sequence, grasping poses, and arm motion integrally. Finally, the robot will follow the planned results to work with the bending machine and bend the metal wire considering spingback compensation predicted using visual feedback and a dynamically updated spingback model.

Our main contribution is the closed-loop exploration considering the task and motion-level constraints. At the task level, we investigate the bending machine's kinematic constraints and the metal wire's changing shapes. These constraints help us to determine several tentative bending sequences. At the motion level, we investigate the logical relations of robotic grasping poses, robot joints' torque limits, and the availability of joint trajectories for moving the metal wire to the next bending pose. The motion-level results will be recorded during exploration to prune the task-level sequences and improve planning efficiency. They together make the impossible exploded combinatorics tractable. We improved the bending results' accuracy with the following techniques: First, we propose an optimization-based algorithm to achieve a better bending set representation. Second, we select robotic grasping poses considering reaction forces from the bending machine to ensure the robot does not bear excessive payload during execution. Third, we include visual estimation and springback compensation to reduce errors caused by a metal wire's elastoplastic properties.

We evaluate the system using both simulated and real-world 3D curving tasks in the experimental section. The results show that the proposed planner can solve robotic 3D curving problems with satisfying time efficiency and precision. It is flexible and applicable to different robots and metal wire materials without a significant change. The method is expected to accelerate the high-variation low-volume manufacture of 3D metal wire curves.

The remaining sections are organized as follows. Section 5.2 reviews the related work. Section 5.3 presents preliminary developments like the bending machine design and bending set paramerization. Section 5.4 presents the combined task and motion planning, with special discussions and analysis spent on force constraints and springback compensation. Section 5.5 presents experiments and analysis. Section 5.6 draws conclusions and discusses the limitation and future work.

5.2 Related Work

5.2.1 Deformable Linear Objects (DLO)

Robotics research communities have paid considerable attention to the problem of automatically shaping Deformable Linear Objects. Unlike rigid objects, a DLO is more complicated due to its infinite degrees of freedom. The methods for automatically shaping such objects can be separated into non-physical (geometric) and physical ones [142][15]. For example, Lv et al. [143] developed a comprehensive dynamic model using a discrete elastic rod model. Yan et al. [144] used Model Predictive Path Integral (MPPI) control for optimizing a sequence of actions. Laezza et al. [145] developed a reinforcement learning-based elastoplastic object shape control method. Takizawa et al. [146] proposed a geometric model-based method to manipulate the shape of a rope to tie it around a pipe using a dual-arm robot. Han et al. [147] developed a sample-efficient reinforcement learning method to resolve the high-dimensional planning problem with a practical number of samples. Since the DLOs' shapes change during robotic manipulation, researchers also developed methods to track the deformation and conduct online collision checking during planning [148]. Tactile [149][150], force [151] and visual [144][152][153] sensing are widely used techniques for the tracking and online collision detection.

Researchers have categorized the DLOs into five categories considering their deformability and whether a force smaller than gravity can result in deformation [154]. This section focuses on curving the elastoplastic DLOs (e.g., metal wire, metal rod, etc.). Unlike the rope or elastic rod, repeated bending of elastoplastic DLOs will result in material fatigue, making their curving task different from that of rope, cable, and elastic beam. The elastoplastic DLOs are generally curved by one or a sequence of discrete bending actions to avoid cracks caused by material fatigue. Previously, Jin et al. [155] designed an automation device to bend an elastoplastic DLO (archwire). The device flexibly curved desired shapes by orchestrating predefined processes. Xia et al. [156] used a robotic arm and an external gripping unit to carry out similar tasks. A sample-based planner was adopted for bending path generation. Kuehl et al. [157] automated hairpin winding with a robotic manipulator. When the material's stiffness is high, it is difficult for a low-payload robot to curve DLOs actively. Lu et al. [158] solved the problem by leveraging a heater to soften metal rods before applying robotic arm actions. Zhang et al. [159] designed a multi-axis special-purpose machine to form high-stiffness titanium alloy strips used in oral and maxillofacial surgery. Another challenge of elastoplastic DLOs bending task is to control the springback, which may cause crucial problem related to accuracy. Wu et al. [160] formulated the springback compensation problem as an optimization of the bending and twisting parameters using Finite Element (FE) simulation. Zhang et al. [161] proposed a springback prediction model for variable curvature tubes, the bending compensation amount was obtained through a reverse process.

This work uses a collaborative robot arm and a bending machine to achieve the elastoplastic metal wire curving task. Unlike previous robotic curving systems, we developed a planner to automatically generate the robot motion for using a bending machine. We parameterized goal curves considering a bending machine's constraints and preserve goal shapes as much as possible. Meanwhile, we developed an online visual estimation-based springback prediction and compensation method to reduce errors caused by a metal wire's elastoplastic properties and achieved satisfying precision.

5.2.2 Bending System

A bending system is a machine or a collection of interconnected mechanisms that work together to bend materials into desired shapes. It can include not only the physical units but also the planning algorithms that guide the process. There is plenty of work done in the automation of bending systems. Raj et al. [162] constructed a bend feasibility matrix to map the entire search space and used the best-first search algorithm to determine the metal sheet bend sequence. Kontolatis et al. [163] and Sen et al. [164] used genetic algorithms and particle swarm optimization to solve the bending sequence planning problem, respectively. Baraldo et al. [165] developed a graph representation and used A* search to find an appropriate bending sequence for a wire bending machine. Aomura et al. [166] used a searching-based method to generate metal sheet bending sequences for a robotic manipulator considering its grasp positions.

Table 5.1 further summarises the systems mentioned above according to three criteria:

- Type of objects being bent: Objects are categorized based on their geometric features.

- Incorporation of action and sequence planning: Action planning generates the parameters for each bending action. Sequence planning generates a series of feasible action poses while considering robotic and environmental constraints.

- Feeding and bending device employed: The feeding device is a robot or machine responsible for translating and rotating an object to achieve the desired pose for bending. The bending device is a robot or machine responsible for shaping the object.

Our study stands out by incorporating both action and sequence planning for wire objects. It uses a general high-DoF manipulator for feeding and a specially designed machine for bending. The planning and the physical setup improve the flexibility of the 3D curving. The proposed system is not only suitable for metal wire. It can also easily be adapted to metal plates and pipes.

5.3 Problem Formulation

5.3.1 Structure of the Bending Machine

We design a bending machine shown in Fig. 5.2 by following the mechanisms of commercial metal plate benders or folders. The bending machine comprises a 23HS45-4204S (Holding Torque 3 Nm) stepping motor as the power source, a 1:10 gearbox to

Table 5.1: Comparison of related works.

		Planning		Physical Device	
Reference	Object Type	Act.	Seq.	Feed	Bend
[156]	Wire	◯	◯	D	M
[157][158]	Wire	◯	×	D	M
[165]	Wire	×	◯	D	D
[159]	Plate	◯	×	D	D
[160][167]	Tube	◯	×	D	D
[162][163][164][168]	Sheet	×	◯	-	D
[166]	Sheet	×	◯	M	D
[169]	Sheet	◯	◯	M	D
Ours	Wire	◯	◯	M	D

* Meanings of abbreviations: Act. - Action planning; Seq. - Sequence planning; Feed - The device used to feed the object; Bend - The device used to feed the object; M - General high-DoF manipulator; D - Specially designed machine or unit.

increase the output torque, an optical photo-interrupter for initialization (calibrating the zero position), and several rollers for pressing or fixing target metal wires. Two needle-roller bearings (Fig. 5.2(b)) are installed under the bending roller and the center roller to allow free sliding motion between them and the metal wire.

A target metal wire is assumed to be placed between the center roller and the other two rollers. The bending roller can rotate clockwise and counterclockwise starting from the calibrated zero position. Thus the machine allows bending from two directions without reversing a wire. The maximum rotation angle of the bending roller is highlighted as a green zone in Fig. 5.3(a). Its effective work range depends on the shape and thickness of the metal wire and is a bit narrower than the maximum rotation range. Fig.5.3(b) and (c) exemplify a bending process. Fig.5.3(b) is the initial configuration where the wire is straight and tangent with all three rollers. The rotation angle of the wire (φ_i) and the start angle of the machine (γ_i) can be computed considering the three rollers' radii and the thickness of the metal wire. Fig. 5.3(c) is the configuration after a bending action. The end angle of the machine can be computed using $\gamma_i + \theta_i$. When the wire is not straight, it is difficult to figure out

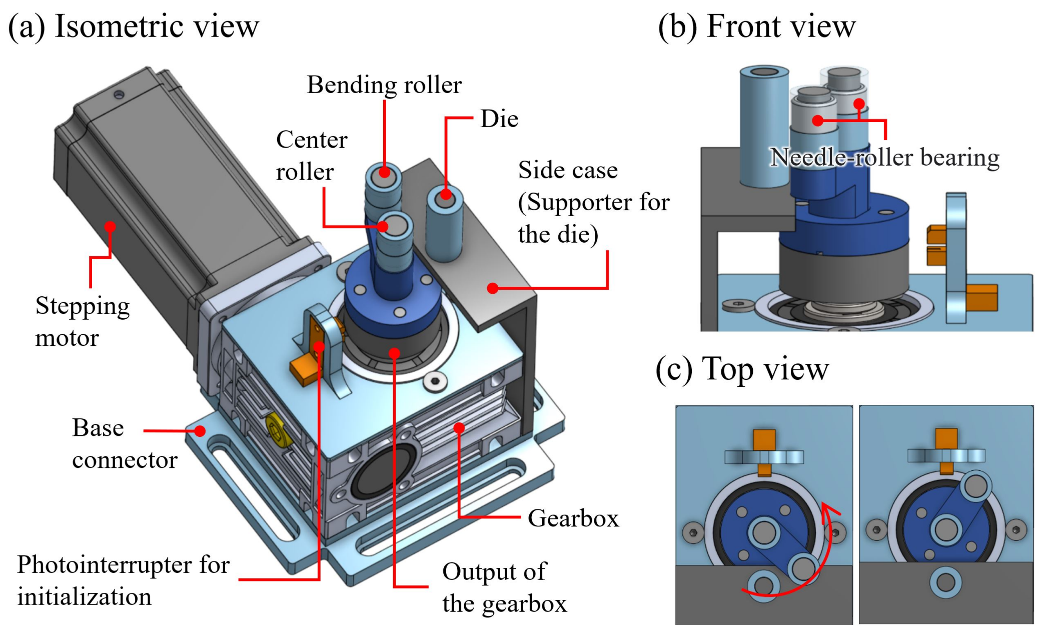


Figure 5.2: CAD model of the designed bending machine.

an explicit equation for computing this angle. In that case, we leverage a kinematic simulator to examine the collision between the wire and the rollers and measure φ_i and γ_i based on the collisions. Notably, we have the following assumptions for the kinematic simulator. First, we do not consider the accumulation and release of kinetic energy and thus cannot estimate and simulate the spring back of elastoplastic materials. Second, the wire begins to bend when the bending roller starts to collide with it and stops bending when the bending roller reaches a given goal rotation angle. Third, we assume the bending arc couples with the center roller during bending. The wire will wind around the center roller to form an arc. Fourth, if two bending actions overlap, the newer arc will override the older arc sections. Under these assumptions, we can compute the changed shape after each bending action by replacing related wire sections with arcs determined by collisions.

Note that although we use this design to practice our task and motion planning implementation, the developed method is not limited to this specific machine. It can be adapted to other alternative mechanisms, such as those presented in [170], [171], etc.

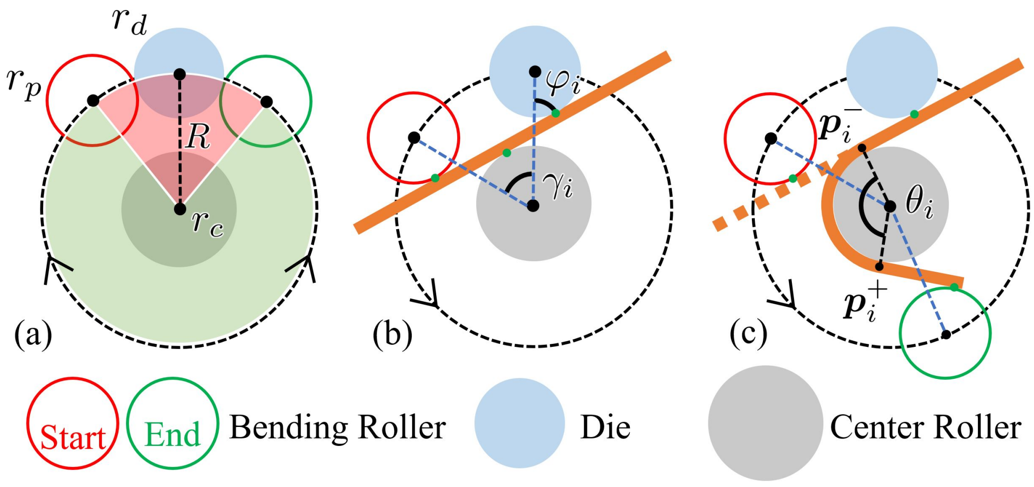


Figure 5.3: Illustration of the bending machine's action from a 2D top view. (a) Maximum rotation range of the bending roller. (b) Initial contact between a bending roller and a straight metal wire. (c) End configuration of a bend action.

5.3.2 Representing a 3D Curve Using Bending Paramerization

We assume a desired 3D curve is initially given as a list of dense points. Since the bending machine's center roller has a non-negligible radius which leads to a maximum bending curvature, it is difficult to generate bending actions considering all dense points. Bending actions close to each other will be canceled by the maximum bending curvature. Instead of directly using the dense points, we propose a coarse-to-fine fitting algorithm to approximate the 3D curve with fewer points. The algorithm's pseudo-code is shown in Algorithm 4. Like its name, the algorithm is carried out progressively from an initial coarse representation to a later optimal refinement. The details will be presented below.

Coarse Representation

In coarse representation, we combine polyline approximation and arc adaption to simplify the dense points. The objective of polyline approximation is to reduce the number of points in a dense point set while maintaining the original shape as closely

Algorithm 4: Coarse-to-fine Bending Parameterization

```
Input:  $\mathcal{D}$, desired curve represented by dense points;
        $r_c$, center roller radius of the bending machine;
        $r_w$, metal wire radius;
        $\epsilon$, error tolerance of the representation;
Output: $\mathbb{B}$, a set of bending parameters;
1  begin
2      $r \leftarrow r_c + r_w$
3      $\mathcal{P} \leftarrow \{\mathcal{D}[0], \mathcal{D}[-1]\}$
4      $\widetilde{\mathbb{B}} \leftarrow \{\}$
5      $error,\ min\_error \leftarrow +\infty$
6      while $error > \epsilon$ do
7          $\boldsymbol{p}_{\text{tmp}} \leftarrow \texttt{farthest\_point}(\mathcal{D}, \texttt{polyline}(\mathcal{P}))$;
8          $\widetilde{\theta}_{\text{tmp}}, \widetilde{\alpha}_{\text{tmp}}, \widetilde{l}_{\text{tmp}} \leftarrow \texttt{arc\_adaption}(\boldsymbol{p}_{\text{tmp}}, r)$;
9          $\widetilde{\mathbb{B}}_{\text{tmp}} \leftarrow \widetilde{\mathbb{B}} \cup \{\widetilde{\theta}_{\text{tmp}}, \widetilde{\alpha}_{\text{tmp}}, \widetilde{l}_{\text{tmp}}\}$;
10         $\mathcal{C} \leftarrow \texttt{merge\_overlap}(\widetilde{\mathbb{B}}_{\text{tmp}}, r)$;
11         $error \leftarrow \texttt{error}(\mathcal{D}, \mathcal{C})$;
12         if $error > min\_error$ then
13             break;
14         $min\_error \leftarrow error$;
15         $\mathcal{P} \leftarrow \mathcal{P} \cup \boldsymbol{p}_{\text{tmp}}$;
16         $\widetilde{\mathbb{B}} \leftarrow \widetilde{\mathbb{B}}_{\text{tmp}}$;
17     $\mathbb{B} \leftarrow \texttt{minimize\_average\_error}(\widetilde{\mathbb{B}}, \mathcal{D})$;
18     return $\mathbb{B}$
```

as possible. This process, known as line simplification, can be achieved through various methods, such as preserving critical points (Douglas-Peucker method, DP) [172], maintaining important bends [173], or retaining significant areas [174][175]. In comparison to the DP method, the latter two methods are more suitable for handling jagged curves or curves with noisy bending curvatures. They do not apply to our scenario, as the goal was to authentically approximate a curve rather than overlook noises. Thus, instead of them, we employ the DP method for line simplification in this work. Fig. 5.4(a) shows the polyline approximation process. The smooth black curve ($\mathcal{D}$) is the desired curve. The orange dashed line ($\mathcal{P}$) is the polyline approximation result. It comprises a sequence of pivot points $[\boldsymbol{p}_0, \boldsymbol{p}_1, ..., \boldsymbol{p}_n]$ plus their normal $[\boldsymbol{n}_0, \boldsymbol{n}_1, ..., \boldsymbol{n}_n\}$. Fig. 5.4(b) and (c) show the arc adaption process. It essentially fits the central roller (circles in the figure) to the pivot points of the approximated polyline and converts the sections tangential to the rollers ($\overrightarrow{\boldsymbol{p}_i^- \boldsymbol{p}_i}$ and $\overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_i^+}$ in the figure) to arc sections ($\widehat{\boldsymbol{p}_i^- \boldsymbol{p}_i^+}$ in the figure). Like polyline approximation, the adapted curve suffers from a representation error. However, there is a trade-off between the number of pivot points and the representation error.

Due to the center roller's non-negligible radius imposing a maximum bending curvature, increasing the number of pivot points has a non-monotonic and non-linear relation with the arc adaption accuracy. The arc adaption in the shadow regions of Fig. 5.4(b) and (c) illustrates an example.

The details are shown in Fig. 5.5. The curve $\mathcal{D}$ is approximated using a polyline in Fig. 5.5(a.1) and adapted with an arc considering the center roller's radius in Fig. 5.5(a.2). The green double-arrow line in Fig. 5.5(a.2) denotes the representation error after adaptation. Increasing a pivot point to it leads to the result shown in Fig. 5.5(b.1). There are two overlapped arc sections, as shown in the enlarged frame box on the right side. If bent sequentially, the two arc sections will result in the curves shown in Fig. 5.5(b.2) and (b.3), making the wire direction different from the goal curve. To keep the wire direction, we merge the two overlapped arc sections into a single arc (thus a single bending action) as shown in Fig. 5.5(b.4). The representation error after merging is denoted by the green double-arrow line in Fig. 5.5(b.4). The error is not necessarily smaller than the one before increasing a pivot point. To minimize the representation error, we incrementally add new pivot points while comparing the representation error with the previous minimum value to an optimal number of pivot points. Lines 6 and 12 of Algorithm 4 respectively show the incremental loop and the comparison of the errors. The *min_error* parameter saves the previous minimum representation error. The current *error* is compared against it in line 12 to determine a break. The threshold error ϵ in line 6 is additionally designed to avoid non-converging iteration. If the minimum representation error always decreases, the loop will stop when the minimum error is lower than ϵ.

Since $\mathcal{D}$ is a 3D curve, an adapted arc section $\widehat{\boldsymbol{p}_i^- \boldsymbol{p}_i^+}$ will also be 3D. The section could be described using a bending angle θ_i and a twisting angle α_i, as shown in Fig.

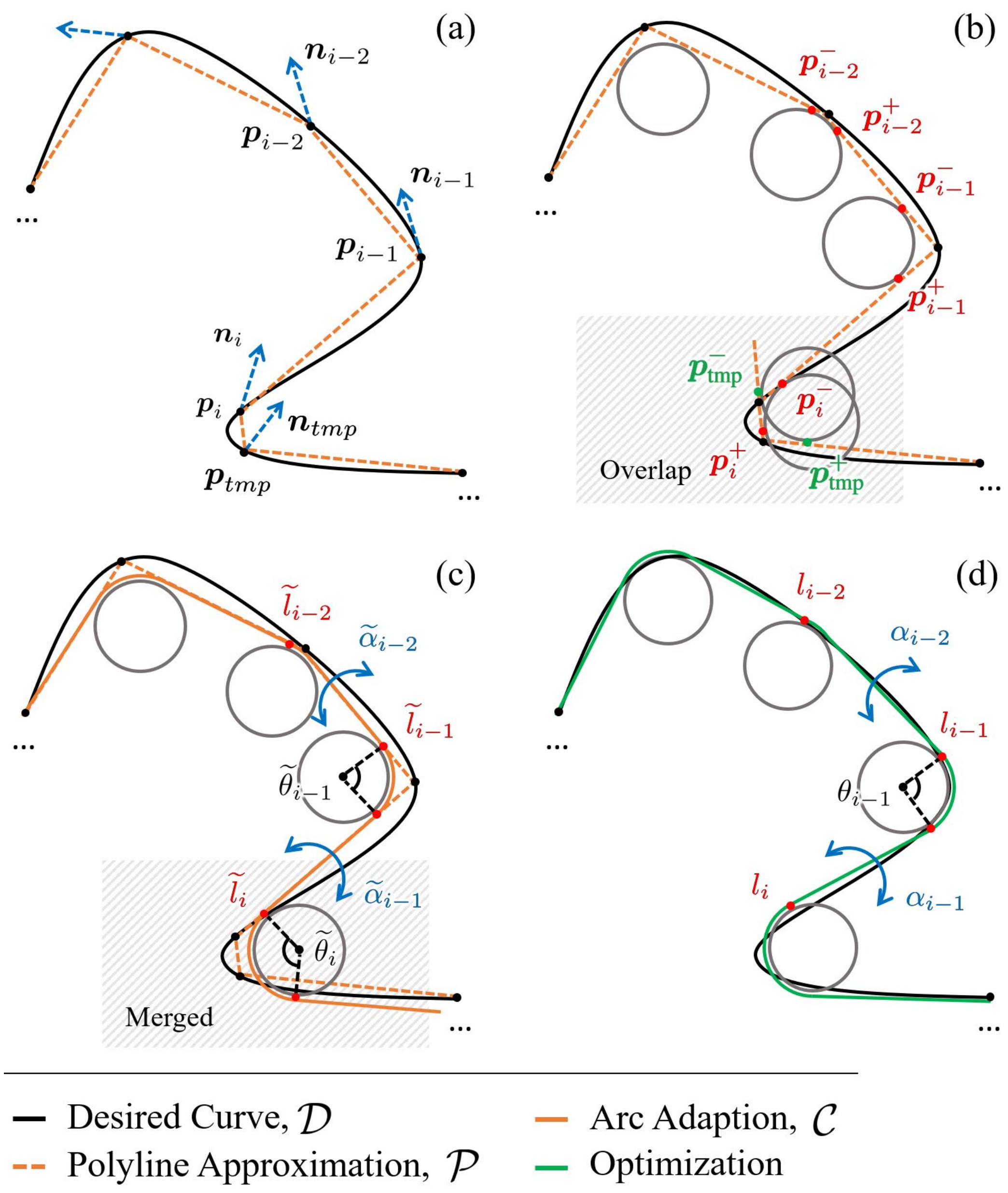


Figure 5.4: (a) Polyline approximation. (b) Fit the bending roller to the pivot points of the approximated polyline. (c) Merge overlapped arcs and parameterize the bending actions. (d) Minimize the representation error by non-linear optimization.

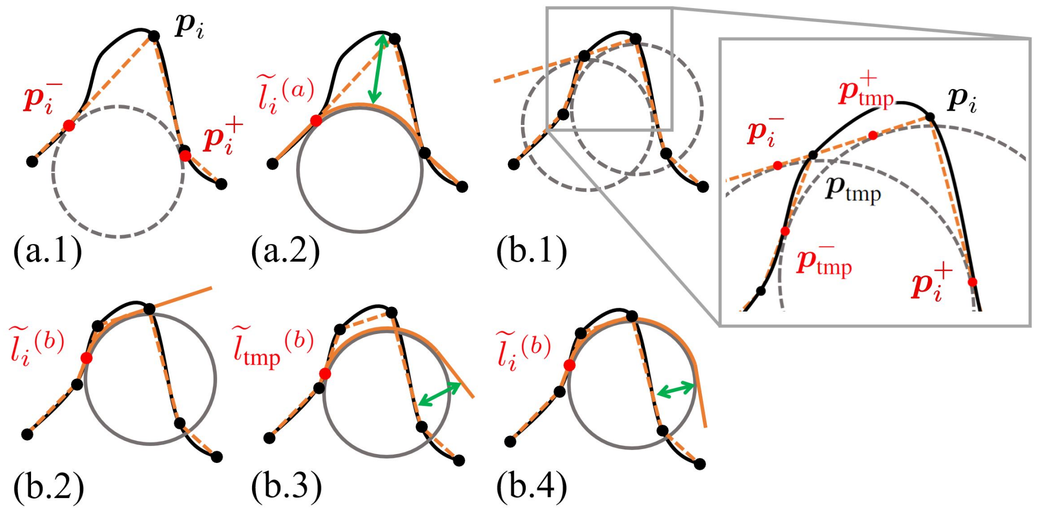


Figure 5.5: (a.1) Polyline approximation with four pivot points (black dots). (a.2) Adapting to an arc considering the center roller's radius. (b.1) Polyline approximation with six pivot points. (b.2-3) Two step arc adaption without merge. In this case, δl is larger than 0. B_i overlaps with B_{i-1}. The adapted arc results into a smaller bending angle. (b.4) The bending angle conforms with expectation after merging B_{i-1} to B_i.

5.6. Based on the description, we define an arc $\widehat{\boldsymbol{p}_i^- \boldsymbol{p}_i^+}$ by using bending parameters $\widetilde{B}_i = \{\widetilde{\theta}_i, \widetilde{\alpha}_i, \widetilde{l}_i\}$. The values of $\widetilde{\theta}_i$ and $\widetilde{\alpha}_i$ are measured in the local coordinate system $\{\boldsymbol{p}_i^-, \mathbf{R}_i\}$ shown in Fig. 5.6[1]. $\widetilde{l}_i$ indicates the 1D position of $\boldsymbol{p}_i^-$ measured along the metal wire starting from its origin. Their formal definitions are presented in equation (5.1). Note that we included a $\widetilde{\cdot}$ hat on top of the set elements to indicate that the parameters are temporal to this algorithm stage and will be refined later. The bending parameters are also labeled in Fig. 5.4(c) for easy understanding. Note that the figure is illustrated in 2D and the angle $\widetilde{\alpha}_i$ may be difficult to read.

$$\widetilde{\theta}_i = \arccos(\frac{\overrightarrow{\boldsymbol{p}_{i-1}\boldsymbol{p}_i} \cdot \overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_{i+1}}}{\|\overrightarrow{\boldsymbol{p}_{i-1}\boldsymbol{p}_i}\| \cdot \|\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_{i+1}})\|} \tag{5.1a}$$

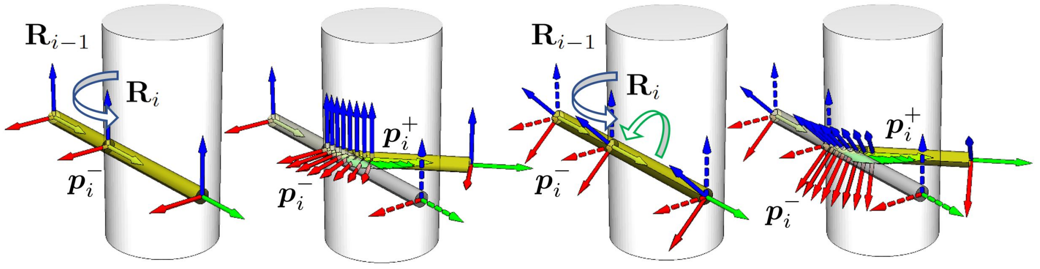
$$\widetilde{\alpha}_i = \arccos(\frac{\boldsymbol{n}_i \cdot \boldsymbol{n}_{i+1}}{\|\boldsymbol{n}_i\| \cdot \|\boldsymbol{n}_{i+1}\|})) \tag{5.1b}$$

[1]The y direction of the local coordinate system is the wire axial direction ($\overrightarrow{\boldsymbol{p}_i^- \boldsymbol{p}_i}$). The z direction is the normal direction of a local plane ($\boldsymbol{n}_i$).

$$\widetilde{l}_i = \begin{cases} \left\| \overrightarrow{\boldsymbol{p}_0 \boldsymbol{p}_1^-} \right\|, i = 1 \\ \widetilde{l}_{i-1} + \widetilde{\theta}_{i-1} \cdot r + \left\| \overrightarrow{\boldsymbol{p}_{i-1}^+ \boldsymbol{p}_i^-} \right\|, 1 < i < n \end{cases} \tag{5.1c}$$

$$\begin{cases} \boldsymbol{p}_i^- = \boldsymbol{p}_i - r \cdot \tan \frac{\theta_i}{2} \cdot \frac{\overrightarrow{\boldsymbol{p}_{i-1} \boldsymbol{p}_i}}{\|\overrightarrow{\boldsymbol{p}_{i-1} \boldsymbol{p}_i}\|} \\ \boldsymbol{p}_i^+ = \boldsymbol{p}_i + r \cdot \tan \frac{\theta_i}{2} \cdot \frac{\overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_{i+1}}}{\|\overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_{i+1}}\|} \end{cases} \tag{5.1d}$$

$$r = r_c + r_w \tag{5.1e}$$



(a) $\theta_i = \pi/4, \alpha_i = 0$ (b) $\theta_i = \pi/4, \alpha_i = \pi/4$

Figure 5.6: We use a bending angle θ_i and a twisting angle α_i to represent a bending action. Their values are defined in the local coordinate system of the bending action point $\boldsymbol{p}_i^-$.

A complete 3D curve is made of many arc sections and can thus be represented as a bending set $\widetilde{\mathbb{B}} = \{\widetilde{B}_i\}$. Each element in the bending set holds the bending parameters of an individual arc section. They will be converted to the bending action commands of the bending machine. For an ith arc section, the metal wire will be placed between the center roller and the bending roller & die, with the wire in contact with the center roller at $\boldsymbol{p}_i^-$. The wire will be rotated by the robot and pressed by the bending roller to form the desired shape. The magnitude of $\mathbb{B}$ is determined by the aforementioned lines 6 and 12 of Algorithm 4. In the following sections, we use m to denote $\mathbb{B}$'s magnitude and differentiate it from the number of pivot points n.

Line 9 and 10 of Algorithm 4 determine the overlap and merge the arc sections. For a newly inserted pivot point and arc section B_{tmp} after B_i, the algorithm recursively examines its positional difference δl with adjacent arc sections following equation (5.2). When δl is larger than 0, the algorithm merges the two arc sections and their respective bending parameters following equation (5.3).

$$\delta l = \widetilde{l}_i + \widetilde{\theta}_i \cdot r - \widetilde{l}_{\text{tmp}} \tag{5.2}$$

$$\begin{cases} \widetilde{l}_i \leftarrow \widetilde{l}_i \\ \widetilde{\theta}_i \leftarrow \widetilde{\theta}_i + \widetilde{\theta}_{\text{tmp}} \\ \widetilde{\alpha}_i \leftarrow \arccos(\dfrac{\boldsymbol{n}_i \cdot \boldsymbol{n}_{i+1}}{\|\boldsymbol{n}_i\| \cdot \|\boldsymbol{n}_{i+1}\|})) \end{cases} \tag{5.3}$$

Optimal Refinement

The straight-line sections of coarse representation will either be coincident (before merging) or parallel (after merging) with a polyline approximation, which is a strong constraint and will lead to unavoidable residual errors. In the second half of the algorithm, we further reduce the residual error by performing an optimal refinement.

Particularly, we allow adjusting the parameters of each bending action point to reduce the errors. We formulate the adjustment as an optimization problem shown by equation (5.4). The optimization goal is to minimize the average error between the target curve ($\mathcal{D}$) and simplified curve ($\mathcal{C}$) as shown by equation (5.4a). The x and y variables in the equation represent dense points in $\mathcal{D}$ and $\mathcal{C}$. They are discretized from respective continuous curves. Notations ξ_θ, ξ_α and ξ_l in equations (5.4c) are manually defined to limit the searching space. Equation (5.4d) ensures the order of the bending action sequence is not changed, and there is no overlap between two bending action points.

Line 17 of Algorithm 4 shows the position of optimization in the pseudo code. It produces a set of optimal bending parameters. The parameters will be used for modeling and solving a combined task and motion planning problem. Fig. 5.4(d) illustrates the results before and after adjustment.

$$\underset{\mathbb{B}=\{B_i\},B_i=\{\theta_i,\alpha_i,l_i\}}{\operatorname{argmin}} \quad \frac{1}{n}\sum_{x\in\mathcal{D}} \min_{y\in\mathcal{C}(\widetilde{\mathbb{B}})} \|x-y\|_2 \tag{5.4a}$$

$$\text{s.t.} \quad \widetilde{B}_i = \{\widetilde{\theta}_i, \widetilde{\alpha}_i, \widetilde{l}_i\} \in \widetilde{\mathbb{B}} \tag{5.4b}$$

$$\begin{cases} \left|\theta_i - \widetilde{\theta}_i\right| \leq \xi_\theta \\ \left|\alpha_i - \widetilde{\alpha}_i\right| \leq \xi_\alpha \\ \left|l_i - \widetilde{l}_i\right| \leq \xi_l \end{cases} \tag{5.4c}$$

$$l_i + \theta_i \cdot r \leq l_{i+1} \tag{5.4d}$$

5.4 Combined Task and Motion Planning Considering Springback

Next, we carry out combined task and motion planning based on the obtained bending parameters. The flowchart of the proposed method, with a particular focus on the combined task and motion planning part, is shown in Fig. 5.7. The planner accepts the $\mathbb{B}$ obtained in the last section and some pre-annotated grasp poses for a metal wire primitive as the input. In the first step, it generates a bending sequence by permuting the order of the optimized bending action points while pruning the searching tree considering previously detected invalid actions. In the second step, the generated bending sequence is sent to the motion generation process to determine grasp poses and plan manipulation motion. The method iterates through the pre-annotated grasp poses to find the candidates that are kinematically feasible to reach all bending action points. It sorts the feasible grasp candidates considering the torque limit of the robot arm to ensure safe executions. The springback of the metal wire is considered during execution. A linear model with online visual estimation and dynamic updates is used to predict additional bending angles and compensate for the springback after releasing the bending roller. The details of the planner are presented below.

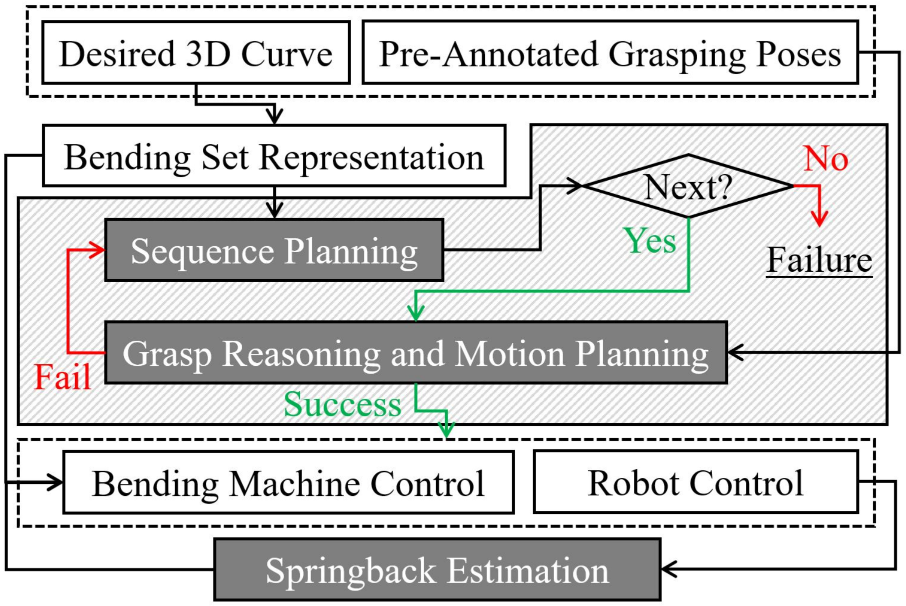


Figure 5.7: Workflow of the proposed planner. The shadowed box in the middle represents the combined task and motion planning. The details of the three steps will be presented in the remaining part of this section.

5.4.1 Bending Sequence Planning

The bending sequence planning process is at the task level. It is carried out without considering robotic constraints. A feasible bending sequence must meet the following two conditions: (1) The wire does not collide with the machine or any other obstacles during bending; (2) The wire is long enough to contact the die and bending roller. If the metal wire cannot touch one of the rollers at a certain bending action, the bending parameter B_i pertaining to the action will be considered infeasible. An example of an infeasible scenario is illustrated in Fig. 5.8.

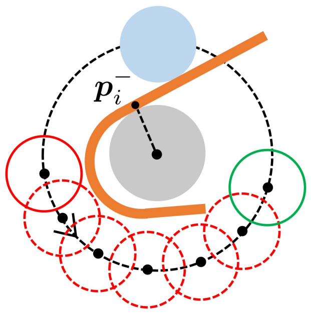


Figure 5.8: Infeasible.

As the sequence that fulfills the conditions above may not be unique, we further introduce two criteria to find the optimized solution: (1) Number of pick-and-places (N_{pnp}); (2) Number of unstable bending actions (N_{unstable}). Consider a bending set

Algorithm 5: Prune Search

```
Input: $\mathbb{B} = \{B_0, B_1..., B_n\}$, a bending action set;
       $\boldsymbol{t}$, permutation tree;
       $\boldsymbol{s} = [0, 1, .., n]$, a bending sequence where each element is an index to
       $\mathbb{B}$;
begin
    $\boldsymbol{t} \leftarrow$ select($\boldsymbol{t}$);
    while $\boldsymbol{s}$ is not empty do
        if $\boldsymbol{s}$ is feasible then
            $G_{com} \leftarrow$ grasp_reasoning($G$);
            motion_planning($G_{com}$);
            if success then
                return $\boldsymbol{s}$
        $\boldsymbol{t} \leftarrow$ update($\boldsymbol{t}$, permutate($\boldsymbol{s}[: i]$));
        $\boldsymbol{s} \leftarrow$ Find a new sequence from $\boldsymbol{t}$;
    return False
```

with $m = 2$ for example. As illustrated in Fig. 5.9, if the bending sequence is $l_0 \rightarrow l_1$ (depicted in (b.i)), the wire between l_0 and the next bending point l_1 will not be straight and the robot will need to reset the wire (i.e., pick the wire up from the rollers, adjust its pose, and place it back for the second bending action, $N_{\text{pnp}} = 1$). Additionally, the wire between l_1 and $\boldsymbol{p}_g$ is not straight. The friction force between the wire and the bending roller is not collinear with the force provided by the gripper, which may cause unexpected deformation of the wire ($N_{\text{unstable}} = 1$). More details will be introduced in Section 5.4.2. On the other hand, if the bending sequence is $l_1 \rightarrow l_0$ (shown in (b.ii)), the wire between $\boldsymbol{p}_g$ and the bending position remains straight. The values of N_{pnp} and N_{unstable} will be 0, and the robot will not need to reset the wire.

The candidate sequences in the permutation tree $\boldsymbol{t}$ are firstly selected using the routine shown in Algorithm 6, as denoted by line 2 of Algorithm 5. The N_{pnp} and N_{unstable} used to compare and select candidates are computed in lines 5∼14 of the select routine. The candidates are sorted in ascending order according to the values of N_{pnp} and N_{unstable}, with a priority given to N_{pnp}.

Next, we employ pruning to find a feasible sequence from the sorted candidates. Finding a sequence that fulfills the above constraints is costly as a set of n bending

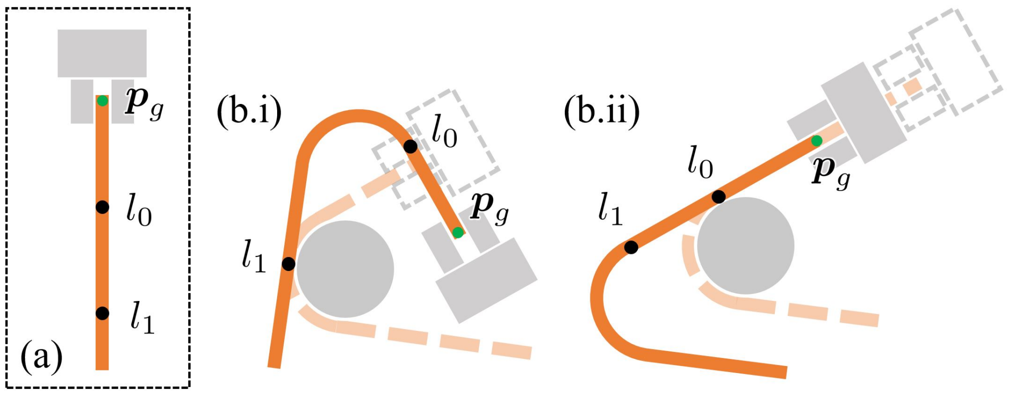


Figure 5.9: Example of a curve with $m = 2$. (a) Bending positions. (b.i) Sequence $l_0 \to l_1$. $N_{\text{pnp}} = 1$, $N_{\text{unstable}} = 1$. (b.ii) Sequence $l_1 \to l_0$. $N_{\text{pnp}} = 0$, $N_{\text{unstable}} = 0$.

actions can be arranged in $n!$ ways. Instead of brutally evaluating all permutations of bending actions, we leverage pruning in this work to accelerate the search. Algorithm 5 shows the details. The algorithm starts with a bending sequence initialized as $\boldsymbol{s} = [0, 1, ..., n]$. Each element of $\boldsymbol{s}$ represents an index of $\mathbb{B}$. The algorithm examines the sequence's validity by checking the action parameters indexed by the sequence elements one by one. Suppose a parameter leads to a collided, infeasible, or unreachable bending action. In that case, the sequence is judged invalid, and the remaining elements no longer need to be explored. The sub-sequence up to the invalid bending parameter, namely $\boldsymbol{s}[: i]$, and all permutations of paths passing through the same nodes (`permutate`$(\boldsymbol{s}[: i])$) are removed from $\boldsymbol{t}$, using the `update`() function (line 9 of Algorithm 5). After that, the algorithm will generate a new sequence from the sorted $\boldsymbol{t}$ while avoiding repeating the invalid sub-sequences. The algorithm will repeat the examination routine with the newly generated sequence.

5.4.2 Grasp Reasoning and Motion Planning

The grasp reasoning and motion planning process is at the motion level. We use an example with two bending action points to explain this process. Like its name, the process comprises grasp reasoning and motion planning. Grasp reasoning accepts (1) pre-annotated grasp poses for a metal wire primitive ($\boldsymbol{G}$) and (2) wire poses generated

Algorithm 6: Compare and select candidates

```
Input: $\boldsymbol{t}$, permutation tree;
1  begin
2      $N_{\text{pnp}}, N_{\text{unstable}} \leftarrow \{\}, \{\}$;
3      while $\boldsymbol{s}$ is not empty do
4          $N_{\text{pnp}}, N_{\text{unstable}} \leftarrow 0, 0$;
5          for $j \in [0, n)$ do
6              if $\boldsymbol{s}[j+1] > \boldsymbol{s}[j]$ then
7                  $N_{\text{pnp}} \leftarrow N_{\text{pnp}} + 1$;
8              if $\texttt{is\_intersec}(\boldsymbol{s}[:j], [\boldsymbol{s}[j+1], \boldsymbol{s}[j]])$ then
9                  $N_{\text{pnp}} \leftarrow N_{\text{pnp}} + 1$;
10             if $\boldsymbol{s}[j] > \texttt{min}(\boldsymbol{s}[:j])$ then
11                 $N_{\text{unstable}} \leftarrow N_{\text{unstable}} + 1$;
12         $N_{\text{pnp}} \leftarrow N_{\text{pnp}} \cup N_{\text{pnp}}$;
13         $N_{\text{unstable}} \leftarrow N_{\text{unstable}} \cup N_{\text{unstable}}$;
14         $\boldsymbol{s} \leftarrow$ Find a new sequence from $\boldsymbol{t}$;
15     $indexes \leftarrow \texttt{ascending\_sort}(N_{\text{pnp}}, N_{\text{unstable}})$;
16     $\boldsymbol{t} \leftarrow \boldsymbol{t}[indexes]$;
17     return $\boldsymbol{t}$
```

based on the bending parameters obtained from sequence planning to reason IK-feasible and collision-free grasps. The robot will hold a wire using reasoned grasping poses. Motion planning accepts the robot's holding configurations and generates the joint motion among the reasoned grasps. The robot will move the wire following the generated motion.

Fig. 5.10(a-c) visualizes the example. Fig. 5.10(a.i) and (a.ii) illustrate a single grasping pose and all pre-annotated grasping poses defined based on a metal wire primitive, respectively. Fig. 5.10(b.i) and (b.ii) illustrate the optimal bending sequence obtained by the task-level planner. Fig. 5.10(c) shows the grasp reasoning details. The pre-annotated grasps are transformed to the local metal wire primitives at the holding positions of the two bending actions. The reasoner examines their collisions and feasibility, as illustrated in (c.i). The grasps that are simultaneously accessible to all holding positions in the sequence will be kept as the "common grasps", as illustrated in (c.ii). The robot can kinematically hold the wire using a "common grasp" at one wire pose and move it to another pose without changing the

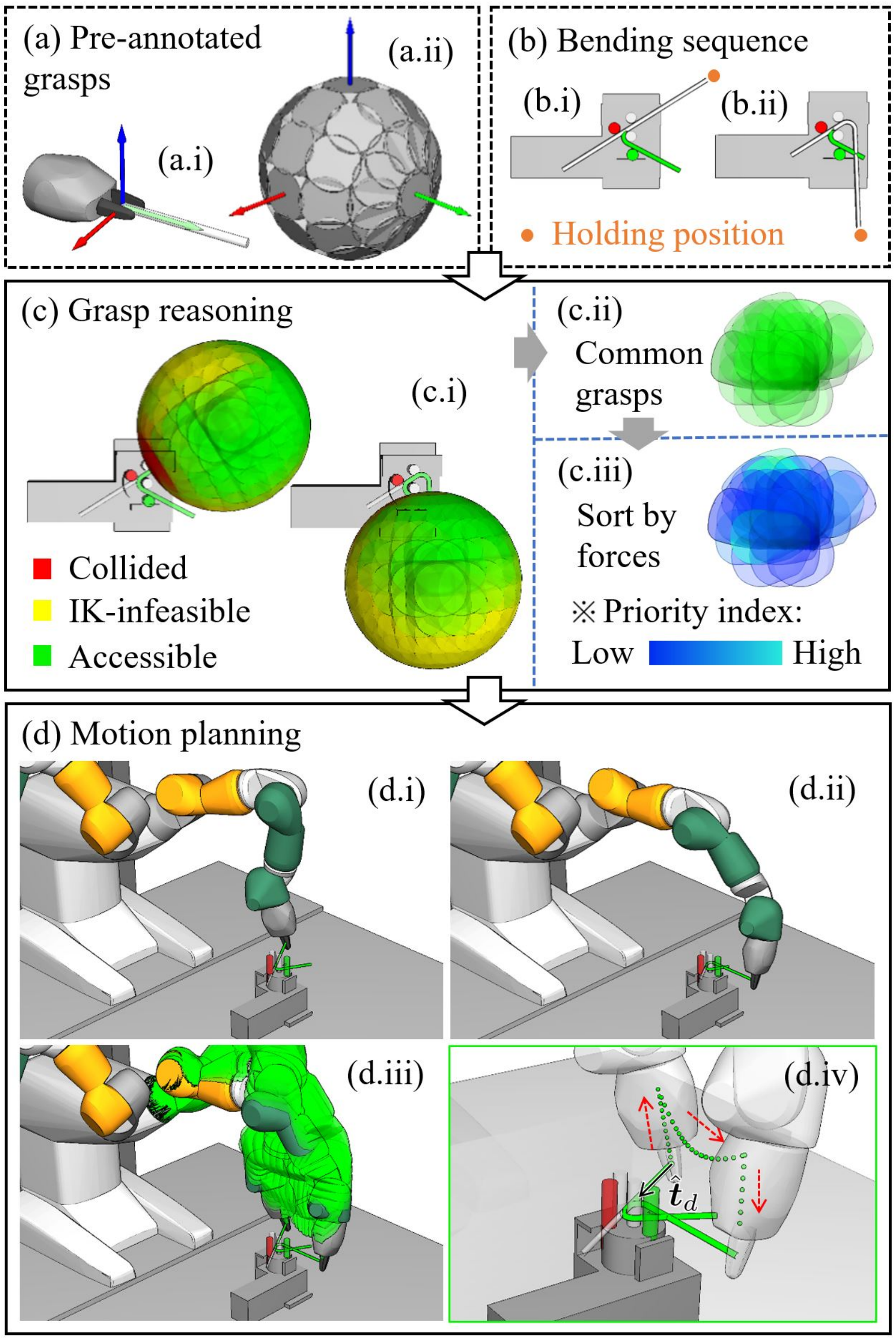


Figure 5.10: Workflow of grasp reasoning and motion planning. The inputs are shown in dashed boxes on the top. The results are in the solid boxes below. (a) Pre-annotated grasps. (b) Planned bending sequence. (c) Grasp reasoning. (d) Motion planning.

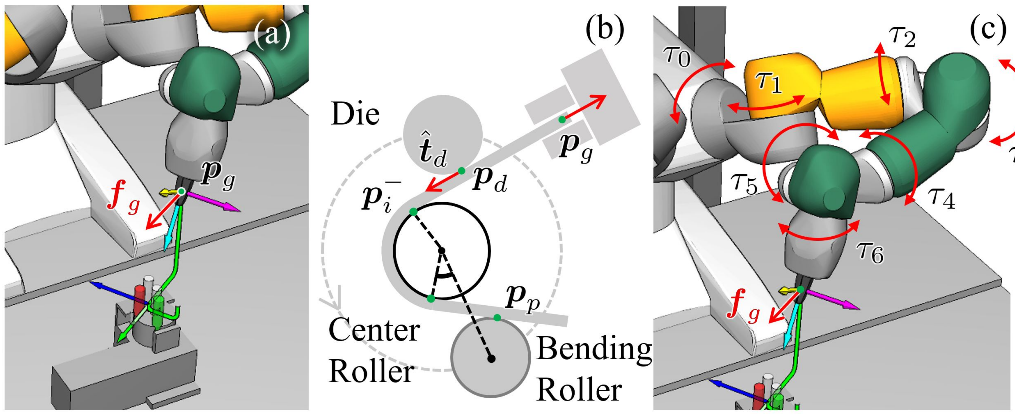


Figure 5.11: (a) A robot configuration at one bending action point. (b) Free-body diagram for a metal wire at an intermediate motor configuration.

grasp.

Despite the kinematic feasibility of the "common grasps", they do not necessarily meet force requirements. As shown in Fig. 5.11, the metal wire is supported by the die and center rollers and pressed by the bending roller. Although we used rolling mechanisms to reduce friction, a large pressing force may still induce friction forces and result in a heavy reaction load on the robot. The heavy reaction load may cause the robotic system to fall into an emergency. Thus, after obtaining the common grasps, we further rate them, considering the maximum force the robot can bear along the bending direction to ensure safe robot executions. The highly ranked "common grasp" will be selected for motion planning.

The force vector $\boldsymbol{f}_g$ in Fig. 5.11(a) and (b) illustrates the reaction load induced by the machine. The robot grasps the metal wire at $\boldsymbol{p}_g$. The wire contacts with the die, bending roller, and center roller respectively at $\boldsymbol{p}_d$, $\boldsymbol{p}_p$, and $\boldsymbol{p}_i^-$. The robot bears an reaction load $\boldsymbol{f}_g$ at $\boldsymbol{p}_g$. Its direction is along the tangential vector $\hat{\boldsymbol{t}}_d$ of the die at $\boldsymbol{p}_d$. For each "common grasp", we rated it considering the maximum force that the robot can bear along $\hat{\boldsymbol{t}}_d$. The highly ranked "common grasp" allowed bearing a large force. We chose it to resist the reaction load and avoid an overloaded emergency.

Specifically, we formulate the maximum force that the robot can bear along $\hat{\boldsymbol{t}}_d$ as the following optimization problem. The goal is to maximize equation (5.5a)

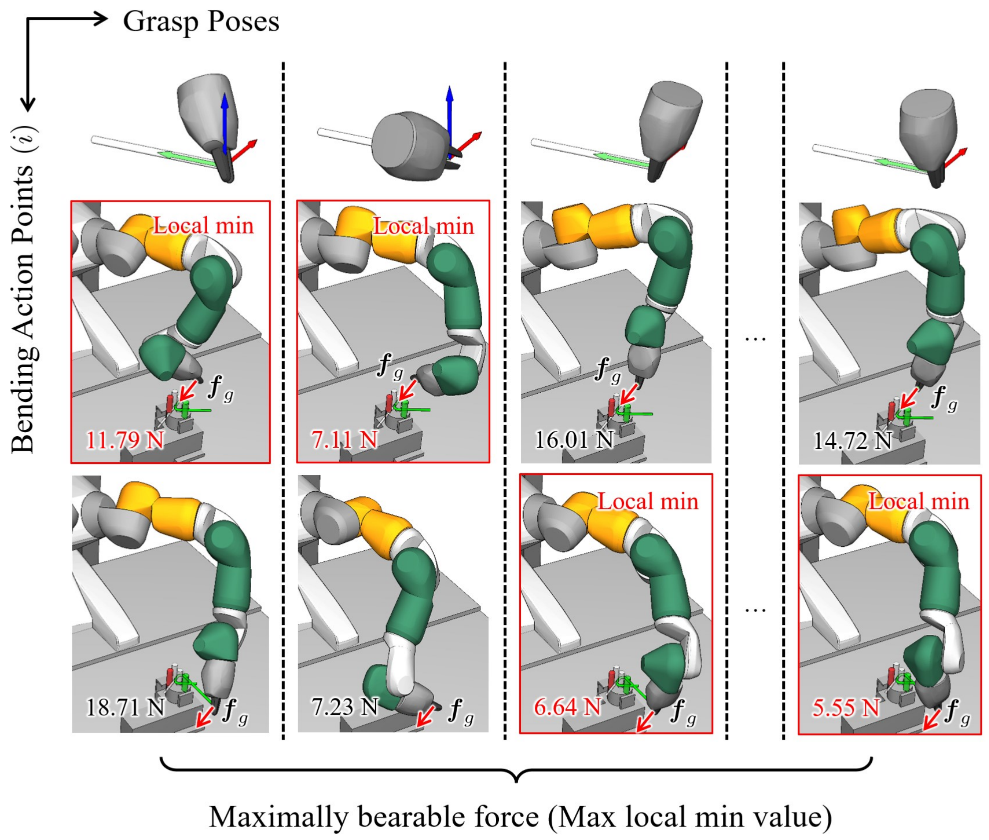


Figure 5.12: The robot will hold and move the wire to all bending poses with a "common grasp". Each column in the figure represents such a "common grasp" and the correspondent bending poses. The best grasping pose is selected as the one that has the maximally bearable force across the columns, which is the largest "Local min".

considering the "common grasp" pose, wire pose, and the torque limits of each robot joint. Here, ${}^{w}\mathrm{F}$ is the robot's output TCP (Tool Center Point) force described in the world coordinate system. ${}^{\hat{\boldsymbol{t}}_d}\mathrm{P}_w$ is the projection matrix that projects the TCP force on to vector $\hat{\boldsymbol{t}}_d$. ${}^{m}\mathrm{T}_g$ is the "common grasp" described in the coordinate system of a metal wire primitive. It is a homogeneous matrix containing the position and rotation of a grasping pose. ${}^{w}\mathrm{T}_m$ is the transformation matrix from the coordinate system of a metal wire primitive to the coordinate system of the local metal wire primitive at the holding position. $J(\boldsymbol{q})$ is the jacobian matrix. n_j is the number of robot joints.

τ_k^-, τ_k^+, q_k^-, q_k^+ are respectively the torque and rotational limits of each joint. Link weights and joint frictions are ignored for simplicity.

$$\underset{\boldsymbol{q}}{\text{argmax}} \quad \|{}^{\hat{\boldsymbol{t}}_d}\mathrm{P}_w{}^w\mathrm{F}\| \tag{5.5a}$$

$$\text{s.t.} \quad {}^w\mathrm{T}_m{}^m\mathrm{T}_g = J(\boldsymbol{q})\boldsymbol{q} \tag{5.5b}$$

$$ {}^w\mathrm{F} = J(\boldsymbol{q})\boldsymbol{\tau} \tag{5.5c}$$

$$\begin{cases} \boldsymbol{\tau} = \{\tau_k\}, \tau_k \in [\tau_k^-, \tau_k^+] \\ \boldsymbol{q} = \{q_k\}, q_k \in [q_k^-, q_k^+] \\ k \in [0, ..., n_j] \end{cases} \tag{5.5d}$$

Note that a "common grasp" is not used for a single bending pose. Instead, the robot will hold the wire with the "common grasp" and move it to all bending poses. The maximally bearable force of a "common grasp" is thus the minimum of all bending poses. Fig. 5.12 shows an example. In each column, the robot holds and moves the wire to different poses using the same "common grasp". Each column's minimum bearable force ("Local min") determines the performance of the grasping pose shown at the beginning. The maximally bearable force of all grasping poses is the largest "Local min" across the columns.

After rating the "common grasps", a motion planner will iterate through them sequentially according to their rank until a feasible joint space motion is found. The following prescribed motion primitives are defined to improve motion planning efficiency:

i) Pulling motion primitive. When the segment between the current bending position and the next bending position is straight, the robot does not need to lift the metal wire. Instead, it re-poses the metal wire by pulling and rotating between the rollers.

ii) Pick-and-place motion primitive. When it is difficult to reason a motion in-between the rollers, the robot performs pick-and-place motion planning to re-pose the metal wire.

Fig. 5.10(d) exemplifies a planned pick-and-place motion. A zoomed-in view is shown in Fig. 5.10(d.iv). The green dots show the end-effector position during the motion, and the dashed arrow indicates the moving direction. We required the robot to incline to $\hat{\boldsymbol{t}}_d$ to avoid large frictional resistances during the planning.

5.4.3 Online Springback Compensation

Since the metal wire used in this study is made of an elastoplastic material, it will exhibit springback when released from the forces of the bending roller. Previous studies [176] concluded that an elastoplastic metal wire's springback angle and bending angle have a positive relationship when the bending radius is fixed and developed analytical methods to compute the springback angle of a metal wire with known physical properties. In this work, we assume the physical properties of the target elastoplastic metal wire is unknown and propose using online visual sensing to estimate the springback. Notably, we represent the springback with a linear model following previous conclusions and leverage ridge regression to dynamically estimate and update the model. The linear model is formulated as

$$\delta\theta = f(\theta) = a\theta + b, \tag{5.6}$$

where $\delta\theta$ is the springback angle, θ is the bending angle. The ridge regression is formulated as

$$\underset{a,b}{\operatorname{argmin}} \sum_{i=0}^{n} \|\delta\theta_i - (a\theta_i + b))\|_2 + \lambda\|a\|, \tag{5.7}$$

where i is the id of a bending action. λ is the regularization weight of ridge regression. The parameters a, b are re-estimated each time a new $(\theta_i,\ \delta\theta_i)$ pair is obtained. The machine will be adjusted to bend an additional springback angle estimated using the latest model to improve precision.

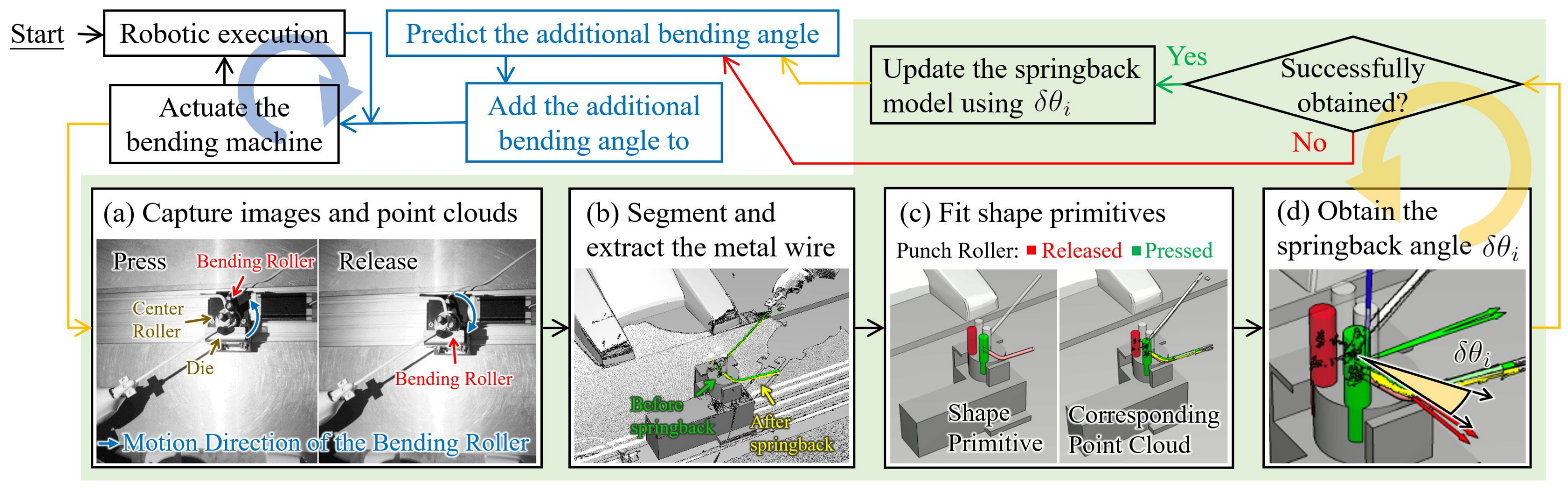

Start
Robotic execution
Actuate the bending machine
Predict the additional bending angle
Add the additional bending angle to
Update the springback model using $\delta\theta_i$
Yes
Successfully obtained?
No
(a) Capture images and point clouds
Press
Bending Roller
Center Roller
Die
Release
Bending Roller
Motion Direction of the Bending Roller
(b) Segment and extract the metal wire
Before springback
After springback
(c) Fit shape primitives
Punch Roller: 🟥 Released 🟩 Pressed
Shape Primitive
Corresponding Point Cloud
(d) Obtain the springback angle $\delta\theta_i$
$\delta\theta_i$

Figure 5.13: Workflow of online springback compensation using visual estimation. The green shadowed region indicates the main workflow to update the springback model. The blue box indicates springback prediction and compensation. The blue and yellow arrows are the same as the ones with the same color in Fig. 5.7.

Fig. 5.13 illustrates the workflow of dynamic visual estimation and bending angle adjustment. It is carried out as a closed loop during execution. The additional bending angle is adjusted dynamically considering the latest springback model. Especially in Fig. 5.13(c), we fit the shape primitive (mesh model from $\boldsymbol{p_i}$ to $\boldsymbol{p_n}$) to the extracted point cloud and compute the transformations before and after releasing the bending roller as $[R_p, t_p]$, $[R_r, t_r]$. We ignore the displacement from t_p to t_r and compute the springback angle as $\delta\theta_i = \arccos(\frac{Tr(R_p \cdot Rr)-1}{2})$, where $Tr(\cdot)$ indicates the trace of a matrix. When occlusion happens, we may not be able to fit shape primitives and obtain a springback angle. In that case, the system will not update the springback model but jump to additional bending angle prediction with the most recent update. The red arrow in Fig. 5.13 shows the workflow in the presence of occlusion. The blue and yellow arrows are coherent with the ones that have the same color in Fig. 5.7. The blue arrows and boxes indicate the dynamic compensation. The yellow arrows indicate the update. They represent two loops as shown by the two circled arrows. The two loops run in parallel and yellow loop is one step late than the blue loop.

An exception is the first bending action. In the beginning, we do not have a good guess about the springback model and thus cannot predict an additional bending angle. We solve the problem by performing a two-time bending action. The first time, we estimate a springback angle and use it to update the springback model. We further bend an additional angle predicted using the updated springback model the second time. The two-time bending action is dedicated to the first bending action point and does not apply to others.

5.5 Experiments and Analysis

In this section, we carry out experiments to examine the proposed methods. The section comprises two parts, where the first part focuses on individual processes, and the second part studies the overall performance on real-world bending tasks. The computation device used in our experiments is a PC with an Intel Core i7-10700 CPU and 16 GB memory. We tested the proposed methods on two robot platforms, a self-assembled robot platform made of two UR3e arms (Max payload of each arm:

3 kg) and two Robotiq Hand-E two-finger parallel grippers, and an ABB YuMi dual-arm robot (Max payload of each arm: 0.5 kg). A Photoneo PhoXi 3D Scanner M with 1032×772 resolution is mounted on the top of the table for visual perception. The bending machine is fixed on a work table under the robot's arms. The Photoneo scanner detects the pose of the machine.

5.5.1 Performance of Individual Processes

Bending Set Representation

First, we evaluated the performance of the coarse-to-fine bending parameterization algorithm use a specific curve shown in Fig. 5.14. We were especially interested in the relation between the number of pivot points (n) and the representation errors in the presence of different center roller radiuses (r_c). Thus, we carried out experiments to observe the changes of representation errors under varying n (5∼50) and r_c (10 mm, 7.5 mm, and 5 mm). The first two rows of Fig. 5.15 show the results. We particularly compared two representation errors, including the maximum difference between $\mathcal{D}$ and $\mathcal{C}$, as shown in the first row of Fig. 5.15, and the average difference between $\mathcal{D}$ and $\mathcal{C}$, as shown in the second row. The results indicate that smaller center roller radiuses lead to more minor representation errors. On the other hand, the representation errors do not decrease monotonically with an increasing n. The observation is coherent with our analysis in Section 5.3.2. The last row of Fig. 5.15 shows the relation between the n and the number of bending action points (m). The curves indicate that the pivot points (and their adapted arcs) were merged more heavily when the center roller radius was larger.

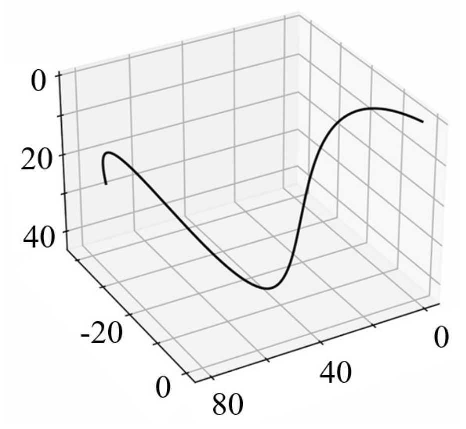


Figure 5.14: Desired curve.

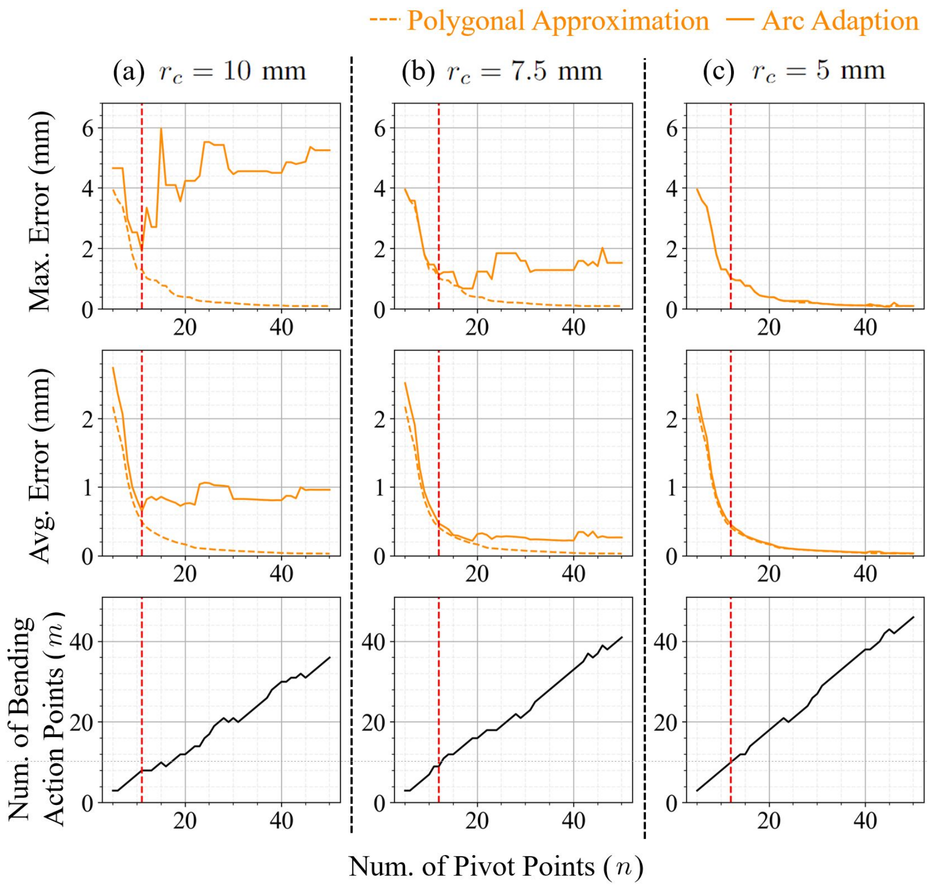


Figure 5.15: The relation between the number of pivot points (n) and max/average fitting errors, and also the relation between the n and the number of bending action points (size of $\mathbb{B}$, m), with different radius of the center pillar (r_c). r_c is 10 mm, 7.5 mm, and 5 mm from left to right, respectively. The wire radius (r_w) is 0.75 mm. The red dashed line highlights the best n found by our algorithm ($\epsilon = 0.5$ mm).

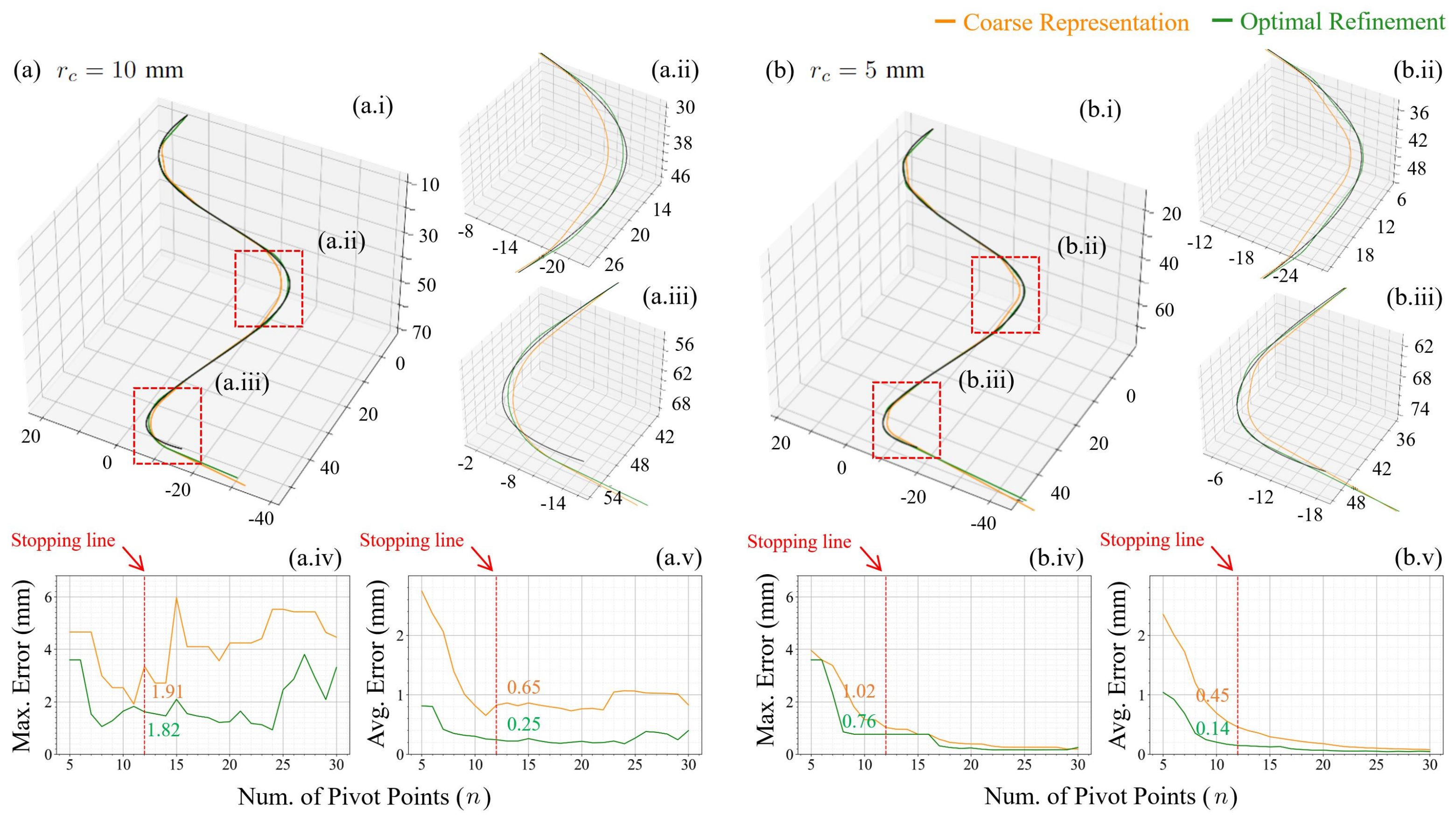


Figure 5.16: The 3D curves in the upper part show the coarse representation result and optimization result, respectively. The line graphs below show the max/average errors of different numbers of pivot points, respectively. Meanings of colors: Orange – coarse representation; Green – optimization. Values of variables: (a) $r_c = 10$ mm, $r_w = 0.75$ mm. (b) $r_c = 5$ mm, $r_w = 0.75$ mm.

Table 5.2: Evaluation of Bending Set Representation.

		$r_c = 5$ mm	$r_c = 7.5$ mm	$r_c = 10$ mm
	n	12.90 ± 2.10	13.25 ± 2.32	13.65 ± 2.63
Uniform Sampling	m	12.90 ± 2.10	13.25 ± 2.32	11.40 ± 1.32
	avg(e_c)	0.50 ± 0.12	0.50 ± 0.13	0.70 ± 0.22
	avg(e_o)	**0.18 ± 0.04**	0.19 ± 0.16	**0.19 ± 0.04**
	max(e_c)	1.81 ± 0.74	1.79 ± 0.73	2.34 ± 0.85
	max(e_o)	**0.72 ± 0.31**	0.88 ± 0.84	**0.86 ± 0.44**
Douglas-Peucker	m	**11.65 ± 1.49**	**11.40 ± 1.43**	**11.30 ± 1.14**
	avg(e_c)	**0.46 ± 0.03**	**0.48 ± 0.13**	**0.52 ± 0.16**
	avg(e_o)	**0.18 ± 0.03**	**0.18 ± 0.04**	**0.19 ± 0.07**
	max(e_c)	**1.19 ± 0.21**	**1.23 ± 0.39**	**1.41 ± 0.58**
	max(e_o)	0.78 ± 0.30	**0.75 ± 0.32**	**0.86 ± 0.48**

* Less number of bending action points (m) and lower error are highlighted in bold. Meanings of abbreviations: e_c - Error of coarse representation result; e_o - Error of optimized result; n - Number of pivot points. m - Number of bending action points.

We also evaluated the performance of optimal refinement. Fig. 5.16 shows the results. We especially compared r_c = 10 mm and 5 mm in Fig. 5.16(a) and (b) to see if 10 mm was acceptable, as we preferred using a large center roller to increase machine stiffness. Fig. 5.16(a.1) and (b.1) respectively show the goal curve, coarse representation result, and optimization result in black, orange, and green colors under the two different r_cs. The details of some sections with significant differences are zoomed-out in detail in (a.ii), (a.iii), (b.ii), and (b.iii). Fig. 5.16(a.iv), (a.v), (b.iv), and (b.v) respectively show the changes in maximum errors and average errors as the number of points increases. The red dashed line highlights the best n determined by our algorithm dashed lines (stopping line). The decrease in errors becomes less remarkable or inverses after the stopping line. The results indicate that the optimal refinement helped to reduce both maximum and average errors. Using a center roller with a larger radius was not significantly worse when the n was limited. The average errors were competitive with the results of a thinner roller (0.25 mm vs. 0.14 mm).

To further understand the efficiency of the proposed coarse representation method, we compared it with a baseline method, in which the points are uniformly sampled on the goal curve. We generated 20 random B-splines for the comparison, with each of them having the same length of 200 mm. The tolerance ϵ is set as 0.5 mm. The statistical information regarding the fitting results can be found in Table 5.2. Less number of bending action points and lower error are highlighted in bold. The results show that the DP method outperformed the baseline method with smaller errors and fewer bending action points (m).

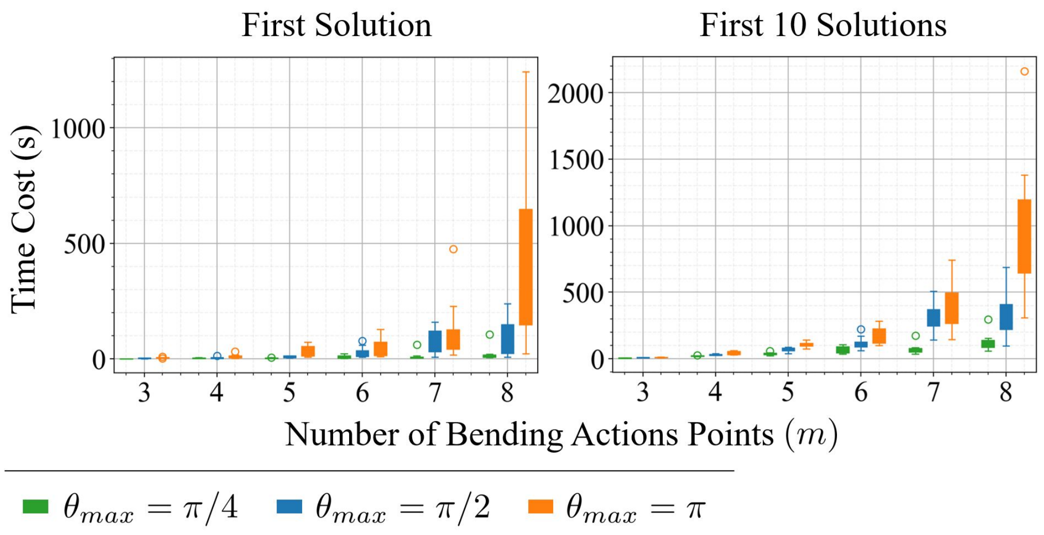
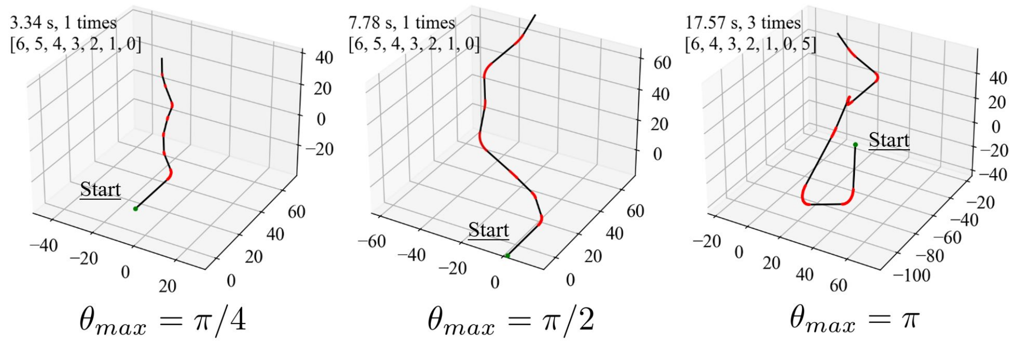


Figure 5.17: Average time costs of sequence planning vs. the number of bending action points (m). The green, blue, and orange colors correspond to random goal curves with $\lambda = 0.2$, $\theta_{\max} = \pi/4, \pi/2, \pi$, respectively. (a) Time costs to find the first solution. (b) Time costs to find the first ten solutions. If there were fewer than ten solutions, it indicates the time cost to find all of them.

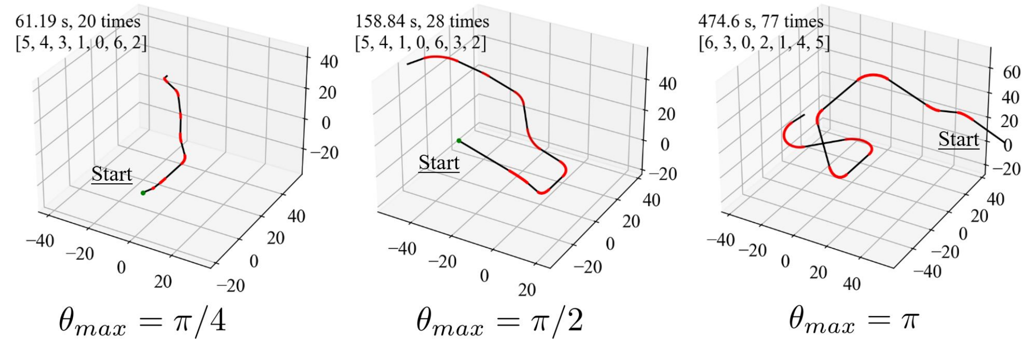
Sequence Planning

In this subsection, we demonstrate some planning results and examine the time efficiency of the sequence planner. The experiments were carried out by randomly generating bending sets and planning their order using the developed method. Especially, we randomized the parameters bending angle $\theta_i \in [\lambda\theta_{\max}, \theta_{\max}]$[2] and twisting

[2]Here, λ represents a percentage that controls the bending angles of the generated shapes.



(a) Random curves that required the minimum time costs



(b) Random curves that required the maximum time costs

Figure 5.18: Some sequence planning results when $m = 7$. The time cost, number of trials, and planned sequence are annotated on the top left of each figure.

angle $\alpha_i \in [-\pi, \pi]$. Larger bending $\theta_{\max}$ angles will result in more complicated curves. The start position of each bending action point, l_i, was set at a random position that did not overlap with others. Five groups of curves with the number of bending action points (m) in randomized bending sets ranging from 3 to 8 were evaluated. Each group included ten random desired curves with $\theta_{\max}$ of $\pi/4, \pi/2$ and π. The statistical cost for different numbers of bending action points is shown in Fig. 5.17. The time cost of finding the first and the first ten solutions are plotted separately. If there are fewer than ten solutions, it indicates the time cost to find all of them. Fig. 5.18 visualized some planning results when $m = 7$. The 3D curves visualize the goals. The time costs and planned sequences are annotated on the upper-left corner of each curve. Each value in the sequence represents an id of the red section. They start from the “start point” and the values increase incrementally as the sections get further. The start point is the holding positions used for grasp reasoning. Each column represents the result with different $\theta_{\max}$ values and comprises two rows. The upper row shows the curve required the minimum time cost. The lower row shows the curve required the maximum time cost.

Fig. 5.19 shows the planned bending result of a 3D curve with five bending action points. The (a.i) and (a.ii) lists shown below the diagram indicate two feasible planned bending sequences. The (a.i) list has smaller N_{pnp} and N_{unstable} and thus has higher priority for the following planning compared to (a.ii). The two lists are graphically illustrated in (b.i) and (b.ii), which provides a more intuitive understanding of the priority. The planned robot actions for (a.i) are illustrated in (c).

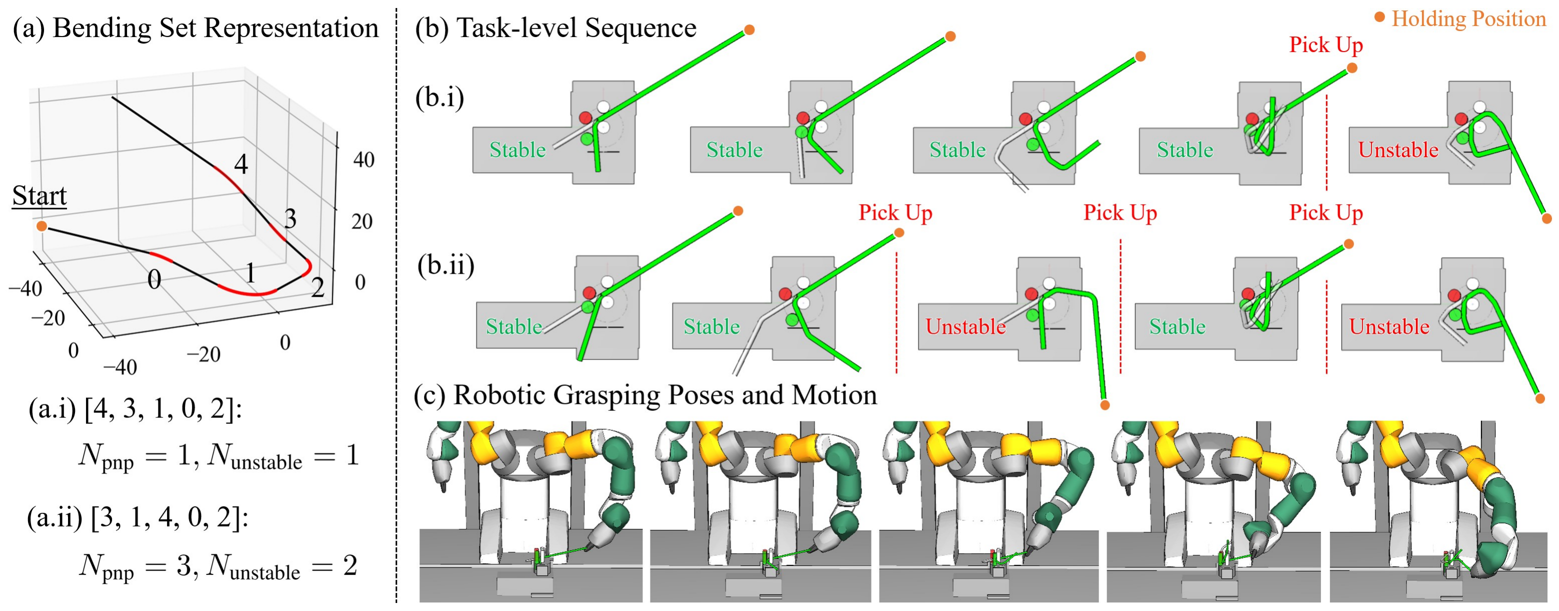


Figure 5.19: An exemplary sequence planning result. (a) The planned orders. Compared with (a.ii), (a.i) has smaller N_{pnp} and N_{unstable}, thus it has higher priority. (b) Detailed acting points and wire poses for each bending action of two candidate bending sequences. (c) Robot motion generated by the grasp reasoning and motion planning process. (The robot motion is not part of sequence planning. It is shown here to provide a complete view of the results).

Performance of Springback Compensation

To verify the performance of the proposed springback compensation model, we performed experiments using two different metal wires, including a 3.3 mm aluminum wire and a 2.6 mm galvanized steel wire, as shown in Fig. 5.20. Both of them wear rubber sleeves around their bodies. The aluminum wire's thickness was 6.0 mm with the sleeve. The galvanized steel wire's thickness changed to 2.9 mm with the sleeve. In the experiments, the robot held a wire using a random grasping pose and placed the wire between the bending machine's rollers horizontally. Then, the bending roller was actuated to a goal angle θ_i^{goal} to observe and update springback. We carried out the experiments 11 times by changing θ_i^{goal} from $\pi/12$ to $11\pi/12$ with $\pi/12$ interval. The bending results were observed after each experiment, and the springback model was dynamically updated with the observation.

The details are shown as follows. The first step was different as we did not have a model:

1. Actuate the bending roller of the machine to bend $\theta_0^{\text{goal}} = 15^\circ$ and capture a point cloud;

2. Release the roller and capture a second point cloud;

3. Segment and extract the metal wires from the two point clouds and obtain their difference $\delta\theta_0$.

4. Initialize the springback model $f(\theta)$ using θ_0^{goal} and $\delta\theta_0$. The springback model after initialization is saved as $f_0(\cdot)$

5. Activate the machine to additionally bend $\delta\theta_0$ and capture a third point cloud;

6. Segment and extract the metal wires from the third point cloud and obtain its difference from θ_0^{goal} as θ_0^{err}.

For the remaining steps, we repeated the following procedure and collected a series of $\delta\widetilde{\theta}_i$ (estimated springback), $\delta\theta_i$ (actual springback), and $\delta\theta_i^{\text{err}}$ (residual error between compensated bending result and the original goal):

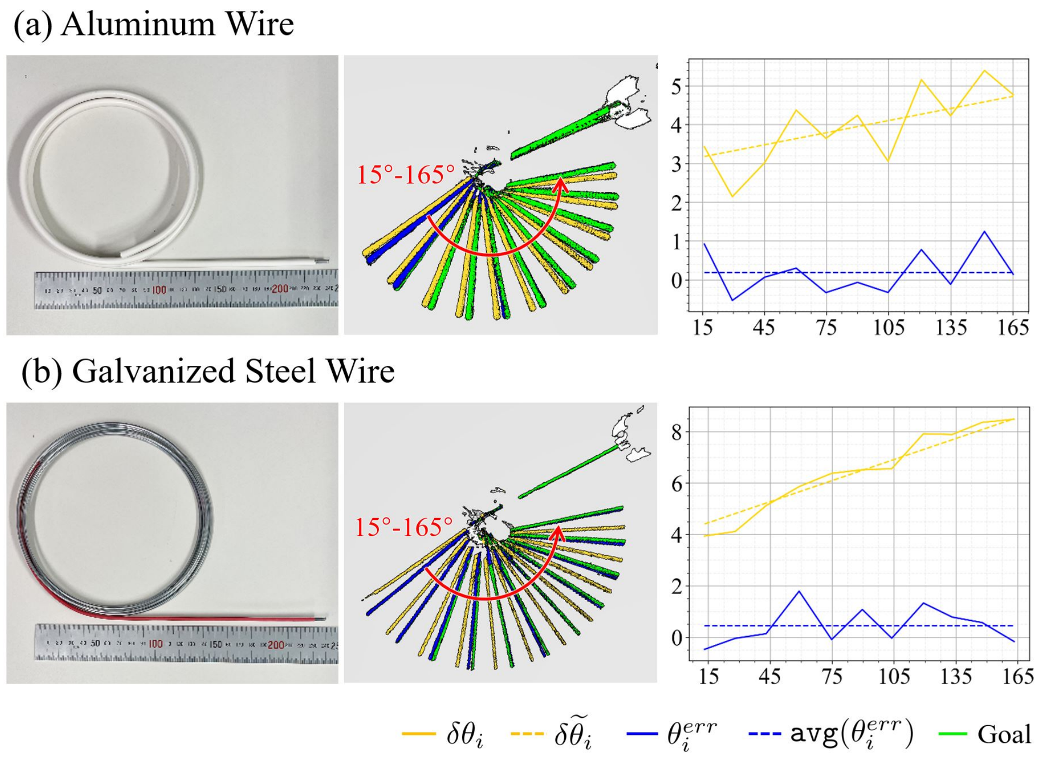


Figure 5.20: We perform experiments on two different metal wires. (a) Aluminum wire with a radius of 3.3 mm (6 mm with rubber sleeve); (b) Galvanized steel wire with a radius of 2.6 mm (2.9 mm with rubber sleeve).

1. Solve the corrected bending angle θ_i^{cor} by $\theta_i^{\text{goal}} = \theta_i^{\text{cor}} - f_{i-1}(\theta_i)$;

2. Actuate the bending roller of the machine to bend θ_i^{cor} and capture a point cloud;

3. Release the bending roller and capture a second point cloud;

4. Segment and extract the metal wires from the second point cloud and obtain its difference from θ_i^{goal} as θ_i^{err};

5. Segment and extract the metal wires from the first point cloud and its difference with the second point cloud as $\delta\theta_i$;

6. Update the model to $f_i(\cdot)$ using θ_i^{cor} and $\delta\theta_i$.

The actual springback angles ($\delta\theta_i$) are shown as the solid yellow line, and the estimated values ($\widetilde{\delta\theta_i}$) are shown as the dashed yellow line. The errors between the compensated bending results and the original goal angles (θ_i^{err}) are shown in the solid blue line. The average errors ($\texttt{avg}(\theta_i^{\text{err}}) = \frac{1}{n}\sum_{i=0}^{n}\theta_i^{\text{err}}$) are shown in the dashed blue line.

By observing the results, we found that the errors θ_i^{err} and $\texttt{avg}(\theta_i^{\text{err}})$ were steady and did not change much with dynamic updates. The observation differs from our expectation that the errors would decrease as we collected more data and improved the springback model. We guess the reason was that the metal wires were even, and the proposed linear model was good enough to predict a precise springback with very few data points. The observation was coherent with the conclusions presented in [176].

5.5.2 Performance of Real-World Executions

We studied the performance of the proposed method by planning the two robot arms, a 6-DoF UR3e and a 7-DoF ABB YuMi to bend different desired shapes using our bending machine. The position of the machine is well-calibrated using AR markers before the experiment. As the gripping force (20N) of the YuMi gripper is not enough to firmly grasp the metal wire at all poses, we fixed the metal wire on the finger using a 3D-printed case.

Three shapes are used to evaluate the complete workflow of the proposed system: (a) A 2D pentagon with a side length of 50 mm; (b) A 3D polygon with a side length of 40 mm; (c) The random generated 3D curve used in Section 5.5.1. The metal wires used were the same as Section 5.5.1, and the center roller's radius (r_c) was 7.5 mm. Fig. 5.21 shows the real-world execution results. The goal curves and planned sequences are shown in (a.i), (b.i), and (c.i). The robot grasping poses and motion are shown in (a.ii), (b.ii), and (c.ii). Pictures of the finished metal wires are shown in (a.iii), (b.iii), and (c.iii). For the first curve, we present the results of both the YuMi and UR3e robots. The YuMi robot was used to bend the thicker aluminum wire. The UR3e robot was used to bend the galvanized steel wire. We only show the results of the YuMi robot for the second and third curves since the UR3e robot failed to find a solution due to its lower DoFs. We encourage readers to watch the supplementary video for more details.

The various costs of the planner for each task are shown in Table 5.3 and 5.4. The "ABB YuMi (7-DoF)" and "UR3e (6-DoF)" tables correspond to the two robots. Under each robot, there are three shape names. They correspond to the three tasks shown in Fig. 5.21. The "Seq. Planning", "Grasp Reasoning", and "Motion Planning" sections of the table show the trials and failures of our planner's three planning processes and their time costs. The "Total Costs" row shows the total time to find the first solution. The "Max Bearable Force" row shows the maximum minimally bearable force encountered during grasp selection. Although the maximum payload of the YuMi robot is 0.5 kg (4.90 N), the force could reach 27.09 N for the "Pentagon" shape. The "Bend Angle vs. Springback" section shows the goal bending angle and the observed springback angle. Each item is represented as a sequence of numbers. They correspond to the sequential bending action points shown in Fig. 5.21. The $\times$ symbol means the vision system failed to observe a springback value due to occlusion. The last two columns of the table do not have valid data as the UR3e robot failed to find a solution for the "3D Polygon" and "3D Curve" shapes due to its lower DoF.

Complete Workflow

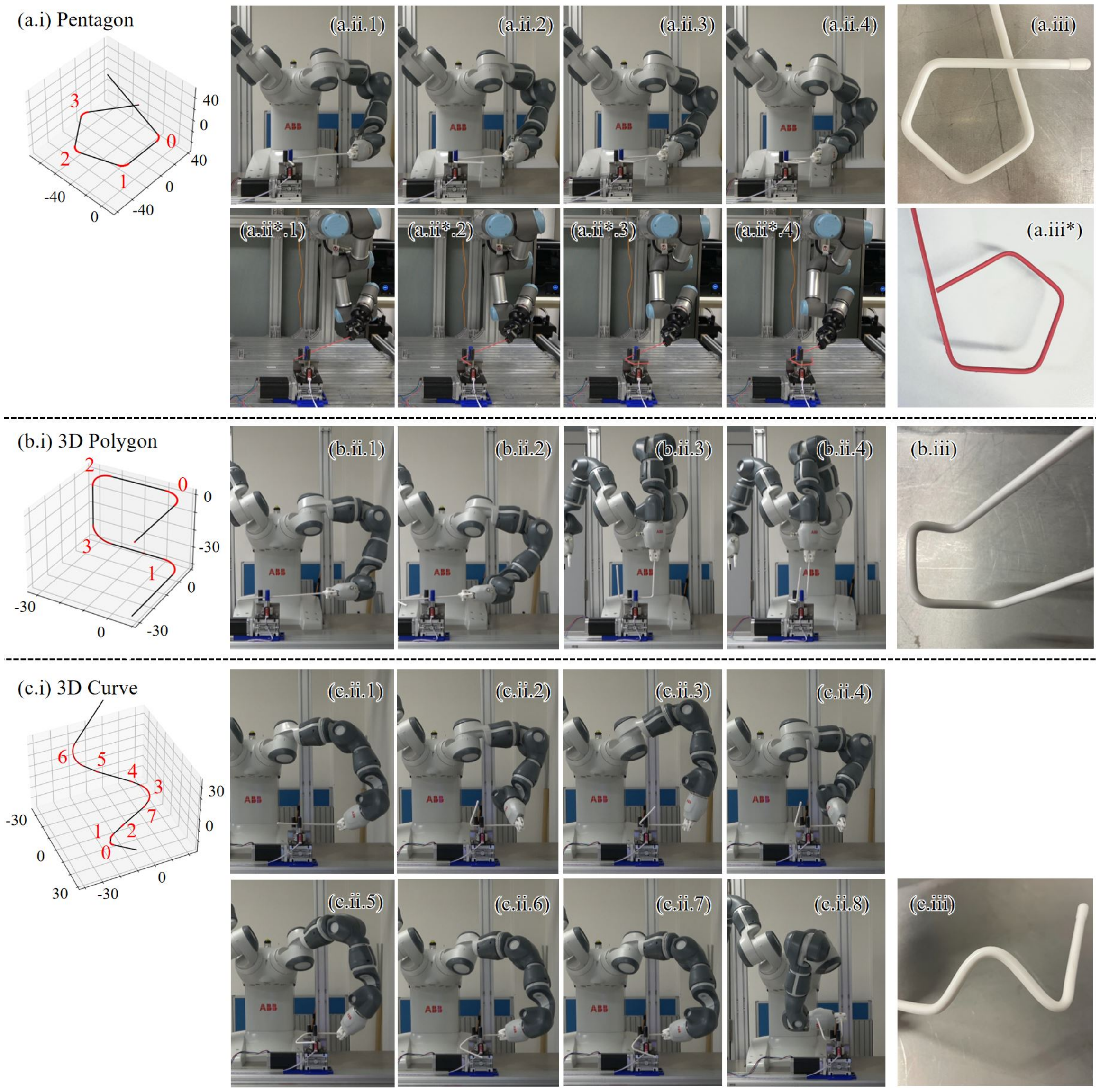


Figure 5.21: Real-world experiment of bending task. The goal shape is shown in the first column, and the planned bending sequence is annotated on it. Following are the corresponding robot motion and final result. (a-b) A 2D pentagon with a side length of 50 mm; (b) A 3D polygon with a side length of 40 mm; (c) The random generated 3D curve shown in Fig. 5.14. When the robot held the wire using the pose in (b.ii.3) and (b.ii.4), the friction force from the rollers not only pulled the robot but also caused an undesired deformation to the vertical section of the wire. The deformation resulted in a large bending error to the 3D polygon (Fig. 5.22(c)).

Table 5.3: Various Costs of the Planner for ABB YuMi (7-DoF), Aluminum Wire

		Pentagon	3D Polygon	3D Curve
Seq. Planning	# Att.	1	5	3
	Cost (s)	1.43	12.88	21.19
Grasp Reasoning	# Grasp	269/624	141/624	15/624
	Cost (s)	84.48	97.00	158.69
Motion Planning	# Att.	1	1	12
	Cost (s)	0.73	3.05	48.51
Total Cost (s)		86.65	112.93	228.40
Max Bearable Force (N)		27.09	19.66	6.13
Bend Angle vs. Springback	θ_i (°)	72.0/72.0/72.0/72.0	90.0/90.0/ 90.0/90.0	109.63/8.0/14.18/14.33/ 13.41/29.42/35.25/79.11
	θ_i^{cor} (°)	72.0/75.88/ 75.84/75.82	90.0/94.12/ 95.03/95.03	109.63/13.71/19.89/20.04/ 19.12/33.19/38.82/83.55
	$\delta\theta_i$ (°)	3.88/3.81/3.79/2.23	4.12/5.94/×/×	5.71/×/×/×/ 3.22/2.88/3.09/×

* Meanings of abbreviations: # Att. - Number of attempts to find the first solution; # Grasp - Number of accessible grasping poses. It is shown as a fraction number where the denominator represents the total number of pre-annotated grasps. The numerator represents the number of accessible ones. The last two rows show the goal bending angle vs. the observed springback. Each item is represented as a sequence of numbers. They correspond to the sequential bending action points shown in Fig. 5.21. The × symbol means the vision system failed to observe a springback value due to occlusion.

Table 5.4: Various Costs of the Planner for UR3e (6-DoF), Galvanized Steel Wire

		Pentagon	3D Polygon	3D Curve
Seq. Planning	# Att.	1	19	-
	Cost (s)	1.54	-	-
Grasp Reasoning	# Grasp	25/600	-	-
	Cost (s)	47.94	-	-
Motion Planning	# Att.	1	-	-
	Cost (s)	1.08	-	-
Total Cost (s)		50.56	508.53	>2000
Max Bearable Force (N)		186.22	-	-
Bend Angle vs. Springback	θ_i (°)	72.0/72.0/72.0/72.0	-	-
	θ_i^{cor} (°)	72.0/76.51/ 77.04/76.71		
	$\delta\theta_i$ (°)	4.51/5.57/4.06/4.17	-	-

* Meanings of abbreviations: # Att. - Number of attempts to find the first solution; # Grasp - Number of accessible grasping poses. It is shown as a fraction number where the denominator represents the total number of pre-annotated grasps. The numerator represents the number of accessible ones. The last two rows show the goal bending angle vs. the observed springback. Each item is represented as a sequence of numbers. They correspond to the sequential bending action points shown in Fig. 5.21. The × symbol means the vision system failed to observe a springback value due to occlusion.

We also conducted experiments to evaluate the resultant shapes of the above experiments globally. We captured the point clouds of resultant metal wires and compared them with the goal shapes to judge if the results were satisfying. In detail, we used the Iterative Closest Point (ICP) [112] algorithm to match the captured point clouds to the dense point set used to represent the bending goals and used the fitness (Root Mean Square Error, RMSE) of the ICP matching to evaluate judge their similarity. Fig. 5.22 shows pictures of the observed point clouds (red) and the dense representation points (green). The matched RMSE values are shown below each picture.

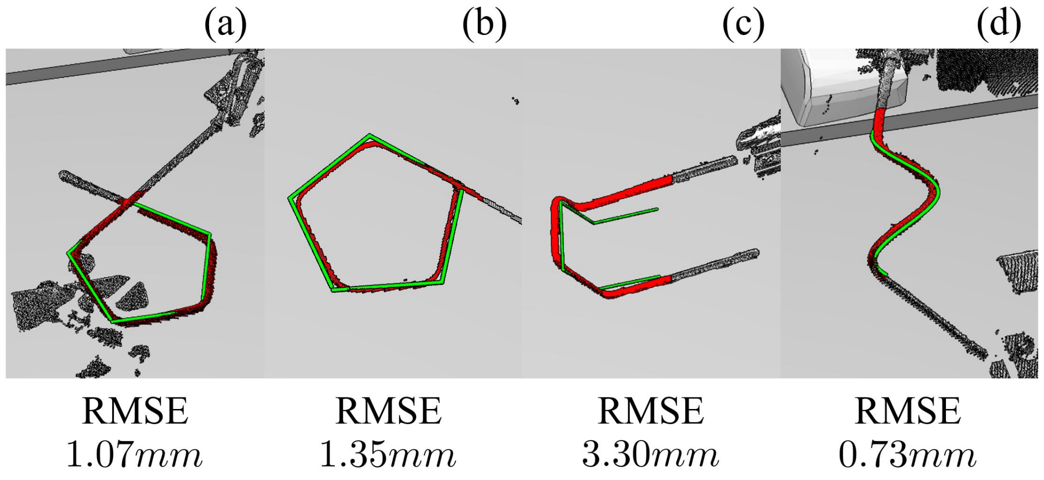


Figure 5.22: Evaluating the resultant shapes of the real-world experiments. The pictures compare the 3D point clouds of the bending results (observed point clouds, represented in red) and the goal shapes (dense points set in green color). The RMSE values of the ICP matching algorithm between them are shown below the pictures.

The matching results show that most RMSE errors were less than 1.50 mm and satisfying. However, Fig. 5.22(c) exhibited an exception. Its error (3.30 mm) was more significant than the others. We analyzed the reasons and found that when a robot held the wire using the pose shown in (b.ii.3) and (b.ii.4), the friction force from the rollers not only pulled the robot but also caused an undesired deformation to the vertical section of the wire. The deformation further resulted in a positional displacement of the bending point ($\boldsymbol{q}^{-}$) along the direction of $\hat{t}_d$. This is defined as N_{unstable} in Section 5.4.1. Including a fixture unit in the bending machine or employing

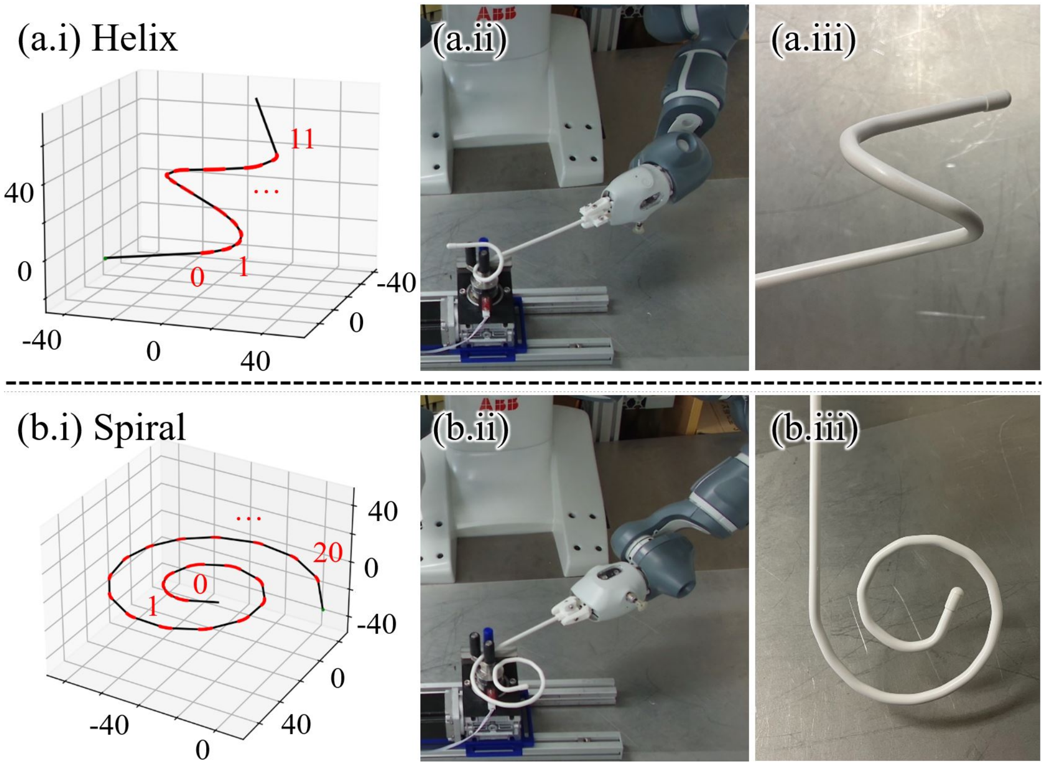


Figure 5.23: Other results. (a) A 3D helix; (b) A 2D spiral.

a robot with compliant control policies may help solve the problem.

Other Results

The proposed system also works for other shapes like spirals. In this part, we conducted two additional experiments to bend a helix and a spiral, respectively. The results are shown in Fig. 5.23. The geometry of the robot and machine may limit the curvature and sizes of target wires that can be processed. However, there are no constraints to a particular shape. Note that the springback model estimated by previous experiments is directly adopted without online updates in these experiments.

It is important to note that the smoothness of the bending results for shapes such as helix and spiral depends on the number of bending action points. We set the tolerance parameter in these experiments to be $\epsilon = 0.2$ mm. By adjusting this tolerance to a lower value, we can achieve a higher level of smoothness in the bending results.

5.6 Conclusions

We presented a combined task and motion planning-based planner for a robot arm to bend metal wire while collaborating with a bending machine. It enabled a low-payload robot arm to automatically curve metal wire with high stiffness. Visual estimation with dynamic update and prediction was used to compensate for the springback caused by elastic deformation during the bending process. We examined different bending results of the developed system and confirmed that our system was flexible and robust for generating robotic motion to corporate with the bending machine. It also exhibited satisfying precision with the help of springback compensation.

Besides these positive conclusions, we learned several lessons and understood several systematic limitations. First, as the metal wire was not firmly fixed on the machine, it might be pulled or dragged by the machine's bending rollers and thus cause an unaffordable payload to the robot. Although careful planning could help avoid large payloads, the deformation it caused led to significant bending errors. Including a fixture unit in the bending machine or employing a robot with compliant control policies may help solve the problem. Second, the metal wire must be strictly straight in the beginning to ensure that ensuing planning, execution, and feedback can be correctly performed. It will be helpful to further improve the bending machine by including a straightener unit. Third, the deformed wire was difficult to be captured by a single view, and the springback estimation may fail due to self-occlusion and robot-object occlusion. Developing a high-precision and active vision system will be helpful to fully capture the curving status and improve the springback compensation performance.

We are interested in working out the above problems to obtain better bending precision in future work. We would also like to extend the method to other types of objects, i.e., a metal plate. In addition, due to the limitation of our current system configuration, such as lacking a straightener unit, it remains difficult for us to ensure 100% precision. In particular cases, we have to invite a human to assist in positioning the wire correctly and preventing unexpected collisions. Entirely eliminating human intervention would be an important topic to explore. Finally, we are interested in using dual-arm coordination to perform the bending tasks. In the dual-arm configuration, the bending machine would be removed, and the robot manipulators will take all the roles. However, implementing such a system requires robots and grippers to have satisfying payloads and holding forces. We may work on it in the future when we have a more suitable dual-arm robotic platform.

Appendix

In this appendix, we discuss the mechanism details of the proposed system.

A challenge of the proposed system lies in the maximum gripping force of the ABB IRB 14000 gripper, which is 20 N. Given this constraint, the gripper is difficult to firmly grasp the metal wire at all poses using force closure-based grasping [177]. This difficulty arises due to the metal wire's thin and long nature, resulting in small contact surfaces and soft-finger contact friction. To address this issue, we fixed the metal wire on the finger using a 3D-printed case, as shown in Fig. 5.24. At the beginning of each experiment, we manually adjust the grasping pose before robotic executions.

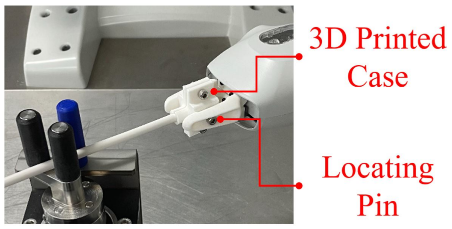


Figure 5.24: Wire fixture.

Additionally, it is vital that the wire is accurately positioned within the bending machine before the operation commences. To this end, we primarily worked on the following two steps to accomplish satisfying positioning. First, we manually calibrated the machine's position using AR markers before the experiment. The poses of the robot hand and bending machine were precisely known through this step. Second, we used tapered rollers to constrain the metal wire's pose. As shown in Fig. 5.25 ($d_1 \leq d_2 = d_3$), the wedge of such tapered rollers provided an error tolerance of $d_2 - d_1$. In the real setting, this tolerance was 2 mm.

It is also possible to check the contact state using force feedback. However, we

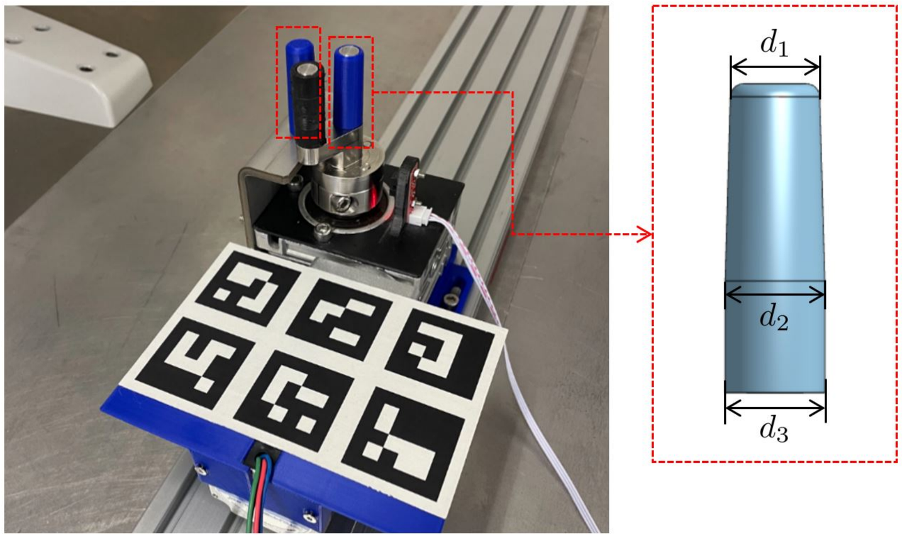


Figure 5.25: Two steps to achieve satisfying positioning: (1) We manually calibrated the machine's position using AR markers before the experiment; (2) We used tapered rollers to constrain the metal wire's pose.

did not implement it as the force sensor is not equipped for the robot. Joint current sensors were available, but they needed to be more precise to meet the requirements.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

6.1.1 System Overview

In this thesis, my primary focus lies in choosing and designing efficient algorithms for planning robot manipulation using general-purpose robot arms. This was achieved through the implementation of two specific robot manipulation systems. Visual and force perception, task and motion planning, and uncertainty management are leveraged for achieving reliable and robust manipulation systems. By employing general-purpose robot arms and grippers, the adaptability and flexibility of robots in accomplishing complex tasks were improved.

The robotic drawing system maps 2D strokes to 3D surfaces, plans pen picking and manipulation motion, and performs drawing on 3D surfaces considering vision and force feedback. The system showcases its flexibility by treating pens as tools that can be picked up and manipulated in real-time. Its robustness is illustrated through a closed-loop operation that utilizes high-quality-stroke mapping, in-hand pose estimation, motion refinement, force control, and error detection and recovery mechanisms to mitigate potential failures. Experimental results demonstrated excellent expected performance.

The robotic bending system integrates a view planning algorithm and a task and motion planner. The view planning component uses a point completion network-based NBV/NBC estimation method that can reconstruct the shape of the metal plate with a minimum number of scans. The task and motion planner facilitates the interaction of a low-payload robot arm with a bending machine, enabling it to autonomously curve high-stiffness metal wire. The system employs dynamic visual estimation and prediction to adjust for the springback effects caused by elastic deformation during the bending process. Different bending results of the developed system are examined to confirm that the proposed system was flexible and robust for generating robotic motion to corporate with the bending machine. It also exhibited satisfying precision with the help of springback compensation.

6.1.2 Planning Method Overview

The selected algorithms for task and motion planning across different systems are listed in Table 6.1. The summary is as follows:

1. **Drawing Task (Chapter 3):** The sub-tasks in this system present both discrete and continuous constraints. The task of picking up a pen, for instance, is discrete in nature. The initial and end poses can be effectively solved using an optimization-based method, while the intervening motion can be managed with a probabilistic method like the RRT*. And the task of drawing presents a more constrained, continuous problem where the path is predetermined and must be followed precisely. Probabilistic-based methods might not be as effective in this case due to their inherent randomness and the large solution spaces they generate. Hence, more deterministic methods that can precisely manage the constraints are generally preferable. Additionally, the choice of poses for recalibrating in-hand pose is handled through a sampling-based method due to two primary reasons: First, the constraints of this task are relatively flexible; Second, the calculations can be performed offline, significantly boosting efficiency.

Table 6.1: Planning Algorithm Overview.

	Optimization-based	Probabilistic-based
Chapter 3	Drawing Motion Planning	Grasp Reasoning; Other Motion Planning
Chapter 4	View Planning; IK Solving	Motion Planning
Chapter 5	Bending Action Representation; Bending Sequence Planning; Pulling Motion Planning	Grasp Reasoning; Lift-and-Place Motion Planning

2. **View Planning Task (Chapter 4):** A probabilistic-based method is employed to represent the object, which allows for effective handling of the uncertainties. The view pose is calculated using an optimization-based method, which ensures the most optimal solution is found. By combining these two methodologies, the task successfully balances efficiency and precision.

3. **Bending Task (Chapter 5):** The fully constrained bending action representation makes it a suitable candidate for an optimization-based method. On the other hand, the sub-task of selecting a grasp pose that meets the force constraint requires a broader exploration of the solution space, which correlates with the number of bending actions. This exploration is effectively conducted using a sampling-based method. The motion connecting bending actions is bifurcated: the lift-and-place motion is addressed by RRT*, whereas the pulling motion, constrained by rollers and necessitating only linear end-effector movement, is best handled by optimization-based methods.

In conclusion, this thesis shows the balance of planning algorithms can significantly enhance the performance of complex robotic manipulation systems. By delving deep into the difference between discrete and continuous constraints, and harnessing the strengths of various planning methodologies, it shows that probabilistic and deterministic strategies are not just stand-alone solutions. When synergistically integrated, they possess the ability to present efficiency, precision, and adaptability in robot manipulation systems.

6.2 Furture Work

While the robotic manipulation systems developed in this thesis demonstrate promising results, they are primarily tailored for specific tasks. Their adaptation to a wide array of tasks without substantial reprogramming or redesign poses a challenge. Future research may aim to enhance the versatility of these robotic manipulation systems.

- **Amodal Scene Representations:** Shifting from a specialized to a generalized approach, one research focus may be to develop amodal scene representations. This involves leveraging complete 3D representations of scenes regardless of occlusions and image field of view (amodal) [178] and utilizing contextual information from the entire scene as a whole (scene). This will essentially equip robots with a level of autonomy to perceive and interpret their environments in a more human-like manner.

- **General Task Planning Strategy:** Despite the substantial advancements made in automatic planning for high-level tasks, there is yet a significant gap to be bridged. The complexity of these tasks is often underpinned by numerous variables that our current computational models struggle to fully grasp and efficiently operationalize. And it is time-consuming to design algorithms for all real-world tasks. With the advent of Artificial General Intelligence (AGI), an increasing number of researchers are working towards the creation of a universal model capable of addressing complex task and motion planning challenges. Recent studies have started to leverage Large Language Models (LLM) for task and motion planning [179][180].

References

[1] Daniel Belanche, Luis V Casaló, Carlos Flavián, and Jeroen Schepers. Service robot implementation: a theoretical framework and research agenda. The Service Industries Journal, 40(3-4):203–225, 2020.

[2] Zhi Lon Gan, Siti Nurmaya Musa, and Hwa Jen Yap. A review of the high-mix, low-volume manufacturing industry. Applied Sciences, 13(3):1687, 2023.

[3] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez De Santos, and Manuel Armada. The evolution of robotics research. IEEE Robotics & Automation Magazine, 14(1):90–103, 2007.

[4] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. Science, 364(6446):eaat8414, 2019.

[5] Weiwei Wan. Efficient caging planning under uncertainty based on configuration spaces of target object and finger formation. PhD thesis, Tokyo University, 2013.

[6] Matthew T Mason. Toward robotic manipulation. Annual Review of Control, Robotics, and Autonomous Systems, 1:1–28, 2018.

[7] Kaidi Nie, Weiwei Wan, and Kensuke Harada. A hand combining two simple grippers to pick up and arrange objects for assembly. IEEE Robotics and Automation Letters, 4(2):958–965, 2019.

[8] Zhengtao Hu, Weiwei Wan, and Kensuke Harada. Designing a mechanical tool for robots with two-finger parallel grippers. IEEE Robotics and Automation Letters, 4(3):2981–2988, 2019.

[9] Deepayan Bhowmik and Kofi Appiah. Embedded vision systems: A review of the literature. In Applied Reconfigurable Computing. Architectures, Tools, and Applications: 14th International Symposium, pages 204–216. Springer, 2018.

[10] Roxana Leontie. Proprioceptive and Kinesthetic Feedback for Robotic Manipulation. PhD thesis, The George Washington University, 2019.

[11] YF Li and XB Chen. On the dynamic behavior of a force/torque sensor for robots. IEEE Transactions on Instrumentation and Measurement, 47(1):304–308, 1998.

[12] Wei Chen, Heba Khamis, Ingvars Birznieks, Nathan F Lepora, and Stephen J Redmond. Tactile sensors for friction estimation and incipient slip detection—toward dexterous robotic manipulation: A review. IEEE Sensors Journal, 18(22):9049–9064, 2018.

[13] Yanmei Li and Imin Kao. A review of modeling of soft-contact fingers and stiffness control for dextrous manipulation in robotics. In IEEE International Conference on Robotics and Automation, volume 3, pages 3055–3060. IEEE, 2001.

[14] Yu Wang and Matthew Mason. Modeling impact dynamics for robotic operations. In IEEE International Conference on Robotics and Automation, volume 4, pages 678–685. IEEE, 1987.

[15] Veronica E Arriola-Rios, Puren Guler, et al. Modeling of deformable objects for robotic manipulation: A tutorial and review. Frontiers in Robotics and AI, 7:82, 2020.

[16] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. Annual review of control, robotics, and autonomous systems, 4:265–293, 2021.

[17] Raymond C Goertz. A force-reflecting positional servomechanism. Nucleonics, 10(11):43–45, 1952.

[18] Stephanie L Kilian. Coordination of Continuous and Discrete Components of Action. PhD thesis, Cleveland State University, 2014.

[19] Caixia Cai, Ying Siu Liang, Nikhil Somani, and Wu Yan. Inferring the geometric nullspace of robot skills from human demonstrations. In IEEE International Conference on Robotics and Automation, pages 7668–7675. IEEE, 2020.

[20] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation, 12(4):566–580, 1996.

[21] Steven M LaValle, James J Kuffner, BR Donald, et al. Rapidly-exploring random trees: Progress and prospects. Algorithmic and computational robotics: new directions, 5:293–308, 2001.

[22] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7):846–894, 2011.

[23] Jeffrey Ichnowski, Michael Danielczuk, Jingyi Xu, Vishal Satish, and Ken Goldberg. Gomp: Grasp-optimized motion planning for bin picking. In 2020 IEEE International Conference on Robotics and Automation, pages 5270–5277. IEEE, 2020.

[24] Jürgen Hess, Gian Diego Tipaldi, and Wolfram Burgard. Null space optimization for effective coverage of 3d surfaces using redundant manipulators. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1923–1928. IEEE, 2012.

[25] Siyu Dai. Probabilistic motion planning and optimization incorporating chance constraints. PhD thesis, Massachusetts Institute of Technology, 2018.

[26] Abubakar Sulaiman Gezawa, Yan Zhang, Qicong Wang, and Lei Yunqi. A review on deep learning approaches for 3d data representations in retrieval and classifications. IEEE Access, 8:57566–57593, 2020.

[27] Michael E Mortenson. Mathematics for computer graphics applications. Industrial Press Inc., 1999.

[28] Carl De Boor. On calculating with b-splines. Journal of Approximation theory, 6(1):50–62, 1972.

[29] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, and Christian Röossl. Geometric modeling based on polygonal meshes. 2007.

[30] Jules Bloomenthal and Chandrajit Bajaj. Introduction to implicit surfaces. Morgan Kaufmann, 1997.

[31] Michael Lounsbery, Stephen Mann, and Tony DeRose. Parametric surface interpolation. IEEE Computer Graphics and Applications, 12(5):45–52, 1992.

[32] Les Piegl. On nurbs: a survey. IEEE Computer Graphics and Applications, 11(01):55–71, 1991.

[33] Vadim Shapiro. Solid modeling. Handbook of computer aided geometric design, 20:473–518, 2002.

[34] Aristides G Requicha. Representations for rigid solids: Theory, methods, and systems. ACM Computing Surveys, 12(4):437–464, 1980.

[35] Worthy N Martin and Jagdishkumar Keshoram Aggarwal. Volumetric descriptions of objects from multiple views. IEEE transactions on pattern analysis and machine intelligence, (2):150–158, 1983.

[36] Michael Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. Computer Vision, Graphics, and Image Processing, 40(1):1–29, 1987.

[37] Aristides AG Requicha and Herbert B Voelcker. Constructive solid geometry. 1977.

[38] Hüseyin Erdim and Horea T Ilieş. Classifying points for sweeping solids. Computer-Aided Design, 40(9):987–998, 2008.

[39] Ray A Jarvis. A laser time-of-flight range scanner for robotic vision. IEEE Transactions on pattern analysis and machine intelligence, (5):505–512, 1983.

[40] Max Yiye Cao, Stephen Laws, and Ferdinando Rodriguez y Baena. Six-axis force/torque sensors for robotics applications: A review. IEEE Sensors Journal, 21(24):27238–27251, 2021.

[41] Liang Zou, Chang Ge, Z Jane Wang, Edmond Cretu, and Xiaoou Li. Novel tactile sensor technology and smart tactile sensing systems: A review. Sensors, 17(11):2653, 2017.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International journal of computer vision, 115:211–252, 2015.

[43] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. Computational intelligence and neuroscience, 2018, 2018.

[44] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. IEEE Transactions on Robotics, 33(6):1273–1291, 2017.

[45] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence, 33(5):898–916, 2010.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015.

[47] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3):257–276, 2023.

[48] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. Robotics: Science and Systems XIV, 2018.

[49] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3706–3715, 2020.

[50] Matthew T Mason. Compliance and force control for computer controlled manipulators. IEEE Transactions on Systems, Man, and Cybernetics, 11(6):418–432, 1981.

[51] Fuyuki Sato, Tatsuya Nishii, Jun Takahashi, Yuki Yoshida, Masaru Mitsuhashi, and Dragomir Nenchev. Experimental evaluation of a trajectory/force tracking controller for a humanoid robot cleaning a vertical surface. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3179–3184. IEEE, 2011.

[52] Daeun Song, Taekhee Lee, and Young J Kim. Artistic pen drawing on an arbitrary surface using an impedance-controlled robot. In IEEE International Conference on Robotics and Automation, pages 4085–4090, 2018.

[53] Jianhua Li, Siyuan Dong, and Edward Adelson. Slip detection with combined tactile and visual information. In IEEE International Conference on Robotics and Automation, pages 7772–7777. IEEE, 2018.

[54] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. Sensors, 17(12):2762, 2017.

[55] Jasper Wollaston James and Nathan F Lepora. Slip detection for grasp stabilization with a multifingered tactile robot hand. IEEE Transactions on Robotics, 37(2):506–519, 2020.

[56] Rocco A Romeo, Clemente Lauretti, Cosimo Gentile, Eugenio Guglielmelli, and Loredana Zollo. Method for automatic slippage detection with tactile sensors embedded in prosthetic hands. IEEE Transactions on Medical Robotics and Bionics, 3(2):485–497, 2021.

[57] Malik Ghallab, Dana Nau, and Paolo Traverso. Automated planning and acting. 2016.

[58] Kai Zhang, Eric Lucet, Julien Alexandre Dit Sandretto, Selma Kchir, and David Filliat. Task and motion planning methods: applications and limitations. In International Conference on Informatics in Control, Automation and Robotics, pages 476–483, 2022.

[59] Jason Andrew Wolfe, Bhaskara Marthi, and Stuart J Russell. Combined task and motion planning for mobile manipulation. In International Conference on Automated Planning and Scheduling, pages 254–258, 2010.

[60] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In IEEE International Conference on Robotics and Automation, pages 639–646, 2014.

[61] Rachel Holladay, Tomás Lozano-Pérez, and Alberto Rodriguez. Planning for multi-stage forceful manipulation. In IEEE International Conference on Robotics and Automation, pages 6556–6562, 2021.

[62] Marc Toussaint, Jung-Su Ha, and Danny Driess. Describing physics for physical reasoning: Force-based sequential manipulation planning. IEEE Robotics and Automation Letters, 5(4):6209–6216, 2020.

[63] Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning symbolic operators for task and motion planning. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3182–3189, 2021.

[64] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. The International Journal of Robotics Research, 40(6-7):866–894, 2021.

[65] Mohamed Raessa, Daniel Sánchez, Weiwei Wan, Damien Petit, and Kensuke Harada. Teaching a robot to use electric tools with regrasp planning. CAAI Transactions on Intelligence Technology, 4(1):54–63, 2019.

[66] Hao Chen, Weiwei Wan, and Kensuke Harada. Combined task and motion planning for a dual-arm robot to use a suction cup tool. In IEEE-RAS International Conference on Humanoid Robots, pages 446–452, 2019.

[67] Daniel Sanchez, Weiwei Wan, and Kensuke Harada. Four-arm collaboration: Two dual-arm robots work together to manipulate tethered tools. IEEE/ASME Transactions on Mechatronics, 2021.

[68] Hao Chen, Weiwei Wan, et al. Planning to build block structures with unstable intermediate states using two manipulators. IEEE Transactions on Automation Science and Engineering, pages 1–17, 2021.

[69] Weiwei Wan, Takeyuki Kotaka, and Kensuke Harada. Arranging test tubes in racks using combined task and motion planning. Robotics and Autonomous Systems, 147:103918, 2022.

[70] Jean-Philippe Saut, Mokhtar Gharbi, Juan Cortés, Daniel Sidobre, and Thierry Siméon. Planning pick-and-place tasks with two-hand regrasping. In IEEE/RSJ

International Conference on Intelligent Robots and Systems, pages 4528–4533, 2010.

[71] Weiwei Wan, Matthew T Mason, Rui Fukui, and Yasuo Kuniyoshi. Improving regrasp algorithms to analyze the utility of work surfaces in a workcell. In IEEE International Conference on Robotics and Automation, pages 4326–4333, 2015.

[72] Shuo Liu and Stefano Carpin. Grasp quality evaluation with whole arm kinematic noise propagation. In IEEE International Conference on Robotics and Automation, pages 1–7, 2018.

[73] J. Mirabel, S. Tonneau, P. Fernbach, A. Seppälä, M. Campana, N. Mansard, and F. Lamiraux. Hpp: A new software for constrained motion planning. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 383–389, 2016.

[74] Máximo A Roa and Raúl Suárez. Grasp quality measures: review and performance. Autonomous Robots, 38(1):65–88, 2015.

[75] Yu Zheng. Computing the best grasp in a discrete point set with wrench-oriented grasp quality measures. Autonomous Robots, 43(4):1041–1062, 2019.

[76] Tokuo Tsuji, Kensuke Harada, and Kenji Kaneko. Easy and fast evaluation of grasp stability by using ellipsoidal approximation of friction cone. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1830–1837, 2009.

[77] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. The International Journal of Robotics Research, 23(7-8):729–746, 2004.

[78] Steven M Lavalle and James J Kuffner Jr. Rapidly-exploring random trees: progress and prospects. In Algorithmic and Computational Robotics: New Directions, 2000.

[79] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Sampling-based methods for motion planning with constraints. Annual Review of Control, Robotics, and Autonomous Systems, 1:159–185, 2018.

[80] Marc Toussaint, Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In Robotics: Science and Systems, 2018.

[81] Rachel Mara Holladay. Force-and-motion constrained planning for tool use. PhD thesis, Massachusetts Institute of Technology, 2019.

[82] Manuel Beschi, Stefano Mutti, Giorgio Nicola, Marco Faroni, Paolo Magnoni, Enrico Villagrossi, and Nicola Pedrocchi. Optimal robot motion planning of redundant robots in machining and additive manufacturing applications. Electronics, 8(12):1437, 2019.

[83] Solly Brown and Claude Sammut. A relational approach to tool-use learning in robots. In Inductive Logic Programming: 22nd International Conference, ILP 2012, Dubrovnik, Croatia, September 17-19, 2012, Revised Selected Papers 22, pages 1–15. Springer, 2013.

[84] David C Conner, Aaron Greenfield, Prasad N Atkar, Alfred A Rizzi, and Howie Choset. Paint deposition modeling for trajectory planning on automotive surfaces. IEEE Transactions on Automation Science and Engineering, 2(4):381–392, 2005.

[85] Heping Chen, Thomas Fuhlbrigge, and Xiongzi Li. Automated industrial robot path planning for spray painting process: a review. In IEEE International Conference on Automation Science and Engineering, pages 522–527, 2008.

[86] Pål Johan From, Johan Gunnar, and Jan Tommy Gravdahl. Optimal paint gun orientation in spray paint applications—experimental results. IEEE Transactions on Automation Science and Engineering, 8(2):438–442, 2010.

[87] Mayur V Andulkar, Shital S Chiddarwar, and Akshay S Marathe. Novel integrated offline trajectory generation approach for robot assisted spray painting operation. Journal of Manufacturing Systems, 37:201–216, 2015.

[88] Shunsuke Kudoh, Koichi Ogawara, Miti Ruchanurucks, and Katsushi Ikeuchi. Painting robot with multi-fingered hands and stereo vision. Robotics and Autonomous Systems, 57(3):279–288, 2009.

[89] Roger F Malina. Aaron's code: meta-art, artificial intelligence and the work of jarold cohen by pamela mccorduck. Leonardo, 24(5):628–629, 1991.

[90] Sylvain Calinon, Julien Epiney, and Aude Billard. A humanoid robot drawing human portraits. In IEEE-RAS International Conference on Humanoid Robots, pages 161–166, 2005.

[91] Patrick Tresset and Frederic Fol Leymarie. Portrait drawing by paul the robot. Computers & Graphics, 37(5):348–363, 2013.

[92] Youngbum Jun, Giho Jang, Baek-Kyu Cho, Joel Trubatch, Inhyeok Kim, Sang-Duck Seo, and Paul Y Oh. A humanoid doing an artistic work-graffiti on the wall. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1538–1543, 2016.

[93] Kazuma Sasaki, Kuniaki Noda, and Tetsuya Ogata. Visual motor integration of robot's drawing behavior using recurrent neural network. Robotics and Autonomous Systems, 86:184–195, 2016.

[94] Jörg Marvin Gülzow, Patrick Paetzold, and Oliver Deussen. Recent developments regarding painting robots for research in automatic painting, artificial creativity, and machine learning. Applied Sciences, 10(10):3396, 2020.

[95] Otoniel Igno-Rosario, Claudia Hernandez-Aguilar, Alfredo Cruz-Orea, and Arturo Dominguez-Pacheco. Interactive system for painting artworks by regions using a robot. Robotics and Autonomous Systems, 121:103263, 2019.

[96] Chao Guo, Tianxiang Bai, Yue Lu, Yilun Lin, Gang Xiong, Xiao Wang, and Fei-Yue Wang. Skywork-davinci: A novel cpss-based painting support system. In IEEE International Conference on Automation Science and Engineering, pages 673–678. IEEE, 2020.

[97] Lorenzo Scalera, Stefano Seriani, Alessandro Gasparetto, and Paolo Gallina. Non-photorealistic rendering techniques for artistic robotic painting. Robotics, 8(1):10, 2019.

[98] Josh HM Lam, Ka Wah Lo, and Yeung Yam. Robot drawing techniques for contoured surface using an automated sketching platform. In IEEE International Conference on Automation Science and Engineering, pages 735–740, 2007.

[99] Steven Haker, Sigurd Angenent, Allen Tannenbaum, Ron Kikinis, Guillermo Sapiro, and Michael Halle. Conformal surface parameterization for texture mapping. IEEE Transactions on Visualization and Computer Graphics, 6(2):181–189, 2000.

[100] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics, 21(3):362–371, 2002.

[101] Daeun Song and Young J Kim. Distortion-free robotic surface-drawing using conformal mapping. In International Conference on Robotics and Automation, pages 627–633, 2019.

[102] Paul Murphy, Greg Forbes, Jon Fleig, Paul Dumas, and Marc Tricard. Stitching interferometry: a flexible solution for surface metrology. Optics and Photonics News, 14(5):38–43, 2003.

[103] Carmelo Mineo, Stephen Gareth Pierce, Pascual Ian Nicholson, and Ian Cooper. Robotic path planning for non-destructive testing–a custom matlab toolbox approach. Robotics and Computer-Integrated Manufacturing, 37:1–12, 2016.

[104] Ahmet Can and Ali Ünüvar. Five-axis tool path generation for 3D curves created by projection on b-spline surfaces. The International Journal of Advanced Manufacturing Technology, 49(9-12):1047–1057, 2010.

[105] Michael S Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In Advances in Multiresolution for Geometric Modelling, pages 157–186. 2005.

[106] Bo Zheng. 2D curve and 3D surface representation using implicit polynomial and its applications. PhD thesis, University of Tokyo, 2008.

[107] Weiwei Wan, Kensuke Harada, and Fumio Kanehiro. Planning grasps with suction cups and parallel grippers using superimposed segmentation of object meshes. IEEE Transactions on Robotics, 37(1):166–184, 2020.

[108] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In Annual Conference on Computer graphics and interactive techniques, pages 67–76, 2001.

[109] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics, 5(4):349–359, 1999.

[110] Ian H Witten and Eibe Frank. Data mining: practical machine learning tools and techniques with java implementations. ACM Sigmod Record, 31(1):76–77, 2002.

[111] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In European Conference on Computer Vision, pages 766–782, 2016.

[112] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. Autonomous Robots, 34(3):133–148, 2013.

[113] Michael E Mortenson. Mathematics for computer graphics applications. Industrial Press Inc., 1999.

[114] Weiwei Wan and Kensuke Harada. Developing and comparing single-arm and dual-arm regrasp. IEEE Robotics and Automation Letters, 1(1):243–250, 2016.

[115] Weiwei Wan, Kensuke Harada, and Fumio Kanehiro. Preparatory manipulation planning using automatically determined single and dual arm. IEEE Transactions on Industrial Informatics, 16(1):442–453, 2019.

[116] Simon Kriegel, Christian Rink, Tim Bodenmüller, and Michael Suppa. Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects. Journal of Real-Time Image Processing, 10:611–631, 2015.

[117] Sho Kobayashi, Weiwei Wan, Takuya Kiyokawa, Keisuke Koyama, and Kensuke Harada. Obtaining an object's 3d model using dual-arm robotic manipulation and stationary depth sensing. IEEE Transactions on Automation Science and Engineering, 2022.

[118] Cl Connolly. The determination of next best views. In IEEE International Conference on Robotics and Automation, volume 2, pages 432–435, 1985.

[119] Laurana M Wong, Christophe Dumont, and Mongi A Abidi. Next-best-view algorithm for object reconstruction. In Sensor Fusion and Decentralized Control in Robotic Systems, volume 3523, pages 191–200, 1998.

[120] Miguel Mendoza, J Irving Vasquez-Gomez, Hind Taud, L Enrique Sucar, and Carolina Reta. Supervised learning of the next-best-view for 3d object reconstruction. Pattern Recognition Letters, 133:224–231, 2020.

[121] Sören Larsson and Johan AP Kjellander. Path planning for laser scanning with an industrial robot. Robotics and Autonomous Systems, 56(7):615–624, 2008.

[122] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In

IEEE International Conference on Robotics and Automation, pages 5031–5037, 2011.

[123] Sebastian A Kay, Simon Julier, and Vijay M Pawar. Semantically informed next best view planning for autonomous aerial 3d reconstruction. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3125–3130, 2021.

[124] Chenming Wu, Rui Zeng, Jia Pan, Charlie CL Wang, and Yong-Jin Liu. Plant phenotyping by deep-learning-based planner for multi-robots. IEEE Robotics and Automation Letters, 4(4):3113–3120, 2019.

[125] Joseph E Banta, LR Wong, Christophe Dumont, and Mongi A Abidi. A next-best-view system for autonomous 3-d object reconstruction. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 30(5):589–598, 2000.

[126] Marcus Gualtieri and Robert Platt. Robotic pick-and-place with uncertain object instance segmentation and shape completion. IEEE Robotics and Automation Letters, 6(2):1753–1760, 2021.

[127] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In European Conference on Computer Vision, pages 628–644, 2016.

[128] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1955–1964, 2018.

[129] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3d object shape from one depth image. In IEEE Conference on Computer Vision and Pattern Recognition, pages 2484–2493, 2015.

[130] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1886–1895, 2018.

[131] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In International Conference on 3D Vision, pages 728–737, 2018.

[132] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 383–392, 2019.

[133] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In IEEE Conference on Computer Vision and Pattern Recognition, pages 206–215, 2018.

[134] Liang Pan, Tong Wu, Zhongang Cai, Ziwei Liu, Xumin Yu, Yongming Rao, Jiwen Lu, Jie Zhou, Mingye Xu, Xiaoyuan Luo, et al. Multi-view partial (mvp) point cloud challenge 2021 on completion and registration: Methods and results. arXiv preprint arXiv:2112.12053, 2021.

[135] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. SIGGRAPH sketches, 10(1):1, 2007.

[136] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. State of the art in surface reconstruction from point clouds. Eurographics 2014-State of the Art Reports, 1(1):161–185, 2014.

[137] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. International journal of computer vision, 40(2):99, 2000.

[138] Aukje De Boer, Martijn S Van der Schoot, and Hester Bijl. Mesh deformation based on radial basis function interpolation. Computers & structures, 85(11-14):784–795, 2007.

[139] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In IEEE Conference on Computer Vision and Pattern Recognition, pages 605–613, 2017.

[140] Marco Evangelos Biancolini, Andrea Chiappa, Ubaldo Cella, Emiliano Costa, Corrado Groth, and Stefano Porziani. Radial basis functions mesh morphing: A comparison between the bi-harmonic spline and the wendland c2 radial function. In International Conference on Computational Science, pages 294–308. Springer, 2020.

[141] Jérôme Demantké, Clément Mallet, Nicolas David, and Bruno Vallet. Dimensionality based scale selection in 3d lidar point clouds. The international archives of the photogrammetry, remote sensing and spatial information sciences, 38:97–102, 2012.

[142] Patricia Moore and Derek Molloy. A survey of computer-based deformable models. In International Machine Vision and Image Processing Conference, pages 55–66, 2007.

[143] Naijing Lv, Jianhua Liu, and Yunyi Jia. Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations. IEEE Transactions on Robotics, 38:2341–2353, 2022.

[144] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. Self-supervised learning of state estimation for manipulating deformable linear objects. IEEE Robotics and Automation Letters, 5(2):2372–2379, 2020.

[145] Rita Laezza and Yiannis Karayiannidis. Learning shape control of elastoplastic deformable linear objects. In IEEE International Conference on Robotics and Automation, pages 4438–4444, 2021.

[146] Masaru Takizawa, Shunsuke Kudoh, and Takashi Suehiro. Method for placing a rope in a target shape and its application to a clove hitch. In IEEE International Symposium on Robot and Human Interactive Communication, pages 646–651, 2015.

[147] Haifeng Han, Gavin Paul, and Takamitsu Matsubara. Model-based reinforcement learning approach for deformable linear object manipulation. In IEEE Conference on Automation Science and Engineering, pages 750–755, 2017.

[148] Félix Nadon, Angel J Valencia, and Pierre Payeur. Multi-modal sensing and robotic manipulation of non-rigid objects: A survey. Robotics, 7(4):74, 2018.

[149] A Delgado, Carlos Alberto Jara, Damian Mira, and F Torres. A tactile-based grasping strategy for deformable objects' manipulation and deformability estimation. In International Conference on Informatics in Control, Automation and Robotics, volume 2, pages 369–374, 2015.

[150] Yu She, Shaoxiong Wang, Siyuan Dong, Neha Sunil, Alberto Rodriguez, and Edward Adelson. Cable manipulation with a tactile-reactive gripper. International Journal of Robotics Research, 40(12-14):1385–1401, 2021.

[151] Shigang Yue and Dominik Henrich. Manipulating deformable linear objects: sensor-based fast manipulation during vibration. In IEEE International Conference on Robotics and Automation, volume 3, pages 2467–2472, 2002.

[152] Shervin Javdani, Sameep Tandon, Jie Tang, James F O'Brien, and Pieter Abbeel. Modeling and perception of deformable one-dimensional objects. In IEEE International Conference on Robotics and Automation, pages 1607–1614, 2011.

[153] Cheng Chi and Dmitry Berenson. Occlusion-robust deformable object tracking without physics simulation. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 6443–6450, 2019.

[154] Dominik Henrich, Tsukasa Ogasawara, and H Worn. Manipulating deformable linear objects-contact states and point contacts. In International Symposium on Assembly and Task Planning, pages 198–204, 1999.

[155] Jingang Jiang, Yongde Zhang, Mingliang Jin, and Chunge Wei. Bending process analysis and structure design of orthodontic archwire bending robot. International Journal of Smart Home, 7(5):345–352, 2013.

[156] Zeyang Xia, Hao Deng, Shaokui Weng, Yangzhou Gan, Jing Xiong, and Hesheng Wang. Development of a robotic system for orthodontic archwire bending. In IEEE International Conference on Robotics and Automation, pages 730–735, 2016.

[157] Alexander Kuehl and Joerg Franke. Robot-based forming of hairpin winding. In IEEE International Electric Machines & Drives Conference, pages 1–7, 2021.

[158] Yi-Heng Lu, Shih-Yuan Wang, Yu-Ting Sheng, Che-Wei Lin, Yu-Hsuan Pang, and Wei-Tse Hung. Transient materialization–robotic metal curving. In International Conference on Computer-Aided Architectural Design Research in Asia, pages 425–434, 2020.

[159] Jiaji Zhang, Binjun Shi, Guang Feng, Guokun Zuo, and Ye Liang. Design, development and control of a forming robot for an internally fixed titanium alloy strip. Machines, 10(2):68, 2022.

[160] WU Jianjun and Zengkun Zhang. An improved procedure for manufacture of 3d tubes with springback concerned in flexible bending process. Chinese Journal of Aeronautics, 34(11):267–276, 2021.

[161] Shuyou Zhang, Mengyu Fu, Zili Wang, Dingyu Fang, Weiming Lin, and Huifang Zhou. Springback prediction model and its compensation method for the variable curvature metal tube bending forming. The International Journal of Advanced Manufacturing Technology, 112(11):3151–3165, 2021.

[162] D Raj Prasanth and MS Shunmugam. Geometry-based bend feasibility matrix for bend sequence planning of sheet metal parts. International Journal of Computer Integrated Manufacturing, 33(5):515–530, 2020.

[163] Nikolaos Kontolatis and G-C Vosniakos. Optimisation of press-brake bending operations in 3d space. Journal of Intelligent Manufacturing, 23(3):457–469, 2012.

[164] Yang Sen, Kaiwei Ma, Zhou Yi, and Fengyu Xu. A research on bending process planning based on improved particle swarm optimization. In International Conference on Intelligent Robotics and Applications, pages 348–358, 2021.

[165] Andrea Baraldo, Luca Bascetta, Fabrizio Caprotti, Sumit Chourasiya, Gianni Ferretti, Angelo Ponti, and Basak Sakcak. Automatic computation of bending sequences for wire bending machines. International Journal of Computer Integrated Manufacturing, pages 1–17, 2022.

[166] Shigeru Aomura and Atsushi Koguchi. Optimized bending sequences of sheet metal bending by robot. Robotics and Computer-Integrated Manufacturing, 18(1):29–39, 2002.

[167] Hao Zhang, Ali Abd El-Aty, Jie Tao, Xunzhong Guo, Shuo Zheng, and Cheng Cheng. Effect of active deflection on the forming of tubes manufactured by 3d free bending technology. Metals, 12(10):1621, 2022.

[168] Zahid Faraz, Syed Waheed ul Haq, Liaqat Ali, Khalid Mahmood, Wasim Akram Tarar, Aamer Ahmed Baqai, Mushtaq Khan, and Syed Husain Imran. Sheet-metal bend sequence planning subjected to process and material variations. The International Journal of Advanced Manufacturing Technology, 88(1-4):815–826, 2017.

[169] Fengyu Xu, Dawei Ding, Baojie Fan, and Sen Yang. Prediction of bending parameters and automated operation planning for sheet-metal bending orientated to graphical programming. The International Journal of Advanced Manufacturing Technology, pages 1–14, 2023.

[170] Teruaki Ito, Rahimah Abdul Hamid, and Tetsuo Ichikawa. Collaborative design and manufacturing of prosthodontics wire clasp. In Transdisciplinary Engineering: Crossing Boundaries, pages 421–428. 2016.

[171] Rahimah Abdul Hamid and Teruaki Ito. 3d prosthodontics wire bending mechanism with a linear segmentation algorithm. Journal of Advanced Manufacturing Technology, pages 33–46, 2016.

[172] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing, 1(3):244–256, 1972.

[173] Zeshen Wang and Jean-Claude Müller. Line generalization based on analysis of shape characteristics. Cartography and Geographic Information Systems, 25(1):3–15, 1998.

[174] Maheswari Visvalingam and James D Whyatt. Line generalisation by repeated elimination of points. The Cartographic Journal, 30(1):46–51, 1993.

[175] Sheng Zhou and Christopher Jones. Shape-aware line generalisation with weighted effective area. Developments in Spatial Data Handling, pages 369–380, 2005.

[176] Jun Ma, Heng Li, and MW Fu. Modelling of springback in tube bending: A generalized analytical approach. International Journal of Mechanical Sciences, 204:106516, 2021.

[177] Mohamed Raessa, Weiwei Wan, Keisuke Koyama, and Kensuke Harada. Planning to flip heavy objects considering soft-finger contacts. International Journal of Automation Technology, 15(2):158–167, 2021.

[178] Jiayang Ao, Qiuhong Ke, and Krista A Ehinger. Image amodal completion: A survey. Computer Vision and Image Understanding, page 103661, 2023.

[179] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol

Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691, 2022.

[180] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and motion planning with large language models for object rearrangement. arXiv preprint arXiv:2303.06247, 2023.

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor, Professor Harada Kensuke. Despite my transition from a different major, he not only welcomed me into the lab as a doctoral student but also instilled in me the confidence that I could successfully complete the program. This trust has been the initial driving force for my journey. And his guidance, support, and expertise have consistently underpinned my research endeavor.

In addition, my deep gratitude extends to Professor Wan Weiwei, whose careful guidance and detailed comments helps me get to know how to do research step by step. His mentorship has been crucial in my transformation from a novice in robotics to where I stand today. Thanks to Professor Isomura Emiko, who give me an interesting topic that inspired me a lot in research.

I am also thankful to my thesis committee members, Professor Sato Kousuke and Professor Iiguni Youji, for their insightful comments and constructive criticism. Their rigorous feedback has helped me enhance the quality of this work.

I am very glad to spend my Ph.D. years with my great lab mates. Special thanks to Ms. Zhang Xinyi whose encouragement from my undergraduate through to my doctorate, spanning aspects of life and research, has been immensely valuable. To Mr. Chen Hao (Senior) and Dr. Hu Zhengtao, for the guidance me when I first come to the lab as a novice and for the inspiring discussion of research throughout my doctoral journey. I also want to thank all my lab mates who helped me a lot in both life and study, Ms. Wang Zhengting, Ms. Gao Yuan, Ms. Chen Hao (Junior), Mr. Zhang Junbo, Mr. Yu Bowen, Mr. Wang Chenxi, and Mr. Tang Yu, Mr. Prashant Kumar, Mr. Higashi Kazuki, and Mr. Tsuru Masato. And also the graduated students from our lab, Dr. Wang Yan, Dr. Xu Jingren, Dr. Motoda Tomohiro, Dr. Cristian Beltran, Dr. Zhang Ang, Mr. Matsuosa Shogo, Mr. Kobayashi Sho, Mr. Hayakawa Shogo and Mr. Joshua C. Triyonoputro. I am forever grateful for your friendship and support.

Thanks to my parents for giving birth to me in the first place and supporting me spiritually throughout my life. To my partner, Mr. Lu Chenyue, the sacrifices you have made and your unending support have played a significant role in getting me this far.

This journey has been filled with challenges and moments of self-doubt. However, I take this moment to acknowledge my own grit and determination that have driven me through these past four years.

List of Publications

Journal Papers

1. **Ruishuang Liu**, Chuan Li, Weiwei Wan, Jia Pan, Kensuke Harada. Next Best View Planning Considering Confidence Obtained from Shape Completion Learning. IEEE Robotics and Automation Letters, 2023 (Submitted, IF: 5.2).

2. **Ruishuang Liu**, Weiwei Wan, Kensuke Harada. TAMP for 3D Curving – A Low-Payload Robot Arm Works Aside a Bending Machine to Curve High-Stiffness Metal Wires. IEEE Transactions on Automation Science and Engineering, 2023 (Early access, IF: 5.6).

3. **Ruishuang Liu**, Weiwei Wan, Keisuke Koyama, Kensuke Harada. Robust Robotic 3-D Drawing Using Closed-Loop Planning and Online Picked Pens, IEEE Transactions on Robotics, 38(3), 1773-1792, 2021. (Presented at 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IF: 7.8).

International Conference Papers (with Peer-Review)

1. **Ruishuang Liu**, Weiwei Wan, Emiko Tanaka Isomura, and Kensuke Harada. Metal Wire Manipulation Planning for 3D Curving - A Low Payload Robot that Uses a Bending Machine to Bend High-Stiffness Wire. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 8927–8932, 2022.

Local Conference Papers (without Peer-Review)

1. **Ruishuang Liu**, Weiwei Wan, Kensuke Harada, Obtaining a Shiny Metal Plate's 3D Model using Deeply Learned Geometry and Next Best View Planning, The Robotics and Mechatronics Conference (ROBOMECH), 2A2-D27, 2023.

2. **Ruishuang Liu**, Weiwei Wan, Kensuke Harada, Metal Wire Manipulation Planning for 3D Curving with a Bending Gadget, The Robotics and Mechatronics Conference (ROBOMECH), 2A2-N03, 2022.

3. **Ruishuang Liu**, Weiwei Wan, Keisuke Koyama, Kensuke Harada, Optimizaiton-based Planning and Control for 3D Robotic Drawing, The Conference of the Robotics Society of Japan (RSJ), 1I3-01, 2021.

4. **Ruishuang Liu**, Weiwei Wan, Keisuke Koyama, Kensuke Harada, Planning 3D Robotic Drawing, The Conference of the Robotics Society of Japan (RSJ), 1I3-01, 2020.