이화여자대학교 대학원
2022학년도
박사학위 청구논문

Artistic Robotic Pen Drawing System using High-DoF Manipulators

인공지능 · 소프트웨어학부
Daeun Song
2023

Artistic Robotic Pen Drawing System using High-DoF Manipulators

이 논문을 박사학위 논문으로 제출함

2022 년 12월

이화여자대학교 대학원

인공지능 · 소프트웨어학부 Daeun Song

Daeun Song의 박사학위 논문을 인준함

지도교수 김영준 _____

심사위원 오유란 _____

류석창 _____

윤성의 _____

이주행 _____

김영준 _____

이화여자대학교 대학원

Table of Contents

List of Figures

List of Tables

Abstract

Since the Renaissance, artists have created artworks using novel techniques and machines, deviating from conventional methods. The robotic drawing system is one of such creative attempts. Since then, robots have been a means of creation, but with the recent development of artificial intelligence (AI), robots are also becoming the subject of creation. Robots and art raise many social and technological questions. In order to make a robot that draws pictures, two physical elements are needed: 'creation or observation' and 'drawing'. The field of computer graphics, which creates artwork by mimicking the human creation process, has been studied for a very long time. And to reproduce these works in a physical space using robots, it is necessary to solve robotics problems such as robot motion planning and control.

In this dissertation, we propose an autonomous robotic system that creates pen art on canvas surfaces of various sizes and shapes. The proposed system includes the method of creating a painting work through the aforementioned 'creation or observation' and the robot motion planning and control for 'drawing'. Unlike ordinary digital art, which consists of discrete pixels, a drawing that needs to be mapped to a continuous robot motion must be replaced with a series of coordinate values. Vector graphics, consisting of points, curves, and polygons based on mathematical equations, correspond to these characteristics. The stroke-based rendering (SBR) method, one of the computer graphics fields studied to produce non-photorealistic paintings that mimic human drawings, is suitable for robotic drawing as it replaces images with painting in units of strokes. Furthermore, this paper introduces a layered depth painting method that analyzes the object and draws the picture in consideration of depth, similar to the process by which a person paints. On the contrary, we also introduce TSP art, an art form suitable for robots, which we believe is specialized in operating delicate movements effectively and accurately.

This dissertation focuses on the features of robots. Robots are good at repeatedly moving pre-

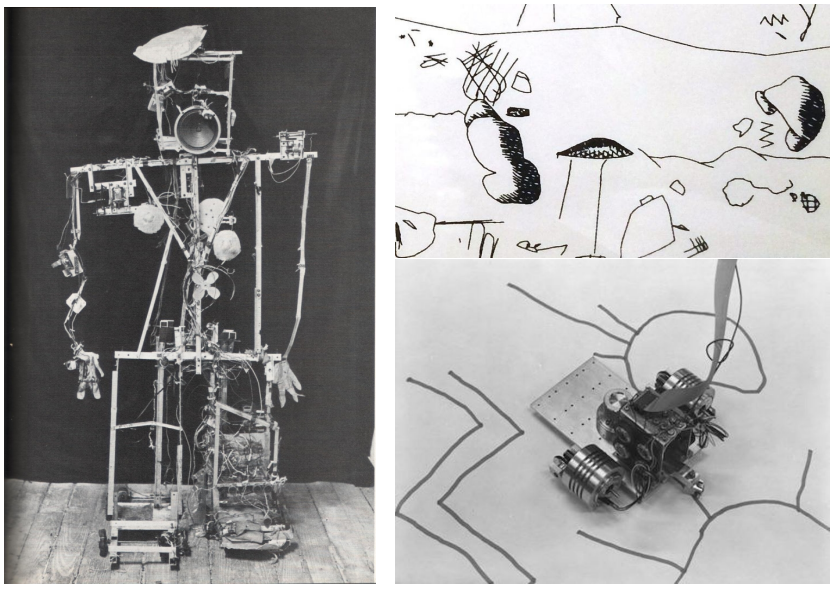
defined motions accurately and precisely. Therefore, we extend the robotic drawing problem to draw on a large, curved target surface without distortion. To this end, conformal mapping that maps the 2D drawing to 3D space while minimizing distortion is applied. We introduce a robust robotic curve rendering method to 'draw' on a non-planar surface. Considering the reachable range of the robot, the drawing is repositioned within the target surface, and the calculated drawing path is carried out with a continuous motion that its end-effector follows the given path. We use impedance control that can correct geometric errors that may result from inconsistencies between the virtual environment and the real environment. Finally, we propose a set cover algorithm to generate pen art on a wider target surface. The algorithm finds a discrete set of mobile platform poses that the robot covers the entire target surface. In addition, we propose a pen tool design and tool-change mechanism to perform fully-automated drawings with colors using dual-arm equipped with adaptive 3-finger grippers.

We demonstrate that our robotic drawing system can create visually pleasing and complex pen art on various surfaces. The proposed system is the first robotic drawing system to draw pen art on large-scale, non-planar surfaces using multi-DoF robots. In this dissertation, we introduce the overall process for the robotic drawing system and propose methods to respond to each element. The technologies applied to this system are not limited to robot drawing but can also be applied to various robotic applications that require continuous contacts with a target surface, such as milling and sanding.

I. Introduction

A. Background

Since the Renaissance, artists have incorporated machines and new technologies into artistic installations and performances to go beyond conventional art forms. Figure 1.1 shows early historical examples of how machines were used as installations or as art creators. Many contemporary artists are radically expanding the possibilities of creative expression using new media and mechanisms. In accordance with the recent impressive advances in computer science and engineering, new possibilities for robots becoming one of the mechanisms are emerging [1, 2, 3, 4]. *Robotic art*, works of art using robotic or automated technology, is becoming popular both for artists and roboticists.

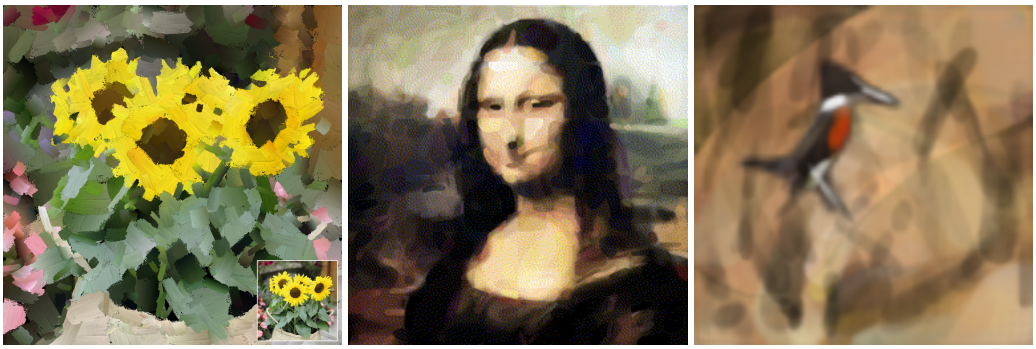


(a) Robot K-456, Nam-June Paik, 1964 (b) AARON, Harold Cohen, 1979

Figure 1.1. Robots in art history

Among many branches of robotic art, the robotic art community has presented many fully-autonomous or semi-autonomous robotic drawing systems [5]. Figure 1.1(b) shows the very first drawing machine, AARON, by the artist Harold Cohen [6]. The initial version of AARON was an artificial intelligence program that created abstract digital drawings. Then those turned more representational over time, able to imitate shapes from nature, and also started to implement colors to the drawings. Later on, Cohen engineered a robot that allowed AARON to act and produce *drawings* in the physical world. Likewise, creating an autonomous robotic drawing system requires interrogation of artistic and technological creativity in digital space, in addition to general robotic technology that brings digital artwork into the physical space.

With the improvements in digital technology, the development offered artists and scientists further exploration of digital creativity. Most of the work focused on investigating the production of artistic images in virtual space that enables a wide variety of expressive and aesthetic styles using computer algorithms. Non-photorealistic rendering (NPR) is one such area in computer graphics that has been studied to capture artists' expressive style of digital art [7]. This includes pen-and-ink illustration, painterly rendering, and cartoon-like rendering style. Today's new developments even incorporate artificial intelligence (AI) and machine learning technologies. Figure 1.2 shows an example of the paintings generated using neural renderer and reinforcement learning.

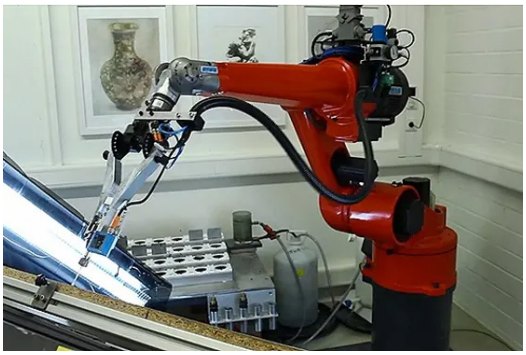


(a) Zou et al. [8] (b) Huang et al. [9] (c) Singh et al. [10]

Figure 1.2. AI-generated non-photorealistic art

Even with the great development of computer graphics technologies in digital space, efforts to transfer digital artwork to physical space continued. Such effort revived more and more with the development of robotic hardware. In the early days, typically, the artwork was just printed on a conventional printer by computing pixel information. But later, robotic drawing systems that distributed real pen strokes or paint strokes on a real canvas using higher degree-of-freedom (DoF) robots appeared (Figure 1.3). Using higher DoF manipulators reduces the spatial and geometrical constraints on the drawing canvas. Besides, having a robot arm drawing can be thought of as an artistic performance in itself. In order to bring digital paintings into a physical space using machines, an input drawing must be comprised of discrete elements such as paint strokes. For instance, stroke-based rendering (SBR) is one of the computer graphics technology in the category of NPR that can be applied for painting machines to place the brush strokes in a physical space [11].

(a) 6 robots named Paul [4], Patrik Tresset, 2012



(b) e-David [12], Oliver Deussen, 2012

Figure 1.3. Robotic drawing systems

Accordingly, this dissertation aims to explore more on the digital rendering methods that suit well on the physical robotic drawing system. Starting with the very intuitive method of reproducing the user drawing input using a vector graphics engine running on a pen-equipped tablet PC, we also propose our scientific understanding of the human drawing process. We extend SBR by integrating panoptic segmentation and depth estimation to mimic the human drawing pro-

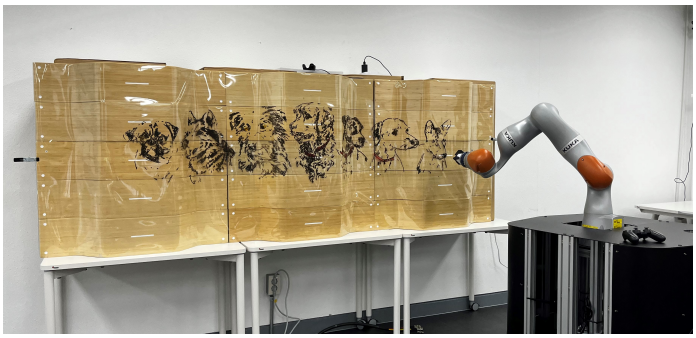
cess that considers the salient object and the depth while observing the target. Furthermore, we explore more on the robotics side, what robots can do well. While humans are good at dexterous manipulation of producing various strokes on the canvas, robots are good at accurately reproducing long, complex strokes. Thus, we propose to apply TSP art that converts input images into a single, path-wise continuous line.

After the drawing sequences and paths are decided, the robotic drawing task can be seen as a robot motion planning problem that moves a pen along a given trajectory while concerning the target surface. As we focus more on *what robots can do well*, we further extend the problem in terms of irregularity and scalability of the target surface. Additionally, we devise to fully automate the system. We explore a fully-autonomous robotic pen drawing system that can draw on a large, non-flat surface.

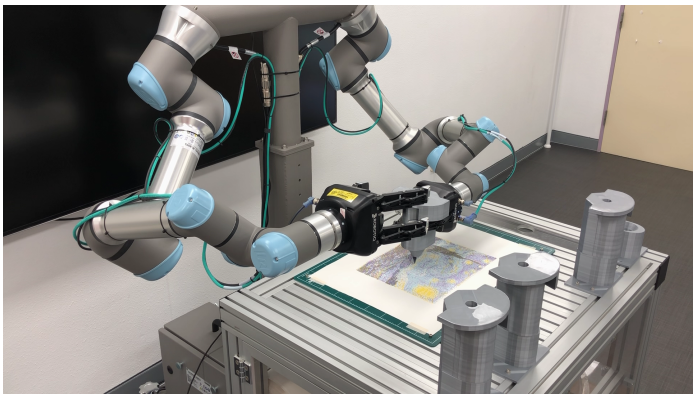
To draw on the non-flat surface, the system not only needs to estimate the target surface but also needs a robust control method to compensate for the estimation error. With the growing demand for robot interaction with the environment, impedance control has evolved [13, 14]. By controlling both the motion of the robot and its contact forces, the robot acts compliant with the external contact with the target surface, which as a result, compensates for surface estimation error. Impedance control in our system plays a large role not only in performing robotic drawing tasks that involve continuous contact motion but also is used for surface estimation error compensation, as well as adaptive surface point collection.

General manipulator-type robots have a limited range of operations. Therefore, in order for the robot to draw in a larger space, it needs mobility that allows the robot to move around freely. With its mobility, the robot must plan a discrete set of mobile base poses to cover the entire drawing space. The set cover problem is a classical question in combinatorial optimization, known to be NP-hard [15]. Given a collection of elements, the problem aims to find the minimum number of sets that cover all of the elements. This problem has many applications in many different industries, including mobile exploration robots. We apply this set cover problem to the large-scale drawing problem.

This dissertation aims to propose a creative robotic pen drawing system that is able to distribute pen strokes on a real canvas using high DoF manipulators. We explore methods for producing drawing commands that ultimately lead to an art form. We further challenge the problem by extending the system into a non-flat and larger target canvas. Thus, we not only deal with the drawing mapping method for distortion-free drawing on non-flat surfaces but also a robust control method that enables continuous contact motion on various surfaces under uncertainties. We propose a set-cover planning method for large-scale drawing using a mobile manipulator. Also, our system aims for colorful drawings. Therefore, we explore fully-automated robotic tool change using dual-arm manipulators, with novel drawing tool design and the tool-change mechanism. Figure 1.4 shows the robotic hardware setups of our proposed system.



(a) Mobile manipulator



(b) Dual-arm with 3-finger grippers

Figure 1.4. Our robotic drawing system

B. Research Objectives and Contributions

The main objective of the dissertation is to propose a robotic pen drawing system that can draw on surfaces with various shapes and sizes while focusing on the robotic ability. In order to build a robotic drawing system, it is necessary not only to understand the drawing creation process scientifically but also to consider robust components of the robotic system, such as control, sensing, and planning. We further challenge the problem by adding scalability and irregularity to the shape of the target surface.

Robots are capable of performing repetitive tasks with speed and accuracy, far exceeding human operators. In particular, having extra degrees of freedom allows for a wider working space and higher accuracy. In addition, if the two manipulators can cooperate with each other with the grippers attached, a higher level of automation, including the tool change, is possible. Thus, in this dissertation, we aim to show that robotic drawing using high-DoF manipulators can enrich artistic creativity by drawing more complicated and sophisticated drawings in a broader choice of target canvas shapes and sizes.

The main contributions of the dissertation are as follows:

- **Drawing Path Generation** We propose drawing path generation methods focused on both the human and robot art creation process.
- **Drawing Path Mapping** We propose a drawing path mapping method to draw distortion-free artwork on a non-flat surface.
- **Robotic Curve Rendering** We propose a method to carry out the robotic drawing task that is robust on geometric errors.
- **Large Canvas Drawing** We propose a set cover planning algorithm to plan a discrete set of mobile base poses for large-scale drawing.
- **Dual-arm Drawing** We propose a novel drawing tool design and the tool-change mechanism for colorful drawing using dual-arm manipulators.

C. Organization

The remainder of this dissertation proceeds as follows. In Chapter II, We survey the related works on robotic curve rendering, digital curve rendering, and an impedance-controlled robot. In Chapter III, we outline our robotic drawing system with the notations that will be used throughout the dissertation. In Chapter IV, we propose various drawing path generation methods depending on stylistic choices. In Chapter V, we describe our method to map a 2D drawing path onto the 3D surface. Our method enables our system to draw on non-flat surfaces with minimal distortion. In Chapter VI, we describe our method for reproducing the planned path as a pen stroke using a robot in physical space. In Chapter VII, we further expand our system in ways that make the most of the characteristics of robotic hardware. We propose a large-scale robotic drawing system using a mobile manipulator that can move around freely and a dual-arm robotic drawing system using dual manipulators equipped with three-finger grippers that can change the drawing tool automatically. We present and discuss our robotic drawing results with statistics in Chapter VIII, and finally, we conclude in Chapter IX.

II. Related Work

In this chapter, we first explore existing robotic drawing systems. We then survey the relevant digital rendering techniques and robotic techniques that can be adapted for robotic drawing.

A. Robotic Drawing System

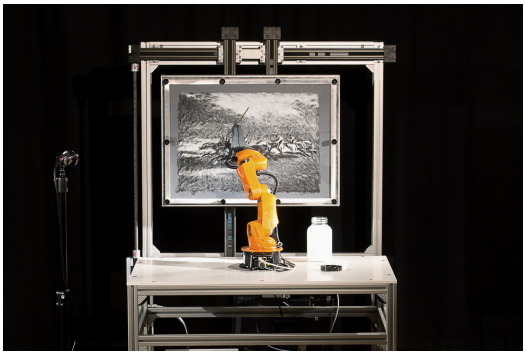
The early work on creating drawing machines can be attributed to artistic work by Jean Tinguely, and Harold Cohens Aaron [6]. In the computer graphics and robotics community, early attempts to create drawing robots were largely based on a plotter-type, special-purposed machine. More recently, research efforts have been applied to using a high-DoF general robot or manipulator for robotic drawing that can span a wide spectrum of artistic expressions.

In the early days, many drawing robots that draw portraits using simple edge detection were introduced. Calinon et al. [2], HOAP-2 humanoid robot draws human portraits by detecting human faces and following human-characteristic styles. Jean-Pierre et al. [16] proposed a robot artist that draws human portraits using KUKA robotic arm with simple edge extraction. Paul the robot [4] is a robotic installation that creates observational portrait drawings that mimic an artist's stylistic signatures. Jain et al. [17] proposed a portrait drawing robot using force control. The authors also demonstrated the ability of the technique to draw over an unknown non-flat surface.

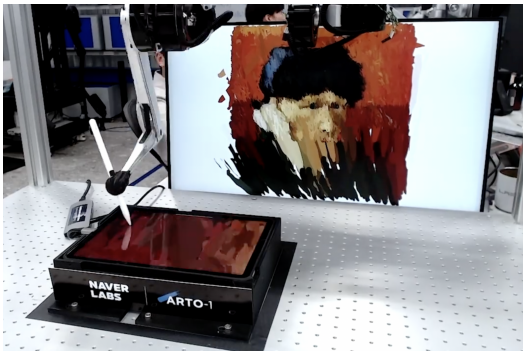
Afterward, drawing robots that applied computer graphics techniques appeared. e-David [3] is a modified industrial robot that can create a wide variety of painting styles from an image input. It relies on visual feedback to generate NPR-type painterly results. Busker robot [18] is the first automatic system that uses the watercolor technique rendering. The authors use a Canny edge detector for contour and Hough Transform for feature extraction technique and implement skeletonization for rendering dark edges. The authors later extended the system to

(a) CloudPainter [21]

(b) Busker robot [18]



(c) Small version of e-David [3]



(d) Arto-1 [23]

Figure 2.1. Robotic drawing systems and their drawings

apply non-photorealistic rendering techniques, cross-hatching, and random splines [19].

Recently, with the development of machine learning technology, reinforcement learning (RL) that enables the painting agent to learn where to place the brush strokes has been studied [20]. Cloudpainter [21] took advantage of the machine learning style transfer method to apply the visual style of a source image to a target image. Bidgoli et al. [22] proposed a method to integrate an artistic style into the brushstrokes and the painting process through collaboration with a human artist. The authors first collect brush strokes and brush motions from an artist and train a generative model to generate brush strokes that pertain to the artist's style. Similarly, recent work by Park et al. [23] used a haptic device to learn stroke-level painting skills from a human demonstration on a digital canvas.

Table 2.1. Overview of drawing robots

References	Year	Kinematic Type	Path Generation	Result
[6]	1979	plotter	N/A	AI-generated art
[2]	2005	humanoid robot	image binarization	pen-drawn portrait
[16]	2012	6-DoF robotic arm	edge detection	pen-drawn portrait
[3]	2013	6-DoF robotic arm	SBR	stroke-wise painting
[4]	2013	3-DoF robotic arm	NPR	pen-drawn portrait
[17]	2015	6-DoF robotic arm	edge detection	pen-drawn portrait
[21]	2018	plotter or robotic arm	deep learning	painting
[18], [19]	2019	6-DoF robotic arm	NPR	stroke-wise painting
[22]	2020	6-DoF robotic arm	data-driven SBR	stroke-wise painting
[20]	2020	5-DoF robotic arm	RL-based SBR	stroke-wise painting
[24]	2021	6-DoF robotic arm	N/A	pen drawing
[23]	2022	4-DoF haptic device	data-driven SBR	stroke-wise digital drawing

While most of the research focused on creating drawing styles on a flat surface, there was recently an attempt to draw a pen drawing on a 3D surface using closed-loop planning and vision-force feedback [24].

In summary, Table 2.1 shows the overview of the existing drawing robots with their actuator, image processing method, and drawing content. Most early works used traditional image processing techniques to detect the edge. As time passed, many drawing robots that applied advanced technology appeared, and with the recent development of machine learning, a lot of research on creating drawing styles has been conducted. However, none of the existing works dealt with creating a pen drawing on a large, non-planar surface using a mobile manipulator. Our robotic drawing system consists of a 9-DoF mobile manipulator and two 6-DoF manipulators. We propose three different image processing methods that will be reproduced as stroke-wise pen drawings on a physical surface.

B. Digital Curve Rendering

1. Vector Graphics

For the robotic drawing system, vector graphics are more suitable than raster graphics, as vector graphics can generate continuous and smooth pen strokes that can be mapped nicely to smooth robotic motions. Vector graphics typically fill pixels inside implicitly-defined curves of a certain width using CPU-based scanline methods [25, 26, 27]. Because vector graphics techniques need to render implicit curves in every frame, the performance of CPU-based rendering methods is slow on the dense screen resolution used by modern display devices.

Loop and Blinn suggested a GPU-based fast resolution-independent rendering method that could render paths and bounded regions [28]. Kilgard and Bolz [29] introduced a fast GPU-based, two-step approach, namely the stencil-and-cover approach. The stencil step determines the stroke path's fill coverage, and the cover step fills the area determined by the stencil step. There exists no vector-graphics method that can adequately reproduce human drawing consisting of free-form lines and curves, even though smooth curve rendering or retrieving methods, such as in [28, 30, 31] and [32], may handle human drawings to some degree.

2. Stroke-based Rendering

As we can see from the survey in Section II.A, NPR is one of the popular choices for the robotic drawing system. Among them, SBR is suitable for human-like drawing that applies sequential strokes to the canvas.

A survey of early SBR research is given in [33]. Early SBR research was mainly concerned with the artistic quality of the final image. Litwinowicz [34] proposed an algorithm that produces an image in an impressionist style. The algorithm generates a set of strokes on a regular

grid with the desired orientation. The orientation is specified as a tangent to the image's gradient. Hertzmann [35] extended the algorithm by adding multiple layers and several style parameters, which resulted in a wider range of possible painting styles. In both methods, the order of the strokes is randomized to avoid consistency.

While early research in SBR mainly considered local image properties, later work integrated the semantic information by introducing user-guided or automatic image segmentation and labeling techniques. Lin et al. [36] create a dictionary of brushes for different material properties of objects and use the semantic content as a guide for stroke placement and rendering. Their system employs user-driven image segmentation. Hertzmann [37] presents a similar system that adds a higher level of automation and incorporates the ability to synthesize and edit strokes. Zeng et al. [38] explore the semantic information through hierarchical decomposition by organizing the components in a parse tree that guides the rendering process.

With the advances in machine learning (ML), new attempts have been made in SBR. The systems can now "learn how to paint" via rules encoded as they maximize the similarity between the input and the rendering output. Zou et al. [8] incorporated a neural style transfer method to generate vivid and realistic painting artworks. Unlike previous methods, they deal with such an artistic creation process in a vectorized environment. Huang et al. [9] use deep reinforcement learning (DRL) with adversarial loss feedback to learn a rendering sequence that minimizes the difference between the target image and the final painting. Singh and Zheng [10] use a similar approach but also incorporate semantic information to create different patterns in processing foreground and background strokes.

We explore more on the SBR technique to apply to robotic drawing systems [39]. The method is guided by the semantic content in the image, and we use depth information to distribute the strokes over the contours to be gradually perceivable as the drawing evolves.

C. Handling Errors in Robotic Curve Rendering

1. Surface Parameterization

Automatic parameterization of a non-parametric surface like polygonal or point-set surfaces is a non-trivial problem and has been extensively studied in geometric modeling and processing [40, 41]. Typical applications of parameterization include texture mapping in computer graphics and mesh editing and re-meshing in geometry processing. In particular, the former application is closely related to our problem, where a curved drawing in 2D should be faithfully reproduced on curved or bump surfaces in 3D.

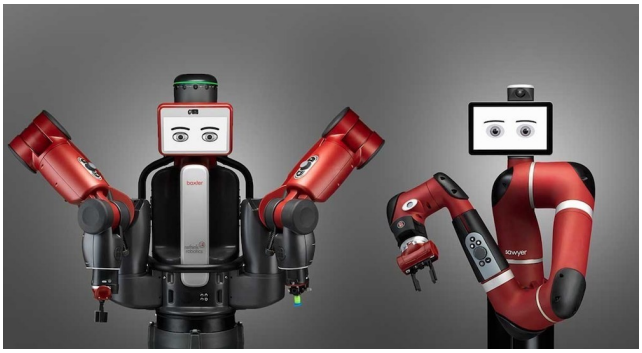
For a polygonal surface with disk-like topology and a convex boundary, Barycentric mapping [42] is the most popular method in the literature. However, this method can induce serious distortion in parameterization if the boundary is highly non-convex. Coping with this problem, conformal mapping has been introduced based on complex analysis. At a high level, conformal mapping can be classified into analytic and geometric methods [41]. The former is relatively easy to implement using energy minimization [43] but can suffer from unbalanced distortion if the underlying surface has high Gaussian curvature, while the latter can resolve this issue [44].

2. Impedance-controlled Robot

Hogan et al. [45] first suggested the idea of using impedance control for controlling the interaction between a manipulator and the surrounding environment. Since then, robot interaction techniques using impedance control have been explored in the robotics community [13, 14]. The popularity of the use of impedance control is partly due to the demand for robotic systems with the ability to interact with uncertain environments in practical applications. Thus, many collaborative robots based on impedance control have been introduced from both industry and

academia, for instance, LBR iiwa from KUKA Robotics, Baxter and Sawyer from Rethink Robotics, and Justin from DLR. These robots are able to handle classically challenging robotic manipulation, such as peg-in-hole assembly. Impedance control can also be used for unscrewing a can for a humanoid robot Justin [46] as well as bimanual tasks [47].

(a) KUKA LBR iiwa, KUKA Robotics



(b) Baxter and Sawyer, Rethink Robotics

Figure 2.2. Impedance-controlled co-bots

III. Robotic Drawing System

In this chapter, we describe the overall architecture of our creative robotic pen drawing system that reproduces the pen art on an arbitrary canvas surface without restricting its size or shape.

A. System Overview

The robotic drawing can be viewed as sequential drawing path planning and control problem that eventually leads to the art form. An artist completes the artwork by placing a series of complex strokes on a canvas space. During the process, the artist performs delicate control of a drawing tool to deliver the intended strokes accurately. Likewise, creating a robotic drawing system requires not only robust computer graphics technology to acquire a set of drawing coordinates but also robust components of control, sensing, and planning. We compose the robotic drawing system in three main modules: drawing path planner, robotic drawing trajectory planner, and robot drawing controller (Figure 3.1).

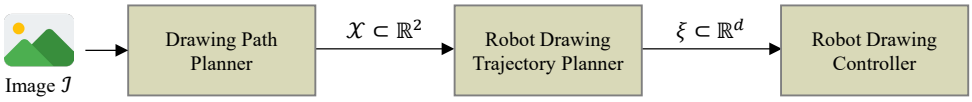


Figure 3.1. Robotic drawing system overview

A drawing path planner converts a given raster or vector image $\mathcal{I}$ into a piece-wise continuous path $\mathcal{X} \subset \mathbb{R}^2$. Our system provides multiple planning methods depending on stylistic choices in Chapter IV, such as TSP art or stroke-based rendering. In order to make the planned drawing path *drawable* for robots, we need to map it into the robot's joint configuration space: $\mathcal{X} \subset \mathbb{R}^2 \rightarrow \xi \subset \mathbb{R}^{DoF}$. To do so, first, we need to map the 2D drawing path into a 3D physical canvas space.

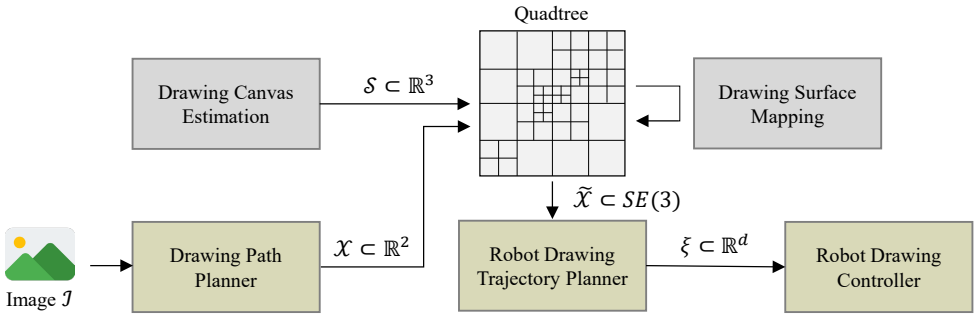


Figure 3.2. Robotic surface drawing system overview

Our robotic drawing system is capable of drawing a pen drawing on a non-flat surface. In order to do that, we consider a target drawing canvas and map the drawing points onto the surface (Figure 3.2). The target surface information $\mathcal{S} \subset \mathbb{R}^3$ can be either given as a 3D mesh model, obtained as a 3D point cloud data using an RGB-D camera, or obtained as a 3D point set adaptively by the manipulator while performing the drawing task (Section V.A). A simple orthographic projection (*i.e.,* projection mapping) can be made to add an additional dimension to the drawing path. However, in this dissertation, we propose a method to map the drawing onto the surface with minimal distortion using conformal mapping (Section V.B). With surface normal estimated from the surface points, we obtain full drawing pose set $\tilde{\mathcal{X}} \subset SE(3)$ by aligning the end-effector orientation perpendicular to the canvas.

Once the target drawing poses $\tilde{\mathcal{X}}$ are decided, robotic drawing is equivalent to finding a robot motion that its end-effector follows the given path. The robot drawing trajectory planner plans a set of continuous robot joint configurations ξ corresponding to $\tilde{\mathcal{X}}$. This problem is also known as path-wise Inverse Kinematics (Section VI.B). Moreover, path-wise IK is suitable for setting a robot's kinematic and dynamic constraints while following the robot trajectory. In the case of robot drawing, achieving a feasible motion with no sudden jumps is crucial. Plus, it is essential to follow the given end-effector poses accurately not to ruin the resulting drawing. Thus, we solve the path-wise IK problem by iteratively solving the IK for the end-effector poses $\tilde{\mathcal{X}}$ with the minimum distance objective in the configuration space.

Finally, our system performs the pen drawing task using an impedance control method (Section VI.C). The robotic pen-art system imposes many new challenges, unlike robotic painting. For instance, robotic pen art requires that the robot maintain constant contact with the target drawing surface to draw the path; this is known to be a difficult task, especially in the presence of uncertainty [48]. Impedance control, an approach for controlling the relationship between the force and the position, has become a popular choice for contact-intensive tasks [49] because it can deal with position uncertainties and adjust the robot's compliance according to the external force [50].

In this dissertation, we also extend the robotic drawing problem into large-scale (Section VII.A) and into bi-manual (Section VII.B) drawing problems. Our system considers the manipulator's reachability to decide the size of the drawable canvas space and apply them to large-scale and bi-manual drawing problems.

B. Definitions and Notations

Table 3.1 defines the notations used throughout the dissertation. The table is divided into three sections with horizontal lines. The first section shows the notations regarding the robotic drawing path. The second section shows the notations regarding the target canvas. The last section shows the notations regarding the coverage of the robot that are mostly used in large-scale robotic drawing problems in Section VII.A. When talking about drawing, the term *stroke* refers to a curve drawn with a brush or a pen until it is removed.

Table 3.1. Table of notations

Notation	Description
Robotic Drawing Path	
$\mathcal{I}$	a target image
$\mathcal{P} \subset \mathbb{R}^2$	a set of points defining a target drawing
$\mathcal{X} \subset \mathbb{R}^2$	a set of strokes composing a target drawing
$\tilde{\mathcal{X}} \subset SE(3)$	a set of drawing poses mapped onto the target surface
$\xi \subset \mathbb{R}^{DoF}$	a set of joint configurations corresponding to $\tilde{\mathcal{X}}$
Target Canvas	
$\mathcal{S} \subset \mathbb{R}^3$	a set of points defining 3D target surface
$\mathcal{S}' \subset \mathbb{R}^2$	a set of points with only x and y values in $\mathcal{S}$
$\Omega \subset \mathbb{R}^2$	a set of points defining a surface in a parametric domain for mapping
Robot Coverage	
$\mathcal{C}_i$	coverage circle of the manipulator
$\mathcal{B}_i$	bounding circle of the mobile base platform
r	radius of the coverage circle
b	radius of the bounding circle
d	distance between the coverage circle and the bounding circle
$\mathcal{Z}_j$	covered surface point sets
$\mathcal{Y}_j$	target mobile platform poses

IV. Drawing Path Generation

In order to perform robotic drawing, first of all, it is necessary to plan the drawing path for the artwork. Unlike digital paintings that consist of colored pixels, machines need to realize a set of sequenced coordinates to place the strokes. Non-photorealistic rendering (NPR) techniques [51] that devise a method to convert digital raster images into painterly or sketched style with a set of stroke coordinates have been studied for many years in computer graphics. In addition, the advent of vector graphics, visual images that are saved as a sequence of vector statements, made the movement of digital images into physical space more feasible. In this chapter, we propose three different drawing path generation methods according to stylistic choices.

A. Vector Graphics

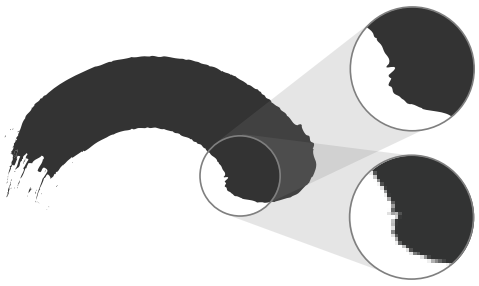


Figure 4.1. Enlarged view of vector graphics image (top) and raster graphics image (bottom)

Vector graphics is a form of computer graphics that creates digital images by a sequence of vector statements. Simplified, vector graphics is like a continuous connect-the-dots drawing. The most striking feature of a vector image is its scalability; it can be enlarged without loss of quality. Such characteristics of a vector image are very suitable for transferring a drawing to a physical space using robotic drawing. It can be reduced or stretched freely to fit the 3D target

surface while considering the ability of the robot. On the other hand, raster images consist of a certain number of squared pixels, and they lose their quality when resized (Figure 4.1). Certain steps must be taken to use for robotic drawing (Section IV.B and IV.C).

Each element composing the vector image includes path information. Figure 4.2 shows a brush shape evenly stretched along a vector path (top) and a vector path with its anchor points in pink (bottom). Likewise, we obtain a set of strokes composing a target drawing, $\mathcal{X} \subset \mathbb{R}^2$, which is a set of coordinates passing through the path separated by stroke units. Thus, we can use vector images already drawn by other artists or draw an artwork directly using a vector graphics editor computer software such as Adobe illustrator [52] and use them for robotic drawing.

Figure 4.2. Vector image of a stroke (top) and its path with anchor points (bottom)

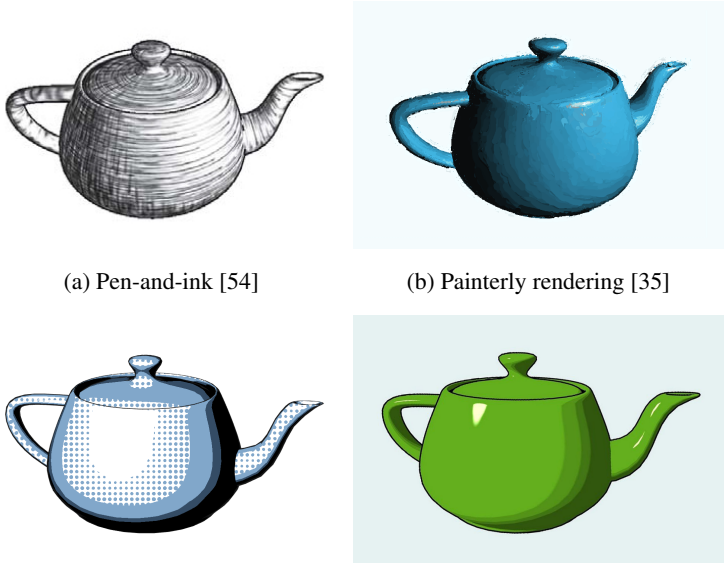
Another intuitive way to get the vector drawing path is to acquire the input drawing path from the artist directly. The desire for more accurate and controllable input on touchscreen devices led to the invention of a stylus pen. Not only does it allow for precise and complex touches on the screen, but also, the recent invention can recognize its pen tip trajectory and even the pen pressure. A vector graphics engine proposed in [53] receives two-dimensional points and corresponding pen pressure using these pen-ready devices such as tablet PC or mobile phones. It takes the artist's pen strokes and generates Bézier spline curves with varying offsets. Figure 4.3 shows a vector graphics engine running on a tablet PC and its rendering result when enlarged. Using this engine, we can obtain drawing paths of various drawings drawn by the artist.

Figure 4.3. Vector graphics engine running on a tablet PC [53]

B. Non-photorealistic Rendering

Vector image shows the characteristics of input required for robotic drawing very well: it has to be divided into stroke units, and all strokes should be continuous so that they can be mapped to robot motion. Humans can instinctively know where to place the strokes when observing the target object. However, it is not easy to define this process scientifically. Vector graphics is a method we can obtain a drawing path without such a scientific definition.

Now we can start questioning: *How do artists create art?* It is still an open question that has been researched in the computer graphics field for many years [51]. Non-photorealistic rendering (NPR) is an area of computer graphics that plays a key role in the scientific understanding of visual art and illustration [7]. NPR techniques devise a wide variety of algorithms to capture artists' expressive styles of digital art. It includes pen-and-ink illustration, painterly rendering, and cartoon-like rendering (Figure 4.4). Most of the algorithms focus on algorithmically defining the artists' drawing skills. Among the various algorithms, we discuss the stroke-based rendering (SBR) method.



(a) Pen-and-ink [54] (b) Painterly rendering [35]

(c) Comic book Style [55] (d) Toon shading [56]

Figure 4.4. Non-photorealistic rendering examples of Utah teapot

1. Stroke-based Rendering

SBR is one of the NPR techniques, also known as painterly rendering. The algorithm uses a 2D source such as a photograph as a target and *paints* a list of spline brush strokes on an empty canvas (Figure 4.7). The idea of the algorithm is to iteratively compare the canvas to the target image and place the stroke onto the canvas where the difference is minimized at every step:

$$\begin{aligned} \forall t, \quad & \textbf{min} \quad \|I - C_t\|^2 \\ & \textbf{s.t.} \quad C_t := paint(C_{t-1}), \end{aligned} \tag{4.1}$$

where the objective is to minimize the difference between the target image I and the current canvas state C_t, and $paint$ defines a function to place a stroke. We use temporal notations only for this Section IV.B.

Because the algorithm finds a set of brush strokes with its 2D coordinates, it perfectly matches the input requirements of the robotic drawing. However, unlike virtual painting, where there is

no limitation in color space, we need to consider a color set when reproducing the image using a physical drawing tool. It is possible that the colors used in the virtual rendering do not exist with the actual drawing tools, and there may not be enough physical space to place the drawing tools for robotic drawing. Therefore, we apply a color clustering algorithm to generate a color palette with a certain number of representative colors for efficient robot drawing.

2. Color Clustering

Generating a color palette can be thought of as partitioning the image color pixels into a certain number of different groups that best represent the image. Each and every pixel of the raster image contains color information, and we can partition them into k different groups. k-means clustering algorithm [57] is a well-known algorithm to solve the problem. Given a set of colored pixels, the algorithm aims to partition the pixels into k clusters to minimize the within-cluster sums of squares. In Figure 4.5, we show color clustering results with different k values. Once the clustering is done, stroke-based rendering is performed with the flattened image. We can generate a robotic drawing path with a limited number of colors to perform robotic drawing on a physical space.

Figure 4.5. k-means clustering result and stroke-based rendering result using [35]

3. Layered Depth Painting

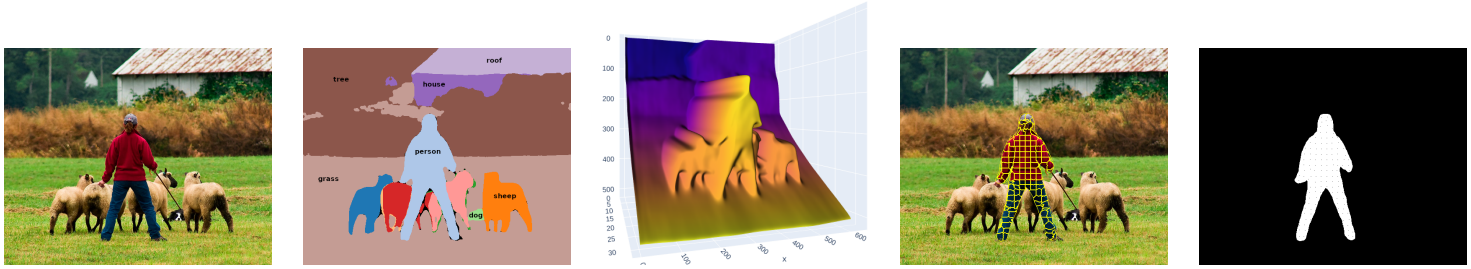
Although a stroke-based rendering algorithm provides a meaningful robotic drawing path, it does not yet fully answer the question of how human artists draw. It does not hold any semantic representation of the image being rendered. However, many human artists observe the target image, determine important features, and decide the stroke sequence accordingly [58]. Likewise, we investigate more on the drawing sequence. We would like to have the sequence of drawing strokes look natural to human observers. Furthermore, the entire process of machine drawing should be regarded as an artistic performance along with the final drawing result.

Our SBR method [39] is guided by the semantic content in the image and monocular *depth* estimation. Given a plausible depth map for a semantic unit in the image (*e.g.,* a face in a portrait), we distribute the strokes over the contours of the layers defined by a plane moving from the back of the face toward the viewer. As the drawing evolves, the viewer quickly perceives the intended shape and also gradually observes detail getting filled in.

The algorithm for generating a stroke plan to produce a painting for image I given a number of strokes N is as follows. We begin by computing the panoptic segmentation [59] to identify the salient objects (*things*) and regions of interest (*stuff*). Objects in the thing class are sorted by *weight* that considers both the size of the item and the confidence score of the prediction. We use semantic ordering for objects in the stuff class. Next, we compute the depth map and its histogram [60], which are the critical components of the proposed method. They enable us to generate a stroke ordering that conveys deliberate planning to an outside observer. For each object in the panoptic segmentation, we compute a sequence of frames of candidate stroke seeds. The back-to-front order ensures that the strokes are reasonably far apart initially, giving the impression that the robot artist is sketching the object outline. As the frames move to the front, variations in the depth map cluster the strokes into separate sub-regions, giving the appearance that the robot is concentrating on the features of the object.

Figure 4.6 shows the output from each stage in the proposed pipeline. (a) is an original image in 641×513 size. (b) shows the panoptic segmentation result into things (person, sheep, dog) and stuff (house, roof, tree, grass). (c) is a smoothed depth map of the image. (d) is a superpixel segmentation result of the *person*, which is one of the salient segmented results from (b). (e) shows the stroke seeds for the *person*, based on superpixel segmentation. Figure 4.7 shows the resulting stroke sequence and the final painting with 2000 strokes. As we can see from the result, our method first draws the salient object starting from the contour of the object and then process it to the background. This resembles the human drawing process.

We clarify that the method presented in this section is a separate work from the dissertation. In this dissertation, we show the drawing results by applying the result of the method to the proposed robotic drawing system.



(a) (b) (c) (d) (e)

Figure 4.6. The output from each stage in the pipeline [39]

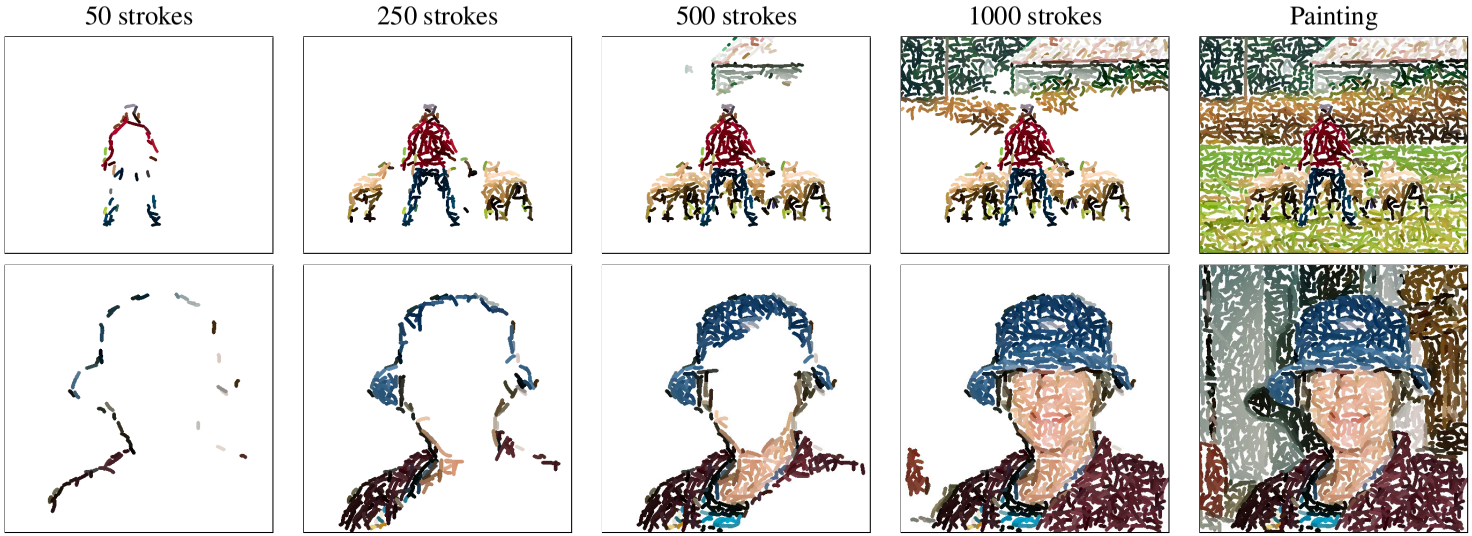
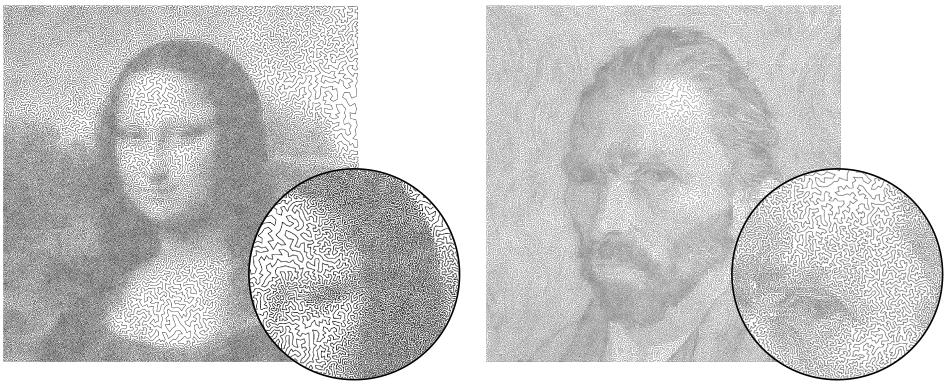


Figure 4.7. Stroke sequence results

C. TSP Art

SBR algorithm is meaningful in the sense that it provides a scientific understanding of human artists creating art. However, applying it to the robotic drawing raises another question: *Is mimicking humans desirable for robotic drawing?* Humans are good at dexterous manipulation of producing various short and long strokes on the canvas. On the other hand, robots are good at accurately reproducing long, complex strokes. In this section, we investigate more about the drawing that robots can do well.



(a) da Vinci, Mona Lisa, 100K points (b) van Gogh, Self Portrait, 120K points

Figure 4.8. TSP art examples [61]

Traveling Salesman Problem Art, abbreviated as TSP art, is one of the representative examples of creating artistic work using computer algorithms. It was first invented by mathematician Robert Bosh, who wished to engage his students in optimization problems [62]. TSP art represents the original digital image with a single long, complex, continuous line. It is obtained by first placing some dots on the image and second connecting the dots with piecewise-continuous line segments. Figure 4.8 shows the example of TSP art, composed of more than 100K points set. TSP art involves not only the creative process of computer algorithms but also fits the nature of a robotic task, whose fundamental mission is to follow a path accurately.

While the original TSP art does not include any color information. It converts the image into grayscale and reproduces the *tone* of the image. In this section, we propose multi-channel TSP art that reproduces not only the tone but also the color of the original image. Figure 4.9 shows the overview of the TSP art generation process. The process includes three main stages as follows:

- **Color Processing:** From $\mathcal{I}$, we separate the color image into multiple color channels $\mathcal{I}_0, \cdots, \mathcal{I}_{k-1}$, where k is the number of colors.
- **Stippling:** For each $\mathcal{I}_i$, $i = 0, \cdots, k-1$, generate and place a set of points $\mathcal{P} \subset \mathbb{R}^2$ to replicate the tone of the original image.
- **TSP Solving:** Find a cycling path $\mathcal{X} \subset \mathbb{R}^2$ that visits every point $\mathcal{P}$ exactly once and returns to the first point.

We will talk about each step; color processing, stippling, and TSP generation in the following subsections. Our approach maps the input image $\mathcal{I}$ into a long, continuous robotic path $\mathcal{X}$. We take the TSP art idea with an additional color processing stage to reproduce the color likeness of the original image.

1. Color Processing

First, we split the color image into multiple color channels $\mathcal{I}_0, \cdots, \mathcal{I}_{k-1}$ and then process each channel as a separate image to create TSP art. When recombined, the resulting pieces of TSP art create color similarity to the original image. How to split the color channels can be controversial. Here we use the CMYK approach, separating a color image into its cyan, magenta, yellow, and black channels. While RGB is a better choice for digital drawing, CMYK is known to be a better choice for physical drawing. CMYK combines four colors to make the desired color, and it is a well-known color separation method for printing. Since a pen drawing also uses ink similar to printing, the CMYK color model is desirable for pen drawing.

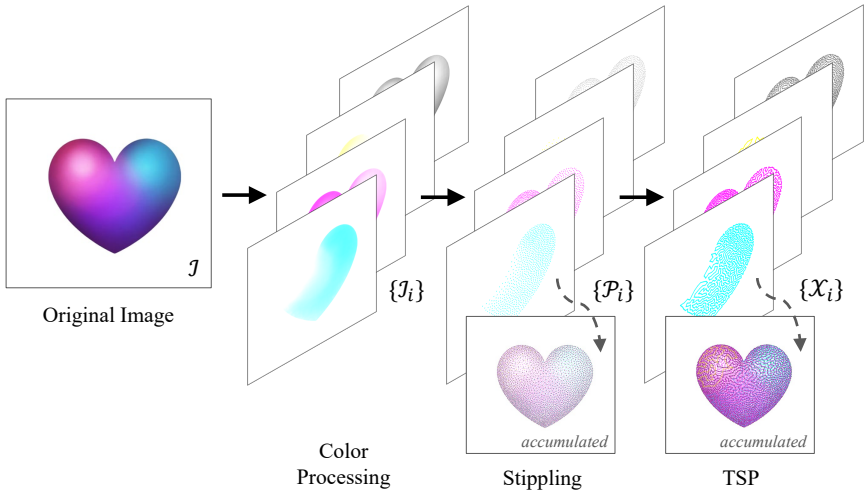


Figure 4.9. TSP art generation process

We can also apply color clustering methods (*e.g.,* K-Means Clustering, Agglomerative Clustering, DBSCAN), also introduced in Section IV.B.2. The algorithms separate a color image into representative colors. Unlike CMYK separation, the color channels obtained by the clustering method have no overlapping area and do not stack colors. Thus, there exist clear advantages and disadvantages between the two methods. Clustering methods extract relatively accurate colors from digital images. It has the advantage of being able to express more accurate colors with fewer points. However, it has the disadvantage that the exact colored physical drawing tools may not exist. Conversely, because the CMYK method represents the desired color by combining four colors, we do not have to worry about the existence of physical drawing tool colors. However, color reproducibility may be very poor in non-dense, sparse drawings.

2. Point Generation

We perform stippling for each color channel individually, converting an image into a set of points $\mathcal{P} \subset \mathbb{R}^2$, such that the points get denser in a darker region while fewer points are placed in a bright region. Many stippling methods have been studied to reproduce the image effectively. In this paper, we adapt the LBG Stippling method [63], which is based on combining a variant of the Linde-Buzo-Gray algorithm [64] with weighted Voronoi stippling [65]. The method splits the Voronoi cells from a single point until it reaches a desired point density with the weight function. It also allows the merging of the neighboring cells once the point density gets too high. The feature of this method is that the number of stippling points is determined according to the image density without specifying the number of stippling points.

3. TSP Generation

The generated points $\mathcal{P}$ can already be an art form, and robotic motions can realize it; however, inefficient and repetitive up-and-down motions may be required for stippling. These motions can be very time-consuming for robots to follow. Instead, we connect all the points in $\mathcal{P}$ in a single path $\mathcal{X} \subset \mathbb{R}^2$, which makes the robot easier to follow. Finding this path is equivalent to solving TSP. TSP is a well-known NP-hard problem. However, much research has been done to solve TSP approximately and efficiently. We use a Concorde solver that relaxes the problem into Linear Program (LP) and iteratively fixes the possible fractional solution by using a cutting plane algorithm [66]. Concorde [67] is an optimization solver widely regarded as the fastest TSP solver for large instances [68]. The resulting single line that resembles the original image's tone is considered an art form called TSP art. We observe that such complex and continuous paths are suitable for robotic drawing systems to reproduce them on a physical surface with its capability of sophisticated maneuverability.

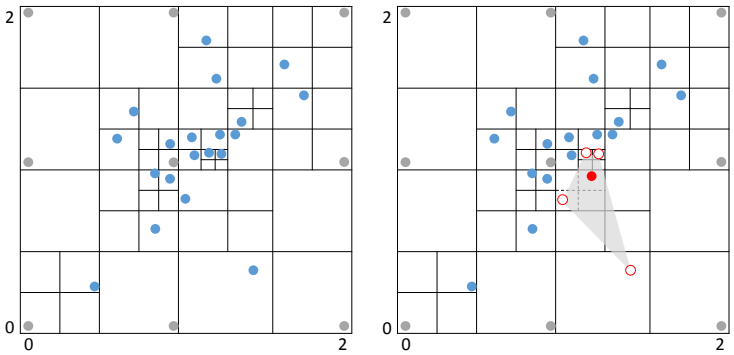
V. Drawing Path Mapping

In order to reproduce the 2D drawing path $\mathcal{X}$ on a 3D physical space, we first need to realize an additional dimension of the drawing points. In this chapter, we introduce our method of how we incrementally estimate the drawing canvas with adaptive sampling (Section V.A), and how we map the 2D drawing onto the surface with minimal distortion (Section V.B).

A. Drawing Canvas Estimation

1. Adaptive Sampling

In this section, we propose an adaptive sampling method that incrementally estimates the drawing canvas space using an impedance-controlled robot when the geometry of the target surface is not given to the robot. Our impedance-controlled robot incrementally samples the surface before and during the drawing and builds an adaptive and implicit representation of the surface in a quadtree data structure; x, y coordinates as a key, and z as a value.



(a) Surface points being stored (b) Estimation for a drawing point

Figure 5.1. Quadtree data structure for sampled points

Specifically, the robot begins by sampling an extent of drawing canvas space before kicking off the actual drawing and keeps adding the control points of every stroke as new samples during drawing. The resulting sampling points constitute a 2.5D height field and are represented as a quadtree data structure $\mathcal{Q}$ as shown in Figure 5.1(a). The gray points represent the initial points covering the extent and the center of the drawing surface. The blue points represent the points incrementally sampled during the drawing. During robotic drawing, for each stroke, the positions of the control points projected to the target surface are estimated using bi-linear interpolation using the quadtree as shown in Figure 5.1(b). The red solid point is a new point being inserted. The four nearest neighbor points are also highlighted in red and form a gray quad. The height of a new red point is then approximated using bi-linear interpolation. Although the estimated projected position is rough, we use a combination of position- and impedance-based control to enable the robot to accurately reproduce the drawing on the target surface.

The control points $\mathcal{X} \subset \mathbb{R}^2$ for drawing strokes, generated by the vector rendering engine, SBR, or TSP art, are defined in 2D and need to be projected onto the target, unknown surface $\mathcal{S}$ in $\mathbb{R}^3$; the height value x_z needs to be determined. To estimate the height of $\mathbf{x} \in \mathcal{X}$ projected on the surface, we first search the four nearest points of $\mathbf{x}$ from the quadtree $\mathcal{Q}$, which forms a quad that includes $\mathbf{x}$ (*e.g.,* the gray quad in Figure 5.1(b)). From the quad annotated with height values, bi-linear interpolation is performed to yield x_z. Since this quad is not always rectangular, we map the quad to a unit square to facilitate the bi-linear interpolation.

Each quadtree node stores a single sampling point (*i.e.,* the control point $\mathbf{x}$) containing its x-y coordinate and the height value (x_z). When a new sampling point is inserted into a node, the node is split into four children if it already contains a sampling point. Note that our quadtree only grows but never shrinks and thus does not require a sophisticated tree re-fitting mechanism. Once the targeted position of $\tilde{x}_z$ is calculated, we add $\tilde{\mathbf{x}}$ to the data structure. The more strokes are drawn to the surface, the more sampled points are being collected; *i.e.,* the quadtree is expanded. As a result, by the time of the completion of the drawing, an estimation of interpolated values would be more reliable, resulting in intended pen pressure during drawing.

2. Surface Points Acquisition

Our robotic drawing system demonstrates sufficient robotic drawing even in the absence of prior knowledge of the surface. However, having sufficient data on the surface enables more robust and efficient drawing on the surface. We try acquiring the surface points $\mathcal{S} \subset \mathbb{R}^3$ using an RGB-D camera or given a 3D mesh format. With the surface points, we estimate the surface normal. We can not only align the drawing tool direction to match the surface normal to perform a more robust drawing but also map the drawing data onto the surface with minimal distortion.

Various types of RGB-D cameras, such as stereo vision, and time-of-flight, can reconstruct the depth information of a 3D surface with an accuracy range of $\pm 5\%$ with a resolution less than $1\mathrm{mm}$. Using this depth data, we generate a point cloud of the scene and calibrate it with the target surface. Further, we also estimate the surface normals for the calibrated points to orient the robot's tool frame with respect to the robot base frame. This enables the robot's pen tip to align toward the surface normal of the target surface and also to exert proper force on the pen tip to draw.

To perform robotic surface drawing on an arbitrary surface, the robot needs to decide both the position $\mathbf{x} \in \mathbb{R}^3$ and orientation $\mathbf{r} \in SO(3)$ of its end-effector. In particular, to calculate the orientation $\mathbf{r}$ that ensures the robot makes stable contact on a surface point, we calculate the surface normals $\mathbf{n}$ of a point cloud and make it parallel to the approaching direction of the end-effector as shown in Figure 5.2.

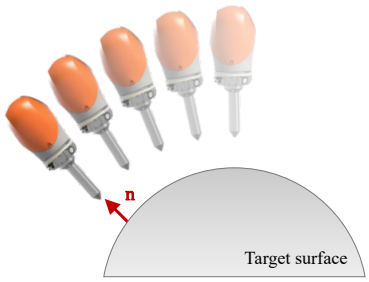


Figure 5.2. Aligning the end-effector direction parallel to the surface normal

One can consider two options to estimate the normals of a point set. A rather straightforward way is to use a surface meshing technique to obtain a mesh representation of S and then extract the per-vertex normals. Alternatively, one can infer per-point normal directly from S by performing least-squared plane-fitting [69]. This is reduced to the principal component analysis for a local neighborhood of points. Even though either way would work in our case, the surface parameterization method we have chosen in Sec. V.B requires surface meshing, and thus opt for the meshing technique. Finally, the updated target surface point data set is defined and stored in a quadtree data structure:

$$\mathcal{S} = \{\langle \mathbf{s}_i, \mathbf{n}_i \rangle | \mathbf{s}_i \in \mathbb{R}^3, \mathbf{n}_i \in \mathbb{R}^3\}. \tag{5.1}$$

B. Conformal Mapping

Before explaining our idea of conformal mapping, we first introduce some theoretical background about conformal mapping. Given two surfaces with similar topology, it is possible to compute a one-to-one correspondence between them [40]. The problem of computing such a mapping is referred to as surface parameterization.

Conformal mapping is one of the surface parameterization techniques that preserves both angles and shapes. Let a continuous surface $\mathcal{S} \subset \mathbb{R}^3$ be parameterized into the parametric domain $\Omega \subset \mathbb{R}^2$. As illustrated in Figure 5.3, a function ψ that is a mapping from 3D surface S to the Ω domain in 2D is said to be conformal if, for each $(u, v) \in \Omega$, the tangent vectors along the horizontal and vertical lines (the red and blue lines in Figure 5.3), which form a regular grid, are orthogonal on S and have the same norm [41]:

$$\mathbf{a} = \mathbf{n} \times \mathbf{b}, \tag{5.2}$$

where $\mathbf{a}$ and $\mathbf{b}$ denote the tangent vectors and $\mathbf{n}$ denotes the unit normal at $(x, y, z) \in \mathcal{S}$. In other words, a conformal mapping locally corresponds to a similarity transform. It transforms an elementary circle on the surface into an elementary circle to the (u, v) domain.

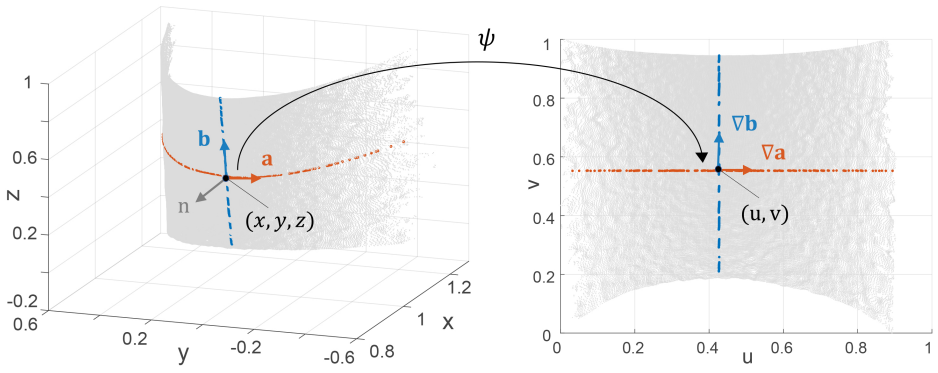


Figure 5.3. Least squares conformal mapping (LSCM). 3D target surface data $\mathcal{S}$ in point cloud (left) and its parameterized representation Ω (right)

To realize conformal mapping in our work, we adapt LSCM [43] to parameterize the target surface. After we *unfold* the target surface into the 2D parameter space $(u, v) \in \Omega$, we search for the proper parameter values of the 2D drawing data in the parameter space and *refold* the surface into 3D space. We choose to compute the gradients using a local orthonormal basis of triangles, which requires that the surface be reconstructed as a triangular mesh. Alternatively, one could use a meshless technique for the conformal mapping of a point cloud without surface reconstruction by using the Laplace-Beltrami (LB) operator [70]. Both results would yield a set of coordinates $(u_i, v_i) \in \Omega$ associated with each point in $\mathcal{X}$ that satisfies Equation 5.2. In our implementation, we have chosen the meshing technique, as robust surface meshing implementations are readily available in [71], and as the meshing technique can be applied to both types of surface input, raw point clouds and meshes.

After the surface parameterization, ψ is complete, we need to decide the proper parameter coordinates for the drawing points set $\mathcal{X}$. Because our target surface data set $\mathcal{S}$ is an unorganized/sparse point set and does not necessarily form a uniform grid, in order to parameterize every drawing point to a corresponding (u, v) coordinate, we need to perform an estimation. Note that this is not a significant challenge for raster-based texture mapping, as one can obtain a clear idea of which rasterized point on the surface needs to be parameterized. On the other hand, in our case, there is no guarantee that all the drawing points in $\mathcal{X}$ can be mapped to the parametrized surface points in Ω. To solve this inverse mapping problem, we simply perform bi-linear interpolation in the parametric domain to estimate the (u, v) coordinates for $\mathcal{X}$ as follows.

Given a desired drawing space $\mathcal{S}$ and a corresponding parameterized space Ω, surface data are stored in a quadtree data structure. Along with the desired drawing scale, we compute the parametric scale in Ω and fit the 2D drawing to Ω. Then, for each drawing point $\mathbf{x}_i \in \mathcal{X}$, we search for the four-nearest points in Ω that form a quadrilateral (Figure 5.1(b)). By performing bi-linear interpolation on this quadrilateral, we can parameterize the $\mathbf{x}_i$ that is mapped to both the position $\mathbf{s}_i$ and the surface normal $\mathbf{n}_i$ on $\mathcal{S}$. Finally, a set of drawing points mapped to the

target surface with the corresponding surface orientations is generated:

$$\tilde{\mathcal{X}} = \{\langle \mathbf{x}_i, \mathbf{r}_i \rangle | \mathbf{x}_i \in \mathbb{R}^3, \mathbf{r}_i \in SO(3)\}. \tag{5.3}$$

Figure 5.4 shows the intuitive explanation of the conformal mapping process of calculating $\tilde{\mathcal{X}}$. Drawing points $\mathcal{X}$ are first mapped onto the parametric space Ω and then by finding the corresponding 3D points in $\mathcal{S}$, $\tilde{\mathcal{X}}$ on the target surface is calculated.

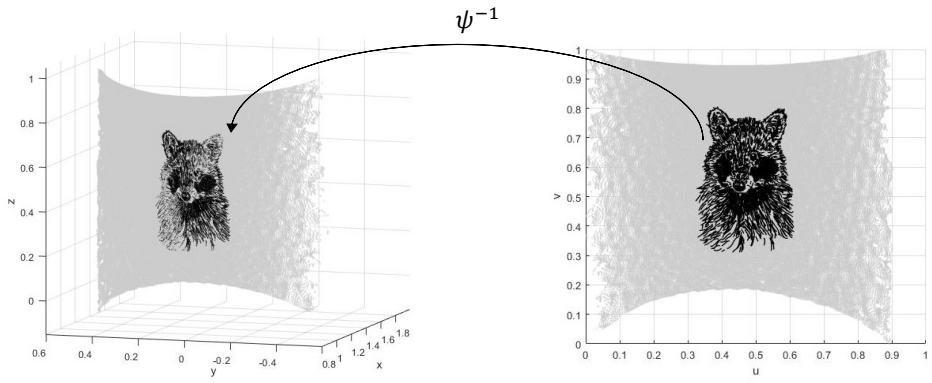


Figure 5.4. Drawing points mapped onto the target surface and its parametric space

VI. Robotic Curve Rendering

Once we have end-effector poses in 3D, robotic drawing can now be viewed as a motion planning problem that its end-effector follows the given poses $\tilde{\mathcal{X}}$. In this chapter, we introduce our approach to delivering a drawing path to the physical canvas space using robotic hardware.

A. Reachability-based Canvas Space Decision

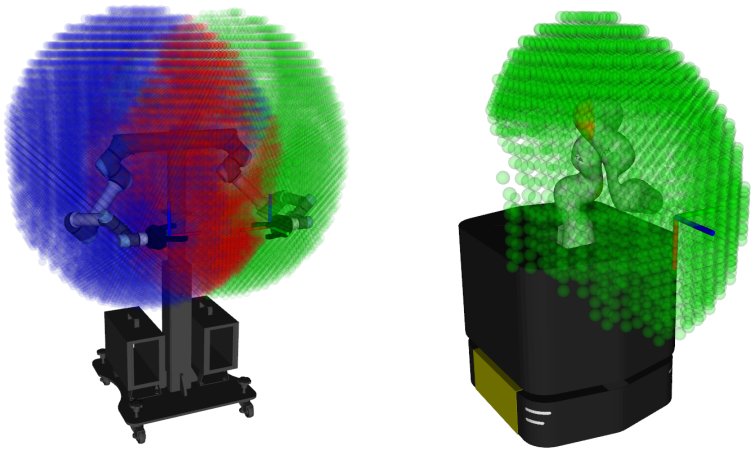
Although the end-effector poses $\tilde{\mathcal{X}}$ are decided, it is not obvious whether the poses are appropriate for the robot motion. In this section, we introduce our approach to deciding the proper canvas dimension considering the reachability of the manipulators and relocating the drawing end-effector poses accordingly.

The robot's reachability naturally decides the maximum size of the drawing canvas, where the reachability of the robot is defined as points reachable by the robot's end-effector [72]. To obtain the reachability map, first, we discretize the robot's Cartesian workspace. For each discrete point, we subdivide the point with sampled orientation. We solve Inverse Kinematics (IK) problem for each sampled pose to check if the pose is reachable by the robot. We alleviate the problem by fixing the sample orientation set. This is possible because we fix the drawing-pen direction always perpendicular to the drawing canvas, and the robot base is generally placed in a facing direction to the surface. Furthermore, we save reachable points instead of the pose when all the sampled orientation is reachable by the robot. The spheres in Fig. 6.1 represent the reachable points by the robots with a predefined fixed orientation. The green and blue spheres represent the points that each manipulator can reach with a fixed end-effector orientation. The red spheres in the dual arm setup show the region that is reachable by both arms.

Once the reachability map is obtained, the canvas dimension is determined. We find a rectangle tightly bounding the intersection with the original image's aspect ratio within the desired depth range based on the robot base. Drawing poses $\tilde{X}$ are then re-scaled and relocated according to the rectangle. Note that this only applies to the flat surface. In the case of non-flat surfaces, we can use the reachability map for verifying the mapped drawing poses, whether they are reachable or not.

For dual-arm drawing, we decide to assign each arm to draw a specific color, sharing a canvas space, instead of sharing the colors. Thus, we find an intersection between the reachable point set by both arms (*i.e.,* red spheres) and the target surface.

In the case of the mobile manipulator, as the system's reachability is limited only perpendicular to the ground (*i.e.,* robot's height), we split the entire canvas into a few sub-canvas by the size the manipulator can draw at its fixed base pose. We repeat drawing a sub-canvas and move to the next sub-canvas. The detailed method will be introduced in Section VII.A.



(a) Dual Arm (b) Mobile Manipulator

Figure 6.1. Reachability for each manipulator

B. Path-wise Inverse Kinematics

Once the target drawing poses $\tilde{\mathcal{X}}$ are decided, they are fed to robots. Robotic drawing is equivalent to finding a continuous path ξ in the robot's configuration space so that its end-effector follows the given path; this problem is also known as path-wise IK [73, 74]. Moreover, path-wise IK is suitable for setting a robot's kinematic and dynamic constraints while following the robot trajectory. In the case of robot drawing, achieving a feasible motion with no sudden jumps is crucial. Plus, it is essential to follow the given end-effector poses accurately not to ruin the resulting drawing. Thus, we solve the path-wise IK problem by iteratively solving the IK for the end-effector poses $\tilde{\mathcal{X}}$ with the minimum distance objective in the configuration space. As a result, we get a set of continuous robot joint configurations $\xi \subset \mathbb{R}^{DoF}$.

C. Impedance-controlled Drawing

Although the robot following joint configurations ξ, calculated from the previous section, will produce successful drawing results, we explore more on robot-surface contact. Impedance-controlled robots interact with the environment by employing a mass-spring-damper-like system that actively controls the robots [49]. Such a system is well-suited to tasks in which contact forces should be kept small, whereas the accurate regulation of contact forces is not mandatory. Thus, impedance control serves nicely for pen drawing tasks.

Using surface estimation and mapping, our drawing robot is fully equipped with a set of drawing poses $\tilde{\mathcal{X}}$ in 3D that can be drawn on the target surface. However, to exert the proper, compliant force at the pen tip (the end-effector), as well as to compensate for possible estimation and sim-to-real errors, we adapt the impedance control method. Impedance control is a hybrid, position- and force-controlled method that was proposed for interaction between a robot and an unstructured environment [45]. By employing a mass-spring-damper-like system, impedance

control allows a robot to react in a compliant manner to external obstacles. By considering a certain offset value (*i.e.,* impedance) for each drawing point, impedance control results in continuous contact with the surface during the entire drawing session.

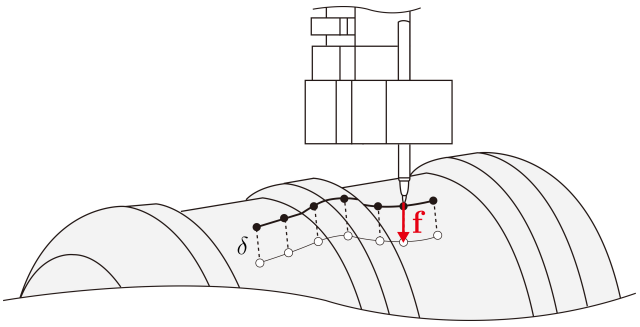


Figure 6.2. Compliant force ($\mathbf{f}$) generated by the impedance-controlled manipulator

The pen tip attached to the robot manipulator is always perpendicular to the wall (Equation 5.3). We configured the impedance controller in such a way that the robot is compliant only in the pen tip heading direction. Additionally, in order to exert an appropriate amount of compliant force at the pen tip, a small deviation between the target position and the physical position of the pen tip needs to be provided to the impedance controller.

Simply having the set of drawing points $\tilde{\mathcal{X}} = \{\langle \mathbf{x}_i, \mathbf{r}_i \rangle | \mathbf{x}_i \in \mathbb{R}^3, \mathbf{r}_i \in SO(3)\}$ define the target drawing poses can result in a lack of sufficient pen pressure and may be very sensitive to surface estimation error. Therefore, the target position $\mathbf{x}_i$ was modified to:

$$\mathbf{x}'_i = \mathbf{x}_i - \delta \mathbf{n}_i, \tag{6.1}$$

where δ is a user-defined gain value that controls the pen pressure, and $-\mathbf{n}_i$ is the direction opposite to surface normal, which is parallel to the z-axis of the pen frame. This deviation results in a compliant force $\mathbf{f} = k\delta$ along the z-axis of the pen frame where k is the spring stiffness.

As a result, the pen-type end-effector traces out the position of spline curves while maintaining contact with the surface, exerting an almost uniform level of compliant forces regardless of the shape of the surface. Figure 6.2 helps to understand how the impedance control works in our system. The thick black line represents spline curves on the physical surface that need to be drawn with a set of black dots for control points, also corresponding to the origin of the pen frame. Meanwhile, the thin line with a set of circular dots represents the target spline curves on the virtual surface, located slightly under the physical surface by δ.

VII. Robotic Drawing Extension

In this chapter, we further extend the robotic drawing problem to large-scale drawing problem and dual-arm drawing problem, fully utilizing the characteristics of the robotic hardware.

A. Large Canvas Drawing

Our system is not limited to the target drawing canvas size, as the robotic platform can move around freely. In particular, if the input drawing requires a large canvas, the manipulator will not be able to draw the whole drawing in one fixed place. Therefore, we need to find a set of robot configurations that can reach the given drawing path.

1. Problem Formulation

The objective of the large canvas drawing problem is to find a path, ξ, for the robot that allows the manipulator to reach the given drawing path, $\tilde{\mathcal{X}}$. This problem can be formally defined as follows:

Problem 1 Drawing Coverage

Find a path ξ defined in the robot's configuration space $\mathfrak{C}$ such that the end-effector of the robot following ξ is able to cover every drawing pose in the drawing path $\tilde{\mathcal{X}}$ in Equation 5.3. In other words,

$$\tilde{\mathcal{X}} \subset \bigcup_{\forall \mathbf{q} \in \xi} \mathcal{F}(\mathbf{q}), \tag{7.1}$$

where $\mathcal{F}$ is the forward kinematics map from $\mathfrak{C}$ to $\mathrm{SE}(3)$.

In our case, $\mathfrak{C} = \mathrm{SE}(2) \times \mathbb{R}^7$. In general, finding ξ in Problem 1 with a high degrees-of-freedom robot is difficult. Adding a constraint or an objective would make the problem even more difficult. Thus, we reformulate the problem based on the notion of *path coverage*.

First, we define the robot's reachability and coverage as follows:

Definition 1 (Reachability) *Given a robot $\mathcal{R}$ with its forward kinematics map $\mathcal{F}_\mathcal{R}$, the reachability $\mathcal{L}_\mathcal{R}(\mathbf{c})$ for a configuration $\mathbf{c} \in$ SE(3) is defined as a mapping from SE(3) to $\{0, 1\}$ that indicates whether $\mathcal{F}_\mathcal{R}(\mathbf{q}) = \mathbf{c}$ for $\exists \mathbf{q} \in \mathfrak{C}$ (1) or not (0).*

Definition 2 (Coverage) *The coverage $\mathcal{C}$ for a robot $\mathcal{R}$ with reachability $\mathcal{L}_\mathcal{R}$ is defined as a set of points that are reachable by the end-effector of the robot. In other words,*

$$\mathcal{C} = \{\forall \mathbf{p} \in \mathbb{R}^3 \mid \exists \mathbf{q} \in \mathfrak{C}, \exists \mathbf{r} \in \mathrm{SO}(3), \mathcal{L}_{\mathcal{R}(\mathbf{q})}(\langle \mathbf{p}, \mathbf{r} \rangle) = 1\}, \tag{7.2}$$

where $\mathcal{R}(\mathbf{q})$ represents the robot placed in a configuration $\mathbf{q}$.

Note that when $\mathbf{q}$ is bounded, $\mathcal{C}$ is compact. Then, instead of solving Problem 1 exactly, we first discretize the drawing path $\tilde{\mathcal{X}}$ with a finite set of points $\mathcal{S}$, and we solve the following coverage problem to approximate Problem 1:

Problem 2 Path Coverage

Given a finite point set $\mathcal{S}$ in 3D that approximates the target drawing path $\tilde{\mathcal{X}}$, find a minimal number of compact sets $\mathcal{C}_i$ such that their union includes $\mathcal{S}$:

$$\min\{n \mid \mathcal{S} \subset \bigcup_{i=1}^{n} \mathcal{C}_i\}, \tag{7.3}$$

where each $\mathcal{C}_i$ denotes a subset of the robot's coverage $\mathcal{C}$ as defined by Definition 2.

While Problem 2 approximates the drawing coverage of Problem 1, it still requires complicated geometric and combinatorial optimization. Thus, we further simplify the problem by restricting each compact set $\mathcal{C}_i$ to be a simple geometric primitive in $\mathbb{R}^3$, such as a sphere, and pre-computing it using the coverage of the robot. Furthermore, we use randomization to reduce the universe of $\{\mathcal{C}_i\}$ and rely on greedy-based combinatorial optimization to solve Problem 2. After a solution to Problem 3 is obtained, we use simple local planning to continuously switch the robot poses from one coverage to another coverage, which yields the entire robot path ξ.

2. Discretization and Reduction

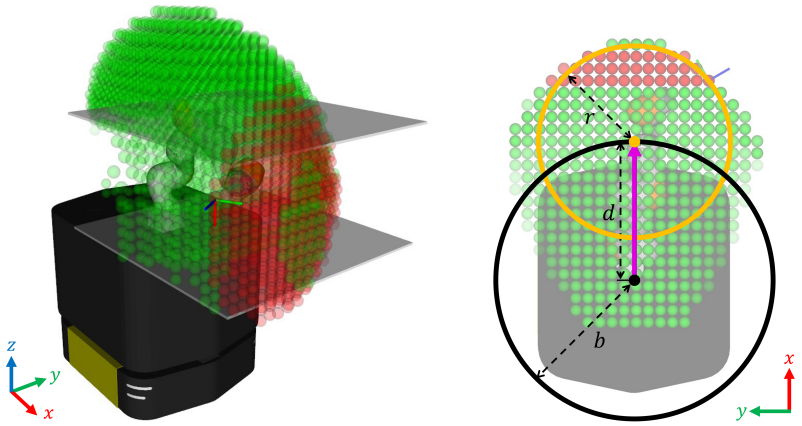
In order to simplify the problem, we first introduce the notion of *coverage map*. The coverage map $\mathcal{M}$ for a robot $\mathcal{R}$ with reachability $\mathcal{L}_{\mathcal{R}}$ is defined as a set of points that are reachable by the end-effector of $\mathcal{R}$ when the base frame of $\mathcal{R}$ is fixed at $\mathbf{p}_{base} \in$ SE(2); *i.e.,* the configuration space $\mathfrak{C}$ in Equation 7.2 is restricted to $\mathbf{p}_{base} \times \mathbb{R}^7$.

The coverage map is bounded and sometimes called the *reachability map* in the literature [72, 75]. In practice, the map is pre-computed by discretizing both the configuration and the workspaces. In our case, we construct the coverage map as follows:

1. Discretize the Cartesian workspace in $\mathbb{R}^3$ with $\{\mathbf{p}_d\}$.
2. Discretize the configuration space in SO(3) with $\{\mathbf{r}_d\}$.
3. For each $\mathbf{p}_d$, if $\mathcal{L}_{\mathcal{R}(\mathbf{p}_{base} \times \mathbb{R}^7)}(\langle \mathbf{p}_d, \mathbf{r_d} \rangle) = 1$ for all $\mathbf{r}_d$ using either forward kine

Figure 7.1 shows the resulting coverage map $\mathcal{M}$ of our robot, where the green spheres correspond to each element in $\mathcal{M}$. Note that $\mathcal{M}$ is a discretized version of $\mathcal{C}_i$ when the base frame is fixed.

In general, the geometry of $\mathcal{C}_i$ may be highly non-convex and may contain holes due to kinematic limits [72]. Even though a simple geometric primitive like a sphere may not closely



(a) (b)

Figure 7.1. Coverage map of the manipulator in 3D with 0.05m resolution

approximate the complicated geometry $\mathcal{C}_i$, we nonetheless opted for the sphere to approximate the geometry due to its simplicity, which will in turn ease the burden of reachability checking—the spherical approximation of $\mathcal{C}_i$ using $\mathcal{M}$ can quickly find reachable points on the target surface.

We also reduce the coverage dimension in Problem 2 from 3D to 2D by assuming a certain geometric characteristic of the target surface. We assume that the geometry of our target canvas is a surface of extrusion (*e.g.,* see Figure 7.3)—the surface is *vertically* extruded along the z-axis from a 2D planar curve (*i.e.,* the profile curve) defined in the x-y plane. Thus, because the geometric characteristics of the target surface still pertain to the 2D profile curve (or the 2D projection of the surface), we can effectively project the spherical coverage of $\mathcal{C}_i$ in 3D to a circle in 2D. We also use another circle in 2D to bound the mobile base, $\mathcal{B}_i$, which will be used in checking collisions with the canvas. The use of the bounding circle $\mathcal{B}_i$ not only allows the mobile base to remain collision-free but also compensates for the area that is unreachable due to the spherical approximation of the geometry of $\mathcal{C}_i$. Similarly, the target drawing path $\mathcal{S}$ is also projected from 3D to a 2D point set $\mathcal{S}' \subset \mathbb{R}^2$.

Now, our path coverage problem in 2D can be stated as follows:

Problem 3 Circular Path Coverage

Given a finite point set $\mathcal{S}'$ in 2D that approximates the drawing path, find a minimal number of coverage circles $\mathcal{C}_i$ such that their union covers $\mathcal{S}'$:

$$\min\{n \mid \mathcal{S}' \subset \bigcup_{i=1}^{n} \mathcal{C}_i,\ \mathcal{S}' \cap \bigcup_{i=1}^{n} \mathcal{B}_i = \emptyset\}, \tag{7.4}$$

where $\mathcal{B}_i$ denotes the circle bounding the robot's mobile base relevant to $\mathcal{C}_i$.

Computing $\mathcal{C}_i$ and $\mathcal{B}_i$

A simple method is employed to decide the geometry of the circles that represent the coverage candidates $\mathcal{C}_i$ and the mobile base $\mathcal{B}_i$ in 2D, as well as to determine the inter-center distance d between $\mathcal{C}_i$ and $\mathcal{B}_i$ (Fig 6.1(b)). First, from the pre-computed geometric distribution of *cells* in $\mathcal{M}$, we observe the following results:

- The overall discrete geometry of $\mathcal{M}$ is similar to a spherical shell, where the inner shell corresponds to the area that is unreachable by the manipulator.
- We orient the mobile base in such a way that the drawing occurs only in the forward-facing direction of the base, *i.e.,* the x-axis of the base frame. Thus, the main radial direction of the spherical shell $\mathcal{M}$ is aligned with the x-axis, and its radial extent is bounded.
- We assume that the target drawing path (or the canvas) has a limited vertical height (along the z-axis). Thus, the spherical shell is truncated by the slabs defined by the vertical height.

The projected geometry of $\mathcal{M}$ onto the x-y plane looks like Figure 6.1(b) and, based on the aforementioned observations, we bound only the first few consecutive rows (the red spheres in the figure) in the projected $\mathcal{M}$ using a circle (the yellow circle in the figure). These rows are contiguous so that they correspond to the internals of the spherical shell. Using the extrema

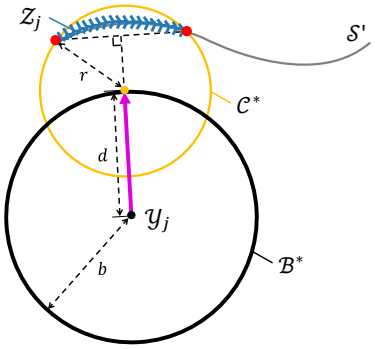


Figure 7.2. Robot base pose calculation from the coverage circle $\mathcal{C}^*$

of these rows, as well as the constraint that the circle's center is aligned with the x-axis, we determine the circle of $\mathcal{C}_i$. The circular bound $\mathcal{B}_i$ of the mobile base with some safety offset is also computed (the black circle in the figure); the inter-center distance d is also obtained from these two circles, $\mathcal{C}_i$ and $\mathcal{B}_i$. Finally, to better approximate the spherical-shell geometry of $\mathcal{M}$, in the next section, we use $\mathcal{C}_i - \mathcal{B}_i$ instead of using just $\mathcal{C}_i$, where "$-$" is the Boolean difference. In fact, Equation 7.4 encodes this Boolean difference operation. Our method for finding $\mathcal{C}_i, \mathcal{B}_i$ is simple, yet conservative and effective in practice.

3. Set-Cover Algorithm

Now we solve the combinatorial optimization of Problem 3. This problem is essentially a set-cover problem, which is known to be NP-hard [15]. In our case, we rely on a rather simple but effective approach based on randomization and greedy algorithms. Moreover, we find the coverage set that covers the entire target surface, not just the drawing path; *i.e.,* $\mathcal{S}'$ in Equation 7.4 approximates the target surface. Thus, this optimization is independent of the drawing input and is pre-computed in our system. Algorithm 1 summarizes an overview of our algorithm.

Algorithm 1: Greedy Algorithm for Set Cover

Input : $\mathcal{S}'$: point sets in $\mathbb{R}^2$ for the target surface

Output: $\mathcal{T} = \{\langle \mathcal{Y}_j, \mathcal{Z}_j \rangle\}$: tuples of base poses in SE(2) and covered point sets in $\mathbb{R}^2$

1 $\mathcal{T} \leftarrow \emptyset$;
2 A set of densely-sampled candidate circles $\{\mathcal{C}_i\}$;
3 **while** $\bigcup \mathcal{Z}_j \neq \mathcal{S}'$ **do**
4 $\quad \mathcal{C}^* = \underset{\mathcal{C}' \in \{\mathcal{C}_i\}}{\operatorname{argmax}} |(\mathcal{S}' - \bigcup \mathcal{Z}_j) \cap \mathcal{C}'|$;
5 $\quad$ Find the base pose $\mathbf{p}_{base}$ and its bounding circle $\mathcal{B}^*$ corresponding to $\mathcal{C}^*$;
6 $\quad$ **if** $\mathcal{S}' \cap \mathcal{B}^* \neq \emptyset$ **then**
7 $\quad\quad X_j = \mathcal{C}^* \cap \mathcal{S}'$;
8 $\quad\quad \mathcal{Y}_j = \mathbf{p}_{base}$;
9 $\quad\quad$ Add $\mathcal{T}_j = \langle \mathcal{Y}_j, \mathcal{Z}_j \rangle$ to $\mathcal{T}$;
10 $\quad$ **end**
11 $\quad$ Remove $\mathcal{C}^*$ from $\{\mathcal{C}_i\}$;
12 **end**

First, we populate the universe of $\{\mathcal{C}_i\}$ by densely random-sampling circles in $\mathbb{R}^2$ until the union of the sampled circles covers the target surface (line 2). From among these candidate circles, we find the circle $\mathcal{C}^*$ that includes the maximum number of points on the target surface that have not been covered by the already-chosen circles (line 4). For such a $\mathcal{C}^*$, the set of all the surface points that are inside the circle $\mathcal{C}^*$ is $\mathcal{Z}^* = \mathcal{C}^* \cap \mathcal{S}'$. Then, we find the configuration $\mathbf{p}_{base}$ of the robot base and its bounding circle $\mathcal{B}^*$ that corresponds to $\mathcal{C}^*$ (line 5); $\mathbf{p}_{base}$ is calculated by moving in the perpendicular direction of the line connecting the two extreme points in $\mathcal{Z}^*$ (Figure 7.2). Two extreme points (red dots) of $\mathcal{S}'$ inside the coverage circle $\mathcal{C}^*$ are identified. $\mathbf{p}_{base}$ is displaced from the center of $\mathcal{C}^*$ by d, the pre-computed distance between the centers of two circles, and the orientation of $\mathbf{p}_{base}$ is normal to the line connecting the extreme points. After verifying that the robot's base is collision-free from the target surface when the robot is placed at $\mathbf{p}_{base}$ (line 6), we add $\mathcal{Z}^*$ to $\mathcal{Z}$, the set of already-covered points. We repeat this until the union of $\mathcal{Z}_j$ includes all the target surface points (line 3). Figure 7.3 shows examples for three different target surfaces. In the result, different colored points on the surface represent the surface area covered by each candidate circle.

As a result of Algorithm 1, we obtain a set of tuples $\mathcal{T}_j = \langle \mathcal{Y}_j, \mathcal{Z}_j \rangle$ consisting of a mobile base pose $\mathcal{Y}_j$ and the point set $\mathcal{Z}_j$ covered by $\mathcal{Y}_j$. We then partition the input drawing points $\tilde{\mathcal{X}}$ into a subset of points $\mathcal{X}_j$ according to the coverage: *i.e.,* $\mathcal{X}_j = \mathcal{Z}_j \cap \tilde{\mathcal{X}}$. $\mathcal{X}_j$ is fed into a manipulator, and the manipulator performs the robotic drawing task using impedance control (Section VI.C). The sequence of $\mathcal{Y}_j$ values is spatially sorted and fed into the mobile base. Next, a linear interpolatory motion is employed to navigate from $\mathcal{Y}_j$ to $\mathcal{Y}_{j+1}$ successively. We repeat the alternation of drawing motion and mobile base navigation until we traverse the tuple set $\mathcal{T}$. Figure 7.4 shows the example of the path coverage result applied to the robotic drawing. Each color represents the drawing split concerning $\mathcal{Z}_j$. The robot base is placed according to $\mathcal{Y}_j$.

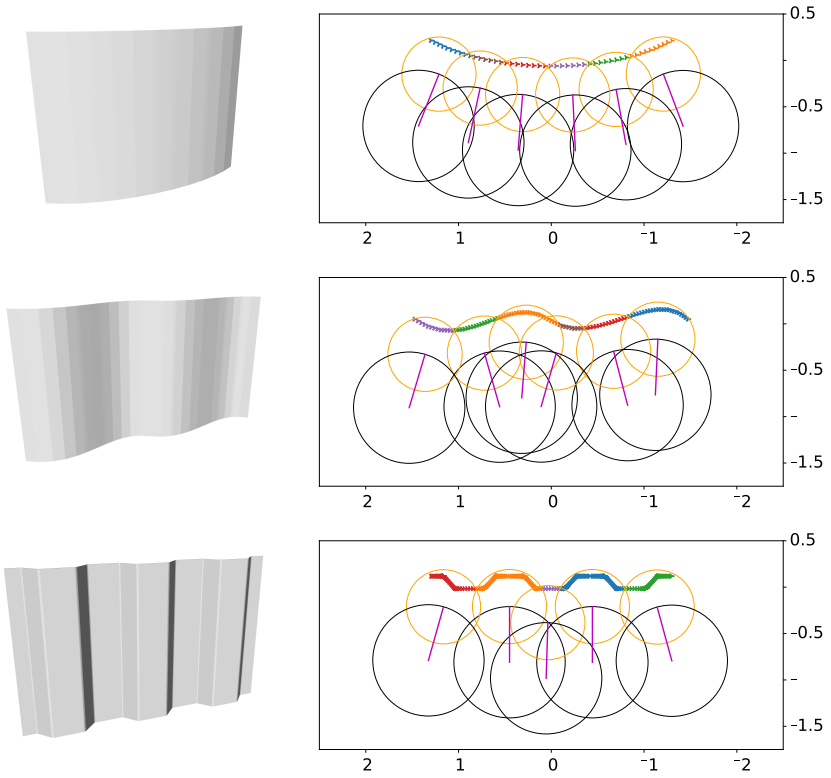


Figure 7.3. Three canvas surface models in 3D (circular, wave, curved) (left column), and the corresponding path coverage results projected in 2D (right column)

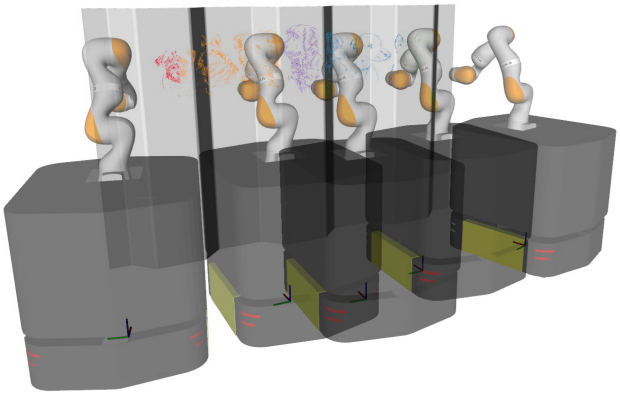


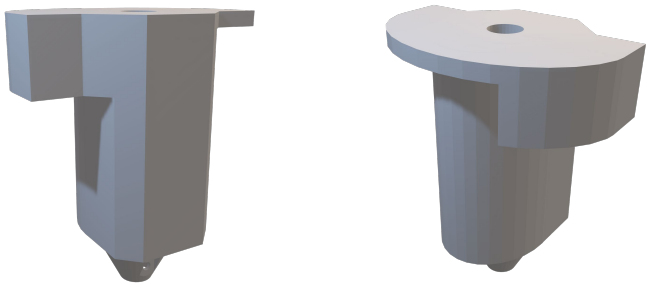
Figure 7.4. Drawing on a curved surface segmented according to the path coverage results and the corresponding robot base poses

B. Dual-arm Drawing

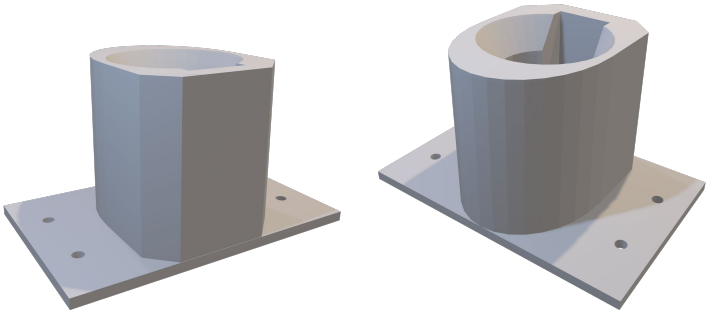
A robotic pen drawing requires firmly attaching the pen to the robot's end-effector as the task needs to resist contact force from the canvas surface. A possible option for this requirement is to rigidly attach the pen to the robot, which makes human intervention inevitable for changing the pen to a different color. On the other hand, one may use a robotic gripper to grab a pen and switch to another if necessary. This pen tool change mechanism is essentially a pick-and-place task, which is very challenging due to various uncertainties of sensing and dynamics in the real world. This section explains our approach to the pen-changing problem that is robust for multi-color drawing systems.

1. Drawing Tool Design

We design a pen-holding tool that the 3-finger gripper can robustly grasp even under slight motion perturbations due to position or motion errors, which also ensures that the gripper always holds the pen in the same pose. Figure 7.5 shows the 3D model of our design. We flatten the side of the tool so that when the gripper is half-closed, it can still adjust the vertical orientation with the fingers. We created upper plates in the holding structure that match the fingers' height so that they can fine-tune the gripper's horizontal orientation when lifted vertically by a half-closed gripper. The tool shape that matches the palm of the gripper once again guarantees to grasp the tool when closing the fingers robustly. The pen tool's docking structure is a rounded concave shape that matches the surface contour of the pen-holding tool. It allows the tool to slide into the bottom when released from the gripper. Another critical and practical aspect of the docking structure is preventing the pen from drying during the long drawing session. Our drawing tool and its docking structure design are experimentally proven robust for many tool changes.



(a) Pen tool



(b) Docking structure

Figure 7.5. 3D model of the drawing tool for dual-arm drawing in two different views

2. Tool-change Mechanism

Along with the novel design of a pen-holding tool, we also devise a robust tool-change mechanism. Figure 7.6 shows snapshots of tool-change sequences using the pen-holding tool and its docking structure. The tool-change steps can be discretized as follows:

1. **Approach**: We first approach toward the tool.
2. **Half-grasp**: We half-close the gripper, which will fit the flat shape of the side of the tool. This will fine-tune the yaw orientation of the tool.
3. **Vertical lift**: We vertically lift the drawing tool. The upper plates in the pen-holding tool match the finger height of the gripper. This again fine-tunes the tool's horizontal orientation.
4. **Full-grasp**: Finally, we fully grasp the tool. The flat shape of the tool matches the shape of the palm of the gripper and once again guarantees to grasp the tool when closing the fingers robustly.

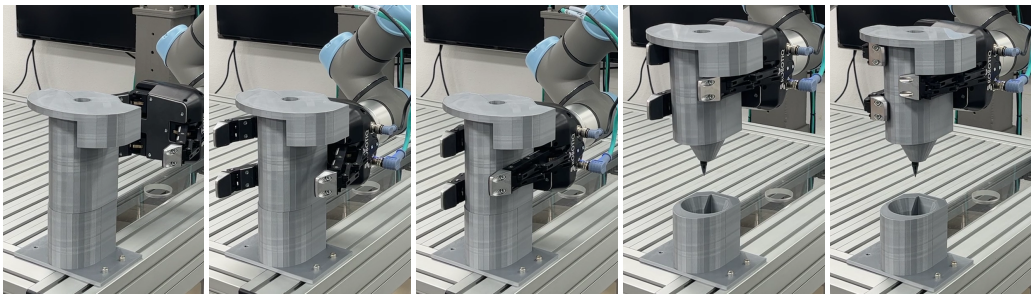


Figure 7.6. Pen tool picking sequences using a 3-finger gripper in five stages

VIII. Results and Discussion

In this section, we describe the implementation details, demonstrate the drawing results of our system and discuss them in detail.

A. Implementation Details

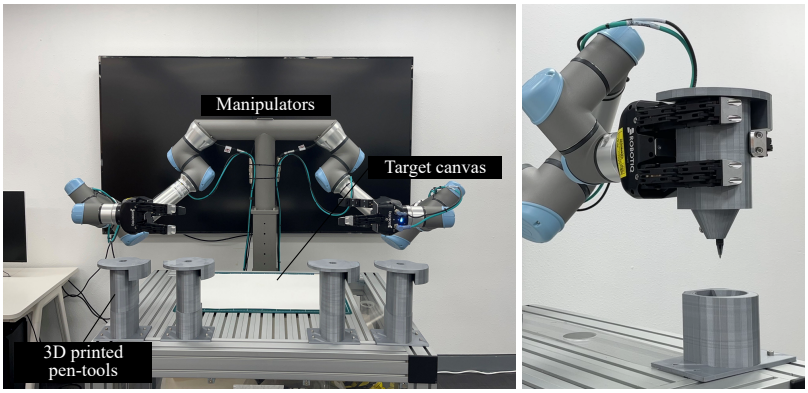
As shown in Figure 8.1, we implemented our robotic drawing system with two different robotic hardware setups:

- **Dual Drawing Manipulators** consist of two UR5e manipulators, each equipped with a Robotiq 3-finger adaptive gripper. We designed a new pen tool that allows our gripper to hold the pen firmly despite sensing and mechanical errors. The drawing process is fully automated thanks to the gripper and our pen tool change mechanism.
- **Mobile Drawing Manipulator** consist of a KUKA LBR iiwa 7 R800 manipulator mounted on top of the omnidirectional mobile platform Ridgeback from Clearpath Robotics. A 3D-printed pen tool is rigidly attached to the robot's end-effector, which requires a manual change of colors. Thanks to its mobility, unlike previous robotic drawing systems, it is not limited to the drawing canvas size.

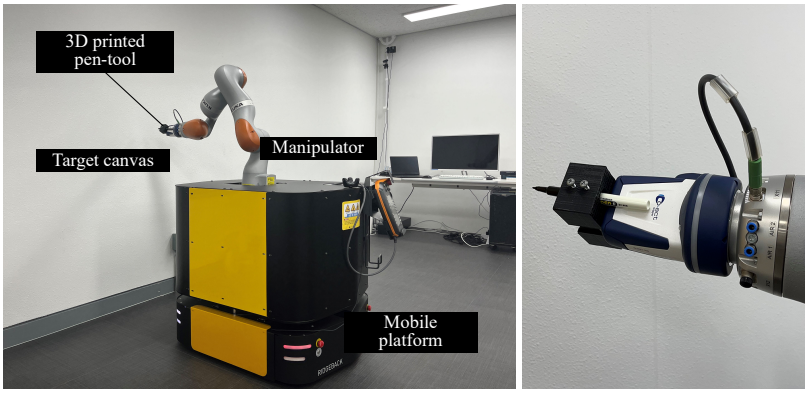
We use Python and C++ for the programming infrastructure, which runs on a PC equipped with an Intel i7-10 CPU and 16-GB RAM. We use Robot Operating System (ROS) Melodic framework under the Ubuntu 18.04 LTS operating system to communicate with the robots(sensors) and to perform the drawing task. We use Samsung Galaxy Tablet PC to run vector graphics under the Android operating system and an Intel RealSense ZR300 RGB-D camera to reconstruct

a 3D point cloud from the target canvas space. We use HTC VIVE tracker 3.0 with OpenVR [76] to localize both the robot and the target drawing wall. We use MoveIt! [73] with TRAC-IK [74] inverse kinematics solver for computing the robot trajectory following the Cartesian path.

To generate TSP art, we adapt the Linde-Buzo-Gray algorithm [63] to generate stippled points. We use Concorde [67] to solve TSP, an optimization solver widely regarded as the fastest TSP solver for large instances [68]. To generate SBR data, we adapt a simple painterly rendering algorithm [35] and a layered depth stroke-based rendering method [39]. Our TSP art and SBR data are generated on a PC equipped with AMD Ryzen 7 CPU and 32-GB memory.



(a) Dual Manipulators and the Pen-holding tools



(b) Mobile Manipulator and the Pen-holding tool

Figure 8.1. Robotic pen-art drawing system hardware setup

B. Qualitative Results

1. Artistic Drawing Results

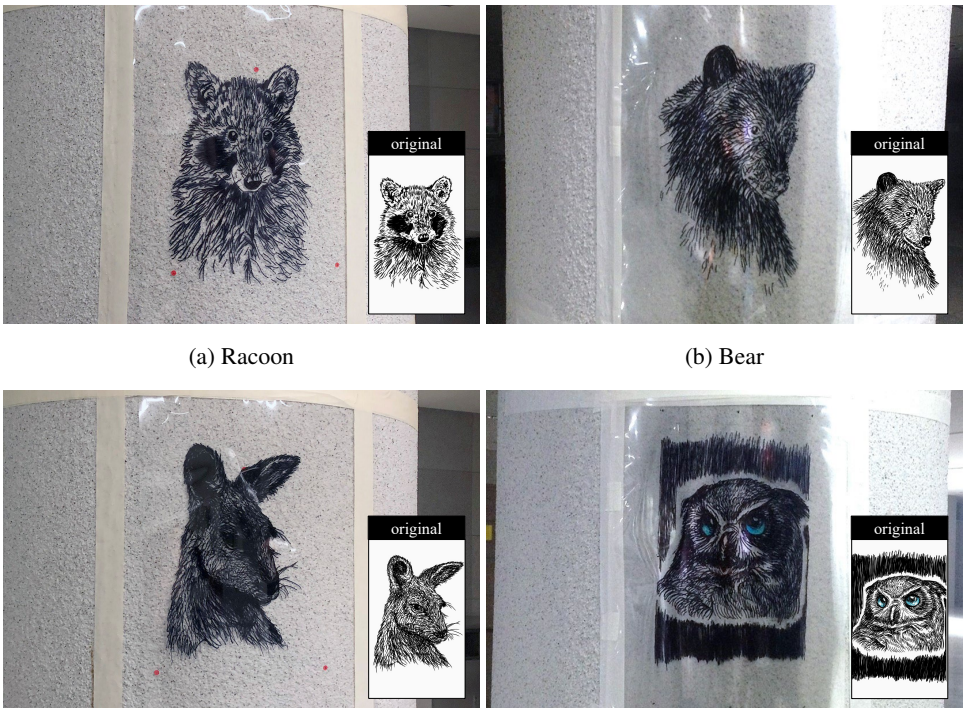
Artistic drawing results show the robotic pen drawings with aesthetic beauty drawn on diverse non-flat surfaces using an impedance-controlled manipulator. Some of them can also be categorized into large-scale drawing results (Violet and EWU graffiti). The original target drawings are generated using three different methods proposed in Chapter IV.

Vector Graphics

Figure 8.2 shows the artistic pen drawing results reproduced on diverse non-flat objects, a water tank, an acrylic half-sphere, a water bucket, and a traffic cone, using our system. The original drawings are drawn by an artist using our vector graphics engine proposed in [77]. The target surface data was not given to the robot. Instead, the robot performed adaptive sampling in order to collect the surface data. A simple orthographic projection (*i.e.,* projection mapping) is used to map a 2D drawing to 3D.

Figure 8.2. Drawing results on Diverse Non-flat objects

Figure 8.3 shows the artistic pen drawing results reproduced on a bumpy circular column wall. The original drawings are drawn by the artist using our vector graphics engine, and the target surface data is reconstructed as a point cloud using the RGB-D camera while the robot base remains fixed. The point cloud of the target surface consists of over 172K points. Our system is capable of drawing multiple color drawings by splitting the drawing tasks into different colored sets: *e.g.,* the Owl drawing consists of two different colors, black and blue, with blue in the eyes. However, because our 3D-printed pen holder has no tool-change mechanism, we manually change the pen when switching the drawing task as needed. The drawings use conformal mapping to map a 2D drawing to 3D, except for the Bear and Owl drawings, which use a simple orthographic projection.



(a) Racoon (b) Bear

(c) Kangaroo (d) Owl

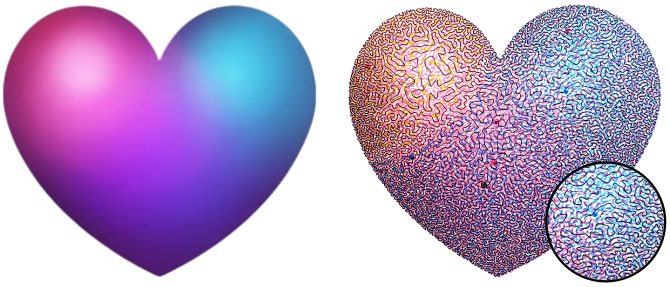
Figure 8.3. Drawing results on a bumpy, circular column wall

TSP Art

Figure 8.5, 8.4 and 8.6 show the artistic pen drawing results reproduced on a planar surface using our system. The target drawings are generated using the TSP Art method introduced in Section IV.C. Figure 8.4 and Figure 8.5 show the robotic pen drawing results reproduced by the mobile manipulator. Heart is drawn in one place with no mobile platform moving. Violet was split into three regions and Ewha Womans University (EWU) graffiti was split into nine regions.

Figure 8.6 shows the robotic pen drawing results drawn with a dual-arm robotic setup. Four color spaces, Cyan, Magenta, Yellow, and Black, are used to reproduce the color of the original input image. Our automatic tool change mechanism introduced in Subsection VII.B.2 is used to change the drawing tool color automatically. The drawing results show that our robotic drawing system performs the drawing task, following the resulting complex and sophisticated path composed of thousands of TSP sites.

Figure 8.4. TSP art results produced by the mobile manipulator, Ewha Womans University Graffiti



(a) Heart

(b) Violet

Figure 8.5. TSP art results produced by the mobile manipulator

(a) Starry Night (b) Big Ben

Figure 8.6. TSP art results produced by the dual manipulators

Stroke-based Rendering

Figure 8.7 shows the robotic pen drawing results drawn with a dual-arm robotic setup. The target drawings are generated using the layered depth stroke-based rendering method introduced in Section IV.B. The maximum number of strokes is set to 1,500, and the images are clustered into four colors. As the physical drawing pen, we used for the robotic drawing does not perfectly match the RGB color from the clustering, the robotic drawing result may differ from the original image. Figure 8.8 and 8.9 show the snapshots of the robotic drawing process. The color chips in the bottom right represent the color palette acquired by the clustering method. They show the color change and tool-change mechanism using the dual-arm well.

Figure 8.7. Stroke-based robotic drawings numbered 1 to 3 from top to bottom. original images (left) and our robotic drawings using four colors (right)

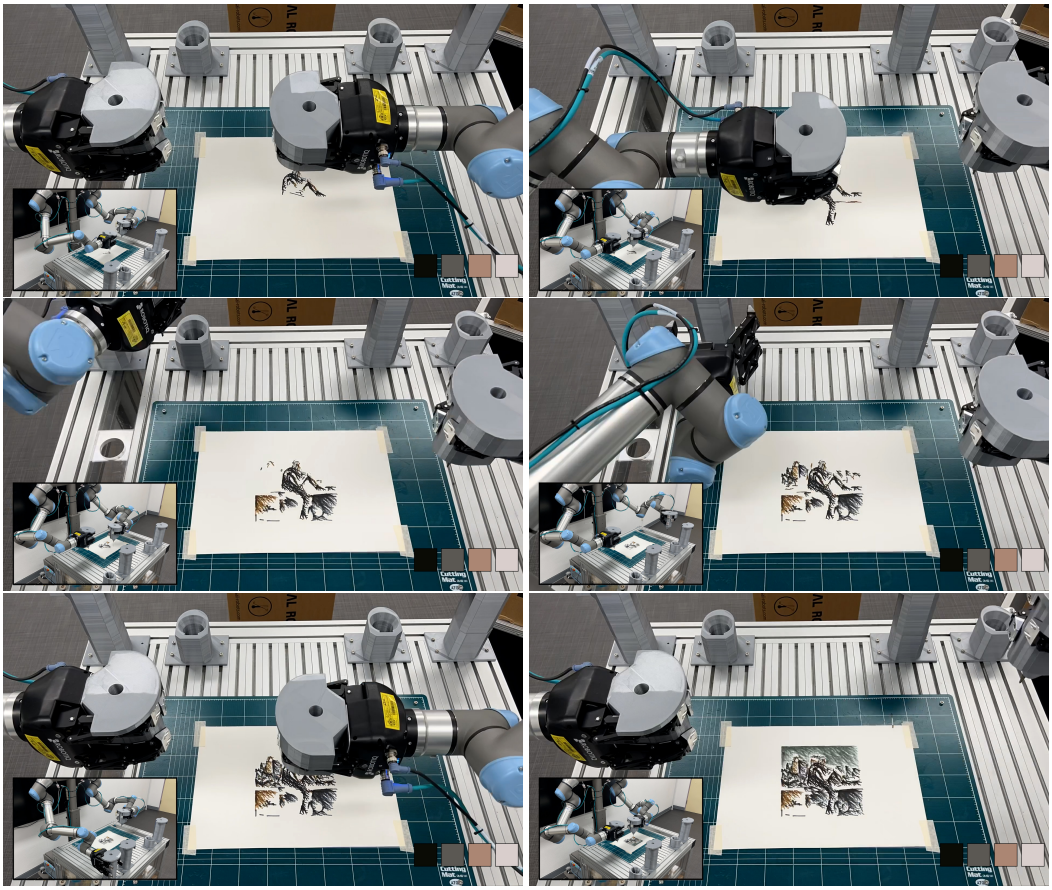


Figure 8.8. Robotic drawing sequences drawing the drawing #1

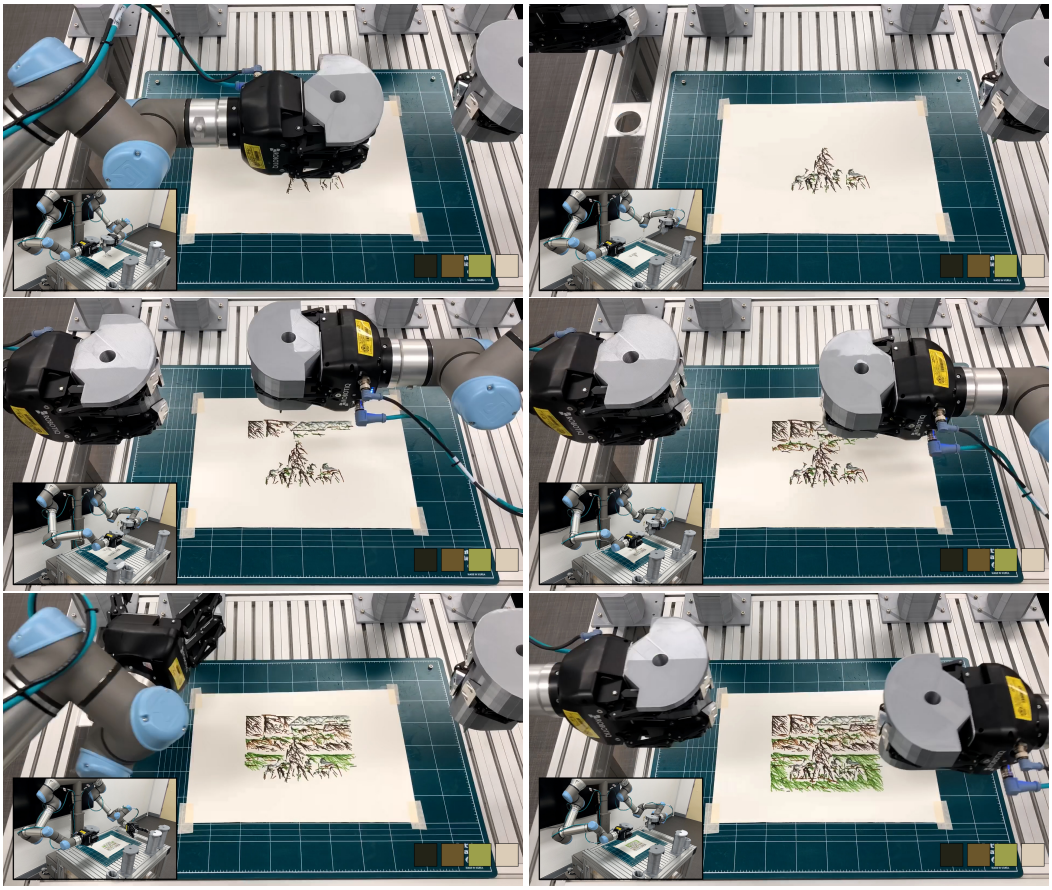


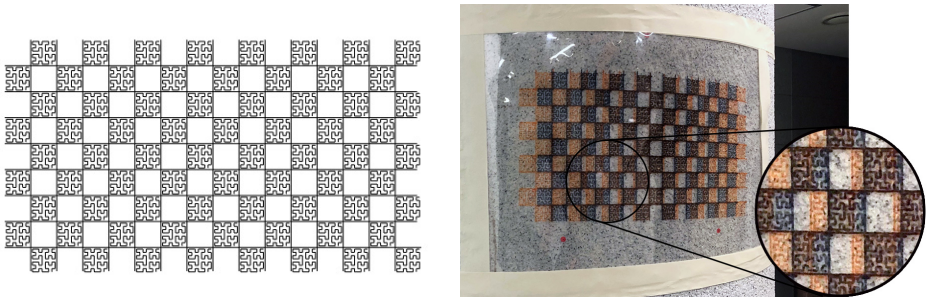
Figure 8.9. Robotic drawing sequences drawing the drawing #3

2. Pattern Drawing Results

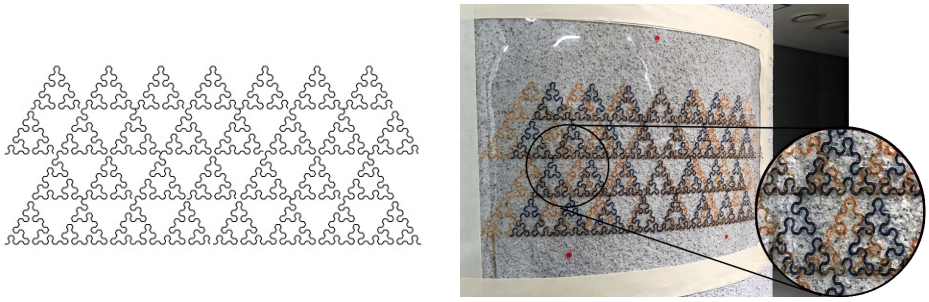
To highlight the effect of conformal mapping, we show comparisons between the results that use conformal mapping and those that use projection mapping by drawing fractal curves on a bumpy, circular column wall (Figure 8.10). We generated several fractal curves, including a Hilbert space-filling curve, a Sierpiński arrowhead curve, and Koch snowflake curves that form uniform squares, triangles, and hexagonal shapes. The black lines represent the robotic drawing results using conformal mapping, and the orange lines represent the results using projection mapping. Compared with the results for projection mapping, the conformal mapping reproduces the original drawing faithfully on the non-planar surface. On the other hand, the projection mapping method does not preserve the original side length, and the length gradually increases as it moves away from the center of the projection. As can be seen from the grid pattern results shown in Figure 8.10(a), the conformal mapping preserves the side lengths of the grid, but with the projection mapping, the length is increased by more than 20% in the worst case.

3. Large-scale Drawing Results

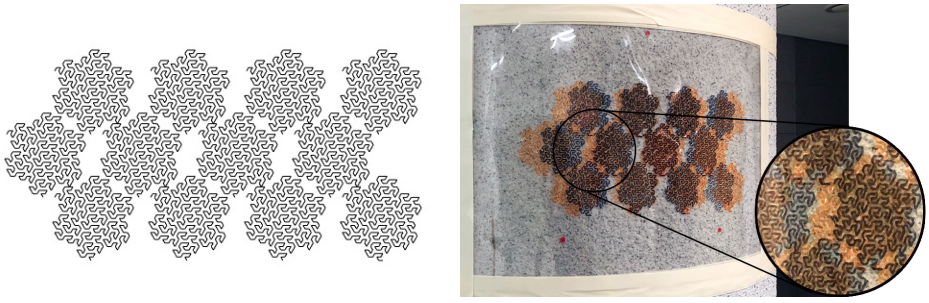
Figure 8.11 shows the artistic pen drawing results reproduced on a large, curved wall by the mobile manipulator. The original drawings are SVG images that are translated into a set of 2D strokes. Dogs and Farm drawings are split into five regions according to the appropriate canvas space measured in Sec. VII.A. Dogs drawing consists of three different colors, yellow in some of the eyes, red on the leash, and black for the rest. The target surface data was given as a polygonal mesh in 3D. There is a geometric discrepancy between the 3D mesh model and the physical wall. We will discuss this error in Section VIII.D. Nonetheless, by using impedance control, our system successfully reproduces artistically-pleasing drawings on a large, non-planar surface by compensating for the model discrepancy.



(a) Grid with Hilbert space-filling curve

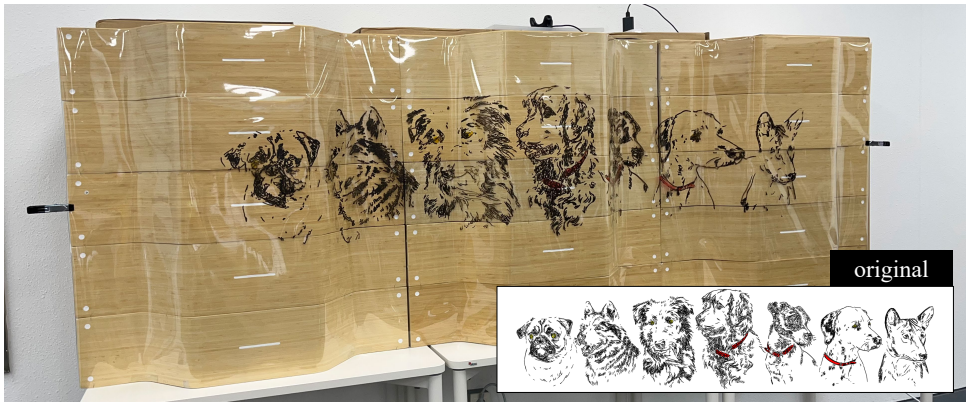


(b) Sierpiński arrowhead curve



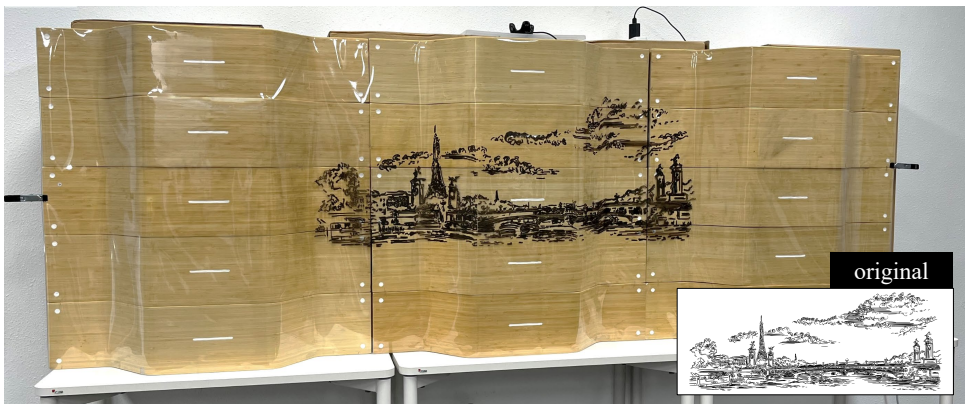
(c) Koch snowflake curve

Figure 8.10. Fractal curve drawing results on a bumpy, circular column wall



(a) Dogs

(b) Farm



(c) Paris

Figure 8.11. Robotic drawing results drawn on a curved surface

C. Quantitative Results

In this section, we provide the statistics of our experimental drawing results that were proposed in the previous section. The statistics include the number of drawings strokes, drawing points, size of drawing surface (*i.e.,* canvas size), and robot execution time.

The statistics for Figure 8.2 and 8.3, the artistic drawing results reproduced on diverse non-planar surfaces, are provided in Table 8.1. There is a difference in the robot execution time according to the difference in the total physical drawing stroke length that results from the shape inconsistency of the curved surface, water tank, and the acrylic half sphere. However, the difference is within 5 minutes, and we provide the average time of the two experiments.

The statistics for Figure 8.5, 8.4, and 8.6, the TSP art drawing results produced on a planar surface using two different robotic hardware, are provided in Table 8.2. The number of stippled points is the number of TSP nodes that are used to solve TSP. The table also provides the time it takes to run the stippling algorithm and to solve TSP. The number of drawing strokes is normally 1 per color task. However, in the case of Violet and EWU Graffiti, the strokes are divided by the robot's coverage.

Table 8.1. Robotic drawing experimental statistics on non-planar surfaces (Figure 8.2, 8.3)

Surface	**Water Tank & Half Sphere**	**Bucket**	**Traffic Cone**	**Circular column wall**			
Drawing	Tiger	Girl-1	Girl-2	Racoon	Kangaroo	Bear	Owl
Canvas size (mm^2)	126×262	126×262	94×184	252×491	252×491	252×491	252×491
# of Drawing Points	72,845	59,205	35,600	69,350	80,580	66,910	159,895
# of Drawing Strokes	523	1,585	966	860	3,147	1,520	1,942
Drawing Time (min.)	216	125	104	186	293	221	317

Table 8.2. TSP Art robotic drawing experimental statistics

Robotic Hardware	**Dual Arm (Figure 8.6)**		**Mobile Manipulator (Figure 8.5, 8.4)**		
Drawing	Starry Night	Big Ben	Heart	Violet	EWU Graffiti
Canvas size (mm^2)	315×250	214×300	400×350	850×300	3600×400
# of Stippled Points	81,591	76,257	21,664	95,155	154,475
Stippling Time (sec.)	32	34	2	35	32
TSP Solving Time (sec.)	6	11	4	17	43
Drawing Time (min.)	124	63	61	585	662

The statistics for Figure 8.7, the SBR drawing results produced on a planar surface using dual-arm, are provided in Table 8.3. The maximum number of strokes is set to 1,500. We use 4 colors to complete the drawings. Our drawing tool design and the tool-change mechanism showed robust tool change under more than 300 color changes.

The statistics for Figure 8.10, the pattern drawing results produced on a circular column wall using the impedance-controlled manipulator, are provided in Table 8.4. The table also includes the mapping calculation time. Even though we parallelize some of the mapping tasks, the drawing time can be further reduced by adopting a few acceleration techniques - for instance, a more efficient neighborhood search method with an efficient data structure such as Delaunay triangulation.

The statistics for Figure 8.11, the large-scale drawing results produced on a wooden curved wall using the mobile manipulator, are provided in Table 8.5. The table also includes the number of coverage poses (*i.e.,* C_i) that are used to split the drawings. Dogs and Farm were split into five regions, and Paris was split into three regions to perform large-scale robotic drawing tasks. The mobile manipulator repeats the drawing and navigates per region to complete the task.

The robot execution time is nearly proportional to the number of control points and size rather than to the number of strokes since the length of strokes may differ by the drawings. Robotic drawing by our system takes time because we executed our robot at a low speed, which was only 20% of the maximum joint velocity, for the safety of both the robot and any drawing collaborators (*i.e.,* humans). Through our experiments, we demonstrate that the drawings can be reproduced on target surfaces of various shapes and sizes.

Table 8.3. SBR drawing robotic drawing experimental statistics (Figure 8.7)

Drawing	1	2	3
Canvas Size (mm^2)	200×200	200×200	246×200
# of Drawing Strokes	1,462	1,489	1,475
# of Color Change	325	320	523
Drawing Time (hrs.)	6	7	10

Table 8.4. Pattern drawing robotic drawing experimental statistics (Figure 8.10)

Drawing	Grid	Arrowhead	Snowflake
Canvas Size (mm^2)	384×216	432×216	384×192
# of Drawing Points	6,930	2,360	4,128
# of Drawing Strokes	95	12	12
Mapping Time (sec.)	635	308	523
Drawing Time (min.)	54	14	19

Table 8.5. Large-scale robotic drawing experimental statistics (Figure 8.11)

Drawing	Dogs	Farm	Paris
Canvas size (mm^2)	2000×500	2000×500	1187×500
# of Coverage Poses	5	5	3
# of Drawing Points	25,727	20,787	20,154
# of Drawing Strokes	2,634	927	1,318
Drawing Time (min.)	582	494	307

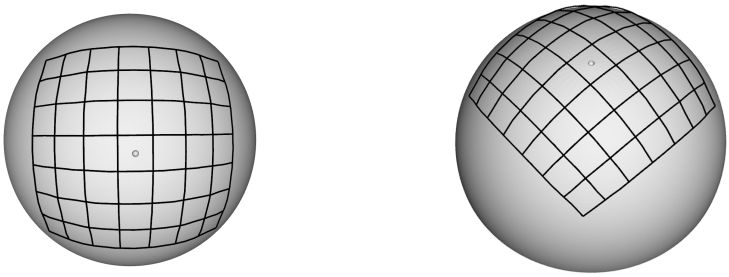
D. Discussion

There are some discussion points concerning the errors. In our system, there exist different sources of error that are attributed to surface deviation, coverage approximation, and localization. Because these errors can affect the quality of the final drawing results, in the following, we address how we compensate for the errors in our system.

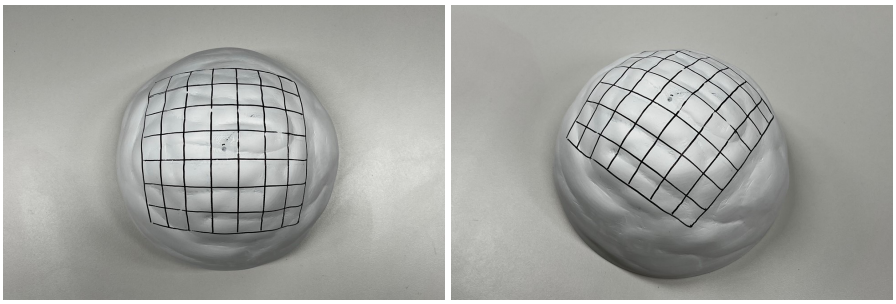
Surface deviation

We reconstruct the canvas surface automatically using an RGB-D camera or manually using geometric modeling software. In both cases, there may exist some discrepancy between the physical and the geometric models. For example, the wooden curved wall used in our experiments is built by stacking several wooden planks vertically and horizontally. From that, the step difference and the aperture between the planks occur, while our 3D model is neat and plain. This type of surface deviation, up to $5\mathrm{cm}$ in our experiment, was successfully managed with the use of impedance control, as explained in Section VI.C.

To further validate the robustness of impedance-controlled drawing, we designed an experiment where we asked the robot to draw a 7×7 square grid, as shown in Fig. 8.12(a) on a hemisphere that was provided as a target surface geometry to the robot. However, the actual hemispheric surface was perturbed by 3.5% of the Hausdorff metric [78] relative to the size of the hemisphere and was 3D-printed as shown in Fig. 8.12(b). Without knowing this change, the robot attempts to draw the grid on the modified hemisphere. We measured how much the two geometric quantities of each grid deviated from the original hemisphere: 1) the 1-1 aspect ratio of the four sides and 2) the right angles of the corners. In our experiments, the robot successfully drew the grid, and the relative deviations of the aspect ratio and the corner angles were measured as 3.0% and 4.4%, respectively. These deviations show a rate of changes similar to that of the 3.5% surface deviation, thanks to the impedance control scheme.



(a) Target input drawing on a half-sphere



(b) Real robot drawing result on a distorted half-sphere

Figure 8.12. Grid drawing experiment result on a half-sphere

Coverage approximation

We made a few assumptions to approximate the set-cover problem in Section VII.A.2. First, we reduce the problem dimension from 3D to 2D by assuming that the canvas-surface geometry is a surface of extrusion. Based on this assumption, we find a set of coverage circles, which yields a sub-optimal solution for the coverage problem. Our greedy set-cover algorithm adds to the sub-optimality. However, the result is still conservative, meaning that we can draw all the strokes with a slightly higher number of robot-base placements. Second, the manipulator coverage map is discretized and pre-computed. This only guarantees the reachability for a fixed drawing pose and not for the continuous drawing motion between poses. However, in practice, the discretization resolution of $0.05\mathrm{m}$ shows no problem in our drawing experiments. The redundancy of the manipulator also contributes to preventing failure.

Localization error

Unlike simulations where the current robot pose can be immediately obtained, in order to operate the physical robot in the real world, a proper localization method is required. Although there are many known methods for robot localization (*e.g.,* AMCL (Adaptive Monte Carlo localization), EKF(Extended Kalman Filter), etc), normally these methods need fine parameter tuning to fit the system and the errors between the components (*i.e.,* map, sensors, odometry) can accumulate. Our drawing system uses IR tracking devices to localize the mobile base as well as to locate the canvas. The IR tracking device (*e.g.,* the HTC Vive tracker) that we use for localization has a sub-millimeter precision [79]. The relatively small tracking error with the impedance control, a drawing error due to the localization was unnoticeable.

IX. Conclusion and Future Work

In this dissertation, we proposed a creative robotic pen-art drawing system that generates artistic pen art on large, non-planar surfaces. We provide diverse drawing path generation methods that plan a discrete set of stroke components on a 2D virtual canvas. Our system uses conformal mapping to project the 2D drawing onto the 3D canvas with minimal distortion. Our navigation algorithm plans the motion for the mobile base to cover the large surface using the coverage map. Our novel drawing tool design and the tool-change mechanism help fully automate our system with dual-arm manipulators. We show a variety of drawing results using our proposed system. The results demonstrate that our system successfully brings the digital image into the real world and is not limited by the canvas shape and size.

Most robot technologies have been developed by "mimicking" humans. This is also true for art, which was considered unique to humans. With the development of technology, robots have become a means of creation, but they are also gradually becoming the subject of creation. This dissertation also deals with 'how to draw like a person using artificial intelligence, but it presents robotic drawing that focuses on robotic ability. Robots are good at accurately following the given complex paths. Especially using high-DoF manipulators, we can perform robotic tasks with a larger working space and better tool positioning ability. Thus, the robotic drawing system using high-DoF manipulators can enrich artistic creativity by drawing complicated and sophisticated drawings on a target canvas with diverse shapes and sizes. Furthermore, by using the robotic gripper and by making the dual-arm manipulators cooperate with each other, it was possible to complete the colorful drawings.

By addressing the following limitations and future work, it is expected to improve our creative robotic pen-drawing system:

- **Creative drawing generation method** Although this dissertation mainly focuses on the robotic ability in robotic drawing systems, our system is limited to reproducing the given source image. More research on AI-based drawing generation methods is needed to generate creative art rather than simple drawing reproduction. Current AI image creation algorithms such as neural style transfer [80], Generative Adversarial Networks (GAN) [81], and diffusion models [82] reached a high level that it is difficult to distinguish whether it is man-made or not. Incorporating such ML-based image generation methods and implementing the system that the robot itself creates an artistic drawing will be an interesting future direction.

- **Closed-loop control** Our current system implementation does not consider any sensory input during the drawing process except for adaptive sampling. By considering additional sensory feedback, such as contact force and visual data, our system can become even more robust on errors. For example, our system relies on an IR-based tracking device for localizing the mobile platform. By doing fine visual alignment using computer vision directly on the canvas, we can reduce the drawing positional error, where even a one-millimeter misalignment in a drawing can be very noticeable in pen art. Additionally, visual feedback can also be applied to stroke placing, like in [12]. SBR method can be applied while directly comparing the physical canvas with the target instead of planning the whole sequence in virtual space. At last, having contact force feedback can catch the possible contact loss during the drawing like in [24] and perform robust pen drawing.

- **Human-robot collaboration** As a long-term future research direction, it would be very interesting to work on human-robot collaborative drawing [83]. This is accompanied by several additional robotic components. In addition to the aforementioned sensory feedback, safe motion planning with human collaborators and advanced artificial intelligence that catches the artist's intention and generates harmonious artwork are needed. In particular, having an artistic intention when creating art is an important feature for a work to be considered art and for a creator to be considered an artist [84]. Understanding the artist's intention and drawing a suitable drawing will be the most important key point.

To the best of our knowledge, we are the first to present the robotic drawing system that draws artistic drawings on large, non-planar surfaces using high-DoF manipulators. Robotic drawing systems come with various difficult computer graphics and robotic research problems. In this dissertation, we address these problems and provide possible solution techniques. The technologies applied to the system are not limited to robot drawing but can also be applied to various robotic applications that require continuous contacts with a target surface, such as milling and sanding.

Bibliography

[1] J. Reichardt, "Machines and art," *Leonardo*, vol. 20, no. 4, pp. 367–372, 1987.

[2] S. Calinon, J. Epiney, and A. Billard, "A humanoid robot drawing human portraits," in *IEEE-RAS International Conference on Humanoid Robots*, 2005.

[3] T. Lindemeier, S. Pirk, and O. Deussen, "Image stylization with a painting machine using semantic hints," *Computers and Graphics*, vol. 37, 2013.

[4] P. Tresset and F. F. Leymarie, "Portrait drawing by paul the robot," *Computers and Graphics*, vol. 37, 2013.

[5] S. Damith Herath, Christian Kroos, *Robots and Art*. Springer, 2016.

[6] P. McCorduck, *AARON'S CODE: Meta-Art, Artificial Intelligence, and the Work of Harold Cohen*. W. H. Freeman & Co, 1990.

[7] A. Hertzmann, "Non-photorealistic rendering and the science of art," in *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, 2010, pp. 147–157.

[8] Z. Zou, T. Shi, S. Qiu, Y. Yuan, and Z. Shi, "Stylized neural painting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 689–15 698.

[9] Z. Huang, S. Zhou, and W. Heng, "Learning to paint with model-based deep reinforcement learning," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8708–8717.

[10] J. Singh and L. Zheng, "Combining semantic guidance and deep reinforcement learning for generating human level paintings," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 382–16 391.

[11] T. Lindemeier, J. Metzner, L. Pollak, and O. Deussen, "Hardware-based non-photorealistic rendering using a painting robot," in *Computer graphics forum*, vol. 34, no. 2. Wiley Online Library, 2015, pp. 311–323.

[12] O. Deussen, T. Lindemeier, S. Pirk, and M. Tautzenberger, *Feedback-guided stroke placement for a painting machine*, 2012.

[13] S. A. Schneider and R. H. Cannon, "Object impedance control for cooperative manipulation: Theory and experimental results," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 383–394, 1992.

[14] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Six-dof impedance control based on angle/axis representations," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 289–300, 1999.

[15] V. V. Vazirani, *Approximation algorithms*. Springer, 2001, vol. 1.

[16] G. Jean-Pierre and Z. Saïd, "The artist robot: A robot drawing like a human artist," in *2012 IEEE International Conference on Industrial Technology*. IEEE, 2012, pp. 486–491.

[17] S. Jain, P. Gupta, V. Kumar, and K. Sharma, "A force-controlled portrait drawing robot," in *2015 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2015, pp. 3160–3165.

[18] L. Scalera, S. Seriani, A. Gasparetto, and P. Gallina, "Watercolour robotic painting: a novel automatic system for artistic rendering," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 3, pp. 871–886, 2019.

[19] L. Scalera, S. Seriani, A. Gasparetto, and P. Gallina, "Non-photorealistic rendering techniques for artistic robotic painting," *Robotics*, vol. 8, no. 1, p. 10, 2019.

[20] P. Schaldenbrand and J. Oh, "Content masked loss: Human-like brush stroke planning in a reinforcement learning painting agent," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 505–512.

[21] P. V. Arman, “Cloudpainter: an artificially intelligent painting robot,” 2018. [Online]. Available: http://www.cloudpainter.com/

[22] A. Bidgoli, M. L. De Guevara, C. Hsiung, J. Oh, and E. Kang, “Artistic style in robotic painting; a machine learning approach to learning brushstroke from human artists,” in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 412–418.

[23] Y. Park, S. Jeon, and T. Lee, “Robot learning to paint from demonstrations,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3053–3060.

[24] R. Liu, W. Wan, K. Koyama, and K. Harada, “Robust robotic 3-d drawing using closed-loop planning and online picked pens,” *IEEE Transactions on Robotics*, 2021.

[25] D. S. Arnon, “Topologically reliable display of algebraic curves,” *SIGGRAPH Comput. Graph.*, vol. 17, no. 3, pp. 219–227, Jul. 1983. [Online]. Available: http://doi.acm.org/10.1145/964967.801152

[26] G. Taubin, “Distance approximations for rasterizing implicit curves,” *ACM Trans. Graph.*, vol. 13, no. 1, pp. 3–42, Jan. 1994. [Online]. Available: http://doi.acm.org/10.1145/174462.174531

[27] A. Quint, “Scalable vector graphics,” *IEEE MultiMedia*, vol. 10, no. 3, pp. 99–102, July 2003.

[28] C. Loop and J. Blinn, “Resolution independent curve rendering using programmable graphics hardware,” in *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 1000–1009.

[29] M. J. Kilgard and J. Bolz, “Gpu-accelerated path rendering,” *ACM Trans. Graph.*, vol. 31, no. 6, pp. 172:1–172:10, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366191

[30] A. Orzan, A. Bousseau, P. Barla, H. Winnemöller, J. Thollot, and D. Salesin, "Diffusion curves: A vector representation for smooth-shaded images," *Commun. ACM*, vol. 56, no. 7, pp. 101–108, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2483852.2483873

[31] D. Sỳkora, J. Buriánek, and J. Zára, "Sketching cartoons by example." in *SBM*, 2005, pp. 27–33.

[32] J.-D. Favreau, F. Lafarge, and A. Bousseau, "Fidelity vs. simplicity: A global approach to line drawing vectorization," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 120:1–120:10, Jul. 2016. [Online]. Available: http://doi.acm.org/10.1145/2897824.2925946

[33] A. Hertzmann, "A survey of stroke-based rendering," *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 70–81, 2003.

[34] P. Litwinowicz, "Processing images and video for an impressionist effect," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97. USA: ACM Press/Addison-Wesley Publishing Co., 1997, p. 407–414. [Online]. Available: https://doi.org/10.1145/258734.258893

[35] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 453–460. [Online]. Available: https://doi.org/10.1145/280814.280951

[36] L. Lin, K. Zeng, H. Lv, Y. Wang, Y. Xu, and S.-C. Zhu, "Painterly animation using video semantics and feature correspondence," in *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 73–80. [Online]. Available: https://doi.org/10.1145/1809939.1809948

[37] P. O'Donovan and A. Hertzmann, "Anipaint: Interactive painterly animation from video," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 3, pp. 475–487, 2012.

[38] K. Zeng, M. Zhao, C. Xiong, and S.-C. Zhu, "From image parsing to painterly rendering," *ACM Trans. Graph.*, vol. 29, no. 1, dec 2009. [Online]. Available: https://doi.org/10.1145/1640443.1640445

[39] I. Ilinkin, D. Song, and Y. J. Kim, "Stroke-based rendering and planning for robotic performance of artistic drawing," *arXiv preprint arXiv:2210.07590*, 2022.

[40] A. Sheffer, K. Hormann, B. Levy, M. Desbrun, K. Zhou, E. Praun, and H. Hoppe, "Mesh parameterization: Theory and practice," *ACM SIGGRAPPH, course notes*, vol. 10, no. 1281500.1281510, 2007.

[41] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*. AK Peters/CRC Press, 2010.

[42] W. T. Tutte, "Convex representation of graphs," in *London Mathematical Society*, 1960.

[43] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," in *ACM transactions on graphics (TOG)*, vol. 21, no. 3, 2002, pp. 362–371.

[44] A. Sheffer and E. de Sturler, "Parameterization of faceted surfaces for meshing using angle-based flattening," *Engineering with Computers*, vol. 17, no. 3, pp. 326–337, Oct 2001. [Online]. Available: https://doi.org/10.1007/PL00013391

[45] N. Hogan, "Impedance control: An approach to manipulation," in *American Control Conference, 1984*. IEEE, 1984, pp. 304–313.

[46] T. Wimbock, C. Ott, and G. Hirzinger, "Impedance behaviors for two-handed manipulation: Design and experiments," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 4182–4189.

[47] J. Lee, P. H. Chang, and R. S. Jamisola, "Relative impedance control for dual-arm robots performing asymmetric bimanual tasks," *IEEE transactions on industrial electronics*, vol. 61, no. 7, pp. 3786–3796, 2014.

[48] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, "A survey of robot manipulation in contact," *arXiv preprint arXiv:2112.01942*, 2021.

[49] N. Hogan, "Stable execution of contact tasks using impedance control," in *IEEE International Conference on Robotics and Automation. Proceedings.*, vol. 4. IEEE, 1987, pp. 1047–1054.

[50] F. J. Abu-Dakka and M. Saveriano, "Variable impedance control and learning—a review," *Frontiers in Robotics and AI*, vol. 7, p. 590681, 2020.

[51] T. Strothotte and S. Schlechtweg, *Non-photorealistic computer graphics: modeling, rendering, and animation*. Morgan Kaufmann, 2002.

[52] Adobe, "Adobe illustrator (version 27.0)," 2022. [Online]. Available: https://www.adobe.com/products/illustrator.html

[53] D. Song and Y. J. Kim, "Distortion-free robotic surface-drawing using conformal mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 627–633.

[54] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-time hatching," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, p. 581.

[55] E. Zhang, "Graphics workshop," https://github.com/ekzhang/graphics-workshop, 2021.

[56] P. Decaudin, "Cartoon-looking rendering of 3d-scenes," *Syntim Project Inria*, vol. 6, no. 4, 1996.

[57] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[58] S. DiPaola, "Painterly rendered portraits from photographs using a knowledge-based approach," in *Human Vision and Electronic Imaging XII*, vol. 6492. SPIE, 2007, pp. 24–33.

[59] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9396–9405.

[60] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, 2022.

[61] R. Bosch. (2022) Tsp art instances. [Online]. Available: https://www.math.uwaterloo.ca/tsp/data/art/

[62] C. S. Kaplan, R. Bosch *et al.*, "Tsp art," in *Renaissance Banff: Mathematics, music, art, culture*. Bridges Conference, 2005, pp. 301–308.

[63] O. Deussen, M. Spicker, and Q. Zheng, "Weighted linde-buzo-gray stippling," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 233:1–233:12, Nov. 2017. [Online]. Available: http://doi.acm.org/10.1145/3130800.3130819

[64] M. A. Joshi, M. S. Raval, Y. H. Dandawate, K. R. Joshi, and S. P. Metkar, *Image and video compression: Fundamentals, Techniques, and Applications*. CRC press, 2014.

[65] A. Secord, "Weighted voronoi stippling," in *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, 2002, pp. 37–43.

[66] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[67] P. J. Campbell, "The traveling salesman problem: A computational study," *Mathematics Magazine*, vol. 80, no. 3, p. 238, 2007.

[68] S. A. Mulder and D. C. Wunsch II, "Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks," *Neural Networks*, vol. 16, no. 5-6, pp. 827–832, 2003.

[69] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.

[70] J. Liang, R. Lai, T. W. Wong, and H. Zhao, "Geometric understanding of point clouds using laplace-beltrami operator," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 214–221.

[71] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[72] O. Porges, R. Lampariello, J. Artigas, A. Wedler, C. Borst, and M. A. Roa, "Reachability and dexterity: Analysis and applications for space robotics," in *Workshop on Advanced Space Technologies for Robotics and Automation-ASTRA*, 2015.

[73] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[74] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 928–935.

[75] A. Makhal and A. K. Goins, "Reuleaux: Robot base placement by reachability analysis," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2018, pp. 137–142.

[76] V. Corporation, "Openvr," https://github.com/ValveSoftware/openvr.git, 2015.

[77] D. Song, T. Lee, and Y. J. Kim, "Artistic pen drawing on an arbitrary surface using an impedance-controlled robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4085–4090.

[78] M. Tang, M. Lee, and Y. J. Kim, "Interactive hausdorff distance computation for general polygonal models," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, pp. 1–9, 2009.

[79] M. Borges, A. Symington, B. Coltin, T. Smith, and R. Ventura, "Htc vive: Analysis and accuracy improvement," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2610–2615.

[80] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[81] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[82] R. Aboulaich, D. Meskine, and A. Souissi, "New diffusion models in image processing," *Computers & Mathematics with Applications*, vol. 56, no. 4, pp. 874–882, 2008.

[83] S. Chung, "Exquisite corpus," 2019. [Online]. Available: https://sougwen.com/project/exquisite-corpus

[84] E. S. Mikalonytė and M. Kneer, "Can artificial intelligence make art? folk intuitions as to whether ai-driven robots can be viewed as artists and produce art," *ACM Transactions on Human-Robot Interaction*, 2022.

국문초록

다자유도 로봇을 이용한 예술적인 로보틱 펜 드로잉 시스템

송다은
인공지능 · 소프트웨어학부
이화여자대학교 대학원

르네상스 이후 예술가들은 기존의 창작 방법에서 벗어나 새로운 기술과 기계를 접목하여 다양한 예술적 시도를 하였다. 로봇을 이용해 그림을 그리는 로보틱 드로잉 시스템(robotic drawing system)은 예술가들의 다양한 창의적인 시도 중 하나이다. 로봇은 인간의 창작 수단이 되기도 했지만 최근 인공지능(AI)의 발달으로 로봇은 창작의 주체가 되고 있기도 하다. 로봇과 예술은 많은 사회적인 그리고 기술적인 물음을 낳는다. 그림을 그리는 로봇을 만들기 위해서는 크게 두가지 요소가 필요한데, '창작 혹은 관찰', 그리고 '그리기'가 그것이다. 그려낼 작품을 창작하거나 대상을 관찰하여 비사실적인(non-photorealistic) 그림 작품으로 구현하는 연구는 컴퓨터그래픽스 분야에서 매우 오랜 기간 진행되어 왔다. 그리고 이러한 작품을 로봇을 이용해 물리적인 공간에 그려내기 위해서는 로봇 동작 계획과 제어와 같은 로봇 공학적 문제 해결을 필요로 한다. 본 논문은 다자유도 로봇을 이용해 넓은 비평면에 펜 아트를 그려내는 로보틱 드로잉 시스템을 소개한다. 이를 구현하기 위한 그림 경로 생성, 그림 맵핑, 그리고 로봇 동작 계획 기술들을 제안한다.

제안하는 시스템은 앞서 언급한 '창작 혹은 관찰'을 통한 그림 작품을 만들어내는 방법과 '그리기'를 위한 로봇 동작 계획과 제어를 포함한다. 이산적인 픽셀들로 이루어진 일반적인 디지털 아트와 달리 로봇의 연속적인 동작으로 사상되기 위한 그림 작품은 일련의 좌표값들로 치환될 수 있어야 한다. 수학 방정식을 기반으로 하는 점, 직선, 곡선, 다각형들로 이루어진 벡터 이미지는 이와 같은 특성에

부합한다. 컴퓨터그래픽스 분야에서 사람이 그린것과 같은 비사실적인 그림을 생성하기 위해 연구된 분야중 하나인 획기반 렌더링(stroke-based rendering) 방법은 이미지를 획 단위로 이루어진 페인팅으로 치환하여 로보틱 드로잉에 적용하기 적합하다. 본 논문은 더나아가 사람이 그림을 그리는 과정과 유사하게, 대상을 분석하고 깊이감을 고려하여 그림을 그리는 레이어화된 깊이 페인팅(layered depth painting) 방법을 소개한다. 그리고 반대로, 효과적이고 정확하게 섬세한 움직임을 동작하는 것에 특화된 로봇에 적절한 예술 형태인 TSP art를 소개한다.

본 논문은 로봇의 특장점에 주목한다. 로봇은 기계산된 동작을 정확하고 정밀하게 반복적으로 움직이는 것에 능하다. 자유도가 높은 로봇은 더 복잡한 동작이 가능한 것 뿐만 아니라, 바퀴가 달린 모바일 로봇은 훨씬 넓은 작업 범위를 가진다. 그리퍼가 부착된 두팔 로봇을 이용하면 두 팔이 상호 협력하고 물체를 파지하여 더 다양한 작업을 할 수 있다. 본 논문은 이러한 다자유도 로봇을 이용하여 넓고 굵곡진 대상 평면에 그림을 왜곡없이 그려내는 문제를 해결한다. 이를 위해 2차원 그림을 3차원으로 왜곡을 최소화하며 사상하는 등각사상 방법을 적용한다. 앞선 계산과정과 가상환경과 실제환경의 불일치에서 올 수 있는 기하학적인 에러를 보정 하며 강건한 '드로잉'을 수행하기 위해 임피던스 제어를 적용한다. 마지막으로, 보다 넓은 대상 표면에 펜아트를 생성하기 위해 모바일 플랫폼이 대상 캔버스를 모두 포함할 수 있는 최소한의 모바일 플랫폼 포즈들을 찾는 알고리즘을 제안한다. 또한, 적응형 3-핑거 그리퍼를 탑재한 두팔 로봇을 이용하여 다채로운 색상의 펜아트를 자동적으로 그려내기 위해 그에 맞는 그림 도구를 고정하는 펜툴 디자인과 툴-체인지 메커니즘을 제안한다.

본 논문은 다양한 표면에 시각적으로 아름답고, 사람이 정확하게 그려내기에 복잡한 펜 아트를 생성하는 로보틱 드로잉 시스템을 제안한다. 제안하는 시스템은 다관절 로봇을 이용하여 대규모의 비평면에 그림을 그려낸다. 본 논문을 통해 로봇이 그림을 그리기 위한 그림 경로 생성 기술, 그림 맵핑 기술, 로봇 동작 계획 기술, 그리고 두 팔 로봇을 위한 드로잉 툴-체인지 메커니즘을 제안한다. 로보틱 드로잉 시스템에 필요한 연구 기술들을 소개하고 다양한 연구 방향성을 제시한다. 나아가 본 시스템에 적용된 기술들은 비단 로봇 드로잉에 그치는 것이 아니라 밀링(milling), 샌딩(sanding)과 같은 대상 표면과의 연속적인 접촉을 필요로 하는 다양한 로보틱 어플리케이션에 적용될 수 있을 것으로 기대된다.

감사의 글

대학원에 입학한 지 6년이라는 시간이 흘러 그 결실이 미흡하게나마 이 논문에 맺히게 되었습니다. 이화여대에서의 10년, 청춘을 쏟았다고 농담처럼 말했지만 얻은 것이 더욱 많은 값진 학교 생활이었습니다. 결코 혼자서 해낸 것이 아닌, 많은 분들의 도움과 지지를 받아 학위를 마칠 수 있었기에, 감사한 분들께 마음을 전합니다.

먼저, 저에게 연구자의 길을 열어 주신 김영준 지도교수님께 감사의 말씀을 드립니다. 호기심에 이끌려 연구실 인턴십을 시작했던 학부생에게 언제나 아낌 없는 지원을 해주신 덕분에 연구에 대한 흥미와 열정을 잃지 않고 박사 학위까지 마무리할 수 있었습니다. 학생들에게 학문에 몰두할 수 있는 환경을 마련해주시고 언제나 가까이서 지도해주시는 교수님께 지도자의 자세를 배웠습니다. 마음처럼 연구가 잘 되지 않아 풀이 죽어 있을 때면, 조금 더 자신감을 가져도 좋다고 위로하고 격려하시던 말씀에 힘을 얻고 다시 연구에 매진할 수 있었습니다. 교수님의 관심과 격려를 거울 삼아 어디를 가더라도 항상 겸손한 자세로, 교수님의 가르침에 빗나가지 않는 연구자가 되겠습니다.

또한, 바쁘신 가운데 학위 논문 심사위원을 맡아주신 오유란 교수님, 류석창 교수님, 윤성의 교수님, 그리고 이주행 박사님께 감사드립니다. 제 연구에 대해 관심을 가져주시고, 아낌 없는 조언과 격려를 나누어 주신 덕분에 마지막까지 잘 정리할 수 있었습니다. 그 외에도 앞으로의 계획에 대해 진심을 다해 생각해주시고 격려해주심에 마음 속 깊이 감사드립니다.

사랑하는 우리 연구실, 컴퓨터그래픽스 식구들에게 감사한 마음을 전합니다. 저의 연구 시작에 큰 영향이 되었던 영은 언니, 여진 언니, 윤형 언니. 즐겁게 연구하던 언니들을 보며 대학원 생활을 꿈 꿀 수 있었습니다. 제게 큰 힘이 되어준 예솔 언니, 언니와 함께 연구하고 고민을 나눌 수 있어 행복했습니다. 현정언니, 정민이, 지수 언니, 지선 언니, 시연이, 민영 언니. 동고동락하며 지내던 그 때가 가장 즐겁게 배우고 가장 많이 성장했던 시기였습니다. 혜정언니, 의정이, 선민이.

제 연구실의 마지막을 또 즐거운 기억들로 쌓아주어서 정말 고맙습니다. 한경민 박사님, 유지연 선생님. 연구실의 어려운 일들을 도맡아 해결해 주시고, 남몰래 항상 힘써주심에 감사드립니다. 부족함이 많은 저를 따라 성실하게 연구에 임해주었던 은정이와 지윤이를 포함하여 연구실을 다녀간 많은 인턴 후배들도 제게 모두 소중한 인연이었습니다. 여러분들을 모두 마음을 다해 응원합니다.

사랑하는 엄마, 아빠, 우리 예은 언니에게 감사의 마음을 전합니다. 먼 타지에서 꿈을 위해 고군분투하는 세상에서 가장 멋있는 우리 언니. 어려서부터 내가 무엇을 하던 소중히 여겨주고 좋아해주던 언니 덕분에 항상 용기를 잃지 않고 앞으로 나아갈 수 있었던 것 같아. 나도 언니가 어디서 무엇을 하던 항상 언니의 1호 팬이 될게. 그리고 오랜 기간 묵묵히 곁을 지켜 주시고 무한히 지지해주신 부모님이 계신 덕에 큰 어려움 없이 무사히 학위 과정을 마칠 수 있었습니다. 항상 변함없이 주신 사랑 보답하며 또 남들에게 베풀며 살겠습니다.

국내·외 많은 분들의 도움으로 이 자리까지 올 수 있었습니다. 그 지지와 사랑이 헛되지 않게 더욱 정진하며 바르게 성장하는 연구자가 될 수 있도록 노력하겠습니다.