Continuous Coverage Motion Planning on 3D Freeform Surfaces

by

Sean M. McGovern

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Robotics Engineering

August 2023

APPROVED:

_____

Dr. Jing Xiao, Professor, RBE (Advisor)

_____

Dr. William Michalson, Professor, RBE

_____

Dr. Berk Calli, Assistant Professor, RBE

_____

Dr. Emmanuel Agu, Professor, CS

Abstract

Many industrial robotic applications (such as spray coating/painting, abrasive blasting, polishing, shotcrete, laser ablation, etc.) require a manipulator's end-effector to fully cover a 3D surface region in a continuously constrained motion. The manipulator must continuously satisfy surface task constraints imposed on the end-effector while maintaining joint constraints. Surface coverage in this context is focused on employing commonly used coverage patterns (such as raster, spiral, or dual-spiral) onto the surface for the manipulator to follow. The overall quality of production is typically evaluated based on the uniformity of tool coverage by the manipulator and the manner in which it was applied to the surface.

While substantial research has been conducted on achieving autonomous coverage on planar surfaces, the available methods for constrained coverage of 3D surfaces are limited to parametric or spline surfaces. Additionally, these methods do not adequately address coverage feasibility given both manipulator and task constraints. This indicates there is a lack of fundamental research to address the general problem: given a manipulator, a 3D surface, and task constraints, whether there exists a feasible continuous motion plan to cover the surface, and if so, how to produce a coverage path for uniform surface coverage which best satisfies task constraints. Furthermore, there is a pressing need for analysis tools that can empower a human worker, who is not a robotics expert, to compare material surface coverage results and identify critical factors (such as task parameters, surface spatial arrangements, spray models, etc.) that contribute to optimal production quality.

This work presents a comprehensive approach to systematically determine continuous coverage feasibility and generate singularity-free manipulator paths to follow uniform coverage patterns on 3D freeform surfaces. We propose a novel method for mapping 3D freeform surfaces to a seamless UV space to facilitate coverage feasibility checking, automatic surface coverage pattern placement, and analysis of material surface coverage. Moreover, we provide an interactive virtual environment to allow a domain expert worker, who is not a robotics expert, to simulate material surface coverage using our methods. Experimental results demonstrate the efficiency of our coverage feasibility checking algorithms and the versatility of our uv grid generation on 3D freeform surfaces for achieving uniform coverage patterns. Finally, we showcase the effectiveness of our approach in achieving uniform material coverage on 3D freeform surfaces compared to other methods through results obtained from a virtual spray painting case study.

Acknowledgements

I am deeply grateful to my Ph.D. advisor, Professor Jing Xiao, for granting me the invaluable opportunity to pursue my Ph.D. in Robotics Engineering at the Adaptive and Intelligent Robotics (AIR) Lab at WPI. I would not have been able to complete my Ph.D. without her wisdom and guidance in both research and writing. Our shared passion for robotics and its potential to enhance everyone's quality of life motivated our work through the years. Additionally, I express my gratitude for the time and assistance provided by my Ph.D. Committee members: Professors Michalson, Agu, and Calli. Their support and feedback have been crucial in shaping my research.

I extend my heartfelt appreciation to the faculty and staff of the WPI Robotics Department for paving the way for individuals with ambitious dreams to pluck such a high-hanging fruit. Their guidance has been instrumental in my journey.

I would also like to acknowledge the remarkable environment cultivated by the students in the WPI Robotics Department. Reuniting with student peers during my visits reminded me of my passion for robotic applications and without them I may not have returned to complete my Ph.D at WPI. I am immensely appreciative of the acceptance I received from them even though I came from a different culture. I admire you all. Also, with the assistance of fellow Ph.D. students, we re-established the WPI IEEE RAS Student Chapter, and I am honored to have served as its first President. I hope the chapter continues to provide a platform for students to develop their leadership skills and establish connections with other Ph.D. students.

To my family, I am thankful for their solace through my struggles on this path and for their support from afar. Our conversations provided the encouragement I needed to persevere and pursue my goals despite the obstacles. This experience has enlightened me to how incredibly important family is for reminding us of the important things in life.

Contents

Contents **v**

List of Figures **ix**

List of Tables **xiv**

1 Introduction **1**

1.1 Industrial Applications for 3D Surface Coverage 1

1.1.1 Constrained Surface Coverage 2

1.1.2 Coverage Path Planning 4

1.1.3 Human-Robot Interaction in Industry 6

2 Related Literature **9**

2.1 Coverage Path Planning Overview 9

2.1.1 Optimal Route: Traveling Salesman Problem 10

2.1.2 View Planning: Watchman Route Problem 12

2.1.3 Cellular Decomposition for Uniform Coverage 14

2.2 Constrained Motion Planning 16

2.2.1 Manipulator Motion Planning 17

2.2.2 Rapidly-Exploring Random Trees 19

2.3 UV Mapping for 3D Surfaces 19
2.3.1 Traditional Texture Mapping 20
2.3.2 Seamless Texture Mapping 22
2.4 Human-Robot Interaction 24
2.4.1 Enhancing Production Through HRI 24
2.4.2 HRI Safety Standards 27

3 Problem Formulation and Objectives 29
3.1 Problem Description 29
3.2 Research Objectives 32
3.2.1 UV Mapping on 3D Surfaces 33
3.2.2 Efficient Coverage Feasibility Checking 33
3.2.3 Surface Coverage Pattern and Manipulator Motion Planning . . . 34
3.2.4 Virtual Interactive Environment for Industrial Surface Coverage . 34
3.3 Overview and Contributions of Remaining Chapters 35

4 UV Grid Generation on 3D Surfaces 37
4.1 Generating uv Grid for 3D Parametric Surfaces 38
4.2 Generating uv Grid for 3D Freeform Surfaces 39
4.2.1 Freeform Surface Types 39
4.2.2 Interactive Determination of Curve C 41
4.2.3 Automatic Generation of uv Space 42
4.2.4 Automatic Discretization of uv Space 50
4.3 Experiments and Analyses 51
4.3.1 Results of uv Grid Generation 53
4.3.2 Discussion 53

4.4 Chapter Summary 55

5 Efficient Coverage Feasibility Checking on 3D Surfaces 57

5.1 Deriving Joint Space Task Constraints 58

5.1.1 Assumptions, Notations, and Constraint formulations 58

5.1.2 Joint Space Task Constraints on 3D Parametric Surfaces 61

5.1.3 Joint-Space Task Constraints on 3D Freeform Surfaces 64

5.2 From Feasible J-cell Transitions to uv-Cell Neighboring Continuity . . . 67

5.2.1 Mapping from Joint Space to uv Space 67

5.3 Experiments and Analyses 72

5.3.1 Results with different sets of joint limits 73

5.3.2 Results with different task parameter values 75

5.3.3 Performance data 78

5.3.4 Discussion 81

5.4 Chapter Summary 82

6 Continuous Coverage Motion Planning on 3D Surfaces 85

6.1 Placing Coverage Pattern on the uv Grid and Mapping to Cartesian Space 86

6.1.1 Generating the uv coverage pattern 87

6.1.2 Choosing a Coverage Pattern 90

6.2 Coverage Path Singularity Detection and Avoidance 92

6.2.1 Generating initial end-effector path 93

6.2.2 Singularity Detection 95

6.2.3 Singularity Avoidance 96

6.3 Experiments and Analyses 98

6.3.1 Surface Coverage Pattern Results 98

6.3.2 Motion Planning Algorithm Results 101
6.3.3 Discussion 103
6.4 Chapter Summary 104

7 Virtual Case Study for Surface Coverage Applications 107
7.1 Virtual Environment for 3D Surface Coverage Applications 108
7.1.1 Interactive GUI for Surface Coverage Planning 109
7.1.2 Manipulator Spray Model 112
7.1.3 3D Surface Analysis Points 115
7.2 Experiments and Analyses 116
7.2.1 Spray Simulation Comparison Results 118
7.2.2 Discussion 121
7.3 Chapter Summary 124

8 Conclusions 127
8.1 Review of Contributions 127
8.2 Future Work 128
8.2.1 Hierarchical Surface Segmentation 129
8.2.2 Virtual Simulation Environment for 3D Surface Application Analysis in Industrial Applications 131
8.3 Broader Impact 132
8.4 Concluding Remarks 132

Bibliography 135

List of Figures

1.1 Various examples of industrial manipulator coverage applications. Image credits: (a) Army Research Lab, https://www.arl.army.mil/ (b) Andulkar *et. al.*, 2015 [1] (c) Liu *et.al.*, 2021 [2]. 3

1.2 Industrial dual spiral coverage pattern on spherical surface. Image credit: Chen *et al.*, 2017 [3]. 5

1.3 Illustration of complementary interaction between domain expert worker (not a robotics expert) and robotic manipulator for surface coverage interactions. . 6

2.1 Artistic depiction of robot traveling salesman holding a map. Image credit: Hackaday, https://hackaday.com/. 10

2.2 Watchman route example with mobile manipulator. Image credit: Wang *et al.*, 2007 [4]. 12

2.3 Commonly used 2D coverage patterns. 14

2.4 Illustrations of grid based decomposition of explorable 2D regions. Image credit: Galceran *et al.*, 2007 [5]. 15

2.5 Franka Emika Panda's end-effector constrained on object surface in simulation environment. 17

2.6 UV mapping segments of Stanford Bunny. Stanford Bunny in foreground with respective UV texture segments in background. Image credit: Yuksel *et al.*, 2019 [6]. 21

2.7 Illustration of visible seams in 3D mesh texture. (a) Example of visible color transitions between two mesh segments in traditional UV texture mapping. (b) Reduced visible contrast between color transitions between two mesh segments. Image credit: Yuksel *et al.*, 2019 [6]. 22

2.8 Example of avoiding vertex seam color transitions in cylindrical and toroidal UV mappings. Image credit: Tarini *et al.*, 2012 [7]. 23

2.9 Illustration of industrial evolution to Industry 5.0. Image credit: Maddikunta *et al.*, 2022 [8]. 25

2.10 Illustration of humans and robot working together using RGB-D devices.Image credit: Gomez *et al.*, 2017 [9]. 26

4.1 Illustrations of 3D freeform surface types (embedded polygon meshes). Curve C_0 for each definition is highlighted in yellow. (a) Freeform surface of translation along the z axis. (b) Freeform surface of rotation about the z axis. (c) Freeform surface of translation (on the xy plane) and rotation (about axis parallel to the z axis). 40

4.2 Illustration of simulation interface for human worker to interactively determine curve C to facilitate automatic uv space generation. 43

4.3 Comparing two different ways of discretization along a curve C. 44

4.4 Illustration of uv parameter calculations along SoT (blue arrow) with axis of translation (green arrow). 45

4.5 Illustration of uv parameter calculations along SoR (blue arrow). 47

4.6 Illustration of uv parameter calculations along SoTR (blue arrow). 48

4.7 Polygon mesh of four different freeform surfaces. 51

4.8 Initial curve C highlighted in yellow for each surface. 52

4.9 Generated uv grid (blue) with applied raster coverage patterns $\mathbf{UV}$ (red) on each freeform surface. 54

5.1 Illustration to compare the effects of (a) position and orientation constraints defined independently, which causes large orientation shift around surface edges, and (b) position constraint defined along the approach vector $\mathbf{a}$, i.e., related to the orientation constraint, which allows for smooth orientation change around surface edges. 66

5.2 Illustration showing the conversion from J-manifold (left), to the E-manifold (center), and then to the uv grid (right). The corresponding regions in each manifold are shown in the same blue/green colors. 68

5.3 Graph G on different surface patches using task parameters value set 1 (in Table 5.2) and different PUMA joint limit sets (in Table 5.1) 74

5.4 Graph G on different surface patches using joint limit set 1 and varied values of task constraint parameters. 77

5.5 Joint-space motion continuity search between two neighboring uv-cells for the spherical surface (in degrees). 80

5.6 Graph G for the spherical surface using task parameter value set 1 and different joint limits, where red dots indicates unreachable nodes. 81

6.1 2D coverage patterns over uv grid. 90

6.2 Different coverage patterns on concave surface in uv space. 91

6.3 Illustration showing examples of suitable and non-suitable raster scan start direction for uv grid surface. 92

6.4 Joint-space motion continuity search between two neighboring E-cells for the spherical surface (in degrees). 94

6.5 (left) Illustration of resulting end-effector path (red) with singularity in desired path.(right) Illustration of end-effector path (red) after orientation smoothing to avoid singularity (right). 97

6.6 Generated uv grid (blue) with applied raster coverage patterns $\mathbf{UV}$ (red) on each freeform surface. 99

6.7 Raster scan coverage patterns on different surfaces. 100

6.8 Snapshots of the Panda manipulator moving along a coverage pattern (red) on the Curvy (SoT) surface. 103

7.1 Virtual environment with Franka Emika Panda and uv-cells (white) on Cone surface (SoR). GUI on right gives human worker interface for surface positioning and for initiating automatic uv grid generation, coverage feasibility checking, coverage path generation, and spray coverage simulation. 110

7.2 Spray cone volume V (orange) projecting from Franka Emika Panda end-effector. 112

7.3 Illustration showing examples of non-circular surface spray intersection with end-effector orientation and surface curvature. (a) Example of non-circular surface spray intersection due to end-effector orientation. (b) Example of non-circular surface spray intersection due to surface curvature. 114

7.4 Analysis points (white) evenly spaced using uv grid on Cone surface (SOR) in virtual environment. 115

7.5 Illustration of Corner Surface (SoTR) with evenly distributed analysis points (white). 118

7.6 Examples of coverage patterns on surfaces using Cartesian coordinates as reference for raster scan line spacing. Image credits: (a) Andulkar *et al.*, 2015 [1] (b) Chen *et al.*, 2019 [10]. 119

7.7 Image of finished surface spray coverage simulation showing Cartesian space coverage pattern (red) and analysis points (white and green) on Corner surface (SoTR). 121

7.8 Image of finished surface spray coverage simulation showing uv space coverage pattern (red) and analysis points (white and green) on Corner surface (SoTR). 122

7.9 Histograms results of spray simulations using different methods for surface coverage pattern placement. 123

8.1 Example of a real world surface approximately segmented using the primitive SoR. Each segment is depicted in orange, accompanied by its corresponding initial curve C in yellow. Notably, each of these segments has a rotation axis that is parallel to the green axes shown for two SoR patches. Image credit: https://www.alamy.com/. 130

List of Tables

4.1 uv parameters for parametrically defined surfaces. 38

5.1 Joint Limits Sets for PUMA. 73

5.2 Task Parameters and Values. 76

6.1 Number of singularities fixed by path planner. 101

6.2 Coverage path/trajectory results. 102

Chapter 1

Introduction

We begin with a review of industrial applications of coverage motion planning and explore the existing methodologies used in this context in Section 1.1. We discuss the current state of constrained surface coverage focusing on its application within industry. Next, we examine the commonly used methods for coverage path planning in industry while acknowledging their limitations. Then, we review the current use of human-robot interaction in relevant industrial applications. Finally, in Section 1.2, we present an overview of this dissertation, outlining each chapter along with its individual contributions.

1.1 Industrial Applications for 3D Surface Coverage

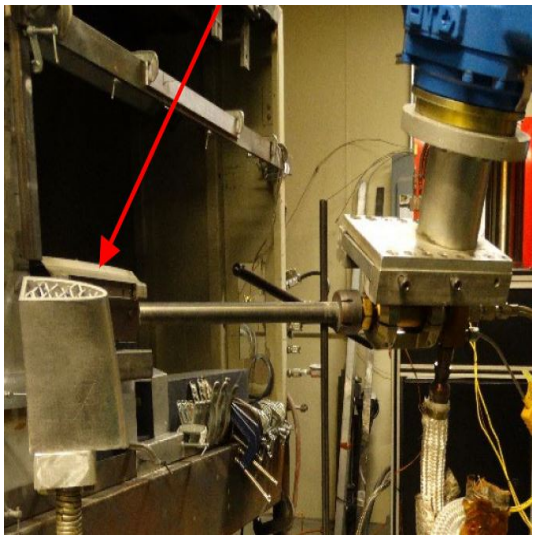
Robots are widely used in various industries such as automotive, furniture, aerospace, construction, agricultural, and household appliances for surface coverage applications. Industrial manipulator coverage applications (such as spray coating/painting, abrasive

blasting, polishing, shotcrete, laser ablation, etc.) require a manipulator's end-effector to traverse the entire surface once while satisfying task criteria in terms of application thickness, cycle time, and material waste [1–3, 10–15]. The quality of production is usually determined by manipulator tool coverage uniformity and how it was applied on the surface (i.e. angle of approach, surface offset, etc.).

Achieving continuous and even coverage manually on a 3D freeform surface poses significant challenges. Manual generation of a tool path on a 3D freeform surface is complex, time consuming, ad hoc, and difficult to optimize. Moreover, it requires the human operator to have substantial knowledge of robotic manipulation. With the increasing need for rapid and efficient production and repairs in related industries, it has become essential to enable automated and optimal robotic coverage on 3D freeform surfaces [16, 17].

1.1.1 Constrained Surface Coverage

Here the problem is concerned with enabling the motion of a robot manipulator that is constrained to a 3D surface for coverage purposes (see Fig. 1.1). In many cases, the manipulator's end-effector is equipped with a tool tip to either apply or remove some material from the surface. In other cases, such as surface inspection, the end-effector is equipped with sensors that must view the surface entirely (referred to as view planning). The main difference lies in the fact that, in the former case, achieving uniform coverage is crucial for the even distribution of the applied substance, and the constraints imposed



(a) Spray coating on aircraft airfoil skin.

(b) Spray painting on automotive panel.

(c) Shotcrete coverage operation in tunnel construction.

Figure 1.1: Various examples of industrial manipulator coverage applications. Image credits: (a) Army Research Lab, https://www.arl.army.mil/ (b) Andulkar *et. al.*, 2015 [1] (c) Liu *et.al.*, 2021 [2].

on the manipulator's movement need to be strictly followed throughout the process. On the other hand, in the latter case, the sensor may pass over the same area multiple times

without negatively impacting the quality of the inspection. This work primarily focuses on addressing the challenges related to achieving continuous and uniform coverage for applications in the former case, although, it may be used for the latter.

However, it is important to note that these methods typically focus on specific industrial applications and do not take into account surface coverage. Additionally, they assume the existence of a feasible path beforehand. It can be advantageous to recognize that a feasible constrained coverage path over a surface may not exist before path planning given certain manipulator and task constraints.

1.1.2 Coverage Path Planning

The objective of coverage path planning (CPP) is to ensure that a robot's sensors or end-effector fully cover an entire surface, either in one continuous motion or through a series of continuous motions. CPP can be implemented in 2D or 3D environments and can be performed online (while the robot is moving) or offline (with prior knowledge of the environment). The specific scenario often dictates the approach taken for CPP. However, most existing CPP approaches do not account for robot constraints.

Previous methods developed for industrial manipulator coverage applications are limited to certain types of 3D surfaces, such as parametric or spline surfaces. These methods can result in coverage paths with uneven spacing, which is not ideal for tasks that require uniform coverage, such as achieving uniform thickness [18]. The commonly used coverage patterns for industrial applications include raster scan or dual spiral coverage

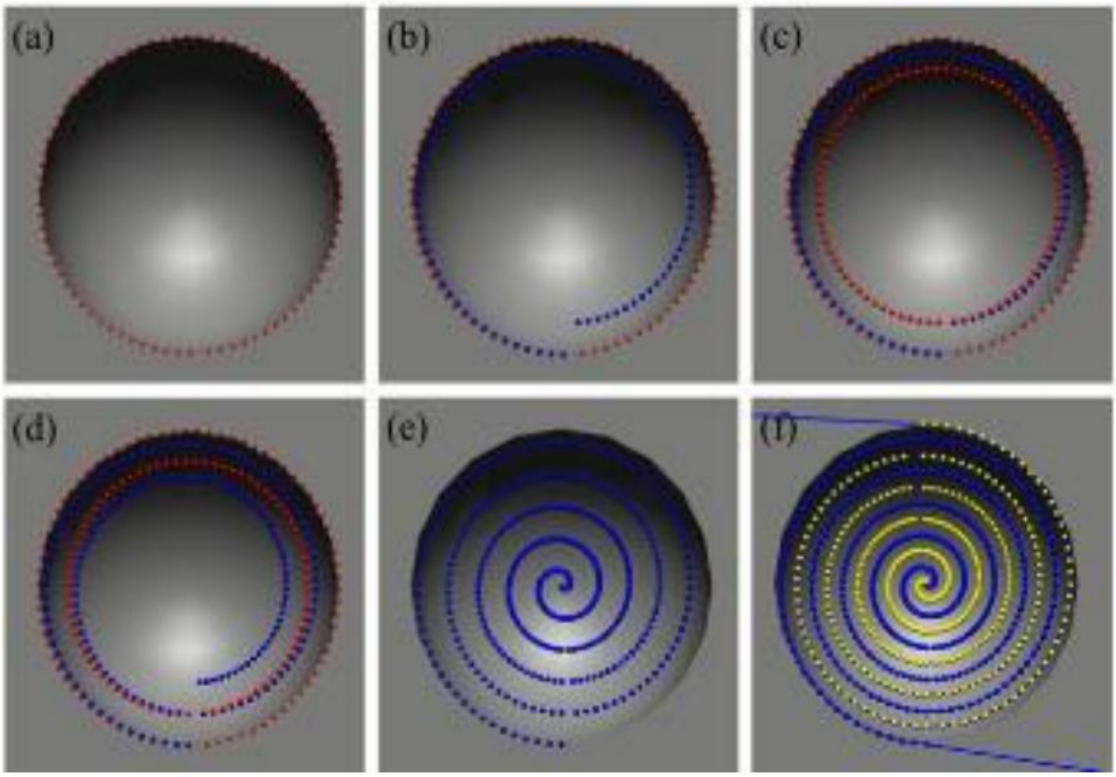


Figure 1.2: Industrial dual spiral coverage pattern on spherical surface. Image credit: Chen *et al.*, 2017 [3].

patterns (see Fig. 1.2) [3, 19].

Most general CPP methods are designed for 2D planar surfaces, often in online settings, and for mobile robots [5, 18, 20–29]. There are also approaches that explore unknown environments [30–33]. Some CPP methods focus on sensor coverage for aerial or submersible robots [34–36], also known as view planning. However, these methods typically result in discontinuous and nonuniform coverage paths. In the field of agriculture, there are

methods proposed to generate uniform coverage paths on 3D landscapes [37–46]; however, these applications do not take manipulator constraints into consideration. Similarly, there are coverage path planning techniques for manipulators in medical applications [47–50]. Nevertheless, there is a scarcity of previous work addressing the placement of a uniform coverage path on general freeform 3D surfaces while considering both task requirements and manipulator constraints.

1.1.3 Human-Robot Interaction in Industry

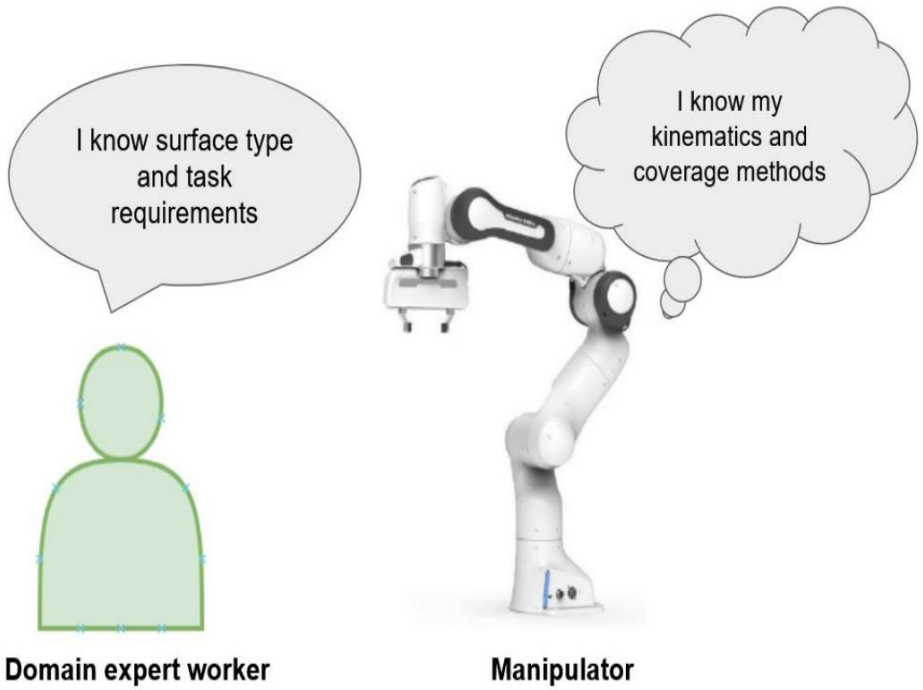


Figure 1.3: Illustration of complementary interaction between domain expert worker (not a robotics expert) and robotic manipulator for surface coverage interactions.

As collaborative robots become more prevalent in industrial applications, it is critical

that robots understand human input and provide feedback [51]. Conventional methods for human-robot interaction (HRI) play a vital role in numerous industries for determining how humans and robots interact in work environments to improve overall production efficiency and quality. It is a rapidly growing field within robotics that aims to create work environments where humans and robots can collaborate effectively and safely [52]. HRI in industrial applications can be classified into three categories:

- Human-Robot Coexistence: This refers to the ability of humans and robots to share a workspace without performing a common task or without mutual coordination.
- Human-Robot Cooperation: In this category, humans and robots share a workspace with limited ability to cooperate and complete a common task together.
- Human-Robot Collaboration: This involves humans and robots working together to perform complex tasks, with direct interaction through physical or non-physical communication. It often requires a higher level of robot intelligence to anticipate human needs or communicate effectively for human understanding.

Industrial robots are increasingly being used to replace or assist humans in performing repetitive, hazardous, or tedious tasks, thereby enhancing the manufacturing process. As robotic technology advances and machines become more intelligent and aware, robots are now capable of working cooperatively and collaborating with human workers to accomplish complex tasks [53]. Despite advancements, practical challenges still exist for which algorithmic solutions have not been developed.

Research in HRI encompasses various topics, including gesture recognition, intrinsically safe collaborative robots (cobots), collision avoidance, cyber-physical safety, virtual and augmented reality, human prediction, and cognitive human-robot interaction. Many applications require a human interface with the robotic system [53–55] (see Figure 1.3). The interface or communication can occur physically through touch or non-physical means such as gestures or voice commands. Trajectory and tooling methods with human-robot interaction have been developed for computer-aided tool path planning, which typically involves automatic coverage path planning [9, 56–58].

Chapter 2

Related Literature

This chapter serves as a comprehensive review of relevant research. We start by discussing the concept of coverage path planning, where we examine both classical and cutting-edge coverage algorithms, as well as explore view planning techniques. Next, we delve into the topic of constrained motion planning and explore related research in this area. Then we discuss traditional and seamless UV mapping methods as applied to texture mapping for computer graphics. Lastly, we close with a review of research pertaining to human-robot interaction for industrial applications.

2.1 Coverage Path Planning Overview

Coverage path planning in relation to this work is about finding a path for either a manipulator's end-effector or mobile robot's sensors to traverse a surface region while achieving certain desired level of coverage [18, 59, 60]. It's important to note that the main objective

is to generate a sequence of points for coverage, regardless of any surface constraints or the kinematics of the robotic system. However, constraints or cost functions are typically taken into account, and the coverage path is optimized accordingly. The exploration or inspection of unknown surfaces is often a fundamental objective in this context, and there may be an additional pursuit of achieving uniform coverage for certain applications.

2.1.1 Optimal Route: Traveling Salesman Problem

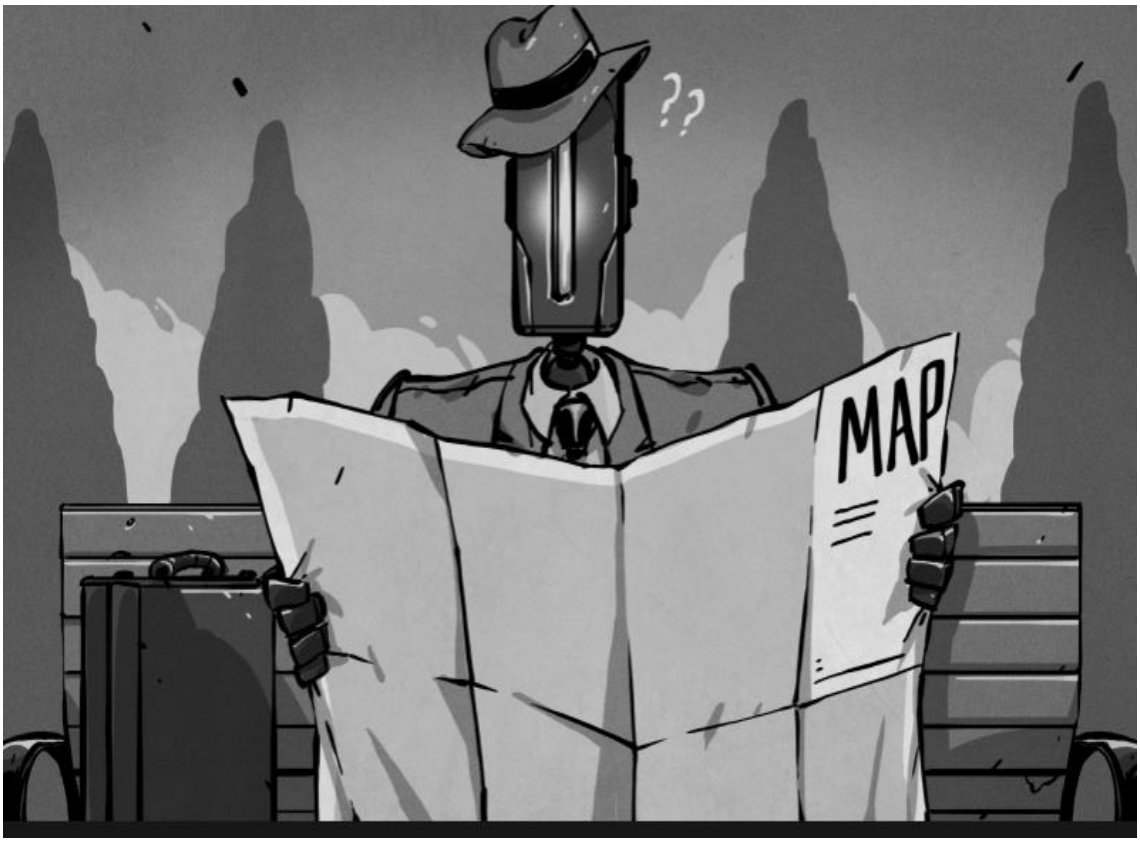


Figure 2.1: Artistic depiction of robot traveling salesman holding a map. Image credit: Hackaday, https://hackaday.com/.

The traveling salesman problem (TSP) is an algorithmic problem tasked to find the

optimal route, represented by a sequence of points, between a given set of locations that must be visited [61]. The problem is inspired by the scenario of a traveling salesman who must visit each city at least once (and sometimes only once) while minimizing travel costs and distance (see Fig. 2.1). TSP is considered an NP-hard problem which means there may be at least some polynomial time approximate to solve it [62, 63]. Other classical variations of the problem include the generalized traveling salesman problem [64] and the covering salesman problem [65]. It is important to note that the solution to TSP solely involves determining a route of points to visit and does not always take into account the area covered.

Various heuristic methods have been employed to address TSP problems efficiently within a short timeframe. These methods include colony optimization [66], simulated annealing [67], and others [68, 69]. Pertaining to robotic manipulator tasks, there have been efforts to solve TSP with kinematic and dynamic constraints, focusing on finding the minimum time route [70, 71]. In addition to these approaches, the concept of multi-goal planning is relevant where the objective is to plan a path of minimum length that visits each configuration in a given list at least once [72–74].

Often described as the covering salesman problem, variants of TSP have been used for the main goal of surface coverage [75, 76]. Boustrophedon coverage path planning has been generalized as a TSP in [77]. Colony optimization has been used again for covering salesman problems [78] and methods have been researched for unmanned aircraft systems [79]. TSP has also been applied to grid maps, leveraging reinforcement learning

methods to achieve optimal coverage [80]. However, these methods are generalized for 2D environments and can lead to coverage path overlaps.

2.1.2 View Planning: Watchman Route Problem

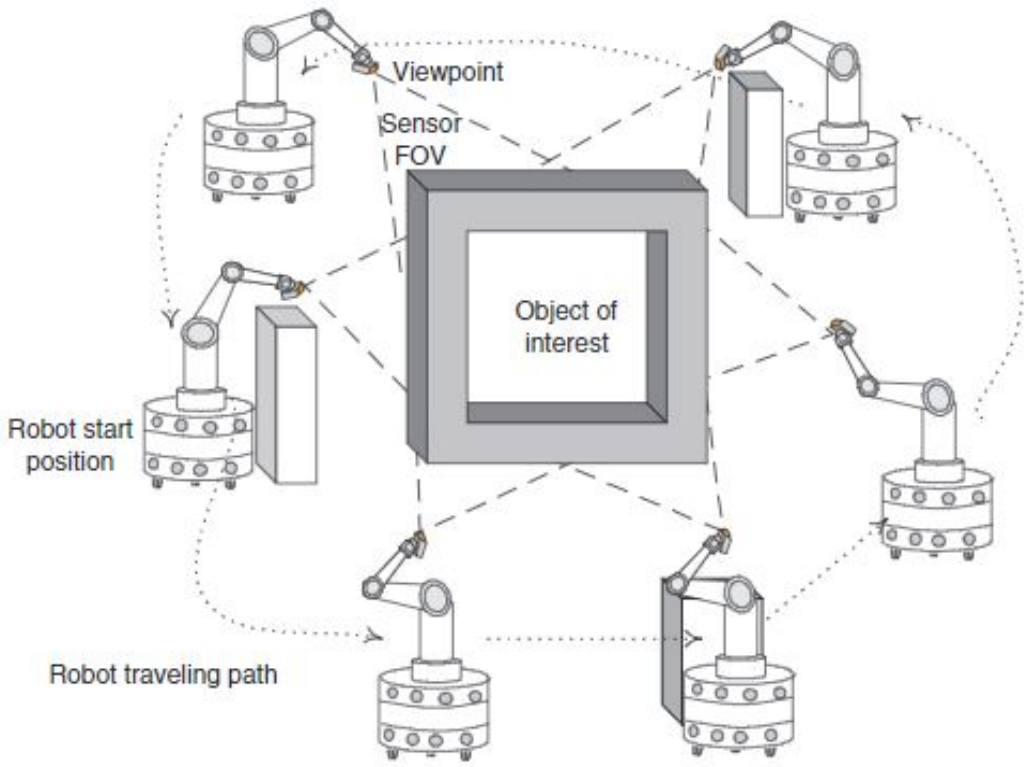


Figure 2.2: Watchman route example with mobile manipulator. Image credit: Wang *et al.*, 2007 [4].

The Watchman Route Problem is an optimization problem in computational geometry that involves determining the shortest route for a watchman to guard an entire area from their visual perspective. The problem is defined by the layout of the area, typically represented as a simple polygon. The objective is to find a path that covers all points

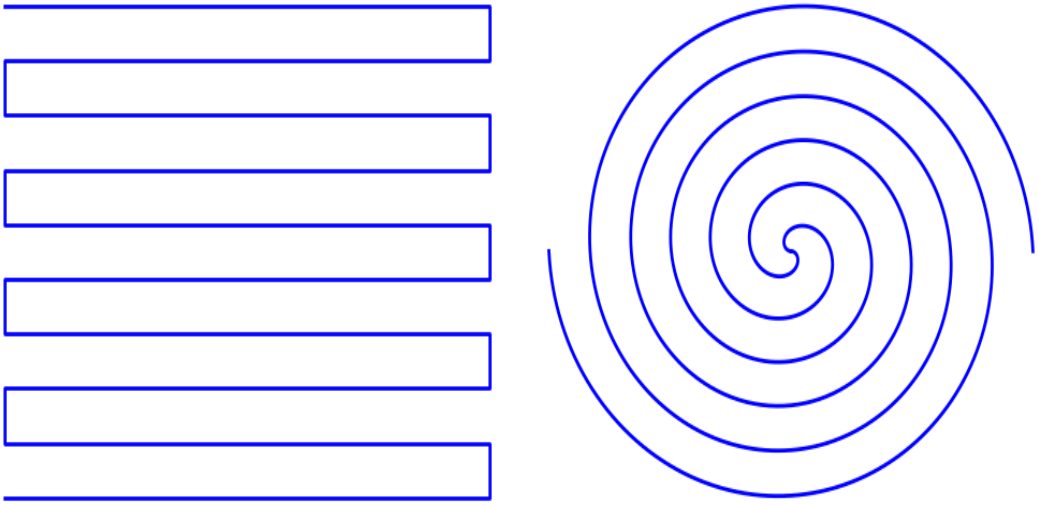
within the polygon and on its boundary, ensuring that each point is visible from at least one point on the path [39, 81–88]. For cases where a starting point is known, the problem can be solved in polynomial time. However, the problem becomes NP-hard when dealing with more complex polygons or when extending it to 3D [89]. In situations involving robots with a limited-range field of view, approximate solutions have been developed that can be found in polynomial time [90]. It's important to note that the Watchman Route Problem does not require specific points within the polygon to be included in the path, and achieving uniform coverage is not the primary objective.

When a known model of the environment to be observed is available, it is possible to generate a set of high-value viewpoints based on the optical parameters of the sensor [91]. By creating a list of viewpoints to be visited along the route and assigning penalizing weights to additional viewpoints, a method can utilize TSP solutions for the traveling view planning problem [4] (see Fig. 2.2). In [69], the TSP component of the algorithm is solved by computing the minimum spanning tree. Other notable methods include those described in [92–95]. An approach that combines model-based view planning, multi-goal planning, and view ordering is the watchman route algorithm developed by Danner and Kavraki [96].

Several methods have been developed to address sensory coverage for mobile robots operating in 2D workspaces. Faigl et al. proposed a method that utilizes the environment boundary as an initial offset for the path and then offsets the robot's maximum sensing range [97]. This approach has been found to generate shorter paths compared to geometric structure partitioning methods [98]. Additionally, teams of robots have been employed for

collaborative view planning problems, as demonstrated in the work by Fazli et al. [99].

2.1.3 Cellular Decomposition for Uniform Coverage

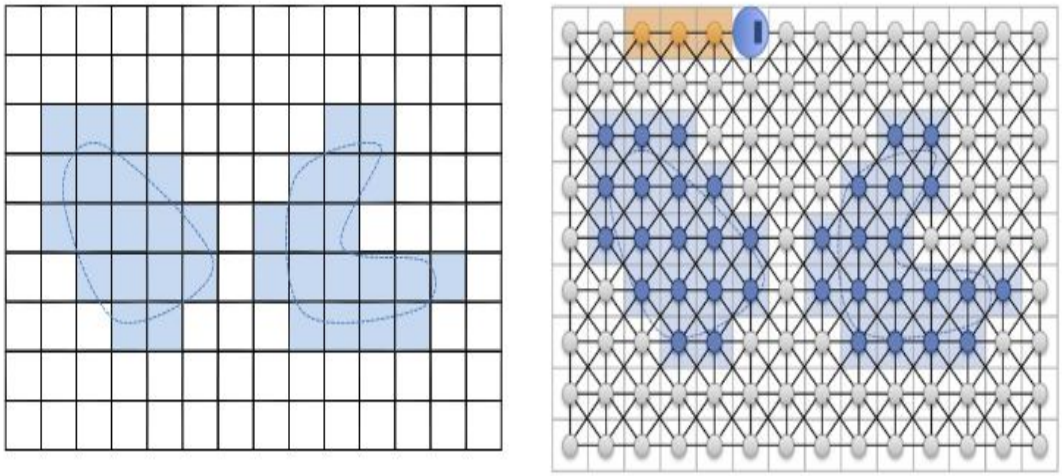


(a) Raster coverage pattern. (b) Dual spiral coverage pattern.

Figure 2.3: Commonly used 2D coverage patterns.

Dividing a region into evenly sized cells is a valuable approach for implementing uniform coverage path planning. Each cell is traversed once, and the sizes of the cells are determined based on specific application parameters. Grid-based coverage methods are commonly employed, involving the decomposition of the target area into uniform cells and subsequently covering each cell using established patterns like raster scans and dual spiral patterns (refer to Fig. 2.3) [100–102]. While a coverage algorithm utilizing a grid of triangle cells was proposed in Oh et al. [103], square-shaped cells, typically representing the size of the robot, are predominantly used. In Fig. 2.4(a), the region is decomposed into

square-shaped cells, with shaded cells indicating the presence of obstacles.



(a) Decomposition of 2D region with shaded cells representing obstacles. (b) Graph representation of grid map on 2D region.

Figure 2.4: Illustrations of grid based decomposition of explorable 2D regions. Image credit: Galceran *et al.*, 2007 [5].

Cellular decomposition of 2D space has been successfully applied in real-time scenarios for mobile robots, as demonstrated in previous studies [32, 104, 105]. One approach to achieve coverage involves utilizing a spanning tree method, which employs a grid with square cells as a graph representation of the environment [106] (see Fig. 2.4(b)). This approach simplifies the environment by transforming it into a graph structure comprising nodes and edges. Leveraging graph theory, various algorithms can efficiently calculate the shortest paths between nodes. Cellular decomposition of 2D regions has also been extended to multi-robot coverage planning, involving parallel paths swept by robots in formation [107, 108].

For 3D surfaces, cellular decomposition techniques have been applied by utilizing planar 2D slices for 3D inspection, as shown in [109]. An alternative approach involves the adoption of hierarchical generalized Voronoi graphs to navigate unknown environments [100]. However, in contrast to the extensive research on cellular decomposition in 2D environments, there is a lack of studies focused on decomposing 3D surfaces to achieve uniform coverage.

2.2 Constrained Motion Planning

Constrained motion planning involves generating the joint trajectories for mobile robots or manipulators while adhering to various task or kinematic constraints imposed by the environment or internal system requirements. These constraints can be directly imposed on the robotic joints (through current or mechanical limits) or on the robotic end-effector (through position and orientation constraints) [110–118]. These constraint groups are typically classified as robotic constraints and task constraints, respectively, and the region of the workspace that satisfies these constraints is referred to as a constraint manifold.

We first provide an overview of manipulator motion planning, focusing on the consideration of task constraints and the use of kinematic solutions to satisfy these constraints. Then we delve into rapidly-exploring random trees (RRT) and their application in discovering constrained motions between two specific kinematic configurations.

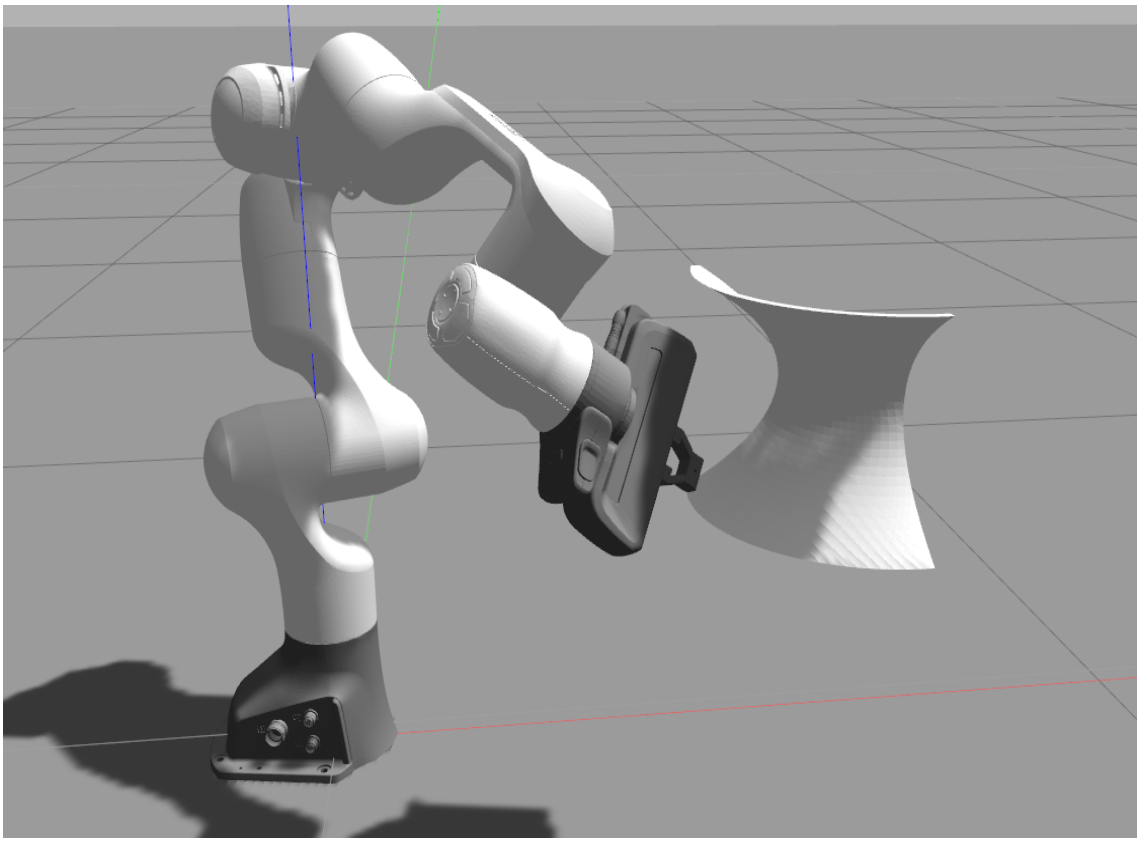


Figure 2.5: Franka Emika Panda's end-effector constrained on object surface in simulation environment.

2.2.1 Manipulator Motion Planning

In the context of robotic manipulators, certain constraints are typically placed on the position, velocity, and acceleration of the manipulator's joints. Additionally, task-specific constraints are imposed on the end-effector's position, velocity, acceleration, orientation, angular velocity, and angular acceleration (refer to Fig. 2.5). It's worth noting that the task constraints often change as the end-effector follows its trajectory due to interactions with the environment and obstacles. Inverse kinematics and Jacobian matrices are commonly

used to calculate the end-effector's position and velocity based on a given joint state and velocity, ensuring that all constraints are satisfied.

Closed-form solutions for manipulator inverse kinematics are superior due to their computational efficiency, as their time complexity is significantly lower compared to iterative numerical solutions, which have at least quadratic time complexity $O(n^2)$. However, closed-form solutions are not always available for manipulators that have six or more degrees of freedom [119–122]. To address this limitation, Shimizu et al. [123] proposed an analytical inverse kinematic solution specifically for 7-DOF redundant manipulators. In most cases with more than six degrees of freedom, algorithms that avoid inverse kinematics when possible for trajectory planning may be better for efficient computation.

Extensive research has been conducted on motion planning for constrained manipulators, as evidenced by several studies [124–128]. The objective of this research is to find a viable path that connects two configurations while ensuring that the end-effector remains constrained (such as when operating on a surface). An end-effector manifold is often used to describe the 3D space where the manipulator can move while adhering to the imposed constraints.

Motion planning methods that do not take constraints into account typically focus on finding collision-free paths between an initial and final robot state configuration [125, 129–132]. These paths provide a geometric representation of the robot's motion without specifying specific velocity or acceleration requirements for trajectory planning. By calculating the robot's workspace and identifying collision-free robot state configurations,

potential paths can be generated for use in various path planning methods. Subsequently, trajectory planning is conducted to determine the desired smooth joint velocity for the robot's motion along the selected path.

2.2.2 Rapidly-Exploring Random Trees

One commonly used algorithm for manipulator motion planning is rapidly-exploring random trees (RRT), which constructs a graph and identifies a path for the end-effector to follow while adhering to the specified constraints [133–137]. RRT is an algorithm specifically developed to efficiently explore and search high-dimensional spaces. It achieves this by incrementally constructing a space-filling tree from randomly sampled points within the search space. RRT algorithm exhibits a bias towards expanding and exploring large unsearched areas of the problem, ensuring comprehensive coverage of the space. There exist various motion planning algorithms designed specifically for redundant manipulators [119–122]. Feasible regions for constrained path planning in RRT are also represented using manifolds [138]. Neural networks are also being used to generate manipulator paths [139].

2.3 UV Mapping for 3D Surfaces

UV mapping for 3D surfaces is a significant area of interest within the field of computer graphics [140–142]. It is a fundamental process in 3D modeling that involves projecting the surface of a 2D image onto a 3D model for texture mapping purposes. In essence, a 3D

surface is a 2D manifold, and thus can be described using 2 position parameters, u and v. The 3D coordinates of a point on the surface can be expressed as the functions of these two parameters u and v.

In this section, we first present the traditional use of texture mapping, which aims to efficiently store texture signals in computer graphics [6]. Then, we discuss the related research on seamless texture mapping, which focuses on addressing issues related to contrasting variations across seams in traditional UV textures. The concept of seamless representations of UV meshes is directly relevant to our research as it facilitates the placement of uniform 2D coverage patterns on 3D surfaces.

2.3.1 Traditional Texture Mapping

The approach of texture mapping has become a cornerstone in computer graphics, with graphics APIs, GPU hardware, 3D model production pipelines, game engines, standard file formats for 3D models, and modeling software all designed around this concept. Texture mapping involves the mapping of various shading parameters, such as diffuse colors, normals, and displacement, onto a 3D polygon mesh from a 2D planar region represented by UV coordinates. These UV coordinates are assigned to the vertices of the mesh, indicating their respective locations in the image space. To create the texture map, the 3D mesh is manually separated into segments and unwrapped onto a 2D planar surface, effectively planarizing the 3D mesh (refer to Fig. 2.6).

However, the traditional method of texture mapping encounters several inherent chal-

Figure 2.6: UV mapping segments of Stanford Bunny. Stanford Bunny in foreground with respective UV texture segments in background. Image credit: Yuksel *et al.*, 2019 [6].

lenges. One of the key issues is the creation of seams in the mesh during the unwrapping process. Seams arise when the 3D mesh is divided into sections for unwrapping, resulting in edges on the 2D mesh segments that connect faces mapped to different locations in the UV space. This can lead to vertex duplication in the UV space and cause discontinuities along the seams. Consequently, visible color transitions may occur between the different mesh segments (refer to Fig. 2.7) [143, 144].

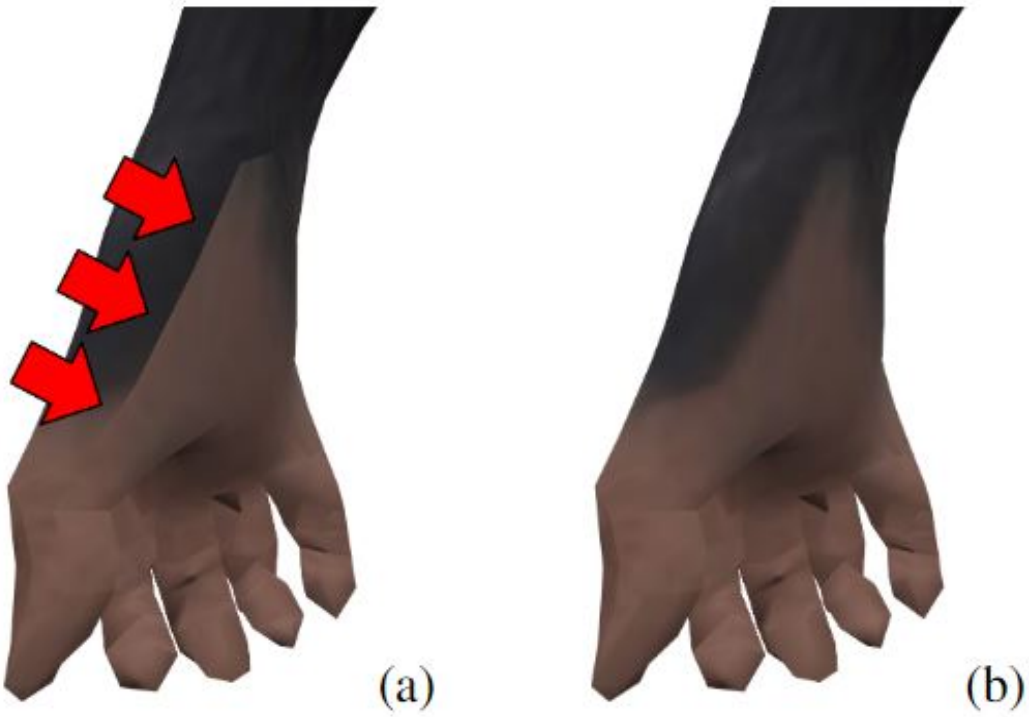


Figure 2.7: Illustration of visible seams in 3D mesh texture. (a) Example of visible color transitions between two mesh segments in traditional UV texture mapping. (b) Reduced visible contrast between color transitions between two mesh segments. Image credit: Yuksel *et al.*, 2019 [6].

2.3.2 Seamless Texture Mapping

One approach to achieving a seamless UV representation of a 3D mesh involves creating an atlas, which consists of charts that map a 2D texture domain to a connected part of the 3D surface patch [143] (see Fig. 2.7(b) [145]). However, the atlas method is primarily designed for efficient texture storage and requires the texture to be present on the surface

Figure 2.8: Example of avoiding vertex seam color transitions in cylindrical and toroidal UV mappings. Image credit: Tarini *et al.*, 2012 [7].

before mapping. An alternative method for avoiding vertex seams in cylindrical and toroidal UV mappings used for texture mapping is presented in [7] (see Fig. 2.8). This method employs a simple rendering technique.

Another approach, introduced in [144], involves computing seamless bijective mappings between two surface meshes by interpolating a given set of correspondences. This is achieved through a novel type of surface flattenings that encode cut-invariance. By optimizing the flattening process using a suitable energy function, a seamless surface-to-surface map is obtained. Several other methods exist for avoiding seams in UV mapping

as well [146, 147].

2.4 Human-Robot Interaction

The emergence of Industry 5.0, often referred to as the Fifth Industrial Revolution, signifies a new phase of industrialization where humans work alongside advanced technology and AI-powered robots to enhance workplace processes. Industry 5.0 extends beyond manufacturing and builds upon the foundation laid by the fourth industrial revolution (Industry 4.0) and focuses mainly on mass customization, where humans will be guiding robots [8](see Fig. 2.9). Our work aligns with the goals and developments of this new stage in the industrial revolution.

In this section, we explore the advancements and practical implementations of human-robot interaction that have improved overall production across various industries. Then, we survey research aimed at ensuring the safe and reliable integration of software interfaces and robots working alongside humans. Drawing guidance from previous research in HRI is crucial in developing methods for industrial surface coverage applications that inherently require a robotic system interface.

2.4.1 Enhancing Production Through HRI

In recent years, the focus of research in human-robot interaction (HRI) has predominantly shifted towards human-robot collaboration (HRC) scenarios in industrial settings (see Fig. 2.10). The introduction of assistive robotics in industries has shown promising potential

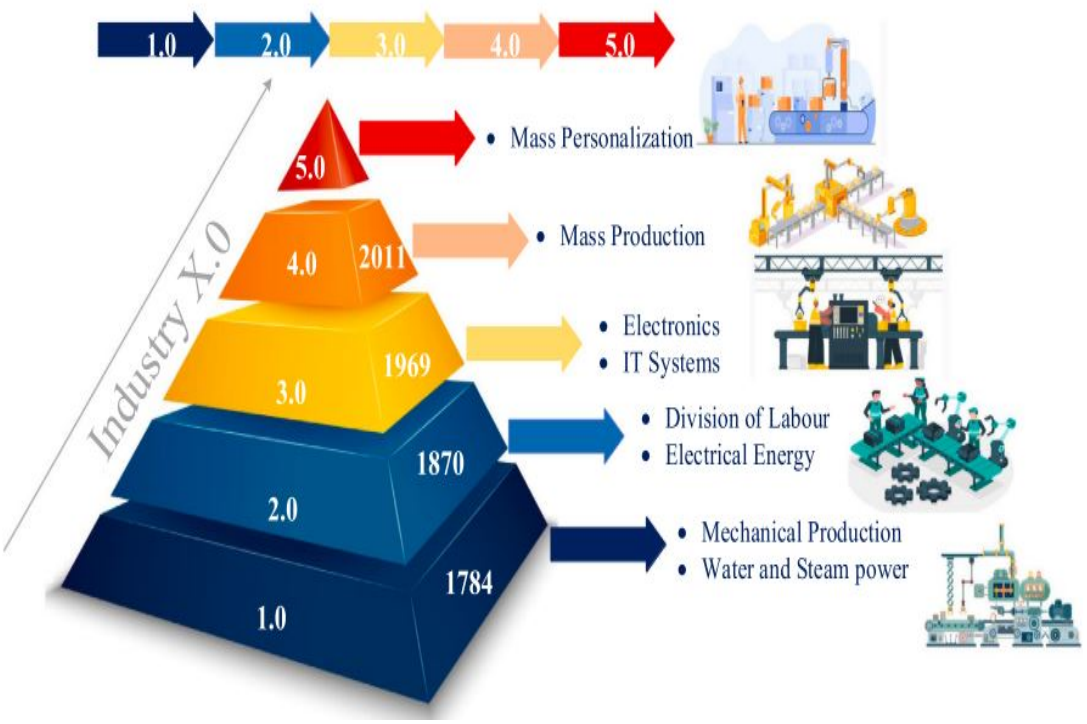


Figure 2.9: Illustration of industrial evolution to Industry 5.0. Image credit: Maddikunta *et al.*, 2022 [8].

to increase production efficiency and enhance product quality [148–150]. While each industry presents unique challenges, there are underlying problems that share similarities, allowing for generalized research and solutions in HRI applications.

One notable study investigates the assembly line balancing problem (ALBP) within the context of HRC. Referred to as ALBP-HRC, this problem arises in advanced manufacturing systems where humans and robots share the same workspace and can perform tasks simultaneously in parallel or in collaboration. To address this challenge, a neighborhood-search simulated annealing (SA) approach is proposed, incorporating customized solution representation and specialized neighborhood search operators designed to align with the

problem characteristics [151].

Moreover, the successful integration of industrial collaborative robots is still an area requiring further investigation. Factors that facilitate or hinder the introduction of such robots are studied in [152]. Klumpp et al. [153] develops an efficiency description for human-computer interaction (HCI) in production logistics. This research involves an interdisciplinary analysis, including a literature review and process study in production logistics, as well as a computer science literature review and simulation study focused on an existing autonomous traffic control algorithm.

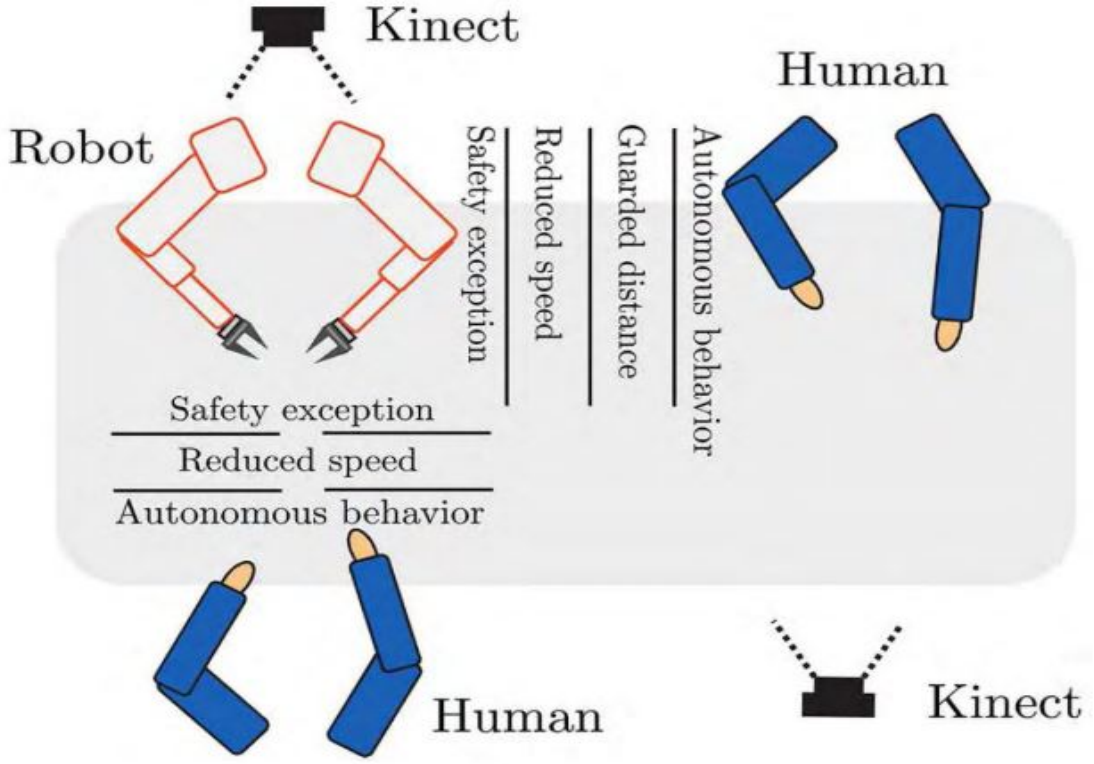


Figure 2.10: Illustration of humans and robot working together using RGB-D devices.Image credit: Gomez *et al.*, 2017 [9].

There has been a significant focus on developing interfaces for robots and manipulators in industrial environments. These interfaces facilitate communication between humans and robots. Physical interaction can take place through touch, either by physically moving the robot or by using a mouse and keyboard. Graphics interfaces have been created to help workers understand the input required from humans for the robot to carry out its assigned task [58, 154–156]. Alternatively, non-physical communication methods such as gesturing or voice commands can be employed [58, 157–159].

These studies highlight the advancements in HRI, specifically in the context of human-robot collaboration within industrial environments. The research aims to address the challenges and explore opportunities to optimize productivity and efficiency through effective human-robot interaction.

2.4.2 HRI Safety Standards

In the age of industrialization and automation, ensuring safety becomes crucial when designing and implementing new systems intended for close collaboration with humans. Within the broader field of human-robot interaction (HRI), ensuring human safety is a significant aspect as it promotes a harmonious coexistence between humans and robots [58, 156, 158, 160–162, 162].

When designing virtual interfaces for human worker interaction, it's important to consider how the software and interface will affect the workers' productivity. Studies by Zhou et al. [163] and Zhang et al. [164] have investigated the impact of software interface

design on user fatigue. Furthermore, previous research on technophobia has explored the main components of computer anxiety and attitudes. Computer anxiety is often studied in educational psychology, and it involves individuals' emotional responses such as fear, apprehension, intimidation, hostility, and worries about embarrassing themselves, looking foolish, or damaging equipment. Attitudes towards computers are closely related to people's perceptions of the impact of computers on society, quality of life, and their overall understanding of computers [165].

If a simulation environment were to be connected to a real robot for implementation, it becomes essential to consider certain safety precautions. In many industrial applications, robot manipulators are used alongside human workers to complete tasks, and significant efforts have been made to establish and maintain safe standards. One crucial aspect is collision avoidance, which aims to prevent manipulators from causing harm to their human coworkers [166–168].

Chapter 3

Problem Formulation and Objectives

In this chapter, we discuss the main problems faced in robotic industrial applications in the context of continuous coverage motion planning on 3D surfaces. This dissertation focuses on the following theme: "Given a manipulator, a 3D surface, and task constraints, how to enable a human worker, who is a task specialist but not a robotics expert, to use automatic algorithms to determine the feasibility for the end-effector to traverse the entire surface continuously while maintaining the task and manipulator constraints, and if it is feasible, produce an evenly spaced, continuous manipulator coverage path that best satisfies task constraints." The problems in this theme are explored and objectives are stated in the following sections.

3.1 Problem Description

The theme of this dissertation can be reflected in the following research questions:

1. *How to develop an interface to facilitate a task specialist, who is not a robotics expert, to interact with an autonomous robotic system to systematically determine optimal surface and manipulator spatial arrangement? (setting up method environment and preconditions for autonomous systems)*

   Efficiency and enhanced interaction between the robot system and the task specialist are crucial for improving cycle time in industrial surface coverage applications. The human worker should be able to efficiently determine the optimal spatial arrangement between the manipulator and the surface, with assistance from the autonomous robot system. Additionally, the human worker should be able to communicate key task parameters to the robotic system. The solution to the problem statement needs to be practical and enable quick testing of multiple spatial arrangements within the manipulator workspace. Furthermore, an interface should provide feedback to the human worker for analysis, particularly when the robotic system faces challenges in finding solutions.

2. *Given a spatial arrangement between a robot manipulator and a 3D surface, does there exists a continuous motion path that enables the manipulator to cover the entire surface while maintaining task constraints?*

   A fundamental question that needs to be answered prior to surface coverage planning is if a feasible coverage path exists given manipulator kinematics, task constraints, and surface positioning relative to the manipulator. The answer to this can provide

insight as to whether the manipulator is suitable for the given task, if task constraints need to be relaxed, or what spatial arrangements are optimal for surface coverage. If coverage feasibility exists, the answer can inform on the extent of feasibility for coverage patterns and, alternatively, if coverage is not feasible, it can provide results for analysis to determine the cause.

3. *Given a point cloud of a 3D freeform surface, how to map uv coordinates to the Cartesian surface s.t. the uv parameters accurately describe the 2D surface manifold?*

   Once coverage feasibility is established, placing a coverage pattern on a freeform surface becomes a non-trivial task. Previous research on this topic has struggled to achieve uniform spacing of coverage patterns across complex freeform surfaces. However, insights from computer graphics offer valuable knowledge on mapping uv coordinates to 3D surfaces for texture storage purposes. By implementing a uv grid on the 3D surface, it becomes possible to facilitate the placement of a uniform coverage pattern within the 2D region and subsequently map it onto the corresponding 3D surface.

4. *Given a uv grid for a 3D surface, how to place a coverage pattern that covers the area, and then how to facilitate the generation of a continuous manipulator motion plan to follow the surface coverage pattern?*

   Discussions on proper coverage pattern placement are needed for the novel approach.

The uv grid on the 3D surface can provide valuable insights into determining the suitable coverage pattern. Additionally, once a coverage pattern is placed on the surface, we need to generate a continuous manipulator motion plan which follows the surface coverage pattern while maintaining task constraints. A manipulator motion plan that best satisfies task constraints is preferred.

5. *Given a semi-autonomous method for surface material application and finishing, how to develop an interface for non-robotic expert workers to interact with and apply the method?*

   A human worker can benefit from a virtual environment designed specifically for testing and analyzing surface coverage applications to determine the optimal task parameters. This virtual environment allows for the simulation of surface coverage scenarios and offers an interface that enables human worker input when required. By leveraging this virtual environment, the human worker can effectively assess and fine-tune the surface coverage task parameters to achieve optimal results.

3.2 Research Objectives

This dissertation has the overarching goal to develop a semi-autonomous method for efficient and optimal continuous coverage motion planning on 3D surfaces. The aim is to enhance the industrial process in this area as well as the quality of work for human workers. Human-robot interaction is inherent within the process and providing a suitable

interface is crucial. The specific objectives for achieving these goals will be discussed in the subsequent subsections.

3.2.1 UV Mapping on 3D Surfaces

Currently, there is a lack of UV mapping methods specifically designed for surface coverage applications on 3D surfaces. Developing a method that effectively maps a 3D surface to a uv space while preserving the surface's geometric properties would be a significant contribution. Such a method will allow general purpose 2D coverage algorithms to be used to cover 3D surfaces. Additionally, the uv representation may provide more insight for coverage pattern placement. Our objective is to provide a semi-autonomous method for uv grid generation on general 3D freeform surfaces. We will provide an interface for the human worker to give input to the semi-autonomous method.

3.2.2 Efficient Coverage Feasibility Checking

There is also no prior work on determining surface coverage feasibility given the surface, manipulator, and task constraints. Our objective is to provide an efficient coverage feasibility checking method that allows a human worker to determine feasibility in a reasonable amount of time in the production pipeline. This systematic method for checking coverage feasibility at different spatial arrangements will allow a human worker to find an optimal spatial arrangement for surface coverage applications.

3.2.3 Surface Coverage Pattern and Manipulator Motion Planning

Even though surface coverage is determined feasible, a feasible coverage path for the manipulator is not yet given. Our objective here is to provide guidance on coverage pattern placement in the uv grid of the surface and to provide an algorithm to determine a coverage motion plan for the manipulator to follow the surface coverage pattern that best satisfies task constraints.

3.2.4 Virtual Interactive Environment for Industrial Surface Coverage

The non-robotic expert worker alongside the semi-autonomous surface coverage system will need to analyze material coverage results on the 3D surface systematically. This will allow the task specialist to discover the best task parameters for their desired application. Our approach involves the development of a virtual environment and interface that empowers the human worker to implement the methods discussed in this work. Additionally, the worker will have the capability to visually review analysis information provided by the robotic system. The analysis information will include both visual and numerical results regarding material coverage, based on the task parameter input provided by the worker.

3.3 Overview and Contributions of Remaining Chapters

Chapter 4 presents the proposed method for UV mapping on 3D surfaces as published in [169]. It covers UV mapping on parametric surfaces and introduces a novel method for UV mapping on 3D freeform surfaces. The contribution here is a human-assisted approach for automatically generating uv grids on 3D freeform surfaces. The uv grids facilitate the placement of uniform coverage patterns, coverage feasibility checking, and material coverage analysis on 3D freeform surfaces.

Chapter 5 proposes a method for efficient coverage feasibility checking as published in [170]. The coverage feasibility algorithm searches through the joint space of the manipulator, utilizing the surface uv grid to check motion continuity. The main contribution of this work is an efficient approach that enables the determination of continuous coverage feasibility for 3D freeform surfaces. Coverage feasibility results can inform on probable causes of infeasibility due to manipulator kinematics, task constraints, or spatial relationships between the surface and manipulator.

Chapter 6 introduces the generation of uniform coverage patterns on 3D freeform surfaces along with the generation of a singularity-free motion plan for the manipulator to follow the surface coverage pattern as published in [171]. The main contribution of this chapter lies in providing insights into the selection and placement of uniform coverage patterns on the uv grid of a 3D freeform surface, as well as presenting an algorithm for

identifying a continuous motion path for the manipulator to traverse the surface coverage pattern which best satisfies task constraints.

Chapter 7 discusses the design of a virtual environment and interface intended for human workers to interact with and implement the methods proposed in this dissertation. The chapter includes a case study on virtual surface spraying and conducts a comparative analysis against previous surface coverage methods explored in related research. The primary contribution of this chapter is the development of a virtual environment and a user-friendly interface that facilitates interactive simulation of continuous coverage motion planning on 3D freeform surfaces.

Chapter 8 presents the conclusions and potential future work. The chapter highlights several directions for future research, such as broadening the scope of freeform surfaces covered by the uv grid generation method. Additionally, it suggests exploring the segmentation of 3D freeform surfaces to address unfeasible areas by employing multiple separate coverage paths. Furthermore, the chapter proposes leveraging the virtual simulation environment to optimize task parameters for surface coverage applications. These avenues represent potential areas for further investigation and improvement in continuous coverage motion planning on 3D freeform surfaces.

Chapter 4

UV Grid Generation on 3D Surfaces

To facilitate coverage feasibility checking and coverage pattern placement on 3D freeform surfaces, we require a method for decomposing the surface into evenly sized cells and parameters for evenly spaced coverage patterns. Previous work for path planning on 3D surfaces [1, 10–12, 109, 172–175] relied on x, y, and z coordinates in Cartesian space as references for surface spacing which can result in the uneven discretization of the surface.

We introduce a novel uv mapping technique on 3D freeform surfaces for continuous surface coverage applications. We define the uv space directly on the freeform surface and discretize it to generate a grid of evenly spaced cells, $uv\text{-}cells$. The uv grid is used in later chapters for coverage feasibility checking, surface coverage pattern placement, and surface material coverage analysis.

This chapter is organized as follows. Section 4.1 discusses uv parameters for parametrically defined 3D surfaces. Section 4.2 derives equations for the automatic generation of uv

grids on 3D freeform surfaces using the topology of the surface polygon mesh. Section 4.3 gives results from generating uv grids on different surface types and Section 4.4 provides conclusions.

4.1 Generating uv Grid for 3D Parametric Surfaces

Table 4.1: uv parameters for parametrically defined surfaces.

Surface Parametric Equations					
S	u	v	x	y	z
Sphere	θ	ψ	$r*sin\theta*cos\psi$	$r*sin\theta*sin\psi$	$r*cos\theta$
Cylinder	θ	z	$r*cos\theta$	$r*sin\theta$	z
Hyperboloid	θ	z	$r(z)*cos\theta$	$r(z)*sin\theta$	z
where $r(v_{min}) \leq r(v) \leq r(v_{max})$					

Various surface patches can be parametrically defined such as those similar to a sphere, cylinder, or hyperboloid surface, S. The parametric equations defining each surface (surface x, y, and z points) are expressed with respect to its origin $[x_c, y_c, z_c]^T$. For each surface frame, its origin is at the center (of symmetry) of the surface, and its orientation is aligned with the robot base frame. The radius of the surface is denoted with r. The spherical surface is a quadrant of a hemisphere, and parameters u and v denote the polar angle and azimuthal angle θ and ψ of the spherical coordinate system respectively. The cylinder and hyperboloid surfaces are constrained by u and v which denote angle θ (about the z-axis of the surface frame) and height z (with respect to origin) respectively (see Table

4.1). These uv equations for parametric surfaces are used again in Chapter 5.

4.2 Generating uv Grid for 3D Freeform Surfaces

In this section, we introduce our novel method for generating a uv grid on 3D freeform surfaces. We first describe the surface types within our scope of work. Then we give the mathematical calculations used for converting the triangle vertices in the surface mesh from Cartesian space to uv space for each surface type.

4.2.1 Freeform Surface Types

We consider a 3D polygon mesh representation of a physical freeform surface, S. The physical surface of an object can be scanned using modern 3D scanning technology and approximated as a 3D polygon mesh, which is the most common way to represent a general surface when there is no other more precise model. Additionally, S must fit within the workspace of a robotic arm to allow the end-effector to cover the surface. A large surface may be segmented to small S's using hierarchical methods as done in [176, 177].

We define and consider the following three types of general 3D freeform surfaces that can be generated from some transformation of a planar, non self-intersecting freeform curve C (with C_0 denoting the initial curve C), along or about some axis called $\mathcal{A}$ (see Fig.

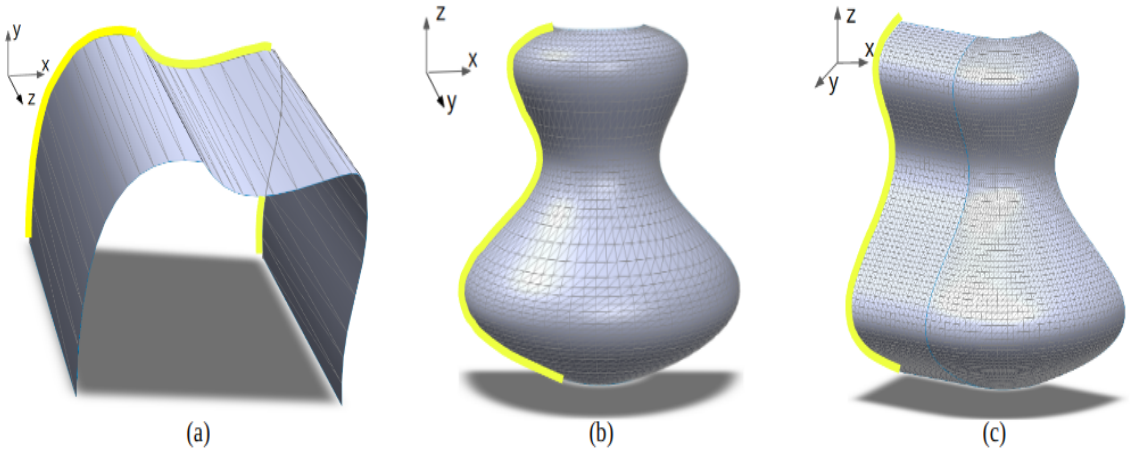


Figure 4.1: Illustrations of 3D freeform surface types (embedded polygon meshes). Curve C_0 for each definition is highlighted in yellow. (a) Freeform surface of translation along the z axis. (b) Freeform surface of rotation about the z axis. (c) Freeform surface of translation (on the xy plane) and rotation (about axis parallel to the z axis).

4.1),

- **Surface of translation (SoT)**: translation of C, along $\mathcal{A}$.
- **Surface of rotation (SoR)**: rotation of C, about $\mathcal{A}$.
- **Surface of translation and rotation (SoTR)**: combinations of SoTs and SoRs from a single C, such that all rotation axes of C are parallel to $\mathcal{A}$ and all translation axes of C are on a plane normal to $\mathcal{A}$.

The curve C can be known or approximated from the input of a human worker.

Note that the above surface types share an important property that curve C is theoretically the same along each axis of translation or about each axis of rotation[1]. Additionally,

[1]Practically, for a freeform surface mesh, C is approximated as a concatenation of (short) straight-line segments as the result of the triangle mesh and can be slightly different at different locations on the surface mesh.

at each point of curve C, the corresponding perpendicular curve maintains a constant curvature. However, as we move along curve C, these perpendicular curves exhibit varying curvature values, contingent upon the specific curve C. Consequently, there is no uniform, constant curvature present throughout each type of these surfaces.

For a SOTR, the surface can be formed through an arbitrary combination of translations and rotations applied to curve C. These transformations may happen simultaneously, with the caveat that the rotation axes remain in the same direction, while the translation axis always remains orthogonal to the rotation axis. It's possible to have different rotation axes, each defining a segment of a different SoR of the surface, and the translations can transpire along various axes. As a result, the SoTR has the potential to encompass a broader array of diverse surfaces than the SoT and the SoR.

The surface types mentioned above can represent various surfaces in the real world, and larger surfaces can be segmented into these types. However, it is essential to acknowledge that there are limitations to surface segmentation based on these defined surface primitives. For instance, highly complex or rough surfaces may not be accurately captured. Our surface types are designed for general industrial product surfaces, which are typically less complex compared to artistic surfaces.

4.2.2 Interactive Determination of Curve C

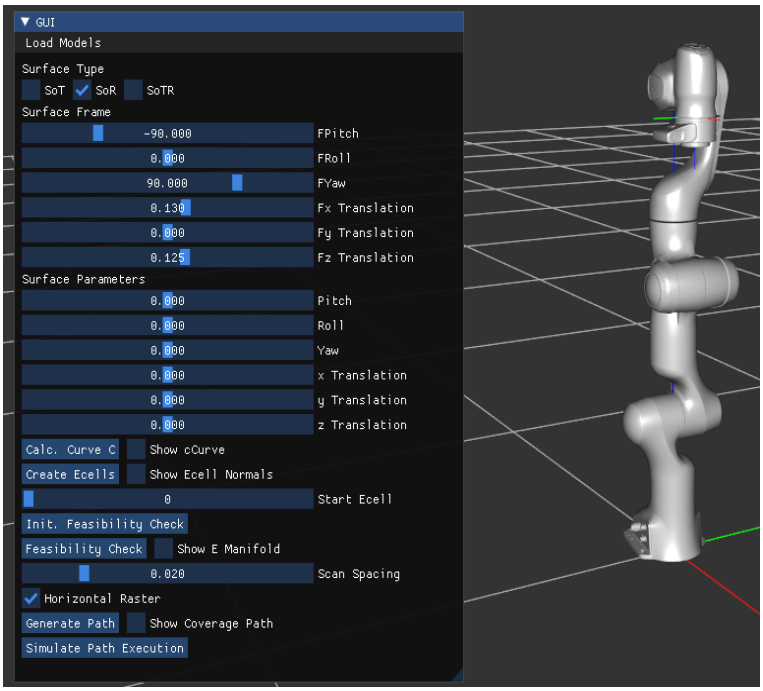
Our virtual environment interface provides a human worker, who is a task-domain expert but not a robotics expert, with a visual and numerical representation of the spatial and

orientation arrangement between the surface S and the manipulator. We denote S_f as the coordinate frame of S. The human worker may adjust the position and orientation of S_f with respect to the manipulator base (world frame), thus changing the arrangement between S and the manipulator. Additionally, S_f is positioned by the human worker with respect to surface mesh vertices to facilitate the automatic generation of the uv grid. Fig. 4.2(a) shows the interface (numerical sliders) and visual display for manipulating S_f.

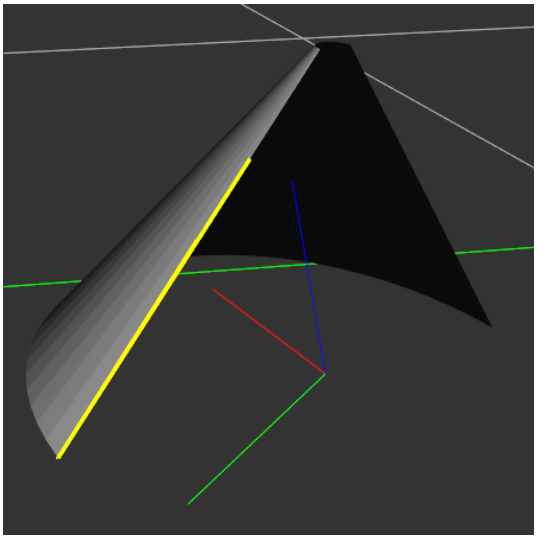
To enable automatic uv grid generation, the human worker first determines the surface S as one of the three types defined. Then, they align the (blue) z-axis of the surface frame S_f with axis $\mathcal{A}$, and our underlying automatic system helps the human worker to verify if S_f is placed accurately by showing the resulting curve C (highlighted in yellow). Fig. 4.2(b) shows an inaccurate placement of S_f, where the curve C is too short, and Fig. 4.2(c) shows the accurate placement of S_f.

4.2.3 Automatic Generation of uv Space

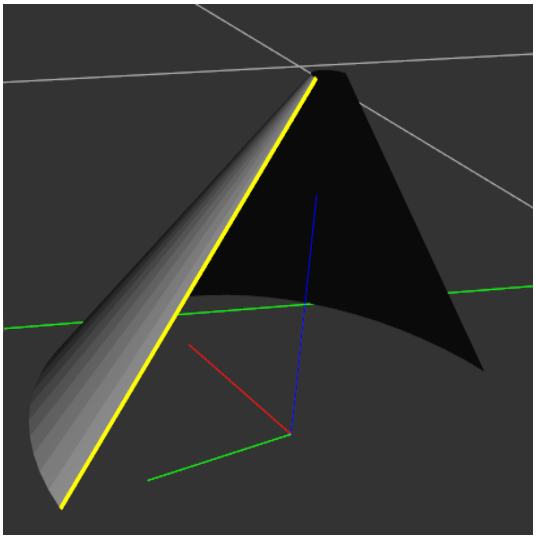
We now describe the automatic generation of uv space for the three types of freeform surfaces, surface of translation (SoT), surface of rotation (SoR), and surface of translation and rotation (SoTR), in turn. Our idea is to let the uv parameters be curve length parameters along the surface to achieve even spacing of the freeform surface along a curve. This concept is illustrated in Fig. 4.3(b) and compared to previous methods using Cartesian coordinates for spacing in Fig. 4.3(a).



(a) Simulation interface for interactive determination of key parameters.

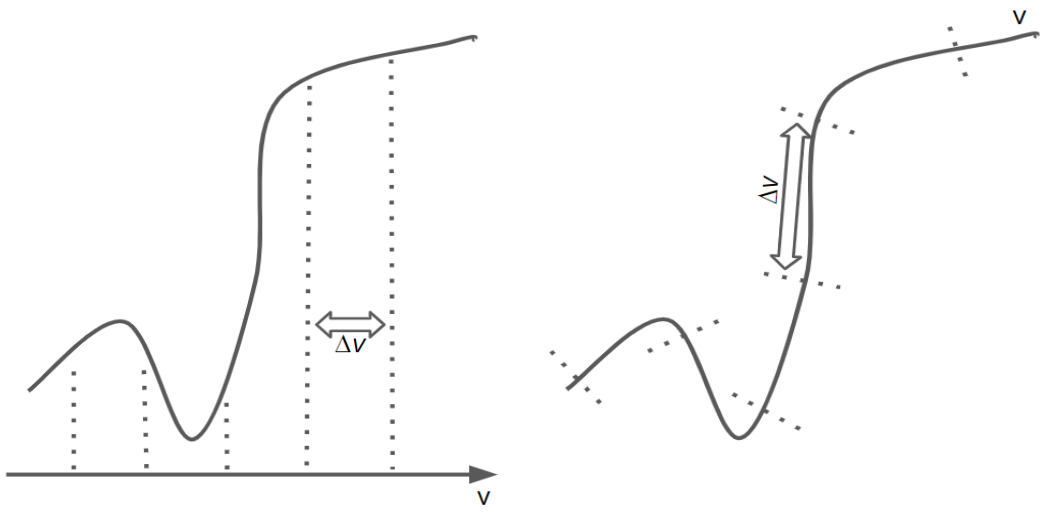


(b) Inaccurate placement of surface frame, S_f, resulting in poor curve C (yellow) calculation.



(c) Accurate placement of surface frame, S_f, resulting in good curve C (yellow) calculation.

Figure 4.2: Illustration of simulation interface for human worker to interactively determine curve C to facilitate automatic uv space generation.



(a) Discretization along an axis results in uneven spacing on curve C.

(b) Even discretization of curve C with v as a length parameter.

Figure 4.3: Comparing two different ways of discretization along a curve C.

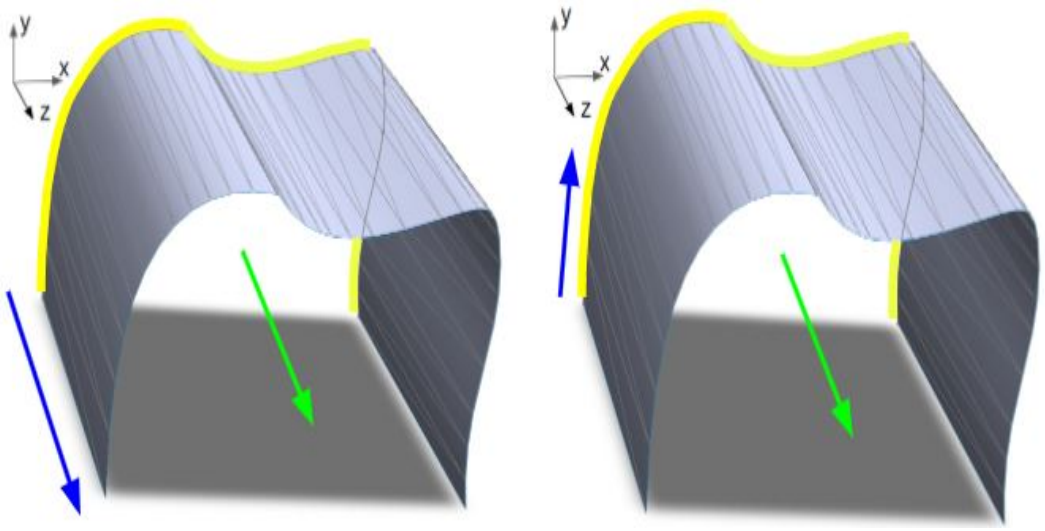
For each type of surface, the surface coordinate frame S_f is set such that the z axis is along the axis $\mathcal{A}$, as shown in Fig. 4.1. The v coordinates are defined along the curve C, recalling that the curve C remains the same theoretically during translation and rotation to form the surface. The starting v coordinate is at one endpoint of C. The u coordinates are defined along curves perpendicular to C.

The following describes how to obtain the $[u, v]$ coordinates for all vertices of the triangle mesh for each type of surface.

Generating uv Space for Surface of Translation (SoT)

Our simplest definition of a freeform surface is a surface of translation (there is curvature only about one axis). The polygon mesh is structured such that the triangle face vertices

are only on the edges of the mesh which are perpendicular to the direction of translation (see yellow highlighted mesh edges in Fig 4.4). Let C_0 be the curve C on the xy plane at the beginning of translation (obtained by the human worker) with z_0.



(a) u equals vertex z-axis value (blue arrow).

(b) v is calculated as curve length along curve C (blue arrow).

Figure 4.4: Illustration of uv parameter calculations along SoT (blue arrow) with axis of translation (green arrow).

Our algorithm sweeps (i.e., translates) a plane perpendicular to the z axis from z_0 to increase z value, starting from the xy plane. The sweeping pauses at every triangle vertex encountered and records the z value as z_i, $i = 1, ..., m$. Our algorithm then obtains the corresponding curve C_i as the intersection of the xy plane with surface S at z_i.

Let $(x_{i,j}, y_{i,j}, z_{i,j})$ be the coordinates of the jth triangle vertex on the curve C_i, $j = 0, ..., n_i$. These points share the same u coordinate:

$$u_{i,j} = z_i. \tag{4.1}$$

For $j = 1, ..., n_i$, let $\Delta x_{i,j} = x_{i,j} - x_{i,j-1}$ and $\Delta y_{i,j} = y_{i,j} - y_{i,j-1}$. We can compute $v_{i,j}$ by initializing $v_{i,0} = 0$ and:

$$v_{i,j} = v_{i,j-1} + \sqrt{\Delta x_{i,j}^2 + \Delta y_{i,j}^2}, \quad j = 1, ..., n_i. \tag{4.2}$$

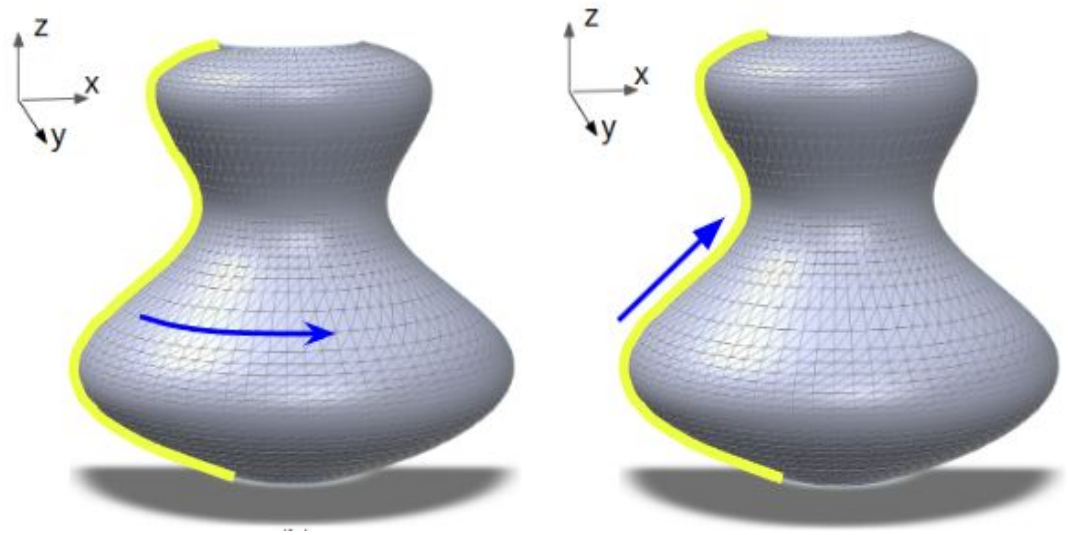
Thus,

$$v_{i,j} = \sum_{k=1}^{j} \sqrt{\Delta x_{i,k}^2 + \Delta y_{i,k}^2}. \tag{4.3}$$

Generating uv Space for Surface of Rotation (SoR)

The surface of rotation, SoR, is a rotation of curve C about an axis. We denote the angle of rotation as ϕ and radius (i.e. vertex distance from the axis of rotation) as $r(z)$, which is an unknown function of z due to the freeform curve C. Let C_0 indicate the curve C on the plane of $\phi = \phi_0$ (see yellow highlighted mesh edges in Fig 4.5).

Now, by increasing ϕ and rotating the corresponding plane (which goes through the z axis), our method pauses the rotating plane at every triangle vertex encountered and records the ϕ value as ϕ_i, $i = 1, ..., n$. Our algorithm then obtains the corresponding curve C_i as the intersection of the plane with surface S at ϕ_i.



(a) u is calculated as the arc curve length about the z-axis (blue arrow).

(b) v is calculated as the curve length along curve C (blue arrow).

Figure 4.5: Illustration of uv parameter calculations along SoR (blue arrow).

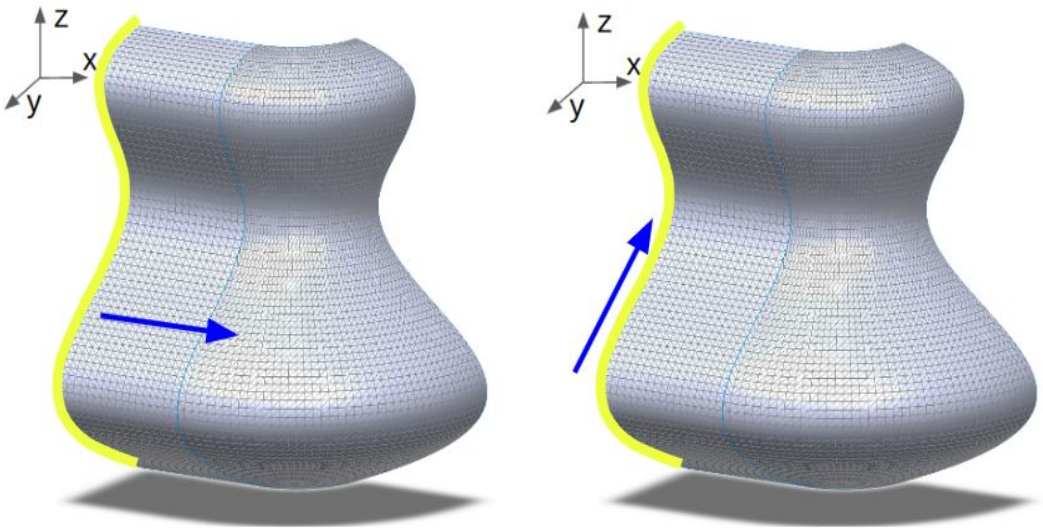
Let $(x_{i,j}, y_{i,j}, z_j)$ be the local coordinates of the jth triangle vertex on the curve C_i with C_i, for $j = 0, ..., n_i$. We compute its u coordinate as:

$$u_{i,j} = r(z_j) \cdot \phi_i. \tag{4.4}$$

For $j = 1, ..., n_i$, let $\Delta z_j = z_j - z_{j-1}$ and $\Delta r_j = r_j - r_{j-1}$. We can compute $v_{i,j}$ by initializing $v_{i,0} = 0$:

$$v_{i,j} = v_{i,j-1} + \sqrt{\Delta z_j^2 + \Delta r_j^2}, \quad j = 1, ..., n_i. \tag{4.5}$$

Thus,



(a) u is calculated as curve length along the direction of translations and rotations.

(b) v is calculated as curve length along curve C.

Figure 4.6: Illustration of uv parameter calculations along SoTR (blue arrow).

$$v_{i,j} = \sum_{k=1}^{j} \sqrt{\Delta z_j^2 + \Delta r_j^2}. \tag{4.6}$$

Generating uv Space for Surface of Translation and Rotation (SoTR)

The surface of translation and rotation (SoTR) can represent many types of complex 3D surfaces. In this case, all rotation axes are parallel to the z-axis, and all translation axes are orthogonal to the z-axis. Let C_0 be on plane $\mathcal{P}$ that also includes the z axis (see yellow highlighted mesh edges in Fig 4.6).

The SoTR and the xy plane intersect at curve U, which intersects C_0 at point p_0. Our method sweeps the plane $\mathcal{P}$ along U while maintaining it perpendicular to U at each point[2],

[2]by translation or rotation or simultaneous translation and rotation depending if the surface portion is

and the sweeping pauses at every triangle vertex encountered on S by $\mathcal{P}$ and records the Cartesian position of the intersection point with U. Let p_i indicate the ith intersection point between the swept $\mathcal{P}$ and U, and let the corresponding C curve be C_i, for $i = 1, ..., m$.

Let $(x_{i,j}, y_{i,j}, z_{i,j})$ be the coordinates of the jth triangle vertex on the curve C_i, for $j = 0, ..., n_i$. For $i = 1, ..., m$, let $\Delta_j x_{i,j} = x_{i,j} - x_{i-1,j}$ and $\Delta_j y_{i,j} = y_{i-1,j} - y_{i,j}$. By initializing $u_{0,j} = 0$, we can compute the u coordinate of the point as:

$$u_{i,j} = u_{i-1,j} + \sqrt{\Delta_j x_i^2 + \Delta_j y_i^2}, \quad i = 1, ..., m. \tag{4.7}$$

Thus,

$$u_{i,j} = \sum_{k=1}^{i} \sqrt{\Delta_j x_k^2 + \Delta_j y_k^2}. \tag{4.8}$$

Now, Let $\Delta_i x_{i,j} = x_{i,j} - x_{i,j-1}$, $\Delta_i y_{i,j} = y_{i,j} - y_{i,j-1}$, $\Delta_i z_{i,j} = z_{i,j} - z_{i,j-1}$, $j = 1, ..., n_i$.

By initializing $v_{i,0} = 0$, we can compute the v coordinate of the point as:

$$v_{i,j} = v_{i,j-1} + \sqrt{\Delta_i x_{i,j}^2 + \Delta_i y_{i,j}^2 + \Delta_i z_{i,j}^2}, \quad j = 1..., n_i. \tag{4.9}$$

Thus,

$$v_{i,j} = \sum_{k=1}^{j} \sqrt{\Delta_i x_{i,k}^2 + \Delta_i y_{i,k}^2 + \Delta_i z_{i,k}^2}. \tag{4.10}$$

SoT or SoR or both.

4.2.4 Automatic Discretization of uv Space

For every triangle $\mathcal{T}$ on the surface S, its vertex with position $\tau_i = [\tau_{ix}, \tau_{iy}, \tau_{iz}]^T$, $i = 1, 2, 3$, now has uv coordinates $[\tau_{iu}, \tau_{iv}]^T$. Note that the uv coordinates at each of those points are real numbers as computed above, even though those discrete points are indexed by integers.

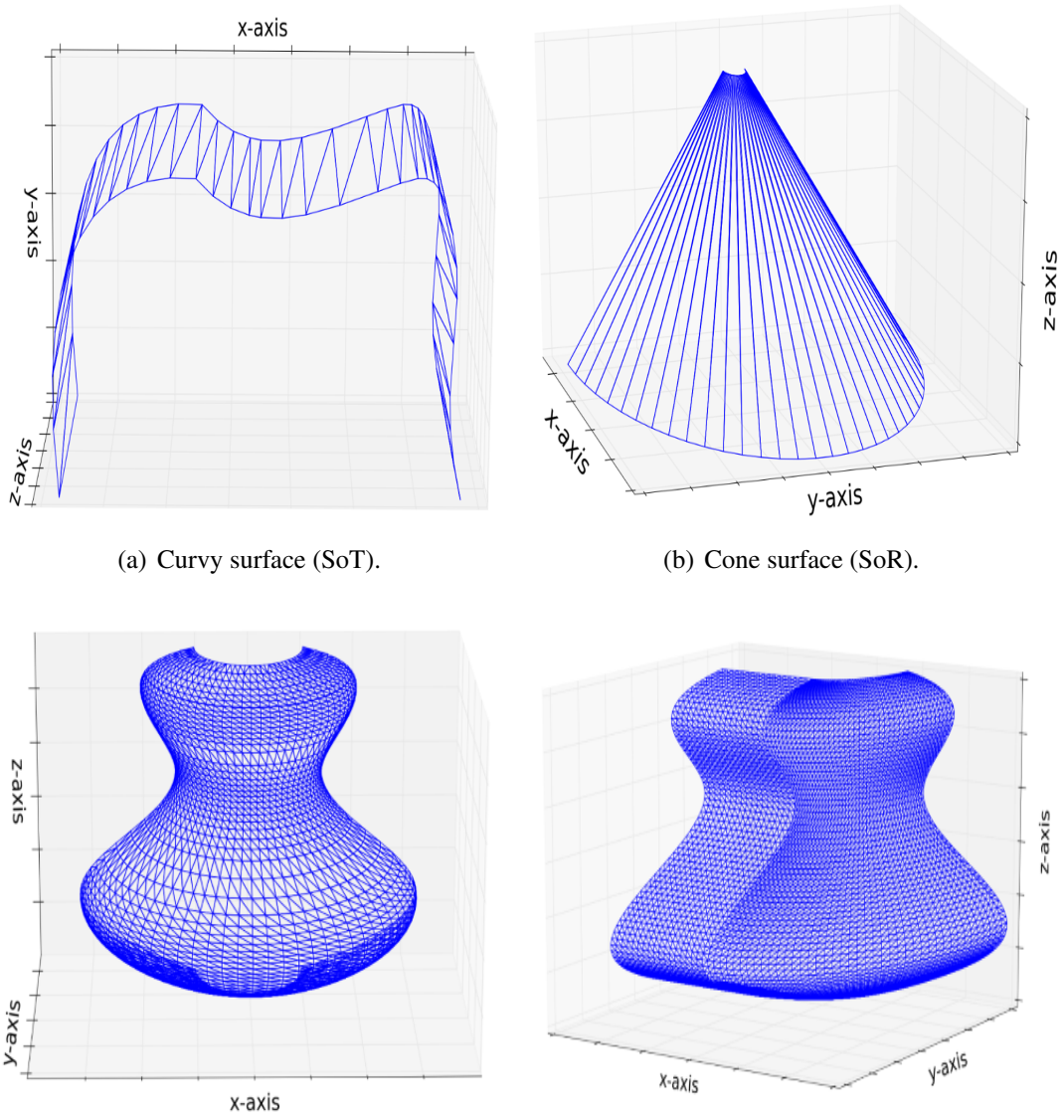
Additionally, for any other point s inside the triangle, its coordinates of either Cartesian space or uv space can be found based on its distance to one of the triangle vertices. Let $\mathbf{p}_s$ indicates point s's position in the Cartesian space, point s's barycentric coordinates, w_i, can be found from the following equations:

$$\mathbf{p}_s = \Sigma_i w_i \tau_i, \;\; 0 \leq w_i \leq 1, \;\; \Sigma_i w_i = 1. \tag{4.11}$$

From the barycentric coordinates, the point s's uv-space coordinates $[u, v]$ can be determined. In general, the Barycentric coordinates w_i's can be used to find either Cartesian or uv-space coordinates of a point inside a triangle from the corresponding vertex coordinates of the triangle.

To discretize the uv space into a grid with uniform cells, let Δu and Δv be the desired dimensions of an uv-cell. We denote a uv-cell with the cell center coordinates $[u_j, v_k]$, where j and k are integer indices for the discretization, even though the uv-space coordinates u_j and v_k are real numbers from the original continuous space.

4.3 Experiments and Analyses



(a) Curvy surface (SoT).

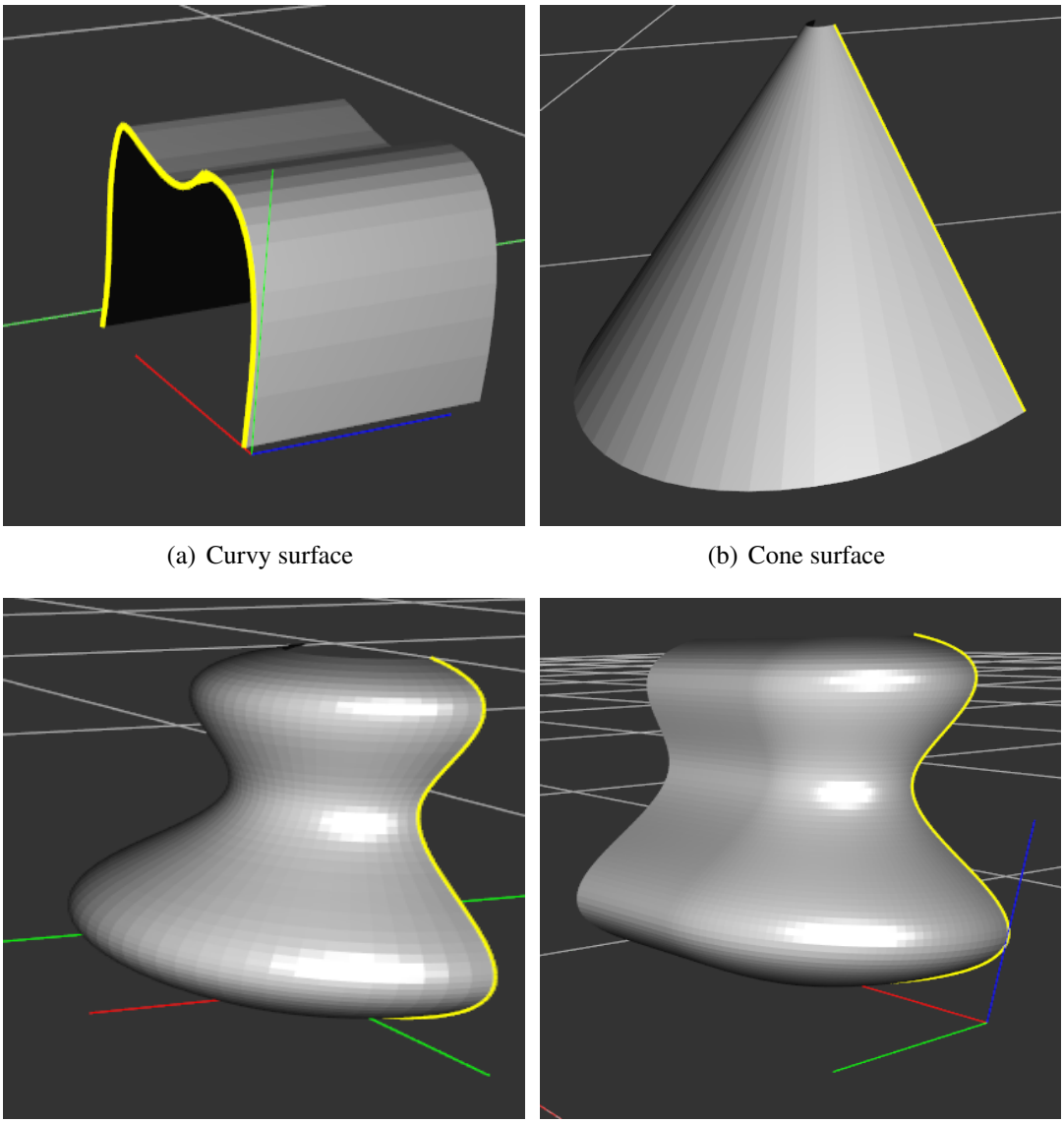
(b) Cone surface (SoR).

(c) Vase surface (SoR).

(d) Corner surface (SoTR).

Figure 4.7: Polygon mesh of four different freeform surfaces.

In the following experiments, we test the interactive determination of curve C through the interface in our virtual environment and the automatic generation of uv parameters on



(a) Curvy surface

(b) Cone surface

(c) Vase surface

(d) Corner surface

Figure 4.8: Initial curve C highlighted in yellow for each surface.

a SoT, SoR, and SoTR. Solidworks is used to generate a polygon mesh of a Curvy (SoT), Cone (SoR), Vase (SoR), and Corner (SoTR) surface (see Fig. 4.7), containing 68, 32, 3743, and 9200 triangle faces respectively. Figure 4.8 shows the results of the interactive placement of each surface frame and feedback from the virtual environment which visually

displays curve C in yellow.

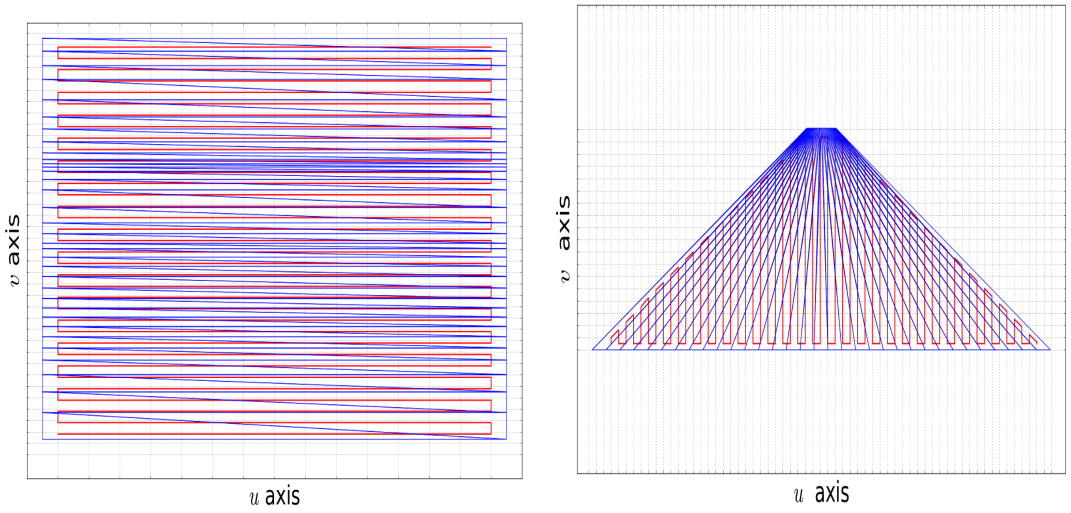
4.3.1 Results of uv Grid Generation

The mesh triangle vertices for each surface were transformed into the uv space using the methods proposed in this chapter and corresponding triangle faces were reconstructed in the uv space to give a 2D representation of the polygon mesh. See Fig. 6.6 for the resulting polygon mesh in uv space.

4.3.2 Discussion

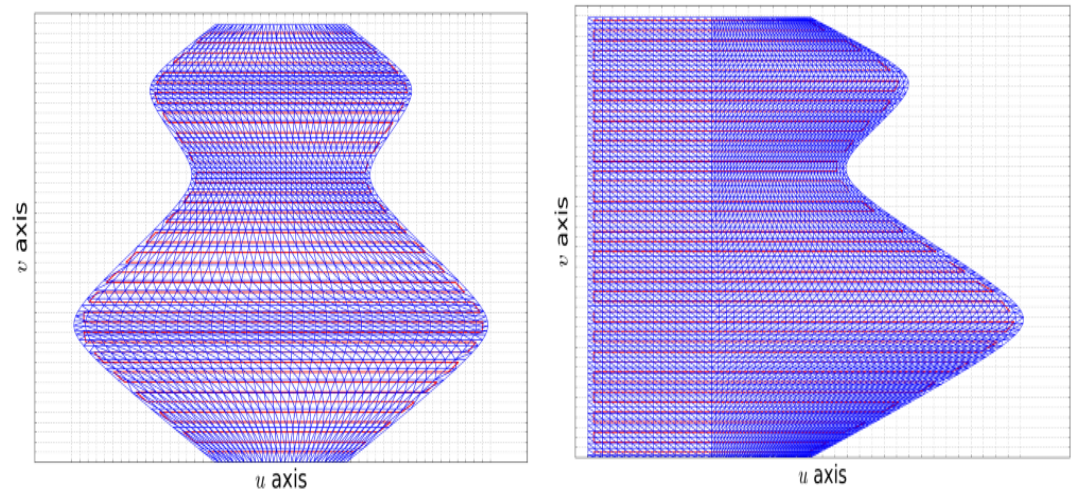
In our approach, we flatten the u coordinate when presenting the uv space. Even if this flattening causes some distortion in the polygon mesh, we are able to maintain an approximate relative surface area. This is achieved by preserving the curve length parameters of the mesh vertices' u and v components, as computed in Section 4.2.3, on the surface.

Currently, there are no available resources or tools for automatically restructuring a polygon mesh on a surface in such a way that the vertices align neatly on the surface, as demonstrated in our surface examples. While this alignment is not strictly necessary, it simplifies the process of converting the vertices to the uv space. Therefore, it may be beneficial to develop a software library that facilitates this restructuring process, potentially



(a) uv gird on Curvy surface (SoT) with coverage pattern.

(b) uv grid on Cone surface (SoR) with coverage pattern.



(c) uv grid on Vase surface (SoR) with coverage pattern.

(d) uv grid on Corner surface (SoTR) with coverage pattern.

Figure 4.9: Generated uv grid (blue) with applied raster coverage patterns $\mathbf{UV}$ (red) on each freeform surface.

providing a more efficient and reliable solution.

Furthermore, it would be valuable to enhance the virtual environment to enable the

worker to partition an arbitrary surface into segments that correspond to specific surface definitions. Each segment could then be targeted individually for coverage feasibility checking and coverage pattern placement.

4.4 Chapter Summary

This chapter introduced a novel method for generating a uv grid on 3D freeform surfaces to facilitate continuous coverage feasibility checking (Chapter 5), the placement of uniformly spaced coverage patterns (Chapter 6), and material coverage analysis (Chapter 7). We first discuss uv parameters for 3D surfaces defined with parametric equations. Then, for 3D freeform surfaces, we introduce a novel method to convert surface polygon mesh vertices from 3D Cartesian space to the uv space. We consider three types of 3D freeform surfaces; a surface of translation (SoT), a surface of rotation (SoR), and a surface of rotation and translation (SoTR). These surface types are generated from some transformation of a planar, non self-intersecting freeform curve C. We provide an interface for a human worker to identify curve C on each surface and to determine accurate surface coordinate frame placement to enable the automatic generation of the surface uv space. The uv space is then decomposed into evenly sized cells (uv-cells) and surface points in the uv space may be mapped back to the 3D Cartesian space.

We show results for accurate C curve identification on surfaces using our virtual

environment and demonstrate surface uv grid generation on four example surfaces. Future research for uv grid generation on 3D freeform surfaces includes expanding the approach to more types of complex freeform surfaces and including methods of human interaction to divide an arbitrary freeform surface into those well-defined types.

Chapter 5

Efficient Coverage Feasibility Checking on 3D Surfaces

In this chapter, we describe our approach to address the fundamental question: given a 3D surface patch and a robotic manipulator, does there exist a feasible path for constrained coverage or not? Note that feasibility refers to the manipulator coverage path satisfying continuously both task and manipulator constraints, which depends on the structure and dimensions of the manipulator links, the types of joints, and the joint limits. Violation of manipulator constraints causes either no solution for inverse kinematics or singularity configurations that prevent the end-effector to move smoothly along the surface. Hence, our approach checks for the existence of a feasible path in the manipulator's joint space. Furthermore, searching for a feasible coverage path in the joint space eliminates the need to perform inverse kinematic checks which can be computationally expensive on redundant

manipulators.

In section 5.1, we derive the joint-space task constraints imposed on the end-effector by parametric and freeform surfaces. Then we introduce our coverage feasibility search in section 5.2. We provide continuity graph results in 5.3 and conclusions in 5.4.

5.1 Deriving Joint Space Task Constraints

In this section, we derive surface task constraints imposed on the manipulator's end-effector for the given application. General surface coverage task requirements are constraints on the end-effector offset from the surface and a limit to the approach angle difference from the surface normal. The task constraints on the end-effector are converted from the end-effector coordinate frame to the joint space (by substituting in joint variables through forward kinematics) to facilitate coverage feasibility checking. We first derive task constraints for 3D surfaces that may be approximated using parametric equations such as spherical, cylindrical, or hyperboloid approximations. We then derive task constraints on 3D freeform surfaces where the end-effector orientation and position are coupled and constrained to a point on the surface.

5.1.1 Assumptions, Notations, and Constraint formulations

We attach a coordinate frame to a target spatial surface S with reference position $[x_c, y_c, z_c]^T$ and orientation R_c, where R_c is the rotation matrix of the surface frame with respect to the world frame. S can be either (a) a discretized freeform surface with indices u and v or

(b) a piece-wise parametric surface with parameters u and v with respect to the surface frame. For (b), it means that S can have a single parametric representation or different parametric representations for its different pieces connected together. Without losing generality, we simply denote every point (x_s, y_s, z_s) on S as a function of (bounded) u and v: $[x_s, y_s, z_s]^T = \mathbf{g}(u, v)$.

The robot end-effector has to satisfy a *position task constraint*, such that every end-effector position $[x_e, y_e, z_e]^T$, also denoted with $\mathbf{p}$, projects to a surface point on S, $\mathbf{p}_s$ and maintains a fixed distance d from the surface; thus the end-effector position is also a function of u and v:

$$[x_e, y_e, z_e]^T = c_p(u, v) \tag{5.1}$$

The end-effector also needs to satisfy an *orientation task constraint*:

$$R_e = c_o(u, v) \tag{5.2}$$

where R_e is the rotation matrix of the end-effector. A common orientation constraint is that the end-effector (or the tool mounted to the end-effector) approaches the surface S in a certain direction (e.g., the normal direction).

Denote the end-effector approach vector as the unit approach vector $\mathbf{a}$ (often along the z axis of the end-effector) and the desired approach direction to the surface as $\mathbf{b}(u, v)$. Then the equation (5.2) can be expressed more specifically as:

$$\mathbf{a} \cdot \mathbf{b}(u, v) = 0 \tag{5.3}$$

The orientation constraint can be further relaxed a bit to allow $\mathbf{a}$ to deviate from $\mathbf{b}$ within a small angle α at each position. Thus, we have the following inequality constraint:

$$\mathbf{a} \cdot \mathbf{b}(u, v) \leq \cos \alpha \tag{5.4}$$

We denote E as the constrained position manifold for the end-effector to cover the surface S while satisfying position and orientation task constraints. S can be discretized into a discrete (surface) grid with u and v being indices for the cells on the grid. Each cell's center is positioned at $\mathbf{g}(u, v) = [x_s, y_s, z_s]^T$ with respect to the surface frame. We further discretize S to form a grid with indices u and v. The resolution of the grid can be depending on the task constraints. After discretization, we call each cell on S a uv-*cell*.

Given an n-dimensional robotic manipulator, we denote its link parameters as $\mathbf{l} = [l_1, l_2, ..., l_n]^T$, a joint space configuration as $\mathbf{q} = [q_1, q_2, ..q_n]^T$, and joint limits as $q_{min,i} \leq q_i \leq q_{max,i}$, $1 \leq i \leq n$. A sequence of joint-space configurations can be noted as $Q = \{\mathbf{q}_1, \mathbf{q}_2.., \mathbf{q}_m\}$.

With forward kinematics, the end-effector position and orientation can be computed as functions of link parameters and joint variables, expressed in the manipulator homogeneous transformation matrix from the end-effector to the base ${}^0_nT$. Now, by substituting end-effector positions and orientations with their expressions $f_p(\mathbf{l}, \mathbf{q})$ and $f_o(\mathbf{l}, \mathbf{q})$ respectively in terms of link parameters and joint variables into equations (5.1), (5.2), and inequality (5.4), we can obtain *joint-space task constraints* that directly relate joint variables to task

parameters:

$$f_p(\mathbf{l}, \mathbf{q}) = c_p(u, v), \tag{5.5}$$

and

$$f_o(\mathbf{l}, \mathbf{q}) = c_o(u, v), \tag{5.6}$$

or a more relaxed orientation constraint:

$$f_a(\mathbf{l}, \mathbf{q}) \leq \cos\alpha. \tag{5.7}$$

We derive the joint-space task constraints for each surface by first defining the task constraints for the end-effector in Cartesian space. To obtain equations (5.22)-(5.7), we substitute end-effector position $[x_e, y_e, z_e]^T$, as defined in equation (5.1), by their functions of joint parameters and variables. For example, for the PUMA manipulator we substitute in the forward kinematic transformation matrix:

$$\begin{aligned} x_e &= c_1(a_2c_2 + a_3c_{23} - d_4s_{23}) - d_3s_1 \\ y_e &= s_1(a_2c_2 + a_3c_{23} - d_4s_{23}) + d_3c_1 \\ z_e &= -a_3s_{23} - a_2s_2 - d_4c_{23} \end{aligned} \tag{5.8}$$

where $s_{123} = sin(\theta_1 + \theta_2 + \theta_3)$, $c_{123} = cos(\theta_1 + \theta_2 + \theta_3)$.

5.1.2 Joint Space Task Constraints on 3D Parametric Surfaces

In the following, we show the derivation of joint space task constraints for three surfaces: spherical, cylindrical, and hyperbolic, whose parametric equations are shown in Table 4.1.

Spherical surface

For the spherical surface, the position task constraint, as presented in equation (5.1), can be found by subtracting the radius r by distance d, which is the end-effector distance from the surface:

$$(x_e - x_c)^2 + (y_e - y_c)^2 + (z_e - z_c)^2 = (r - d)^2 \tag{5.9}$$

The corresponding joint-space task constraint (5.22) can be expressed by replacing x_e, y_e, z_e with their joint functions in equation (5.8).

The orientation task constraint of inequality (5.7) can be expressed in terms of the dot product of the unit vector along the end-effector z axis $\mathbf{a}$ and the normal vector $\mathbf{b}$ at each position on the constrained manifold E, such that:

$$\mathbf{a} = [r_{13}, r_{23}, r_{33}]^T \tag{5.10}$$

where r_{**} is from the rotation matrix of the manipulator transformation matrix, and:

$$\begin{bmatrix} x_e - x_c \\ y_e - y_c \\ z_e - z_c \end{bmatrix} = \vec{b}_s, \quad \mathbf{b} = \frac{\vec{b}_s}{\|\vec{b}_s\|}, \quad \mathbf{a} \cdot \mathbf{b} \leq cos\alpha \tag{5.11}$$

Cylindrical surface

For the cylindrical surface, the position task constraint is as follows:

$$(x_e - x_c)^2 + (y_e - y_c)^2 = (r - d)^2 \tag{5.12}$$

The corresponding joint-space task constraint (5.22) can be expressed by replacing x_e, y_e, z_e with their joint functions in equation (5.8).

The *orientation task constraint* of inequality (5.7) can be expressed in terms of the dot product of the unit vector along the end-effector a axis $\mathbf{a}$ and the normal vector $\mathbf{b}$ at each end-effector position on the constrained position manifold E such that $\mathbf{a}$ is as defined in equation (5.18) and:

$$\begin{bmatrix} x_e - x_c \\ y_e - x_c \\ 0 \end{bmatrix} = \vec{b}_s, \quad \mathbf{b} = \frac{\vec{b}_s}{\|\vec{b}_s\|}, \quad \mathbf{a} \cdot \mathbf{b} \leq cos\alpha \tag{5.13}$$

Hyperbolic surface

For the hyperbolic surface, the end-effector position task constraint is shown in equation (5.14). This is similar to the position task constraint for the cylinder, except that the radius, r, is a function of coordinate z of the surface frame:

$$(x_e - x_c)^2 + (y_e - y_c)^2 = [r(z) - d]^2. \tag{5.14}$$

The orientation task constraint of inequality (5.7) can be expressed in terms of the dot product of the unit vector along the end-effector z axis $\mathbf{z}$ and the unit normal vector $\mathbf{b}$ at each end-effector position on the constrained position manifold E such that $\mathbf{a}$ is as defined in equation (5.18) and:

$$x = x_e - x_c, y = y_e - y_c, z = z_e - z_c \tag{5.15}$$

so that $[x, y, z]^T$ indicates the end-effector position expressed in the surface frame. The unit normal vector $\mathbf{b}$ can be expressed in the following:

$$\mathbf{b} = \frac{\mathbf{c_1} \times \mathbf{c_2}}{\sqrt{\|\mathbf{c_1}\|^2 \|\mathbf{c_2}\|^2 - \|\mathbf{c}_1 \cdot \mathbf{c}_2\|^2}}, \quad \mathbf{a} \cdot \mathbf{b} \leq cos\alpha \tag{5.16}$$

where

$$\mathbf{c_1} = \begin{bmatrix} -y \\ x \\ 0 \end{bmatrix}, \mathbf{c_2} = \begin{bmatrix} \frac{\partial r(z)}{\partial z} \frac{x}{r(z)} \\ \frac{\partial r(z)}{\partial z} \frac{y}{r(z)} \\ 1 \end{bmatrix} \tag{5.17}$$

5.1.3 Joint-Space Task Constraints on 3D Freeform Surfaces

In this section, we introduce a general position and orientation constraint formulation imposed on the end-effector. In general, the end-effector's position and orientation task

constraints are related, in that the distance constraint from the end-effector position to the surface S should be measured along the end-effector approach vector, which is determined by the end-effector's orientation task constraint. In [169], the end-effector's position and orientation constraints were introduced independently, which prohibited deviations of the approach vector from the optimal surface normal in order to satisfy the end-effector position constraint (to maintain constant offset) with respect to the surface. As a result, it was difficult to handle transitions around sharp edges and vertices (see Fig. 5.1(a)).

In order to achieve smoother transitions around edges and vertices of a 3D freeform surface (in polygonal mesh) during surface traversal (see Fig. 5.1(b)), we consider inter-related position and orientation task constraints. We denote the end-effector position in Cartesian space with respect to the surface coordinate frame as **p** and the end-effector approach vector as the unit vector **a** (along the z axis of the end-effector), such that:

$$\mathbf{a} = [r_{13}, r_{23}, r_{33}]^T, \tag{5.18}$$

where r_{**} is from the rotation matrix R of the manipulator end-effector. A task constraint is imposed on the end-effector such that its position must be offset from the surface at a distance d along the approach vector $\mathbf{a}$, satisfying

$$\mathbf{p} + d \cdot \mathbf{a} = \mathbf{p}', \tag{5.19}$$

where $\mathbf{p}'$ is the position of a point on the surface S. The approach vector $\mathbf{a}$ must also

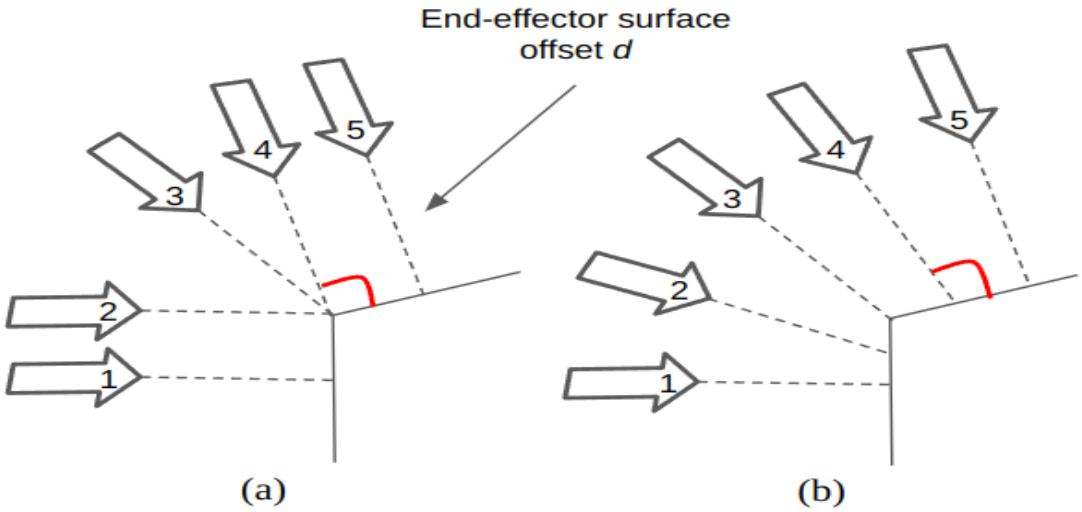


Figure 5.1: Illustration to compare the effects of (a) position and orientation constraints defined independently, which causes large orientation shift around surface edges, and (b) position constraint defined along the approach vector $\mathbf{a}$, i.e., related to the orientation constraint, which allows for smooth orientation change around surface edges.

satisfy the orientation constraint:

$$-\mathbf{b} \cdot \mathbf{a} \leq \cos\alpha, \tag{5.20}$$

where $0 \leq \alpha \leq \epsilon$ is the allowable approach angle deviation from the surface normal, $\mathbf{b}$.

Let $\mathbf{p}'$ be on the uv-cell $[u_j, v_k]$, which is on a triangle $\mathcal{T}$ of the surface with vertex positions τ_1, τ_2, and τ_3. Using the task constraint of Eq. (5.19), we have

$$\mathbf{p} + d \cdot \mathbf{a} = \Sigma_i w_i \tau_i, \;\; 0 \leq w_i \leq 1, \;\; \Sigma_i w_i = 1, \tag{5.21}$$

which expresses the task constraint on the corresponding end-effector position $\mathbf{p}$ in terms of $\tau_\mathbf{i}$'s.

5.2 From Feasible J-cell Transitions to uv-Cell Neighboring Continuity

In this section, we discuss our mapping from the joint-space to the uv space. We then introduce our algorithms for continuity checking in the J-manifold between neighboring uv-cells and the tree search through uv-cells to determine surface coverage continuity.

5.2.1 Mapping from Joint Space to uv Space

With end-effector position $\mathbf{p}$ expressed in terms of the triangle parameters of the uv-cell $[u_j, v_k]$ and the angle α as in equations (5.19)-(5.21), we can relate an n-dimensional robotic manipulator's link parameters, $\mathbf{l}$, and joint variables, $\mathbf{q}$, directly to the task constraint equations using forward kinematics to obtain *joint space task constraints*:

$$f(\mathbf{l}, \mathbf{q}) = c(u_j, v_k, \mathcal{T}, \alpha). \tag{5.22}$$

The manipulator joint configurations that satisfy the joint space task constraints Eq. (5.22) form what we call a *J-manifold*. We can discretize the J-manifold into an n-dimensional grid, where each cell, called a *J-cell*, corresponds to a joint configuration $\mathbf{q}$. The distance between two neighboring J-cell joint configurations is $\mathbf{dq}$ such that each joint variable's value q_i in $\mathbf{q}$ is increased or decreased by δq or remains the same to reach its neighboring J-cell. Thus, a given J-cell has $3^n - 1$ neighboring J-cells. Through forward

kinematics a J-cell corresponds to a feasible end-effector configuration, an $E\text{-}cell$ with position $\mathbf{p}$ and orientation R. We denote the space of all E-cells as the $E\text{-}manifold$.

An E-cell with position $\mathbf{p}$ maps to a point on the surface with position $\mathbf{p}'$, by Eq. (5.19), and $\mathbf{p}'$ can be converted to uv space (see Section III). Note that multiple E-cells with different $\mathbf{a}$ (approach) vectors may map to the same surface position $\mathbf{p}'$ or the same uv coordinates. Mappings between the J-manifold, E-manifold, and the uv grid are illustrated in Fig. 5.2.

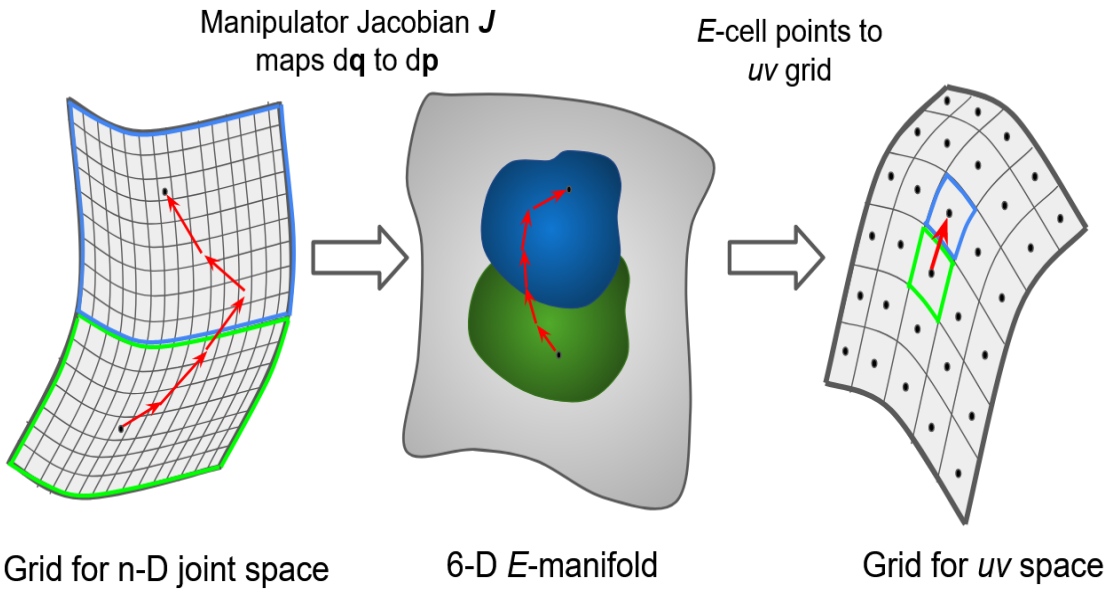


Figure 5.2: Illustration showing the conversion from J-manifold (left), to the E-manifold (center), and then to the uv grid (right). The corresponding regions in each manifold are shown in the same blue/green colors.

Our coverage feasibility checking algorithm in Algorithm 1 explores the uv grid, using a tree search, reaching every uv-cell once. Starting from an initial uv-cell, uv_1, which

Algorithm 1: Continuity check on uv grid

Global $Neighbors$;
Input uv_1 and corresponding jc_1;
Initialize Tree T at uv_1;
Initialize graph G containing all uv-cells with no edges;
T, G = TREESEARCH(T, G, jc_1);
if G *is a connected component* **then**
    return "$\exists$ feasible path"
else
    return "no feasible path"
end

TreeSearch$T$, G, jc:
From jc get corresponding uv-cell called uv;
$Neighbors$ = unvisited neighbor uv-cells of uv;
repeat:
Randomly generate none-zero **dq**;
call Algorithm 2 with jc, **dq**, which returns *continuity* between uv and a neighbor uv_N and the corresponding J-cell jc_N;
$Neighbors$ =$Neighbors$ - $\{uv_N\}$;
if $continuity$ **then**
    Add edge between uv and uv_N if uv_N is not in G;
    if uv_N *is not in* T **then**
        Add uv_N as child to uv in T;
        T, G = TREESEARCH(T, G, jc_N)
    end
end
until $Neighbors = \emptyset$;
return T, G;

inversely maps to a J-cell, jc_1, the algorithm stores neighbor uv-cells in $Neighbors$ and randomly generates a none-zero **dq** by randomly assigning a value from $\{-\delta q, 0, \delta q\}$ for each joint. It calls Algorithm 2 to search the J-manifold moving through neighboring J-cells until a corresponding E-cell is reached, which maps inside a neighboring uv-cell (i.e. the uv-cell with its center closest to the corresponding uv point of that E-cell), and establish that there is a *neighboring feasibility continuity* of the manipulator to cover the

Algorithm 2: Continuity check on J-manifold between neighboring uv-cells

CONTjc, **dq**:
repeat:
From **dq** find the neighbor jc' of jc;
if jc' *satisfies joint-space task constraints and joint limits* **then**
  E-cell ec is obtained from jc via forward kinematics;
  dx = J(q) $\cdot$ **dq**;
  From ec and **dx**, obtain next E-cell ec_N;
  By mapping ec and ec_N to the uv space, get the direction from uv to a neighbor uv_N;
  if $uv_N \in$ *Neighbors* **then**
    if uv_N *is reached* **then**
      $continuity = true$; $jc_N = jc'$;
      return $continuity$, uv_N, *and* jc_N;
    end
    call *CONT(*jc'*,* ***dq****)*
  end
end
Adjust **dq** and find an unchecked neighbor J-cell jc';
until there is no valid J-cell transition from jc;
$continuity = false$;
return $continuity$, uv_N, jc_N;

two uv-cells.

We make the Algorithm 2's joint space search more efficient by allowing the search to continue in a direction, **dq**, instead of a random direction for each J-cell transition, thus narrowing the search space. If following a direction **dq**, the search reaches a joint configuration **q** that satisfies joint space task constraints, then **dq** is used again until the constraints are no longer satisfied. Once these constraints are no longer satisfied, **dq** is adjusted using the Jacobian matrix **J(q)** to identify the order of joint variable q_i which least affects the resulting change **dx** in the end-effector configuration. Starting with the least significant joint variable, the adjustment procedure generates a new value that is not

previously used for that joint in **dq**, trying to obtain a new **dq** that is close to the previous **dq** in the hope of satisfying joint space task constraints with as little deviation from the original search direction as possible.

This process continues in Algorithm 1 until every reachable uv-cell pair continuity is checked via a tree search and a uv space connectivity graph, G, is built based on neighboring feasibility continuity results. If the resulting graph is a single connected component, we determine that there is at least one end-effector path that can cover the surface S continuously while maintaining constraints.

The worst case time complexity of the uv grid search depends on the worst case time complexity of the feasibility search between two adjacent uv-cells, which is essentially a depth-first search (DFS) in the joint space. The worst case time complexity of the DFS is $O(b^m)$, where b is the maximum branching factor and m is the maximum depth of search. $b \leq 3^{n\breve{}}1$, where n is the degrees of freedom of the manipulator. Usually, $b << 3^{n\breve{}}1$ because (1) not all joints move during the transition from one uv-cell to an adjacent one, and (2) many joint configurations cause a violation of constraints. m depends on the joint limits and the value of δq. Essentially, if δq is used to discretize the high-dimensional joint space into a hyper-grid, m is bounded by the number of J-cells. Thus, the worst-case time complexity of the uv grid search algorithm is $O(N \cdot b^m)$, where N is the number of uv-cells. The actual running time for a feasibility check between two adjacent uv-cells takes just a few milliseconds.

5.3 Experiments and Analyses

To test our surface coverage feasibility algorithm as introduced in [170] we use a PUMA 560 robotic manipulator to determine feasibility for autonomous constrained manipulation coverage. We considered spray coating as an example constrained coverage problem. For this problem, the end-effector position task constraint is to keep the end-effector a constant distance d from the target surface as it traverses the surface, and the end-effector orientation task constraint is to maintain its approach vector within a small angle α from the normal of the surface.

The PUMA robot contains six revolute joints, with the joint vector $\mathbf{q} = [\theta_1, \theta_2, ..., \theta_6]^T$ and link parameters $\mathbf{l} = [a_2, a_3, d_3, d_4] = [0.7, 0.1, 0.1, 0.7]$(m). The manipulator transformation matrix ${}^0_6T$, see [178], and joint limits are used to obtain $f_p(\mathbf{l}, \mathbf{q})$, $f_o(\mathbf{l}, \mathbf{q})$, and $f_a(\mathbf{l}, \mathbf{q})$ in the joint-space task constraint equations (5.22)-(5.7). We considered several sets of joint limits for the PUMA, see Table 5.1, and different task constraint parameter values for each surface, see Table 5.2.

We use various surface patches which include a spherical, a cylindrical, and a hyperbolic surface patch. The parametric representation of each surface, as shown in Table 4.1, is expressed with respect to its origin $[x_c, y_c, z_c]^T$. For each surface frame, its origin is at the center (of symmetry) of the surface, and its orientation is aligned with the robot base frame.

We first test our coverage feasibility algorithm on parametric surfaces using different

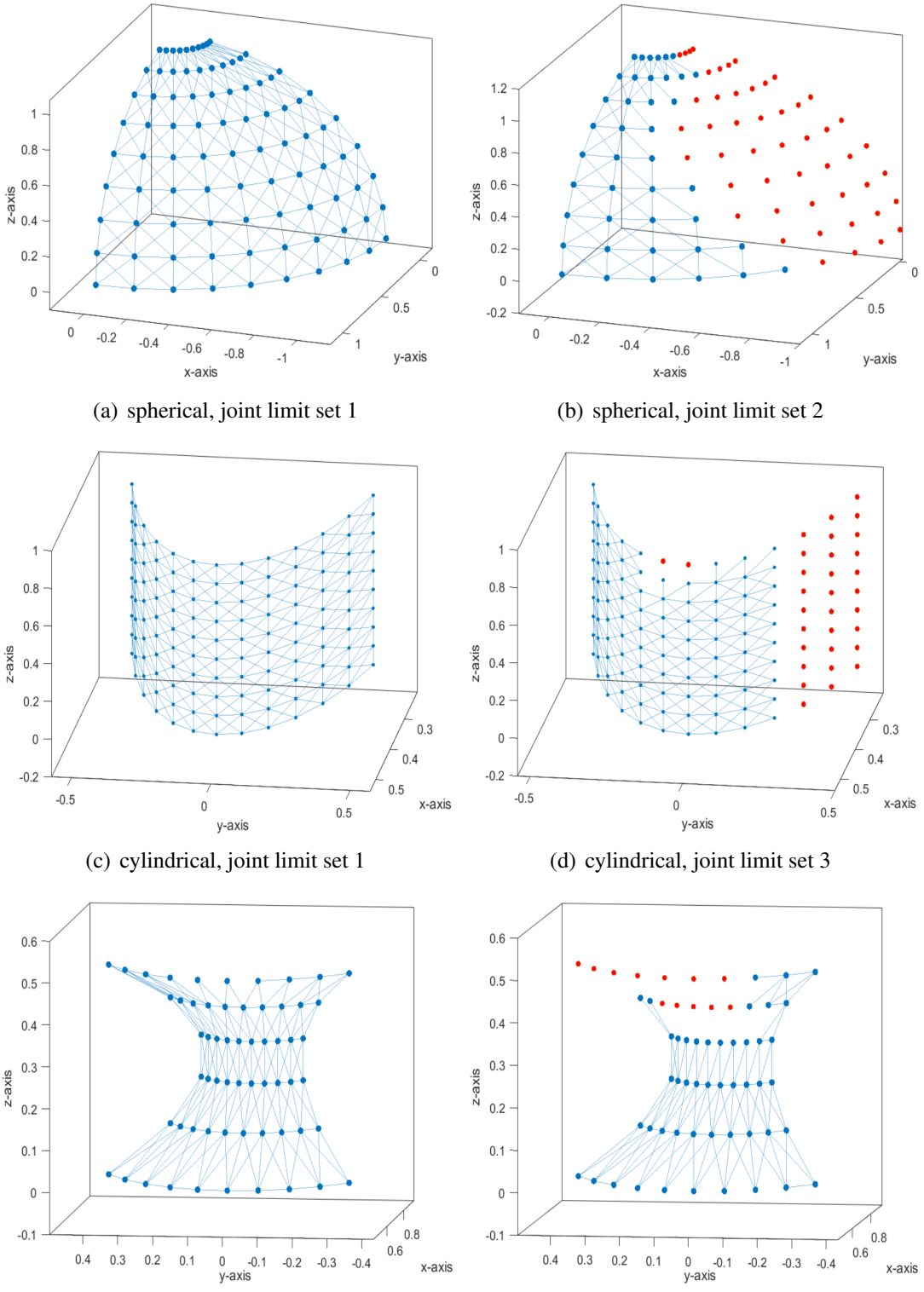
sets of joint limits and then different sets of task parameter values for analysis and comparison of the resulting continuity graphs. We then provide performance data of the continuity check between uv-cells.

5.3.1 Results with different sets of joint limits

We experiment with what the effects of different joint limits would have on the coverage continuity graph G on different surfaces. The different joint limit sets tested can be viewed in Table 5.1. These tests were performed while keeping the task parameter values the same as the parameter set 1 in Table 5.2.

Table 5.1: Joint Limits Sets for PUMA.

Set 1	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-180°$	$-180°$	$-180°$	$-180°$	$-180°$	$-180°$
θ_{max}	$180°$	$180°$	$180°$	$180°$	$180°$	$180°$
Set 2	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-170°$	$-170°$	$-180°$	$-135°$	$-100°$	$-90°$
θ_{max}	$170°$	$75°$	$175°$	$190°$	$90°$	$90°$
Set 3	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-90°$	$-170°$	$-180°$	$-135°$	$-100°$	$-90°$
θ_{max}	$10°$	$75°$	$90°$	$170°$	$90°$	$90°$
Set 4	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{min}	$-110°$	$-110°$	$-110°$	$-150°$	$-45°$	$-180°$
θ_{max}	$145°$	$75°$	$40°$	$120°$	$140°$	$180°$



(a) spherical, joint limit set 1

(b) spherical, joint limit set 2

(c) cylindrical, joint limit set 1

(d) cylindrical, joint limit set 3

(e) hyperbolic, joint limit set 1

(f) hyperbolic, joint limit set 4

Figure 5.3: Graph G on different surface patches using task parameters value set 1 (in Table 5.2) and different PUMA joint limit sets (in Table 5.1)

Figure 5.3 illustrates graph G on the surface for the spherical, cylindrical, and hyperbolic surfaces using different joint limit sets in Table 5.1. For each surface, when joint limit set 1 is applied, the graph G is a connected component, as shown in the left figure. However, when joint limit set 2 is applied, the graph G does not have a connected component that covers all uv-cells, as shown in the right figure, i.e., there is no feasible motion path to traverse the entire S in those cases. Note that joint limit set 1 applies joint limits which are more relaxed than the other joint limit sets.

5.3.2 Results with different task parameter values

We also experiment with different task parameter values which can affect surface continuity graph G. Figure 5.4 illustrates graphs G on the surface for the spherical, cylindrical, and hyperbolic surfaces using joint limit set 1 (in Table 5.1) and task parameter value set 2 then set 3 respectively (in Table 5.2).

Using joint limit set 1 with the spherical surface, which previously resulted in a single connected component in G (see Fig. 5.3(a)), Figure 5.4 (a) and (b) show that decreasing α can greatly reduce the size of the multi-cell connected component in G. These figures also show that when the sphere center is moved a very small distance along the x-axis of the robot base frame, the multi-cell connected component in G shrinks much more even though α is increased. In this case, the spatial arrangement of the surface relative to the robot seems to have a more significant effect on feasibility than the α value. This can be

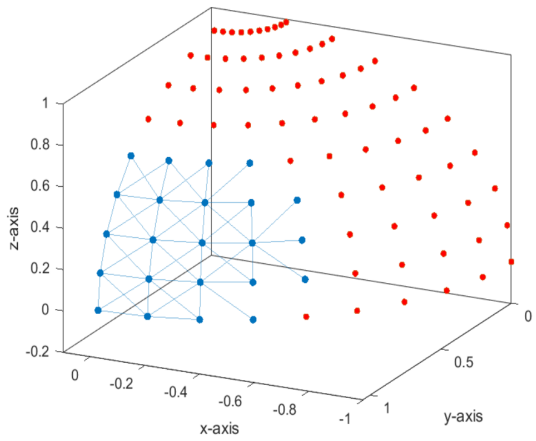
Table 5.2: Task Parameters and Values.

Set 1	x_c (m)	y_c (m)	z_c (m)	α	d (mm)
All Surfaces	0	0	0	$45°$	5
Set 2	x_c (m)	y_c (m)	z_c (m)	α	d (mm)
Sphere	0	0	0	$5°$	5
Cylinder	0	0.055	0	$5°$	5
Hyperboloid	1.095	0	0	$10°$	5
Set 3	x_c (m)	y_c (m)	z_c (m)	α	d (mm)
Sphere	0.005	0	0	$10°$	5
Cylinder	0	0.055	0	$10°$	5
Hyperboloid	1.15	0	0	$20°$	5

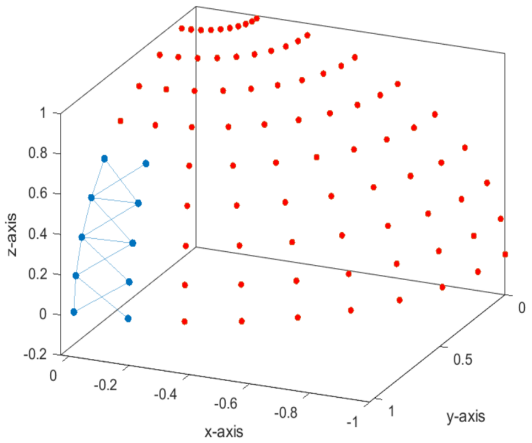
an effect of the surface getting closer to the edge of the manipulator workspace.

Similarly, using joint limit set 1 with the cylindrical surface, previously resulted in a G of a single connected component (see Fig. 5.3(c)), but in Figure 5.4 (c), moving the surface along the y-axis of the robot base frame for a large distance reduced half of the uv-cells from the (multi-cell) connected component of G. After changing α from $5°$ to $10°$, a relatively small amount, we again see a graph G with a single connected component (Fig. 5.4 (d)). In this case, the spatial arrangement of the cylinder with respect to the robot does not seem to have as much of an effect on feasibility as the value of α.

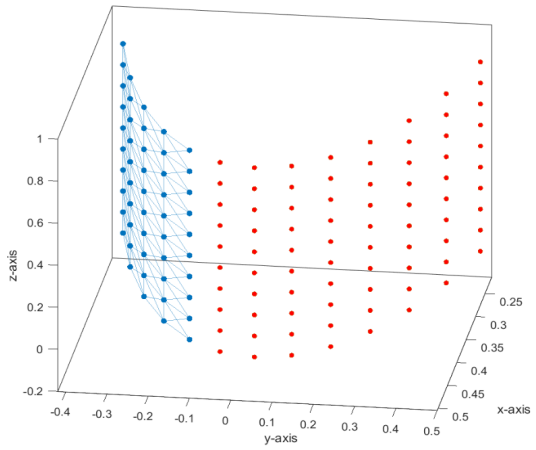
There are also scenarios where a change in parameter values can result in a loss of uv-cells and edges in one section of the connected component in G while gaining uv-cells and edges in another part of the connected component. This is visibly noticeable from changes in Figure 5.4 (e) and (f). The multi-cell connected component of G loses edges



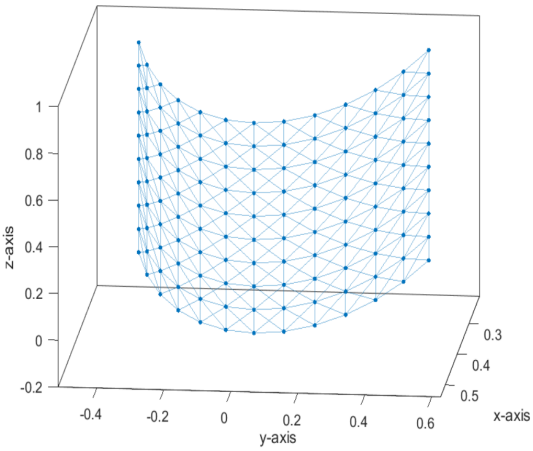
(a) spherical, α=5°



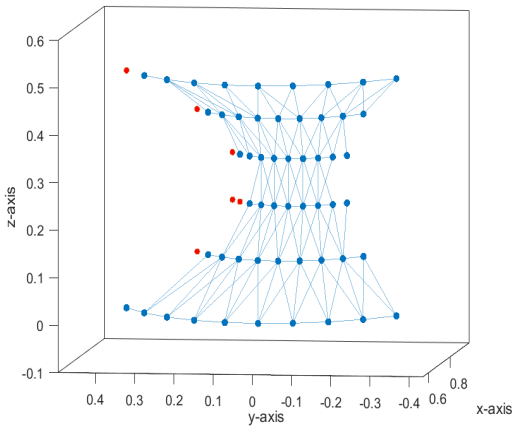
(b) spherical, α=10°, x_c=5mm



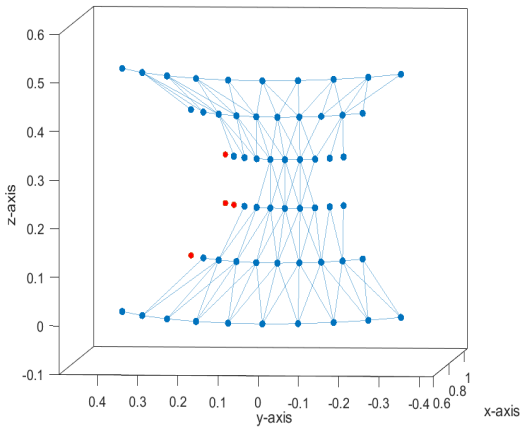
(c) cylindrical, α=5°, y_c=55mm



(d) cylindrical, α=10°, y_c=55mm



(e) hyperbolic, α=10°, x_c=1.095m



(f) hyperbolic, α=20°, x_c=1.15m

Figure 5.4: Graph G on different surface patches using joint limit set 1 and varied values of task constraint parameters.

near the center of the hyperboloid (where the curvature is greatest) as the surface moves further away from the robot base along the x-axis, and at the same time, uv-cells near the corner of the hyperboloid are added to the connected component of G.

The testing results show that the effects of different parameter values on the feasibility of continuous coverage can vary from case to case and may be inter-related, which confirms the need for a systematic approach and the utility of our method to determine whether there is a feasible path to cover a surface continuously given different manipulator and task constraints.

5.3.3 Performance data

Figure 5.5 shows an example of how the coverage feasibility algorithm searches the joint-space grid for a continuous constrained end-effector motion between two uv-cells for the spherical surface, using the set 1 joint limits. Note that θ_1 and θ_2 are the horizontal and vertical axes here since PUMA's first three joint variables θ_1, θ_2, and θ_3, exclusively define the end-effector position and θ_1 and θ_2 are most significant (θ_3 does not change in this example). The figure shows that it took 36 steps in the joint space to complete the neighboring transition in the uv space.

Fig. 5.6(a) shows the graph G results from the coverage feasibility algorithm using joint limits set 1 for the spherical surface. In this case, G is a single connected component with all nodes connected to their neighbors that cover all uv-cells on the surface. This means not only that there exists a feasible path for the robot to continuously move the end-effector

to cover all the uv-cells, but also that there are many feasible paths to do so. Fig. 5.6(b) shows the graph G using joint limits set 2 this time, resulting in one connected component (in blue) that does not include all uv-cells on the surface and single-cell components from the remaining uv-cells (in red). The red dots show uv-cells that cannot be visited by the robot end-effector in continuous constrained motion. This means that there is no feasible continuous motion path to cover all uv-cells. Note how some uv-cells in the (blue) connected component in Fig. 5.6(b) cannot reach some of their neighboring uv-cells. This shows that there are limited directions of feasible motion to reach those uv-cells, which is informative to constrained path planning.

Each test case had a runtime of building graph G well below 1 second. For example, the runtime of building the graph G for the spherical surface with the joint limit set 1 and task parameter set 1 was 540 ms. Hence, the average joint-space search time between two neighboring uv-cells is less than 2 ms. This example represents the worst-case scenario where there is a single connected component in G with the maximum number of edges. For cases where G contains multiple components (such as in Fig. 5.6(b)), i.e., there is no feasible solution to cover the entire surface, the runtime is much lower still. The runtime of the coverage feasibility algorithm will increase as the resolution of the uv grid increases, but the average joint-space search time between two neighboring uv-cells will decrease at the same time.

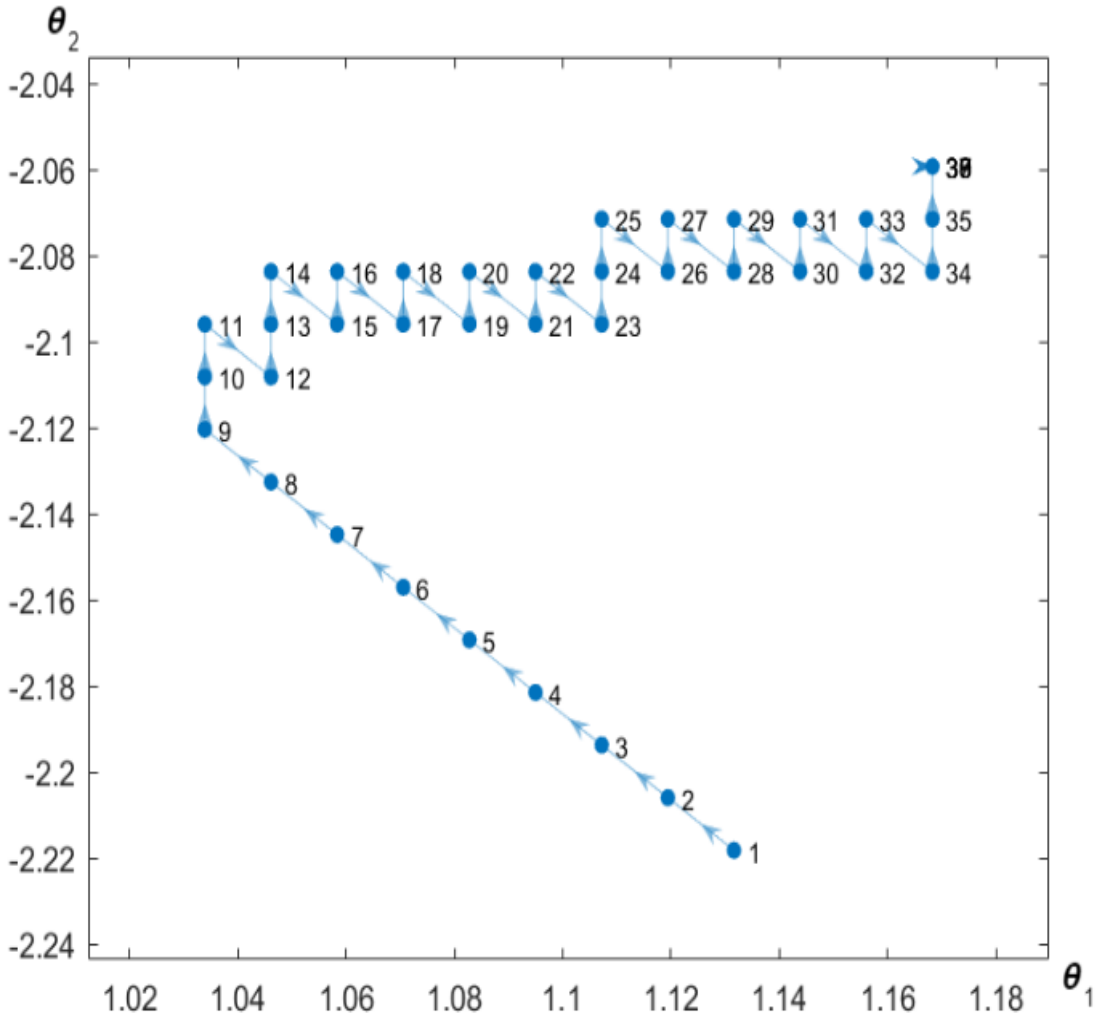
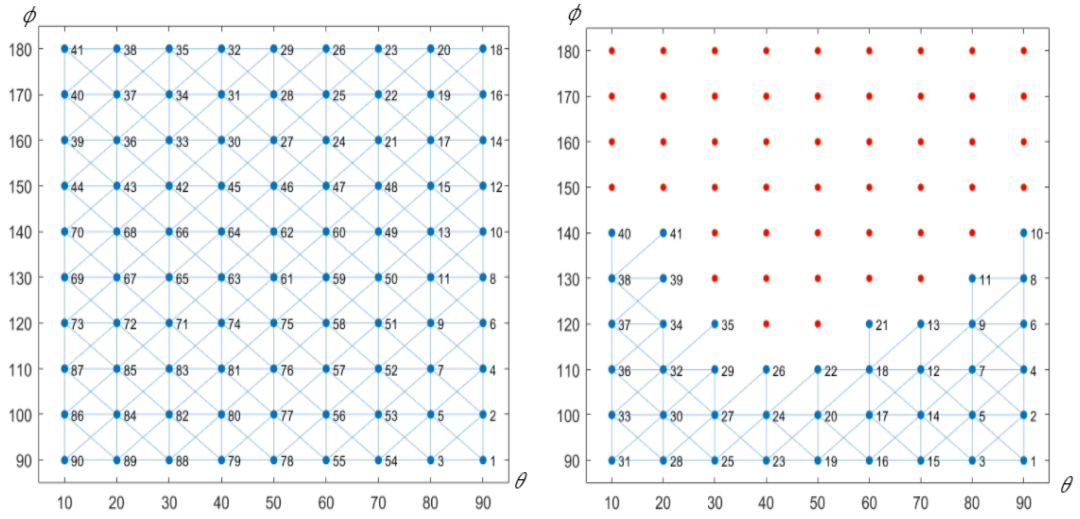


Figure 5.5: Joint-space motion continuity search between two neighboring uv-cells for the spherical surface (in degrees).



(a) based on joint limit set 1 (b) based on joint limit set 2

Figure 5.6: Graph G for the spherical surface using task parameter value set 1 and different joint limits, where red dots indicates unreachable nodes.

5.3.4 Discussion

Although many applications require continuously moving a manipulator's end-effector across a surface patch under some task constraints, the question of whether this is always possible, given a specific manipulator, target surface, and task constraints, has not been answered systematically. The coverage feasibility algorithms may be used systematically at different surface spatial arrangements to analyze surface continuity. Different manipulator kinematics may be tested to analyze suitable parameters for surface coverage.

Furthermore, the continuity graph G may inform on how the surface may be separated into surface patches that are covered individually (non continuously). This would require multiple surface coverage patterns placed on the surface and a single end-effector liftoff

from the surface where task constraints are broken to continue on a different coverage pattern on the surface. This may be performed where the surface maintains its spatial arrangement or changes position in between surface pattern coverage. Large surfaces that may not fit in the manipulator workspace would require surface segmentation and changes in spatial arrangement with the manipulator in between surface segment coverage.

There are surfaces that may present challenges for our coverage feasibility checking algorithm, such as surfaces with holes or rough surfaces. A reasonable approach to any edge case may be to approximate the surface with a more favorable surface definition (i.e. alter the polygon mesh to fill in or smooth the surface).

5.4 Chapter Summary

This Chapter introduced an efficient and systematic approach to address the fundamental question: given a manipulator, a 3D surface, and task constraints, does there exists a feasible continuous motion plan to cover the surface? We derive and convert task constraints from the Cartesian space to the joint space (for parametrically defined and freeform surfaces) and our coverage feasibility algorithm searches for feasible joint space paths directly without inverse kinematics, which can be computationally expensive, especially for redundant manipulators. Manipulator motion continuity is checked between each neighboring uv-cell on the surface and a final continuity graph G (with nodes as uv-cells and edges denoting feasible motion paths) is provided for continuous motion coverage analysis. If graph G is one connected component then we can determine that at least one

feasible coverage motion exists on the surface.

The introduced continuous coverage feasibility algorithm has been implemented and tested on example 3D surfaces involving a PUMA robot and variable task parameters. The results have shown that manipulator joint limits and the value of task constraint parameters both can greatly affect whether or not there is a feasible solution, i.e., a continuous path for the robot manipulator to cover the target surface patch under task constraints. If the answer is yes, graph G generated by applying the introduced method can further convert the problem of planning feasible manipulator paths for coverage into a much simpler graph search problem. If the answer is no, the method can help reveal the cause(s) of infeasibility, which could range from mismatched manipulator and task to a mis-positioned target surface with respect to the manipulator. If the latter is the cause, the method can be used to adjust the relative position and pose of the surface with respect to the manipulator to find the relative arrangement that results in a feasible coverage solution. Alternatively, the method can be used to determine an acceptable manipulator for a given task.

Chapter 6

Continuous Coverage Motion Planning on 3D Surfaces

In this chapter, we inform on how to generate and place coverage patterns on 3D surfaces using our uv space representation of the polygon mesh (introduced in Chapter 4). If coverage feasibility checking determines that a continuous coverage path exists on a 3D surface given the task and manipulator constraints, then the continuity graph G and the surface uv grid can be used to generate a suitable surface coverage pattern. Recall, coverage patterns such as raster scans or Archimedean spirals are generally used for constrained surface coverage applications due to their constant separation between adjacent turns, thus resulting in uniform coverage [3, 15](see Fig. 2.3). Section 6.1 discusses surface coverage pattern algorithms and how to choose suitable coverage patterns based on the surface uv grid.

Next, the surface coverage patterns are converted from the uv space to the Cartesian space using the surface polygon mesh triangles (discussed in Chapter 4). Then, we generate a manipulator end-effector coverage path to follow the surface coverage pattern. Note the surface coverage pattern is where the manipulator's end-effector approach vector should project to the surface (at a distance from the surface given the offset parameter d). Thus, an initial manipulator coverage path is generated by offsetting the surface coverage pattern given the task surface offset parameter, d, and surface normals at each point. We test the initial manipulator coverage path for singularities and smooth the end-effector orientations if singularities are detected. Section 6.2 describes the manipulator coverage path generation and provides algorithms for path singularity detection and avoidance. Section 6.3 provides results from surface coverage pattern generation and manipulator coverage paths on various surfaces. Section 6.4 is chapter conclusions.

6.1 Placing Coverage Pattern on the uv Grid and Mapping to Cartesian Space

First, we provide algorithms to generate 2D raster scan and dual spiral coverage patterns on the surface uv grids. Recall the uv grid is decomposed into evenly sized cells, uv-cells, whose size depends on task parameters such as the application plume width. Thus, spacing between adjacent turns is chosen so that the coverage pattern passes through each uv-cell once. Then, we discuss choosing a suitable coverage pattern based on the uv space representation of the polygon mesh. Factors that affect coverage pattern placement include

the symmetry and concavity of the surface uv grid.

6.1.1 Generating the uv coverage pattern

Given a finite parametric surface S, represented by $[x, y, z]^T = \mathbf{g}(u, v)$, such that (u, v) is within a boundary B, consisting of curve segments in the uv space, we need to fit a given 2D coverage pattern into the 2D region for (u, v) inside the boundary. Each uv-cell is represented by its center (u, v) coordinates. The resolution of the grid is determined based on the hard constraint of the spray nozzle plume width.

We denote **UV** as a 2D coverage pattern on the uv grid with uv_i being the ith uv point of the pattern such that $\mathbf{UV} = [uv_1, uv_2, ..., uv_M]$. Every point in the uv grid that is within a triangle face is a viable location for uv_i. 2D coverage patterns considered include vertical and horizontal raster patterns and clockwise and counter-clockwise dual spirals with different starting positions.

Algorithm 3 outlines how to create a raster scan coverage pattern in the uv space as shown in Fig. 6.1 (a) and (b). Let ω be the interval between two discrete (u, v) points and $\frac{\omega}{2}$ be the distance between viable (u, v) points and polygon mesh edges. ω can be determined based on the task constraint in an application, such as the spray width in spray painting. We can discretize the uv grid again using ω. The algorithm begins setting variables u^* and v^* to those of the top-left viable point coordinates in the uv grid. The algorithm then iterates through each v to put the coordinates of all viable points between the minimum and maximum u values for each v into the **UV** in the order of alternating right and left

Algorithm 3: Create horizontal raster UV in uv space

Input uv grid; $i = 0$; $direction = Right$;
$(u^*, v^*) = (u_{min}(v_{max}), v_{max})\ \forall\ u, v \in uv$ grid;
while $v^* \geq v_{min}$ **do**
  UV $= (u^*, v^*)$; $i++$;
  if $direction == Right$ **then**
    $direction = Left$;
    While $u^* < u_{max}(v^*)$ **do:**
      $u^* = u^* + \omega$ **; UV**$(i) = (u^*, v^*)$**;** $i++$**;**
  else
    $direction = Right$;
    While $u^* < u_{min}(v^*)$ **do:**
      $u^* = u^* - \omega$ **; UV** $= (u^*, v^*)$**;** $i++$**;**
  end
  UV $= (u^*, v^*)$; $i++$;
  $v^* = v^* - \omega$;
end

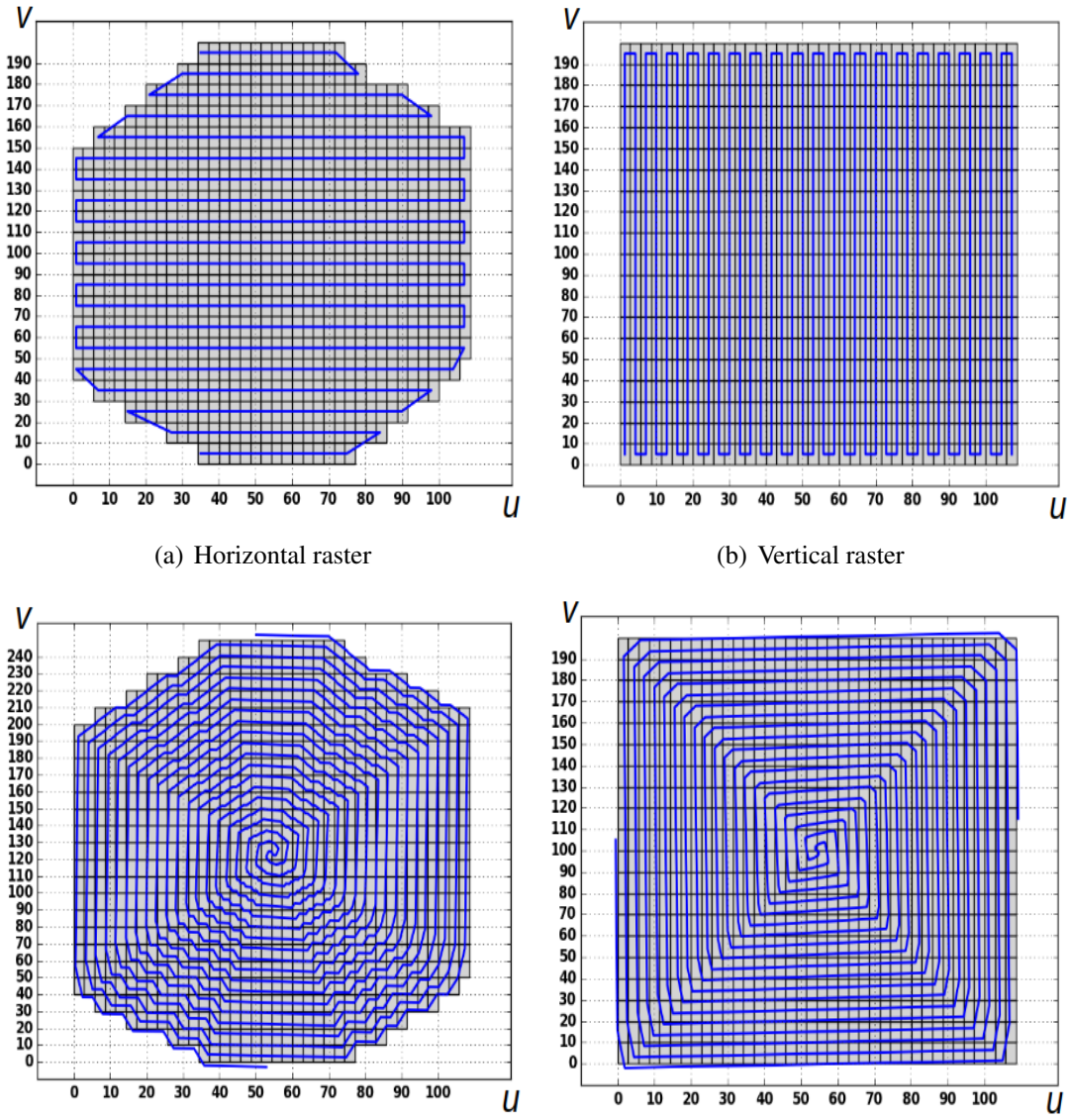
scan directions. To generate vertical raster scans, Algorithm 3 is altered by swapping u and v coordinates values, and after the algorithm finishes, swap the u and v coordinates for each waypoint in **UV**.

Algorithm 4 outlines how a single 2D spiral curve is created. This algorithm is used to create dual spiral coverage curves as shown in Fig. 6.1 (c) and (d). For both spirals in the curve, the algorithm begins by initializing Γ as the sequence of border cells on the uv grid. Next, the center cell, c, on the uv grid and the spiral outer starting cell, s, on the uv grid are defined (note when creating a dual spiral curve, both spiral starting cells should be on opposite sides of the uv grid from each other). $\{\vec{V}\}$ is then initialized as the set of vectors from c to each border cell in Γ and $\vec{V}_s$ is a single vector from c to s. We use these vectors to create the outer layer L_0 of a spiral by ordering cells of Γ in ascending order using each corresponding angle from $\vec{V}$ to $\vec{V}_s$ in the given spiral direction. To determine the number

Algorithm 4: Create 2D spiral curve

Input uv grid, spiral direction and starting cell;
col = number of columns in uv grid;
row = number of rows in uv grid
Γ = set of boundary cells in uv grid;
c, s = center and starting cell in uv grid;
$\{\vec{V}\}$ = set of vectors from c to each cell in Γ;
$\vec{V_s}$ = Vector from c to s;
Initialize outer layer:
sort cells of Γ in ascending order of the angle from $\vec{V}$ to $\vec{V_s}$ in spiral direction;
$L_0 = \Gamma$;
create and attach inner layers:
$n = max(col, row)/4$; $\zeta = 1/n$; $i = 1$;
while $i < n$ **do**
    Create inner layer L_i by scaling outer layer L_0 by $1 - (\zeta * i)$ towards c;
    increment L_{i-1} towards L_i cell by cell;
    append L_{i-1} to **UV**;
end
increment L_{n-1} towards c cell by cell;
append L_{n-1} to **UV**;

of inner layers n, the maximum number of columns or rows in the uv grid is divided by 4 (since a single layer will cover 4 cells along each row or column). The layers are created by iterating $n - 1$ times, scaling L_0 down by a decrementing factor each time. Each layer L_i, except for the last layer, is grown cell by cell towards the next inner layer. The final layer L_{n-1} is grown cell by cell towards c. Finally, the end of each layer is attached to the beginning of the next inner layer to create a single spiral sequence. Once a second spiral sequence is created, the end of the second spiral is attached to the end of the first spiral to create a dual-spiral coverage pattern.



(a) Horizontal raster

(b) Vertical raster

(c) Clockwise dual spiral

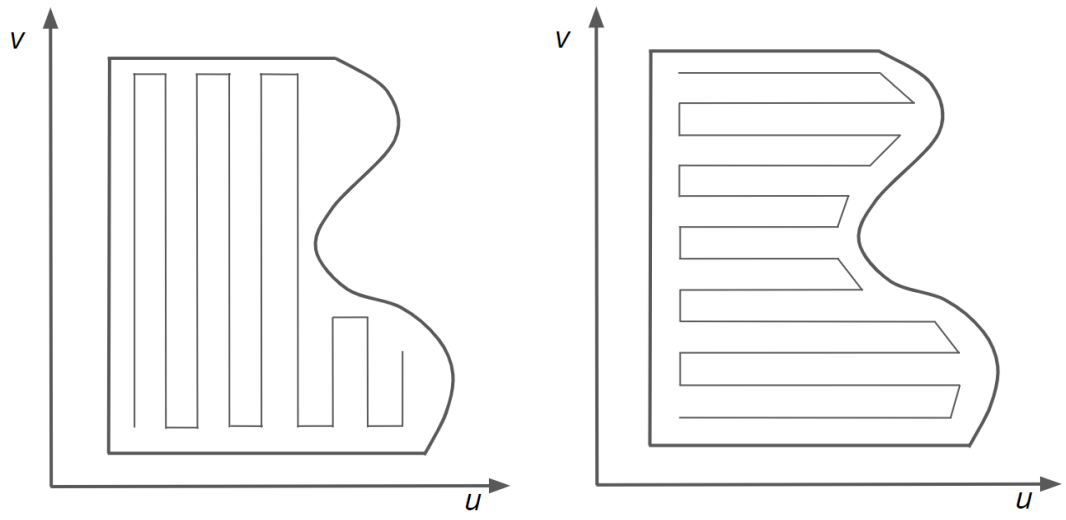
(d) Counter-clockwise dual spiral

Figure 6.1: 2D coverage patterns over uv grid.

6.1.2 Choosing a Coverage Pattern

The uv grid makes it easier to detect if a coverage pattern is suitable than viewing the surface in Cartesian space. The convexity of a flattened (planar) uv space can be used to

decide whether a coverage pattern can be applied, since for some concave planar regions, certain coverage patterns cannot provide uninterrupted coverage for the entire surface (see Fig. 6.2). If no coverage pattern can provide uninterrupted surface coverage, the surface can be decomposed into smaller and preferably convex regions [179, 180] so that each smaller region can be covered by a desired coverage pattern. Alternatively, other 2D CPP methods [5, 18, 20–22] could be adapted to design a coverage pattern in the uv space.

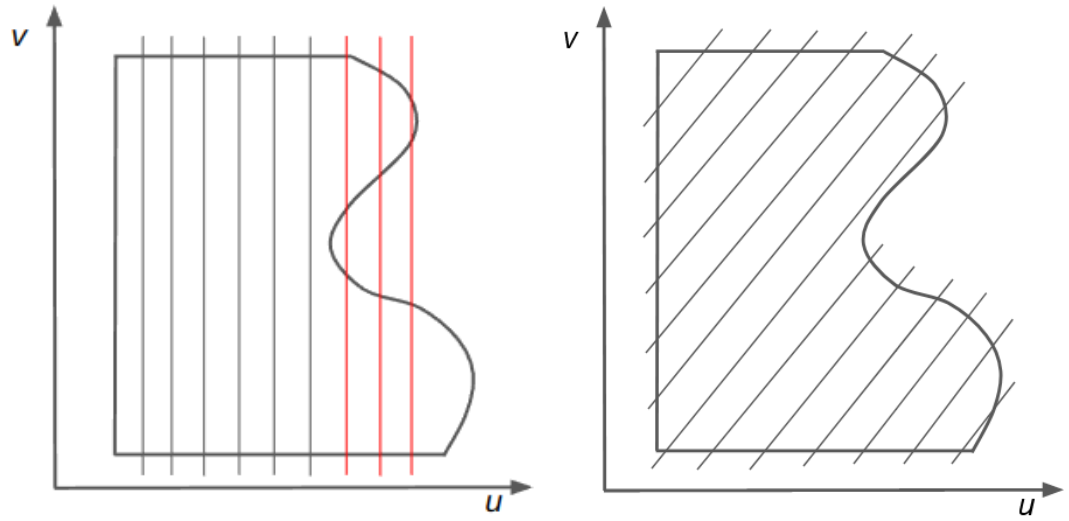


(a) Vertical raster scan pattern cannot cover the entire uv space.

(b) Horizontal raster scan pattern can cover the entire uv space.

Figure 6.2: Different coverage patterns on concave surface in uv space.

A common raster pattern is a set of parallel, straight “scan" lines that are separated perpendicularly by a distance, ω, and connected at their ends in an alternating manner. We denote the direction of the lines as the *start direction*. In general, we can determine if a start direction will result in a raster coverage pattern that covers the surface uninterrupted by using a systematic check on the scan lines. If each scan line intersects the surface



(a) Start direction failed check on raster scan lines (red). (b) Start direction passes check on raster scan lines.

Figure 6.3: Illustration showing examples of suitable and non-suitable raster scan start direction for uv grid surface.

boundaries only twice, then that start direction is suitable for the uv grid surface (see Fig. 6.3). If not, the start direction is rotated to a new direction, and the test is repeated. The lines are shortened to fit within the surface boundaries with endpoints connected in an alternating manner.

6.2 Coverage Path Singularity Detection and Avoidance

Given that a feasible coverage path exists (from feasibility checking) and a coverage pattern, our system next generates the end-effector's coverage path that satisfies the position and orientation constraints. The initial end-effector position can be determined by applying

an offset d from the surface (as required by the position task constraint), and the initial end-effector orientation can be determined as the optimal tool angle to surface (usually the surface normal, as required by the orientation constraint). An initial end-effector path can be created in terms of the sequence of end-effector positions and orientations corresponding to the centers of uv-cells along the coverage pattern. However, such an initial path may not be feasible, and our system finds a feasible path by detecting singularities along the initial path and modifying the end-effector configurations mainly by altering the end-effector orientations to avoid singularities.

6.2.1 Generating initial end-effector path

We denote **H** as the coverage pattern **UV** expressed in Cartesian coordinates, with $\mathbf{h}_i$ being the position of uv_i on the pattern such that $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_M]$. Every point in **UV** is confined by a polygon triangle in the uv space and can be converted to Cartesian space using the corresponding triangle vertice Cartesian components (see Fig. 6.4). Thus, we convert each point in **UV** from uv space to Cartesian space using the surface point, $\mathbf{p}_s$, conversion as described in Chapter 4.

Let $\{R_1, R_2, ..., R_M\}$ denote a sequence of end-effector orientations, such that R_i is the rotation matrix of the manipulator initialized with the end-effector's z-axis along the normal $\mathbf{b}_i$ of the triangle that contains the ith waypoint uv_i of the coverage pattern. This is to best satisfy the orientation task constraint.

We compute the end-effector's position $\mathbf{p}_i$ corresponding to each waypoint uv_i with

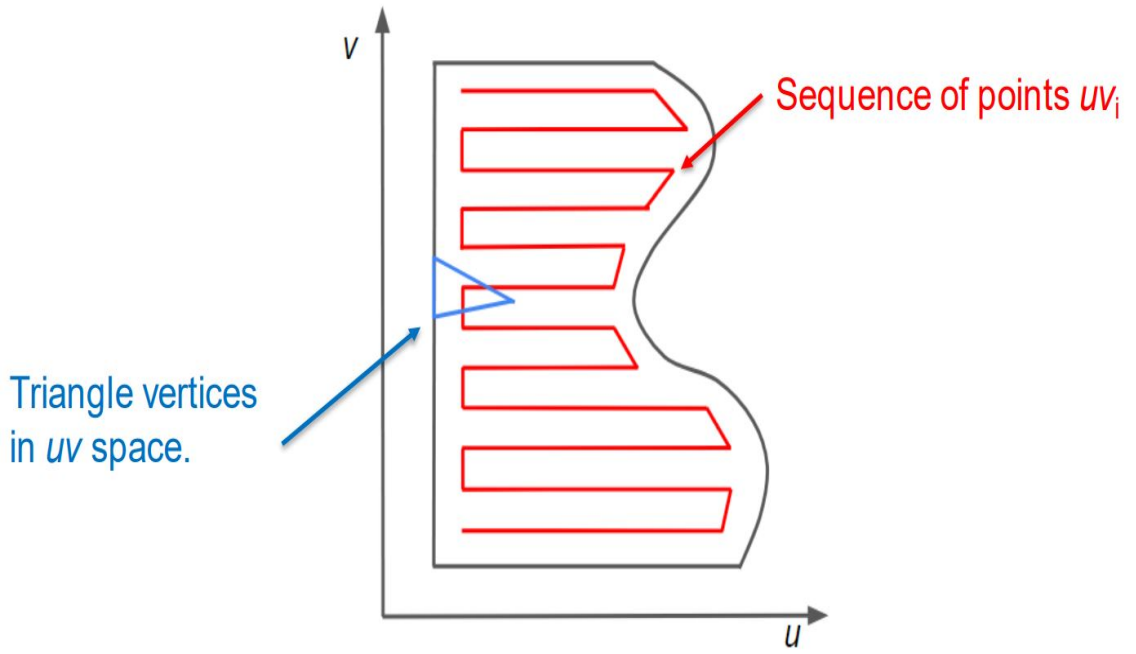


Figure 6.4: Joint-space motion continuity search between two neighboring E-cells for the spherical surface (in degrees).

position $\mathbf{h}_i$ in the Cartesian space as:

$$\mathbf{p}_i = \mathbf{h}_i - d \cdot \mathbf{a}_i, \tag{6.1}$$

where $\mathbf{a}_i$ is the third column of R_i. Now we initialize the end-effector coverage path $\mathbf{P}$, s.t. $\mathbf{P} = [E_1, E_2, ..., E_M]$, where E_i, for $i = 1, ..., M$, is a homogeneous transformation matrix consisting of position $\mathbf{p}_i$ and rotation matrix R_i. We denote $\mathbf{Q}$ as the sequence of corresponding joint states of $\mathbf{P}$, with $\mathbf{q}_i$ being the ith joint state on the path such that $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_M]$.

If a feasible coverage path exists (as determined in Chapter 5), it does not mean the

initial joint trajectory **Q** continuously satisfies task constraints. The following subsections describe singularity detection on the joint trajectory **Q**, and if singularities are detected, how we alter the end-effector path **P** to avoid singularities and best satisfy task constraints.

6.2.2 Singularity Detection

Even if the surface S is placed within the robot's workspace, depending on the shape and size of S, there can be internal singularities to prevent the manipulator from following the initial path. Two kinds of singularities have to be considered: *joint space singularities* and *Cartesian space singularities*.

We refer to joint space singularities along a joint space path as those configurations that exceed a certain joint limit, which result in failures of motion/trajectory planning. These singularities result in an end-effector "lift-off" from the surface, where task constraints are broken during the generation of joint trajectories given the desired end-effector path **P**. We refer to Cartesian space singularities along a Cartesian space path as those end-effector configurations that cannot be reached with the imposed task constraints.

Our method checks the end-effector path **P** for such singularities through the corresponding manipulator joint space path **Q**. As in [181], linear interpolation is performed between joint configurations $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$, for $i = 1, ..., M$, on path **Q**, such that:

$$\mathbf{q}_{i_j} = \mathbf{q}_i + j \cdot \epsilon \cdot (\mathbf{q}_{i+1} - \mathbf{q}_i), j = 1, 2, ... \tag{6.2}$$

for a small ϵ. Forward kinematics yields corresponding Cartesian configurations. If the

end-effector position $\mathbf{p}_{i_j}$ satisfies $|\mathbf{p}_{i_j} - \mathbf{p}_{i+1}| > |\mathbf{p}_{i_{j-1}} - \mathbf{p}_{i+1}|$, then a singularity occurs; otherwise, there is no singularity at $\mathbf{p}_{i_j}$. The joint-space path between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ should result in end-effector positions where $\mathbf{p}_{i_j}$ is closer to $\mathbf{p}_{i+1}$ than $\mathbf{p}_{i_{j-1}}$, otherwise a singularity exists between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$.

6.2.3 Singularity Avoidance

Algorithm 5: Manipulator Path Planner

```
Initialize end-effector orientation sequence {R_1, R_2, ..., R_M};
Initialize joint-space path Q = {q_1, q_2, ..., q_M};
i = 2; count = 0;
while i ≤ M do
    check joint-space singularity and Cartesian-space singularity from q_{i-1} to q_i:
    while Singularity at q_{i-1} ≤ q* ≤ q_i and count < K do
        for j = 0 to i − 1 do
            smooth the orientation R_{i-j} if needed to reduce the difference to the
              orientation R_{(i-j)-1} while satisfying the orientation constraint and
              update p_{i-j} accordingly to satisfy position constraint;
            update q_{i-j} based on the updated p_{i-j};
        end
        count++;
    end
    if no singularity then
        i++;
    else
        exit with "no feasible solution";
    end
end
output Q;
```

For every singularity encountered along the path (either a joint space singularity or a Cartesian space singularity), our planner alters the end-effector orientations in a neighborhood of the singularity configuration, called *orientation smoothing*, until a singularity-free

path is obtained.

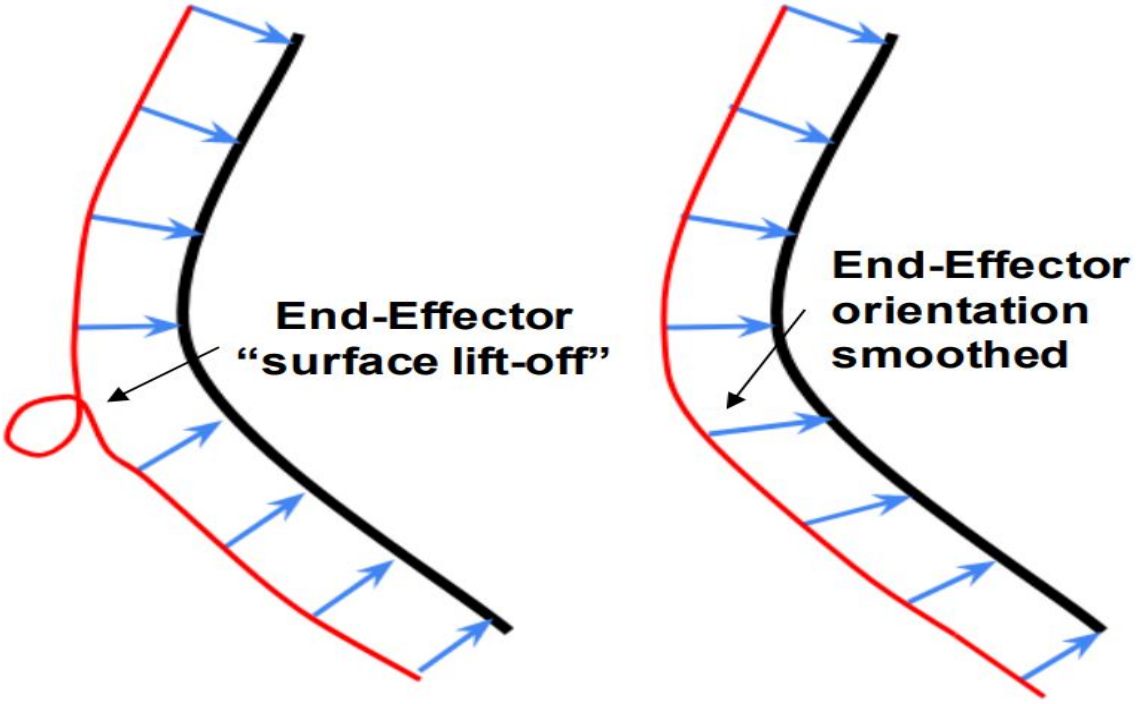


Figure 6.5: (left) Illustration of resulting end-effector path (red) with singularity in desired path.(right) Illustration of end-effector path (red) after orientation smoothing to avoid singularity (right).

For each singularity encountered along the end-effector initial coverage path $\mathbf{P}$, say at the ith waypoint, our method re-orients the end-effector approach vector $\mathbf{a}_{i-j}$ closer to that of the previous waypoint's approach vector $\mathbf{a}_{(i-j)-1}$ with: $\mathbf{a}_{i-j} = \frac{\mathbf{a}_v}{\|\mathbf{a}_v\|}$, where $\mathbf{a}_v = \frac{(\mathbf{a}_{i-j}+\mathbf{a}_{(i-j)-1})}{2}$, for $j = 0$ to $i-1$. The path can be smoothed up to N times, where the difference between neighboring approach vectors is reduced to 2^{-N} original difference. Satisfaction of task constraints is checked after orientation smoothing at each point. The worst case time complexity for each smoothing process is O(M) and the total algorithm

execution time is proportional to how many singularities are detected along the initial coverage path **P**.

The complete manipulator path planner is shown in Algorithm 5, which alters the initial manipulator path to create a singularity-free path, satisfying equality task constraints, and best satisfying inequality task constraints. Figure 6.5 is an illustration of an example singularity in an end-effector path resulting in a surface "lift-off" and a smoothed end-effector path to avoid singularities. If it is not possible to create such a path under the current coverage pattern, the planner returns “no feasible solution". Another coverage pattern can be tried.

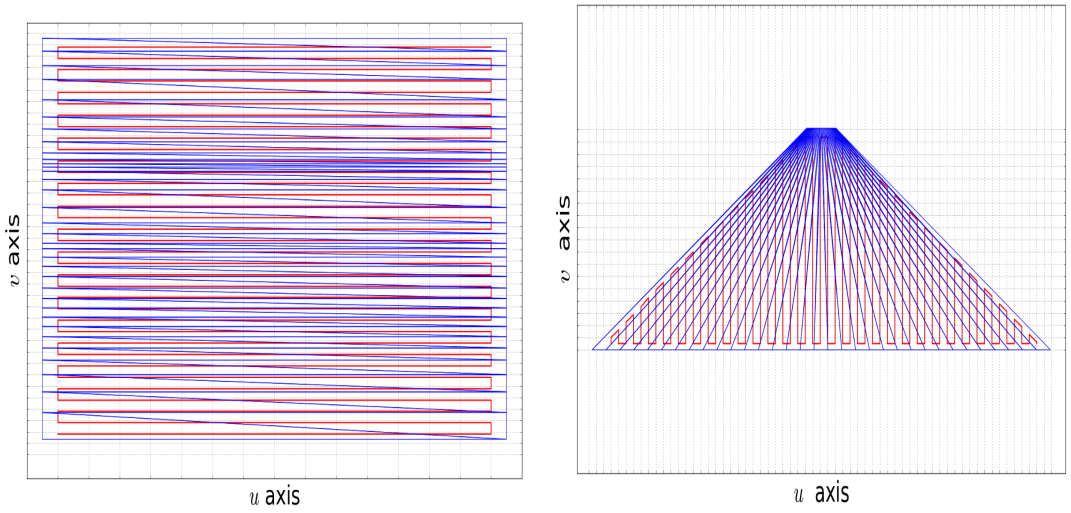
6.3 Experiments and Analyses

In this section, we provide results from converting the uv coverage pattern on different surfaces to Cartesian space and giving an initial offset from the surface. We then implement our motion planning algorithm to provide singularity-free manipulator coverage paths to follow the surface coverage patterns.

6.3.1 Surface Coverage Pattern Results

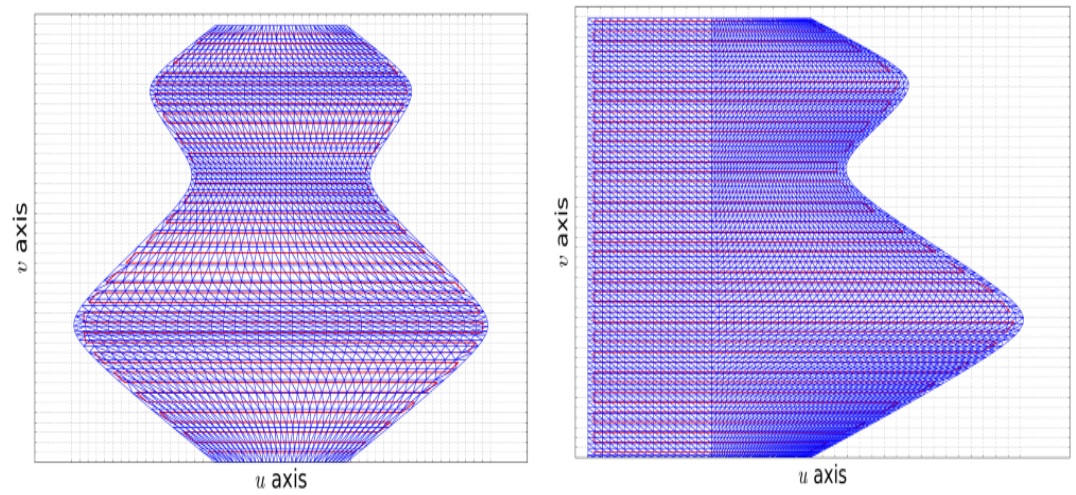
Figure 6.6 shows **UV** raster coverage patterns in red overlayed on the uv grid of four different surfaces. Each of our surface types is represented in these surfaces. Figure 6.7 gives the resulting **H** coverage patterns in Cartesian space for each surface. Note the uniform spacing for each coverage pattern along each surface in Cartesian space even with

a high degree of surface curvature.



(a) uv gird on Curvy surface (SoT) with coverage pattern.

(b) uv grid on Cone surface (SoR) with coverage pattern.

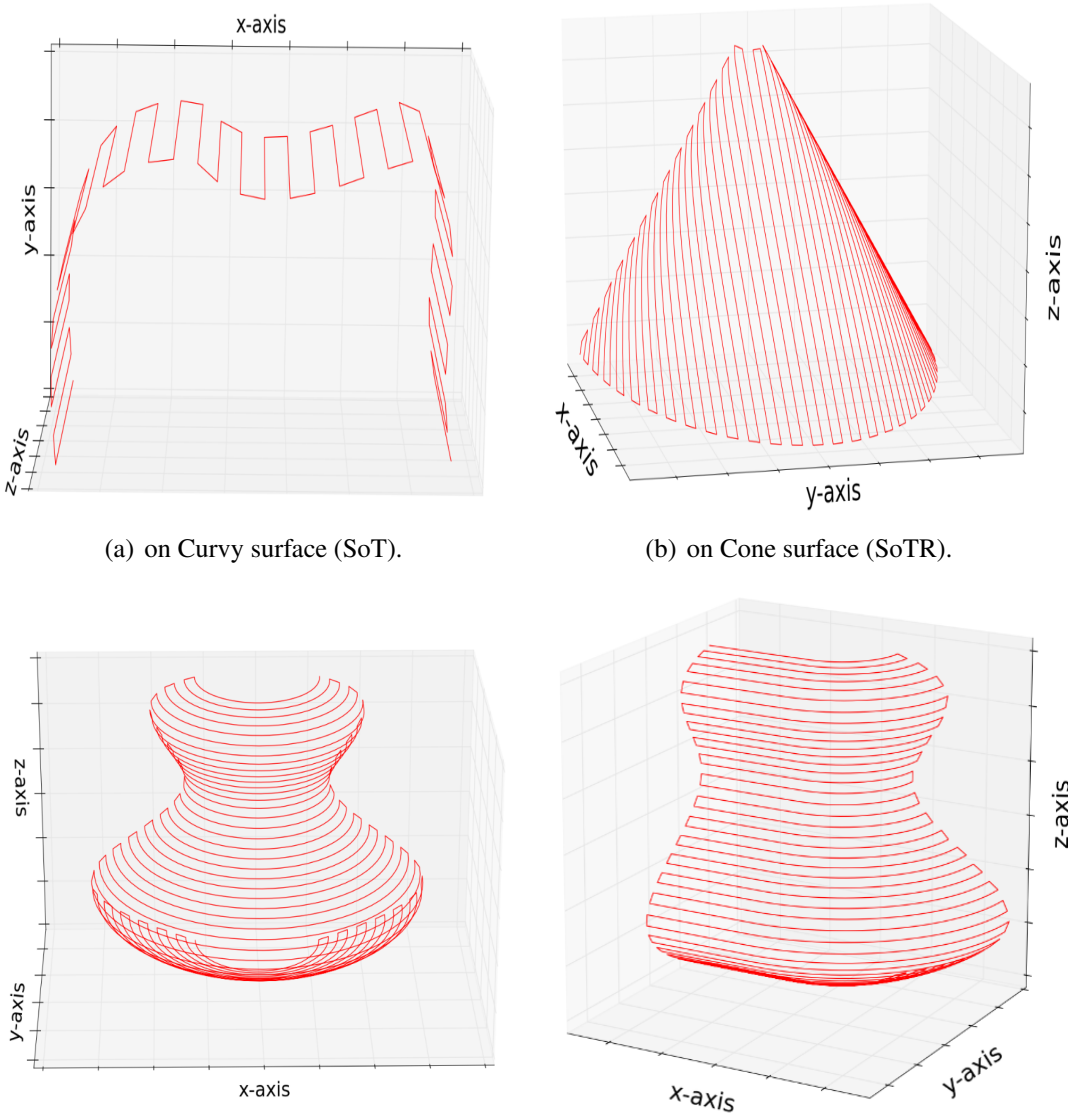


(c) uv grid on Vase surface (SoR) with coverage pattern.

(d) uv grid on Corner surface (SoTR) with coverage pattern.

Figure 6.6: Generated uv grid (blue) with applied raster coverage patterns $\mathbf{UV}$ (red) on each freeform surface.

Fig. 6.7 (b) shows the results of a coverage pattern that was generated on a cone surface



(a) on Curvy surface (SoT).

(b) on Cone surface (SoTR).

(c) on Vase surface (SoR).

(d) on Corner surface (SoTR).

Figure 6.7: Raster scan coverage patterns on different surfaces.

using our method. This pattern is not trivial to generate using other methods but is easily generated and evenly spaced using our uv grid. The “vertical" raster pattern is straight near the center yet spirals on either side. This is a simple example to demonstrate that using the uv grid generated by our method, non-trivial coverage patterns can be produced

on 3D freeform surfaces.

6.3.2 Motion Planning Algorithm Results

We experiment with different surface spatial arrangements with different surfaces and place raster scan coverage patterns on each surface. We then generate initial manipulator coverage paths and run our smoothing algorithm to avoid singularities if they are detected. Table 6.2 shows the corresponding results of the coverage paths and executed trajectories given three different spatial arraignments with respect to the manipulator. Table 6.1 presents the number of singularities fixed by Alg. 5 for each surface in three spatial arrangements relative to the robot.

Table 6.1: Number of singularities fixed by path planner.

	Curvy (SoT)	Cone (SoR)	Vase (SoR)	Corner (SoTR)
Arr.	# Sing.	# Sing.	# Sing.	# Sing.
1)	0	0	0	0
2)	3	4	12	4
3)	16	8	35	22

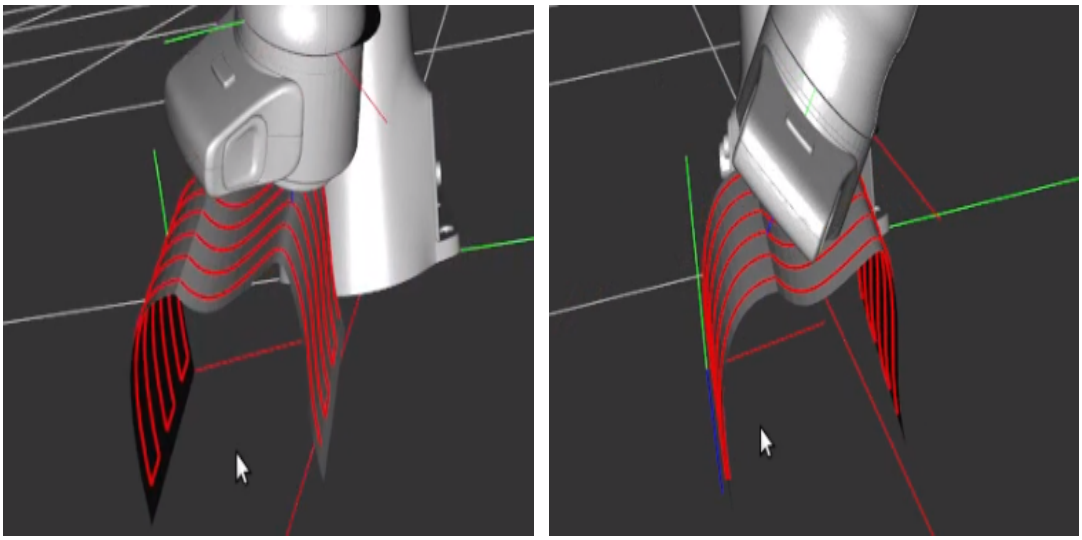
For each surface, the first spatial arrangement resulted in no singularities for the end-effector coverage path. These arrangements are within a more desirable workspace of the given manipulator. The other spatial arrangements resulted in singularities, most likely a result of joint values nearing the manipulator joint limits. Our orientation smoothing method outlined in Alg. 5 successfully smoothed the coverage path orientations within

Table 6.2: Coverage path/trajectory results.

Curvy (SoT)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	48.6	7.1	1.4	1.42
2)	4.9°	0.28°	45.3	6.8	1.3	1.56
3)	5.7°	0.43°	44.3	6.6	1.3	1.74
Cone (SoR)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	47.6	8.3	1.5	0.9
2)	2.2°	0.15°	45.8	7.9	1.2	1.13
3)	6.1°	0.21°	45.2	7.5	0.9	1.24
Vase (SoR)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	43.5	13.6	1.5	0.41
2)	4.7°	0.42°	41.7	12.7	1.4	0.54
3)	8.9°	0.72°	39.2	12.5	1.2	0.61
Corner (SoTR)						
Arr.	Max ψ (deg.)	Avg. ψ (deg.)	Max Speed (cm/s)	Avg. Speed (cm/s)	Avg. Accel. (cm/s^2)	Exec. Time (min.)
1)	0°	0°	33.8	12.3	2.0	0.42
2)	3.4°	0.41°	34.3	11.6	1.9	0.48
3)	5.5°	0.61°	31.2	11.1	1.7	0.53

task constraints and produced a feasible path for each spatial arrangement.

Figure 6.8 shows snapshots in our virtual environment of the Franka Emika Panda end-effector traversing a surface coverage pattern along the curvy (SoT) surface. Note the distance between raster scans, ω, is larger in the video than as shown in Fig. 6.7 to reduce the coverage execution time. This shows that our method can place different coverage patterns on a freeform surface with ease.



(a) (b)

Figure 6.8: Snapshots of the Panda manipulator moving along a coverage pattern (red) on the Curvy (SoT) surface.

6.3.3 Discussion

Different 2D coverage patterns may be placed in the uv grid as long as the pattern path is informed by the continuity graph G where the pattern does not cross between uv-cells who did not have feasible continuity. This is a general assumption that given feasible motion

continuity between uv-cells then the coverage pattern may pass between those uv-cells. There may be exceptions to this rule in rare cases where a straight coverage line between the uv-cells may not be feasible.

The orientation smoothing algorithm avoids singularities caused by joint limits by smoothing the end-effector orientations at the location where a singularity was detected. The orientation is adjusted to be closer to previous orientations along the path (where no singularities were detected) in an attempt to pull joint values away from their respective joint limits. Small adjustments in orientation are preferred to reduce the change in end-effector orientation from the optimal approach vector at each position (depending on task parameters). Thus, we find end-effector orientations that avoid singularities at critical points in the path while reducing the effect on optimally satisfying surface task constraints.

6.4 Chapter Summary

This Chapter informs on the placement of commonly used surface coverage patterns on our uv grid representation of the 3D surface and provides algorithms to generate these coverage patterns. Coverage patterns such as raster scans or Archimedean spirals are generally used for constrained surface coverage applications due to their constant separation between adjacent turns, thus resulting in uniform coverage [3,15]. If continuous coverage feasibility checking determines that a feasible coverage path exists for a 3D freeform surface given task and manipulator constraints, a coverage pattern is placed on the uv grid and mapped to the Cartesian space.

Although we determined that at least one coverage pattern exists on the surface, we still need to find a continuous coverage path. An initial manipulator end-effector coverage path is generated to follow the Cartesian surface coverage pattern and end-effector z-axis vectors are aligned with the surface normal at each point. We introduce a coverage path singularity detection algorithm to determine if the corresponding manipulator joint trajectories cause a surface "lift-off" (breaking surface task constraints) due to a singularity in the manipulator end-effector path. If a singularity is detected, then the end-effector orientations along the path are smoothed (within surface task constraints) to better avoid exceeding manipulator joint limits.

We provide results from converting the surface coverage pattern from uv space to Cartesian space on four example surfaces. We then provide results from singularity-free manipulator path generation on different surfaces given different spatial arrangements. Some spatial arrangements (closer to the manipulator workspace edge) create singularities in the manipulator coverage path and our motion planner removes these singularities by adjusting the end-effector orientations along the path.

Future research for the singularity-free motion planner includes using analysis results from coverage feasibility checking and virtual simulation to better understand key parameters which cause singularities along the manipulator coverage path. Results may inform an improved initial manipulator coverage path generation to avoid singularities.

Chapter 7

Virtual Case Study for Surface Coverage Applications

With so many distinct industrial surface coverage applications, the number of different combinations of robotic manipulators, variable task parameters, and material application models is enormous. Efficient coverage application analysis is essential for each specific application to determine which task parameters and material application models produce the best surface coverage results. The majority of industrial surface coverage case studies are performed in real world environments which require laborious and time consuming experiment setup [1, 10]. A general virtual environment for simulating surface coverage applications given a manipulator, task parameters, and material application models will allow human workers to better understand key contributors for uniform surface coverage.

This chapter presents a general interactive virtual environment for planning coverage

paths on 3D freeform surfaces and simulation of surface coverage applications for uniform coverage analysis given a manipulator, task parameters, and material application model. We demonstrate the effectiveness of our methods with a case study of a spray coverage application using a Franka Emika Panda robotic manipulator. We analyze the uniformity of surface material coverage between previous methods for placing coverage patterns on 3D freeform surfaces and our method for coverage pattern placement using the surface uv representation. Section 7.1 discusses the setup of the virtual environment, the end-effector spray model, and analysis methods for the simulation of 3D surface coverage applications. Section 7.2 provides results from the virtual case study giving visual and numerical comparisons between the methods in this dissertation and previous methods. Section 7.3 concludes the chapter.

7.1 Virtual Environment for 3D Surface Coverage Applications

Our virtual environment for the simulation of 3D surface coverage applications is programmed in C++, with method algorithms programmed in Python, and uses public libraries such as freeGLUT to interface with OpenGL, Assimp, and Dear ImGui. The virtual environment interfaces with ROS and MoveIt! for forward and inverse kinematic services. This virtual environment is currently implemented to work with the Franka Emika Panda manipulator and future work may allow any manipulator to work in the environment. All 3D surface model meshes were generated using Solidworks.

In this section, we first discuss the interactive GUI and systematic process for producing a manipulator coverage path on 3D freeform surfaces. We then introduce a spray model used during the spray simulation. Finally, we introduce analysis points generated on the surface using its uv grid. Analysis points are used for simulated material accumulation measurements on the surface.

7.1.1 Interactive GUI for Surface Coverage Planning

It is expensive and inconvenient to require a robotics expert who understands the manipulator's kinematics and motion planning algorithms to be present during the planning of industrial surface coverage applications. On the other hand, there are many complex environmental challenges that a human worker (who knows the task as a domain expert) is currently more suitable to tackle than automation. Hence, there is a need to enable human domain experts to interact with the automation system to combine their strengths in problem solving. An interactive virtual environment is advantageous for a domain expert worker, who is not a robotics expert, to provide task specific parameters for autonomous algorithms to perform the surface coverage efficiently and optimally (see Fig. 1.3).

Our interactive virtual environment allows a domain expert user to input any 3D surface model and spatially arrange the surface with respect to the manipulator base, which is treated as the origin of the environment. Sliders allow the user to adjust the position of the surface along the x, y, and z Cartesian axes. Sliders also allow the user to rotate the surface about its pitch, roll, and yaw. Additional sliders are provided to adjust the

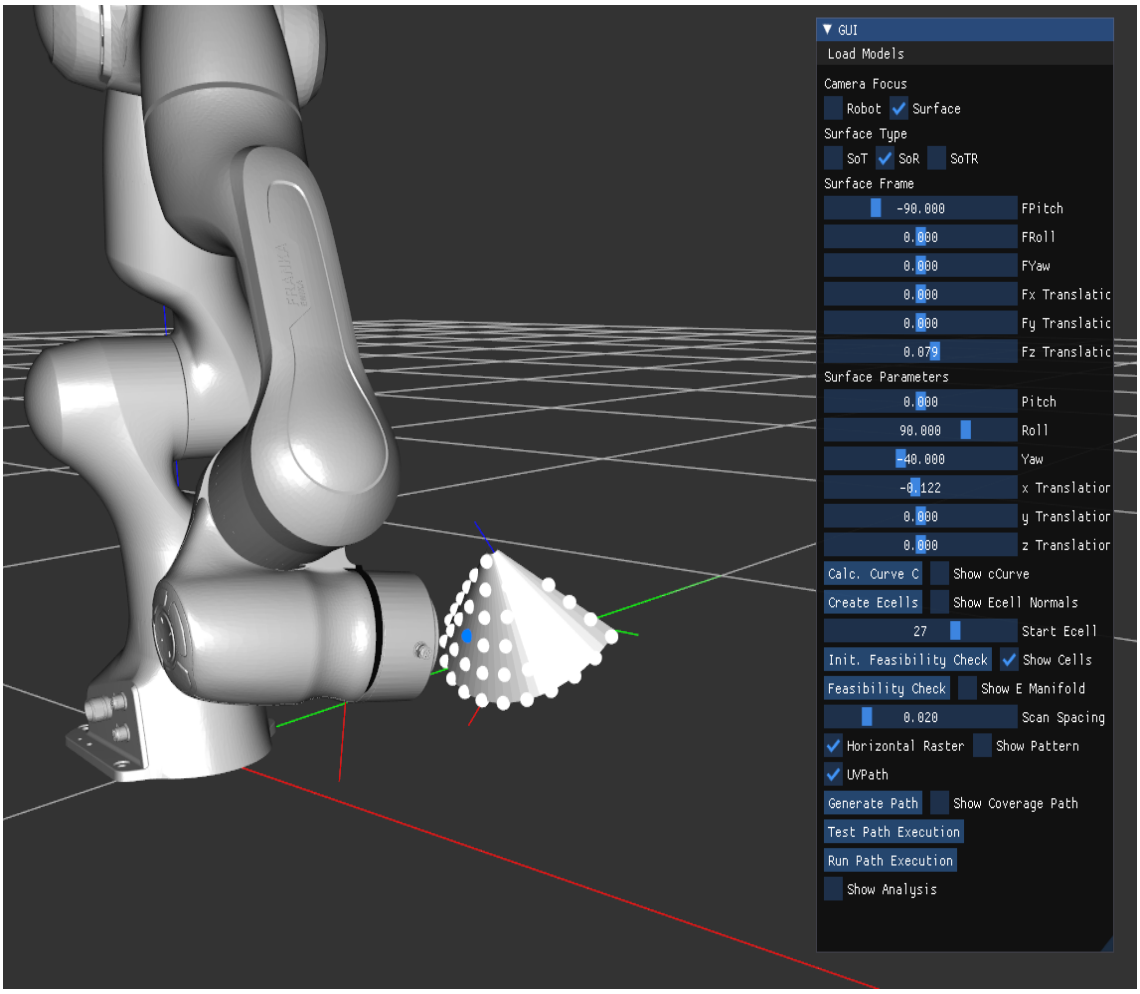


Figure 7.1: Virtual environment with Franka Emika Panda and uv-cells (white) on Cone surface (SoR). GUI on right gives human worker interface for surface positioning and for initiating automatic uv grid generation, coverage feasibility checking, coverage path generation, and spray coverage simulation.

surface coordinate frame as discussed in Chapter 4. Once the 3D surface is in the virtual environment, the worker can adjust the 3D surface coordinate frame until the C curve (yellow) is correctly represented for each surface type. Then, the worker can allow the simulation to automatically generate the uv grid on the surface to facilitate coverage feasibility checking and coverage pattern placement (see Fig. 7.1 for GUI visual).

The human worker can systematically change surface orientations through the interface to let the robotic system check for manipulator workspace feasibility and coverage feasibility as discussed in Chapter 5. The checking results are presented to the human worker visually through the interface in terms of continuity between each neighboring uv-cell in a final continuity graph once complete. The worker can then decide to reposition the surface or move forward to place a coverage pattern on the surface. The human worker can choose to place different coverage patterns on the surface, such as horizontal or vertical raster scan coverage patterns, depending on the surface uv grid representation shape as discussed in Chapter 6. A surface coverage pattern is automatically generated and the human worker can visually verify coverage pattern uniformity on the 3D surface.

Finally, the worker can instruct the virtual environment to generate a singularity-free manipulator coverage path to follow the surface coverage pattern. The virtual environment uses Moveit! to generate an initial manipulator coverage path given the coverage pattern points, task surface offset, and end-effector orientations along the pattern. If Moveit! fails to generate a coverage path due to unavoidable singularities along the proposed coverage path, then the end-effector orientations are adjusted systematically as discussed in Chapter 6. The manipulator path trajectory is adjusted to produce a smooth velocity for point $\mathbf{p}'$ on the surface (recall that is the point that the end-effector's z-axis projects to on the surface). The human worker may then initiate a virtual spray simulation and will be provided with visual and numerical results of material application.

7.1.2 Manipulator Spray Model

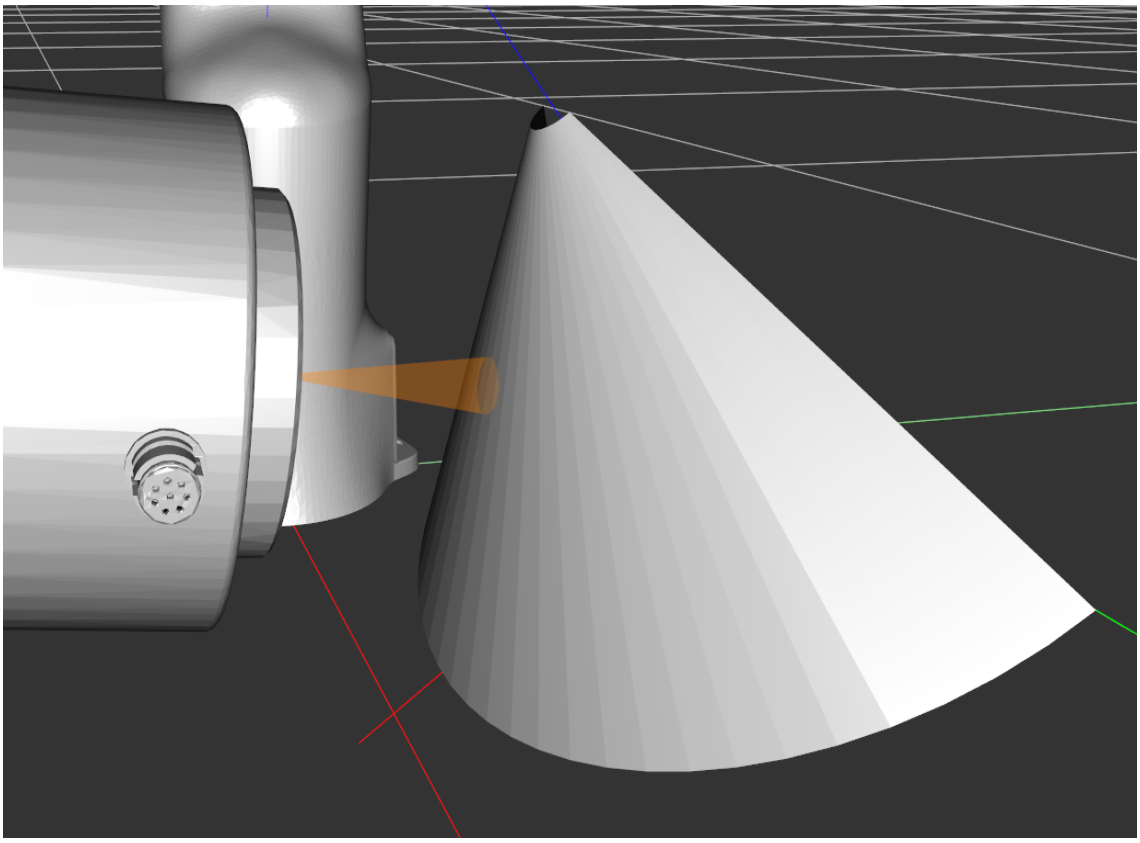


Figure 7.2: Spray cone volume V (orange) projecting from Franka Emika Panda end-effector.

There are many types of physical spray models used in industry depending on the surface coverage application. For this case study, we consider a simple "cone shaped" spray volume, V, protruding from the end-effector tip which widens further along the z-axis of the end-effector coordinate frame (see Fig. 7.2). For the virtual spray simulation, we determine the spray application to be inside the cone shaped volume where the 3D surface mesh intersects the volume. The cone shaped volume has a maximum height,

V_h, (distance from the end-effector tip along end-effector coordinate frame z-axis) and a maximum radius, V_r.

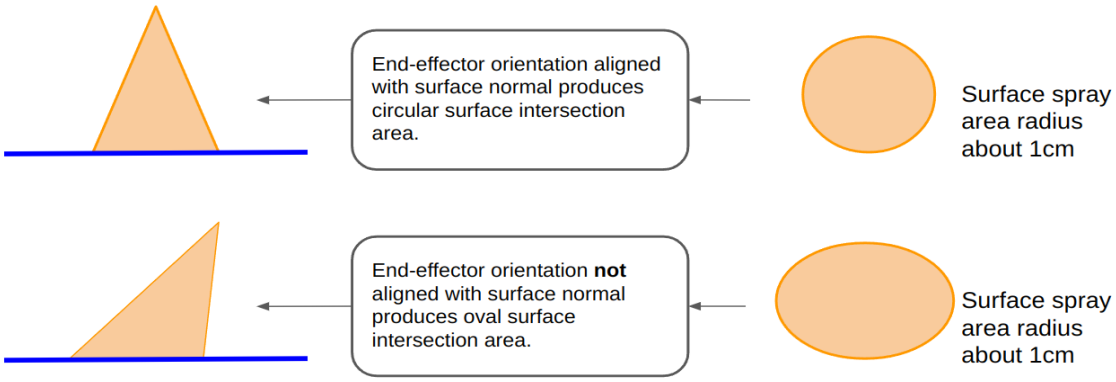
For a point on the 3D surface mesh, $\mathbf{p}_s$, we first calculate the z-axis component of $\mathbf{p}_s$ with respect to the end-effector coordinate frame,

$$D_z = \mathbf{a} \cdot (\mathbf{p}_s - \mathbf{p}), \tag{7.1}$$

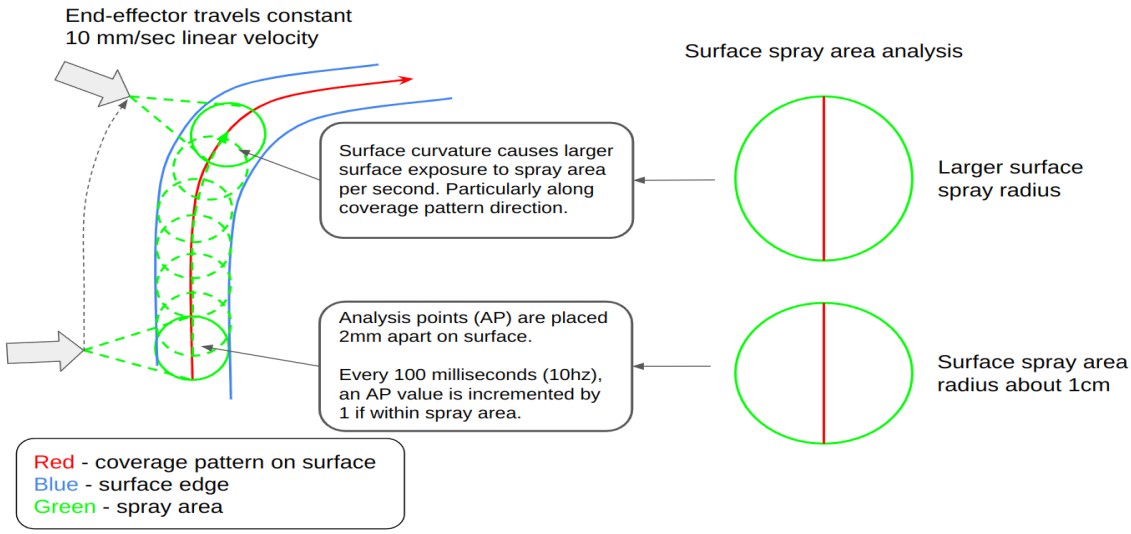
where **p** is the end-effector Cartesian position, **a** is the end-effector approach unit vector, and $0 < D_z < V_h$. We then calculate the tangential distance from the end-effector z-axis to $\mathbf{p}_s$ which must be less than the radius of the cone at D_z along the z-axis:

$$\|(\mathbf{p}_s - \mathbf{p}) - (D_z \cdot \mathbf{a})\| < (D_z/V_h) \cdot V_r. \tag{7.2}$$

A point on the surface, $\mathbf{p}_s$, is determined to be in the application area where the spray volume, V, and surface mesh intersect if equations 7.1 and 7.2 are satisfied. The application area depends on the approach vector of the end-effector, **a**, with respect to the surface normals (see Fig. 7.3(a)). With an end-effector orientation aligned to the surface normal, a circular spray application can be achieved. As the end-effector orientation deviates from the surface normal the application area becomes larger and oval shaped. Furthermore, a similar effect may occur along high surface curvatures along the coverage path (see Fig. 7.3(b)).



(a) Illustration of surface spray intersection along surface with high degree of curvature.



(b) Illustration of surface spray intersection with different end-effector orientations in relation to surface normal.

Figure 7.3: Illustration showing examples of non-circular surface spray intersection with end-effector orientation and surface curvature. (a) Example of non-circular surface spray intersection due to end-effector orientation. (b) Example of non-circular surface spray intersection due to surface curvature.

Finally, we calculate the amount of spray material that accumulates on a surface point, $acc(\mathbf{p}_s)$, by incrementing $acc(\mathbf{p}_s)$ by 1 for every 10 milliseconds $\mathbf{p}_s$ is within the surface application area. Thus, we may use the $acc(\mathbf{p}_s)$ value for each point in the surface to quantify uniform coverage of the application material after simulating manipulator surface spraying in the virtual environment.

7.1.3 3D Surface Analysis Points

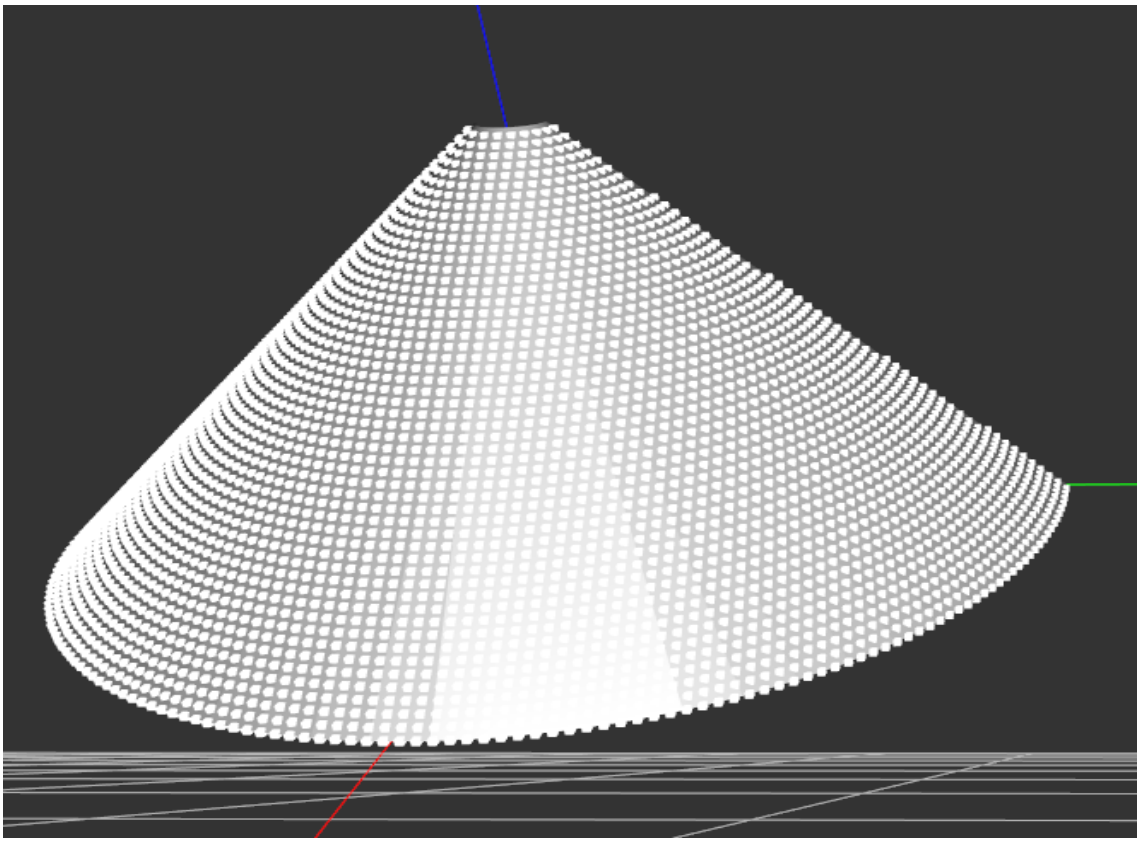


Figure 7.4: Analysis points (white) evenly spaced using uv grid on Cone surface (SOR) in virtual environment.

Analysis of coverage uniformity on 3D surfaces for experimentation in other previous

research has been done using discretized points evenly spaced on the surface. The amount of material at each point is measured and the standard deviation from the mean material accumulation is calculated. A small standard deviation denotes a more uniform application coverage on the surface.

The virtual environment automatically generates analysis points on the surface 2mm apart (see Fig. 7.4) using each surface's uv grid representation to provide evenly spaced points of analysis on the surface. Each analysis point is a Cartesian point on the surface mesh and we calculate material application, acc, at each point as discussed previously in this section. The analysis point colors range from white to green depending on the thickness of the application on the surface (i.e. value of acc for each analysis point). Algorithm 6 outlines the calculation of acc for each analysis point on the surface. We may then calculate the mean, standard deviation, and variation given the acc for every analysis point.

Note that Moveit! and ROS topic communications update manipulator joint values every 10 milliseconds, thus, the joint configurations in $\mathbf{Q}$ are tested in Algorithm 6 every 10 milliseconds. Therefore, we consider an accumulation of 1 unit of material application at each point within V on the surface every 10 milliseconds.

7.2 Experiments and Analyses

Using the virtual spray simulation we compare our method for uniform surface coverage with previous methods for coverage pattern placement on 3D freeform surfaces. This

Algorithm 6: Analysis point value calculation

Input joint-space path $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_M\}$;
Input analysis point positions $\mathbf{AP} = \{p_{s1}, p_{s2}, ..., p_{sN}\}$;
Initialize analysis point values $\mathbf{AC} = \{acc_1, acc_2, ..., acc_N\}$;
i = 0;
while $i \leq M$ **do**
  With $\boldsymbol{q}_i$ *acquire end-effector position* $\boldsymbol{p}_i$ *and approach vector* $\boldsymbol{a}_i$ *through forward kinematics;*
  j = 0;
  while $j \leq N$ **do**
    zDist = $\boldsymbol{a}_i \cdot (\boldsymbol{p}_{sj} - \boldsymbol{p}_i)$
    if $0 < zDist < V_h$ ***and***
     $\|(\boldsymbol{p}_{sj} - \boldsymbol{p}_i) - (zDist \cdot \boldsymbol{a}_i)\| < (zDist/V_h) \cdot V_r$ **then**
      Increment acc_j *by 1;*
    end
    j++;
  end
  i++;
  Return ***AC****;*
end

comparison is to verify improvements to uniform coverage on 3D freeform surfaces. For this case study, we created a new Corner surface (SoTR) whose shape has a high degree of curvature about multiple Cartesian axes (see Fig. 7.5). Analysis points are generated evenly on the surface to calculate spray material accumulation during the simulation of the surface coverage. We next provide comparison results and discussions on multiple simulations of surface spray coverage in the virtual environment.

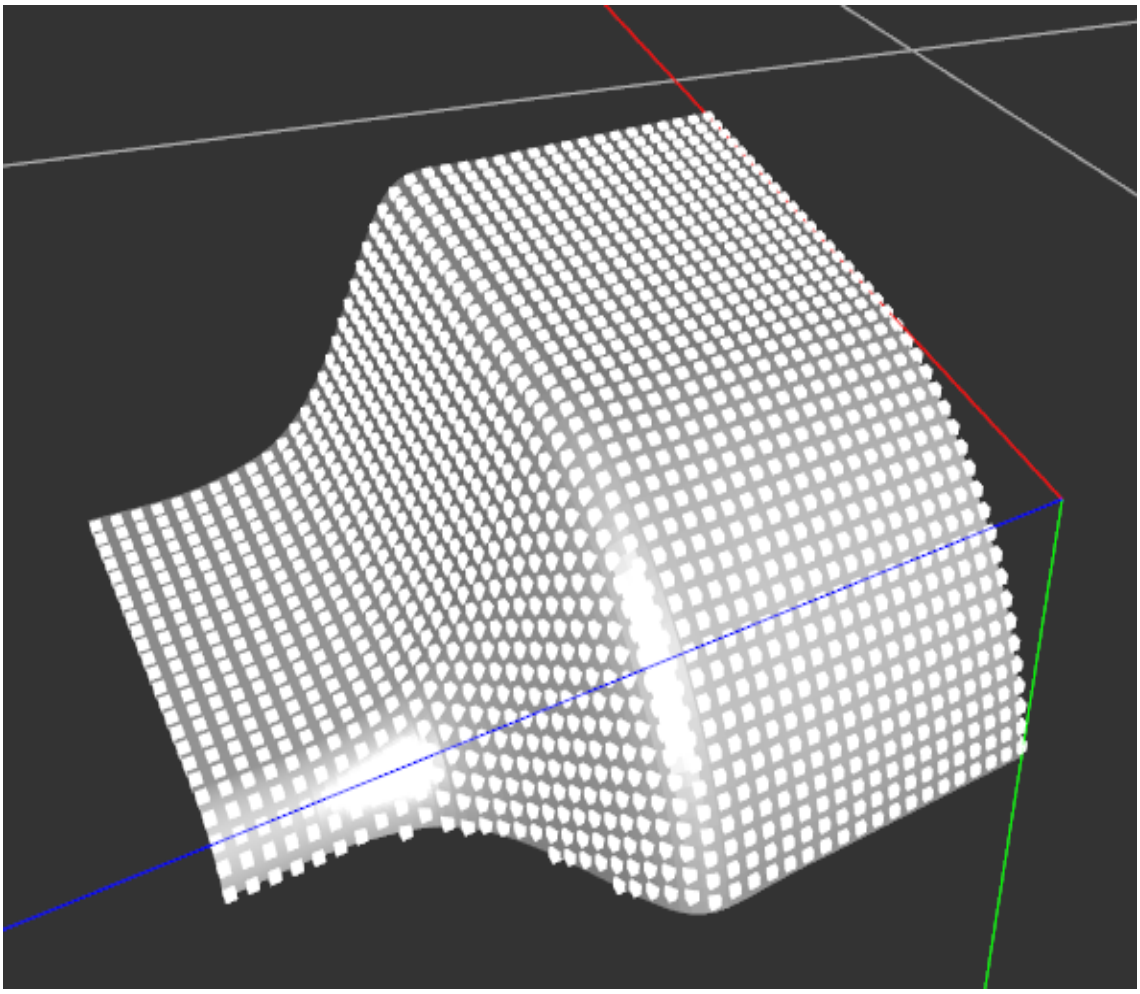
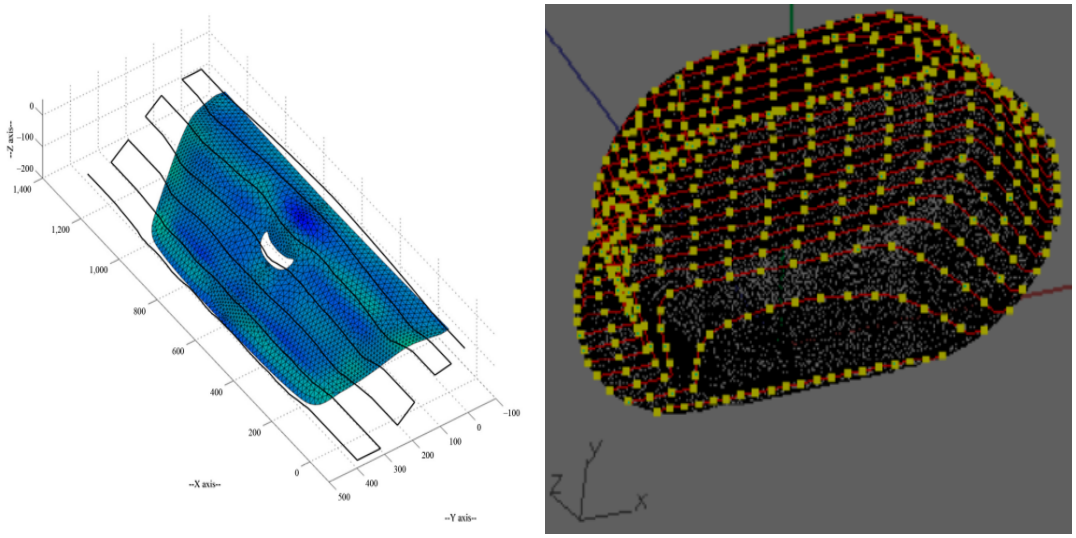


Figure 7.5: Illustration of Corner Surface (SoTR) with evenly distributed analysis points (white).

7.2.1 Spray Simulation Comparison Results

In previous work for coverage pattern placement, the coverage pattern raster scan lines are spaces using Cartesian space coordinates which results in larger gaps along a high degree of curvature on the surface (see Fig. 7.6). No matter how the coverage pattern using Cartesian space is placed on the Corner surface (horizontal or vertical) there are larger

gaps in the raster scan lines due to a high degree of curvature.



(a) Raster coverage pattern on vehicle hood. (b) Raster coverage pattern on freeform surface.

Figure 7.6: Examples of coverage patterns on surfaces using Cartesian coordinates as reference for raster scan line spacing. Image credits: (a) Andulkar *et al.*, 2015 [1] (b) Chen *et al.*, 2019 [10].

For each spray simulation, we use the following task parameters: a surface offset for the end-effector of 4cm, a raster coverage pattern scan line spacing of 8mm, a spray volume height V_h of 6cm, and a spray volume radius V_r of 8mm. The final manipulator coverage paths for each simulation have end-effector approach angles that align with the surface normal at each respective $\mathbf{p}'$. Analysis point colors range from pure white to full green with a av of 0 units displaying a white analysis point and a av of 30 units displaying a full green analysis point.

First, we run the spray simulation using the Cartesian space to place the coverage pattern on the surface as done in previous research. The coverage pattern is actually placed

on the Cartesian plane formed by the x and y axes and projected to the actual surface, which is common in previous work. Figure 7.7 is an image of the final results of this simulation with the coverage pattern on the surface in red. The analysis points that accumulated more material spray during the simulation have turned more green. There are noticeable gaps in the spray coverage on the surface (white analysis points) due to the high degree of curvature on the surface causing uneven spacing between raster scan lines. The histogram in Figure 7.9(a) confirms the visual results in Figure 7.7 where more than 350 analysis points have zero spray application exposure. The distribution of application on the surface is a bit more spread out with a standard deviation of about 6.5 units and a mean of 8.4 units of spray material. For this case, there are many analysis points with zero material accumulation value and only 76 percent of the surface is covered with spray material.

Next, we run the simulation on the same Corner surface (SoTR) with a coverage pattern generated using our methods that place a uv grid directly on the surface. Figure 7.8 is an image showing the results with the surface coverage pattern in red and analysis points turning green as they are exposed to the spray volume V. The histogram in Figure 7.9(b) has a more narrow distribution of spray application on the surface with only about 50 analysis points having zero exposure. As shown in Figure 7.8, there are no gaps of spray coverage (green analysis points) between coverage pattern scan lines giving a smaller standard deviation of about 5 units with a mean of 12 units of spray material. For this case, 97 percent of the surface is covered with spray material.

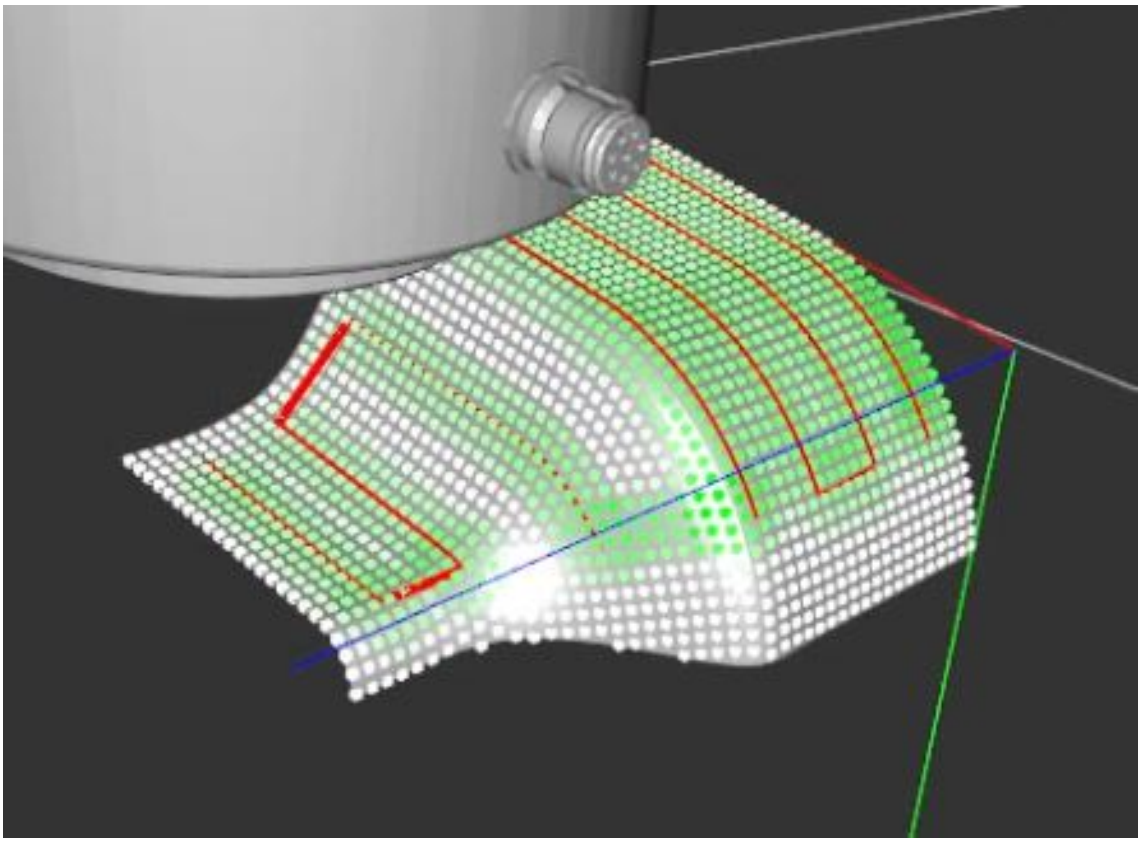


Figure 7.7: Image of finished surface spray coverage simulation showing Cartesian space coverage pattern (red) and analysis points (white and green) on Corner surface (SoTR).

7.2.2 Discussion

The virtual simulation environment provides a suitable solution to analyzing surface coverage feasibility and material application uniformity given a manipulator and task parameters. The interface allows a human worker to conduct a systematic analysis, without a robotics expert, to discover feasible surface coverage solutions and key spray and task parameters values contributing to uniform surface material coverage.

From the results in this chapter, we can determine that 3D freeform surfaces with a

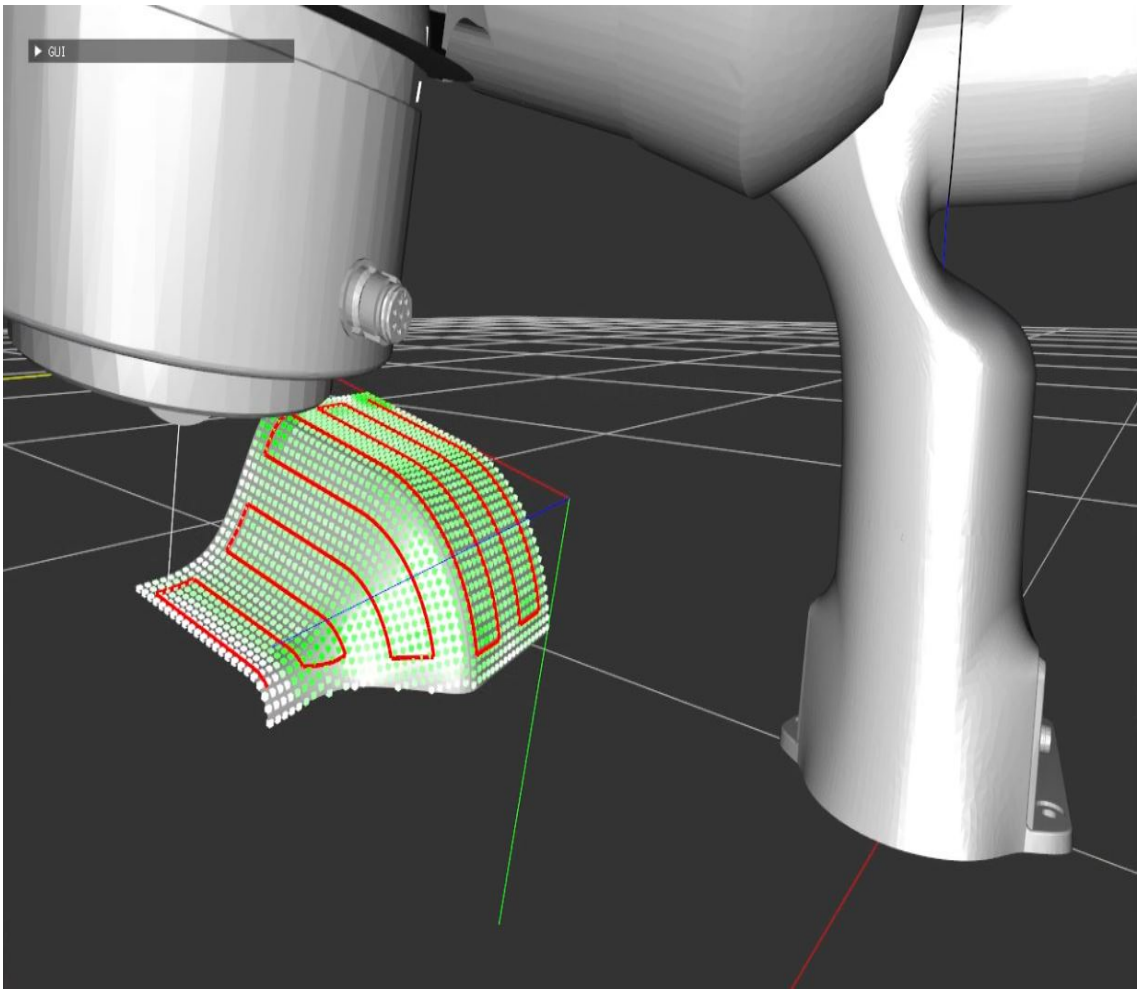
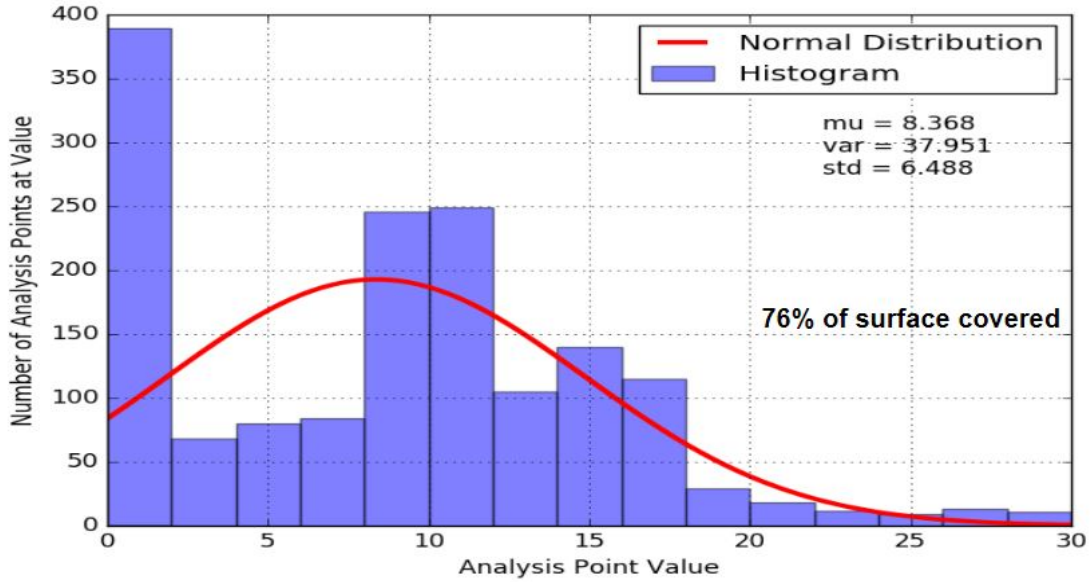
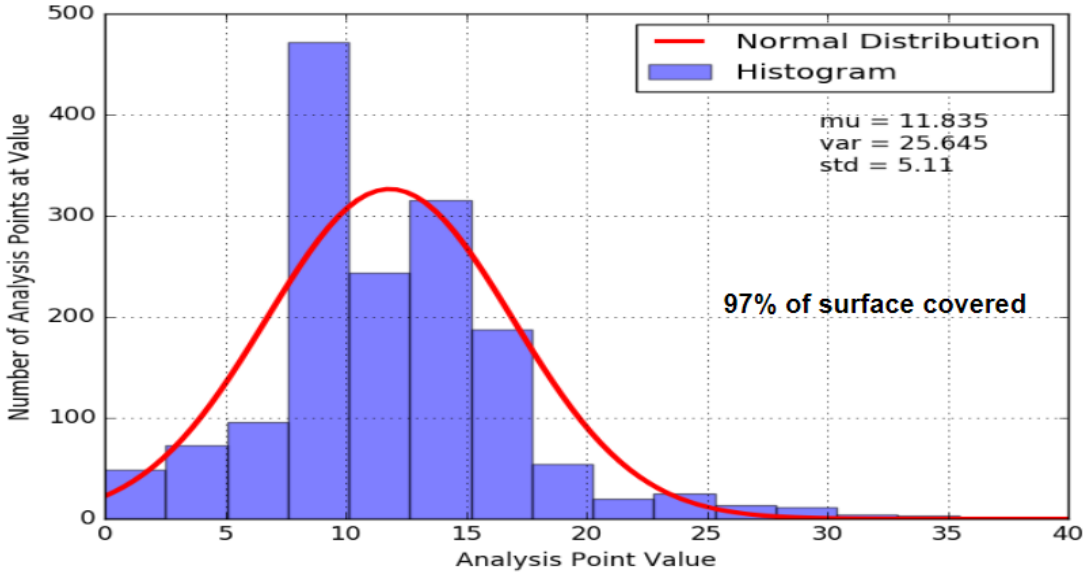


Figure 7.8: Image of finished surface spray coverage simulation showing uv space coverage pattern (red) and analysis points (white and green) on Corner surface (SoTR).

higher degree of curvature (such as a SoR and SoTR) benefit more from our methods of placing a uv grid than the common practice in previous work that places a grid on the Cartesian plane. That practice cannot produce even spacing coverage patterns along complex 3D freeform surfaces. Our method of coverage pattern placement is capable of handling a high degree of curvature about multiple axes simultaneously whereas the previous methods may only work well about one axis.



(a) Histogram plot (blue) and normal distribution (red) of spray application on Corner surface (SoTR) using Cartesian coordinates for coverage pattern placement.



(b) Histogram plot (blue) and normal distribution (red) of spray application on Corner surface (SoTR) using the introduced uv grid for coverage pattern placement.

Figure 7.9: Histograms results of spray simulations using different methods for surface coverage pattern placement.

The simulation environment analysis can be used to determine an appropriate spray model for the specific coverage task. The spray model used in this case study can be compared to other spray models, such as a thin triangular spray volume V versus a cone spray volume. Additionally, different calculations on spray material application in V, such as incrementing av differently based on distance from the end-effector tip, can be analyzed for quality of uniform coverage.

The current simulation environment for surface spray coverage depends on Moveit! and ROS for communicating the generated path over time. Therefore, the current time frame for each simulation is as long as the Moveit! simulated path is meant to take. The simulation time may be decreased substantially by speeding up the simulation clock.

7.3 Chapter Summary

In many industrial applications today, human-robot interaction is necessary for optimal and efficient completion of complex surface coverage tasks. For surface coverage applications, there are task specific parameters a human worker, who is not a robotics expert, can provide to a robot, and the robot can then complete the task autonomously with minimal feedback from the human worker.

In this Chapter, we introduced and demonstrated our virtual environment for the simulation of surface coverage applications using a Franka Emika Panda on a Corner surface (SoTR). The virtual environment provides an interface for a non-robotic expert to systematically check for feasible surface coverage solutions and analyze surface material

coverage efficiently with different task parameters. We describe the manipulator spray model and 3D surface analysis points used to simulate material coverage on the surface.

In the virtual environment, we demonstrate a manipulator spray simulation through a case study of surface coverage using a previous method for surface pattern placement and our method using the surface uv grid. The simulation spray results from analysis points on the Corner surface show that our method for surface coverage pattern placement provides more uniform material coverage on the surface. The results show uniform coverage patterns are crucial on freeform 3D surfaces with a high degree of curvature for uniform material coverage.

The virtual environment may be used by a non-robotics expert to analyze surface material coverage with different manipulators, 3D surfaces, task parameters, and spray models and determine key contributors for optimal production quality.

Chapter 8

Conclusions

In this chapter, the contributions of this dissertation are summarized and state multiple areas of interest for future work. We note the importance of the need to increase the capability of our uv gird to cover more types of 3D freeform surfaces. Additionally, we discuss the potential future use of our virtual simulation environment to improve industrial coverage applications.

8.1 Review of Contributions

We introduce a novel method for a human worker to automatically generate a uv grid on 3D freeform surfaces using our virtual environment. The uv grid allows uniform coverage patterns to be placed on complex freeform surface patches and facilitates coverage feasibility checking.

We then provide two algorithms for efficient coverage feasibility checking which allow

for the surface spatial arrangement to be checked prior to coverage path planning given manipulator kinematics. With the use of our virtual environment, a human worker may move the surface in relation to the manipulator and locate an optimal location for the surface given coverage task requirements.

Once a location is determined for surface coverage by the manipulator, the human worker may place a coverage pattern on the surface and visually verify the results. We have identified how to determine a proper coverage pattern starting orientation given the surface uv grid. Finally, we provide an algorithm to find a coverage path for the manipulator to follow the surface coverage pattern while maintaining constraints (i.e. avoiding manipulator singularities).

Furthermore, we presented a case study focusing on the analysis of spraying results. Our virtual environment enables human workers to simulate and analyze the application of coverage on a 3D surface, considering various task model parameters. The use of virtual simulation offers the advantage of faster analysis compared to real-world spray testing. It allows for efficient repetition of post-coverage analysis in a simulated environment.

8.2 Future Work

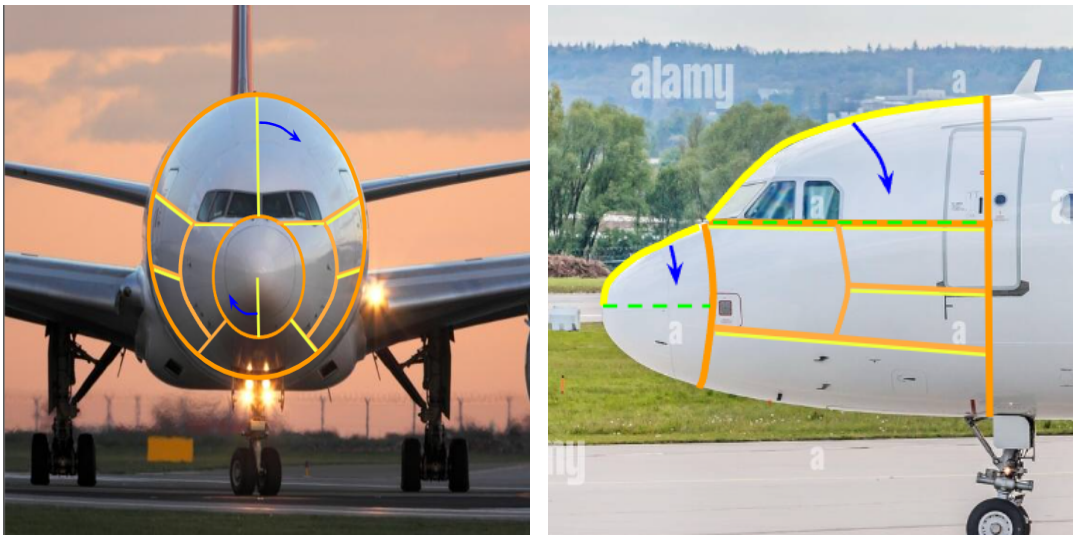
In this section, we discuss potential motivations for future work for this research. There is still much work to be done to improve upon the human-robot interaction for industrial surface coverage applications.

8.2.1 Hierarchical Surface Segmentation

Numerous 3D freeform surfaces exist that cannot be accurately approximated using the three primitive definitions introduced in Chapter 4. Furthermore, there could be surfaces too large to fit within the workspace constraints of the designated manipulator. In such cases, these surfaces could be divided into smaller surface patches based on the outcomes of our methods presented in this work. Hence, we recommend a potential avenue for future research, wherein a hierarchical method is adopted to segment surfaces through the following:

- With an initial surface inspection, a domain expert worker can analyze and decide on potential divisions into multiple patches, where each patch can be approximated using the 3D surface primitive definitions. This process can be aided by a virtual interface to the robotic system, facilitating efficient surface segmentation. Subsequently, each identified surface patch would then undergo the methods for continuous coverage motion planning on 3D freeform surfaces as delineated in this dissertation. See Figure 8.1 for an example illustration of an approximate segmentation of an aircraft for surface coverage.

- Following the coverage feasibility checking for each surface patch, the continuity graph G can visually represent the feasible coverage area on each patch, considering the manipulator's initial configuration. This graph's information can be used to



(a) Front view of aircraft skin segmented into primitive surface patches.

(b) Side view of aircraft skin segmented into primitive surface patches.

Figure 8.1: Example of a real world surface approximately segmented using the primitive SoR. Each segment is depicted in orange, accompanied by its corresponding initial curve C in yellow. Notably, each of these segments has a rotation axis that is parallel to the green axes shown for two SoR patches. Image credit: https://www.alamy.com/.

guide the further segmentation of the surface into sub patches, allowing for the use of multiple coverage patterns. This approach enables the manipulator to cover a portion of the surface, temporarily break constraints, move away from the surface, and subsequently reestablish constraints to cover the remaining area.

- In each surface patch, the coverage patterns can be optimized to minimize the time the manipulator spends lifted off the surface between coverage paths. To achieve this, the starting and ending points of each pattern could be strategically positioned on the surface patches, aiming to minimize the distance between them. Additionally, an iterative methodology might be developed, where coverage pattern placement

is influenced by manipulator motion planning. By constructing a singularity-free manipulator motion plan, it's possible to ensure the coverage of all surface patterns while minimizing the time the manipulator spends lifted off the surface. This approach could lead to reductions in material waste and the cycle time of coverage paths.

- Lastly, it's worth contemplating the scenario where the spatial relationship between the surface and manipulator changes during a surface lift-off. This adjustment could be made to accommodate larger surfaces that exceed the manipulator's workspace. During the lift-off phase (when the manipulator is not bound by surface task constraints), the surface might be translated or rotated. Consequently, a surface patch that was initially beyond the manipulator's workspace could be repositioned into a feasible coverage location. Achieving this would likely involve an iterative strategy, incorporating surface coverage feasibility checks across various spatial arrangements of the surface.

8.2.2 Virtual Simulation Environment for 3D Surface Application Analysis in Industrial Applications

We provide a virtual environment to implement our method for surface coverage application and, as a case study, simulate spray application on surfaces for coverage analysis. The simulation is given key spray parameters to determine the spray volume and surface

coverage. The virtual environment parameters, including the manipulator, may be changed to analyze optimal spray parameters for different surface types. These parameters include surface spray offset, surface spray orientation, and spray volume and in future work can include manipulator velocity and additional spray model features. Other industrial applications with various types of surface coverage can also be simulated for analyses.

8.3 Broader Impact

Our work provides support for industrial manufacturing in human-robot interaction for manipulator surface coverage applications. In industry for surface coverage applications, a human worker is required to laboriously generate a coverage path for a given surface and manipulator. This is not an optimal manipulator path and usually results in material waste. If the surface or manipulator should change the process repeats. Our virtual environment interface and semi-autonomous methods provide human workers in industry with robotic tools convenient to use to improve efficiency and product quality. This can improve the quality of work for human workers as well.

8.4 Concluding Remarks

There is more work to be done than initially thought for autonomous industrial coverage of freeform surfaces. Completing more of this work can greatly improve product quality and quality of work for workers, and reduce production costs and cycle time in industry. Our work is one step forward toward more autonomy for coverage on freeform surfaces in

industry.

This work was partially funded by the Future of Robots in the Workplace - Research and Development Program (FORW-RD). Due to current trends in labor shortages and industrial needs for increased production, society would benefit greatly from more robotic solutions for more laborious tasks.

Bibliography

[1] M. V. Andulkar and S. S. Chiddarwar, "Incremental approach for trajectory generation of spray painting robot," *Industrial Robot*, vol. 42, pp. 228–241, 2015.

[2] G. Liu, X. Sun, Y. Liu, T. Liu, C. Li, and X. Zhang, "Automatic spraying motion planning of a shotcrete manipulator," *Intelligent Service Robotics*, vol. 15, pp. 115–128, 2021.

[3] C. Chen, S. Gojon, Y. Xie, S. Yin, C. Verdy, Z. Ren, H. Liao, and S. Deng, "A novel spiral trajectory for damage component recovery with cold spray," *Surface and Coatings Technology*, vol. 309, pp. 719–728, 2017.

[4] P. Wang, R. Krishnamurti, and K. Gupta, "View planning problem with combined view and traveling cost," *Proceedings in IEEE International Conference on Robotics and Automation*, pp. 711–716, 2007.

[5] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, pp. 1258–1276, 2013.

[6] C. Yuksel, M. Tarini, and S. Lefebvre, "Rethinking texture mapping," *Computer graphics forum*, vol. 38, pp. 535–551, 2019.

[7] M. Tarini, "Cylindrical and toroidal parameterizations without vertex seams," *Journal of Graphics Tools*, vol. 16, pp. 144–150, 2012.

[8] P. K. R. Maddikunta, Q. V. Pham, P. B, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, 2022.

[9] S. Gomez, V. M. Becerra, J. R. Llata, E. Gonzalez-Sarabia, C. Torre-Ferrero, and J. Perez-Oria, "Working together: A review on safe human-robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26754–26773, 2017.

[10] W. Chen, J. Liu, Y. Tang, and H. Ge, "Automatic spray trajectory optimization on bézier surface," *Electronics (Switzerland)*, vol. 8, 2019.

[11] H. Chen, W. Sheng, N. Xi, M. Song, and Y. Chen, "Cad-based automated robot trajectory planning for spray painting of free-form surfaces," *Industrial Robot*, vol. 29, pp. 426–433, 2002.

[12] T. Girbacia, F. Girbacia, and G. Mogan, "Virtual planning of robot trajectories for spray painting applications," *Applied Mechanics and Materials*, vol. 658, pp. 632–637, 2014.

[13] G. Trigatti, P. Boscariol, L. Scalera, D. Pillan, and A. Gasparetto, "A new path-constrained trajectory planning strategy for spray painting robots - rev.1," *International Journal of Advanced Manufacturing Technology*, vol. 98, pp. 2287–2296, 2018.

[14] X. Ye, L. Luo, L. Hou, Y. Duan, and Y. Wu, "Laser ablation manipulator coverage path planning method based on an improved ant colony algorithm," *Applied Sciences (Switzerland)*, vol. 10, pp. 1–19, 2020.

[15] H. Chen, T. Fuhlbrigge, and X. Li, "A review of cad-based robot path planning for spray painting," *Industrial Robot*, vol. 36, pp. 45–50, 2009.

[16] V. Champagne and D. Helfritch, "Critical assessment 11: Structural repairs by cold spray," *Materials Science and Technology (United Kingdom)*, vol. 31, pp. 627–634, 2015.

[17] A. Moridi, S. M. H. Gangaraj, S. Vezzu, and M. Guagliano, "Number of passes and thickness effect on mechanical characteristics of cold spray coating," *Procedia Engineering*, vol. 74, pp. 449–459, 2014.

[18] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119310–119342, 2021.

[19] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Uniform coverage of automotive surface patches," *International Journal of Robotics Research*, vol. 24, pp. 883–898, 11 2005.

[20] C. H. Chen and K. T. Song, "Complete coverage motion control of a cleaning robot using infrared sensors," *Proceedings of the 2005 IEEE International Conference on Mechatronics, ICM '05*, vol. 2005, pp. 543–548, 2005.

[21] T. K. Lee, S. H. Baek, Y. H. Choi, and S. Y. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robotics and Autonomous Systems*, vol. 59, pp. 801–812, 2011.

[22] J. Song and S. Gupta, "E*: An online coverage path planning algorithm," *IEEE Transactions on Robotics*, vol. 34, pp. 526–533, 2018.

[23] Z. Cai, S. Li, Y. Gan, R. Zhang, and Q. Zhang, "Research on complete coverage path planning algorithms based on a* algorithms," *The Open Cybernetics Systemics Journal*, vol. 8, pp. 418–426, 2014.

[24] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots," *IEEE Access*, vol. 7, pp. 94642–94657, 2019.

[25] P. M. Hsu, C. L. Lin, and M. Y. Yang, "On the complete coverage path planning for mobile robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 74, pp. 945–963, 2014.

[26] B. Luo, Y. Huang, F. Deng, W. Li, and Y. Yan, "Complete coverage path planning for intelligent sweeping robot," *Proceedings of IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers, IPEC 2021*, pp. 316–321, 2021.

[27] C. N. Macleod, G. Dobie, S. G. Pierce, R. Summan, and M. Morozov, "Machining-based coverage path planning for automated structural inspection," *IEEE Transactions on Automation Science and Engineering*, vol. 15, pp. 202–213, 2018.

[28] B. Nasirian, M. Mehrandezh, and F. Janabi-Sharifi, "Efficient coverage path planning for mobile disinfecting robots using graph-based representation of environment," *Frontiers in Robotics and AI*, vol. 8, 2021.

[29] P. Zhou, Z.-M. Wang, Z.-N. Li, and Y. Li, "Complete coverage path planning of mobile robot based on dynamic programming algorithm," *2nd International Conference on Electronic Mechanical Engineering and Information Technology*, pp. 1837–1841, 2012.

[30] A. Bouman, J. Ott, S. K. Kim, K. Chen, M. J. Kochenderfer, B. Lopez, A. A. Agha-Mohammadi, and J. Burdick, "Adaptive coverage path planning for efficient exploration of unknown environments," *In IEEE International Conference on Intelligent Robots and Systems*, pp. 11916–11923, 2022.

[31] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," *International Conference on Robotics and Automation*, pp. 1071–1078, 2015.

[32] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *The International Journal of Robotics Research*, vol. 21, pp. 331–344, 2002.

[33] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, pp. 718–725, 2004.

[34] K. Schmid, H. Hirschmüller, A. Dömel, I. Grixa, M. Suppa, and G. Hirzinger, "View planning for multi-view stereo 3d reconstruction using an autonomous multicopter," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 65, pp. 309–323, 2012.

[35] M. Na, H. J. Jo, and J. B. Song, "Cad-based view planning with globally consistent registration for robotic inspection," *International Journal of Precision Engineering and Manufacturing*, vol. 22, pp. 1391–1399, 2021.

[36] J. G. Mooney and E. N. Johnson, "A comparison of automatic nap-of-the-earth guidance strategies for helicopters," *Journal of Field Robotics*, vol. 33, pp. 1–17, 2014.

[37] J. Jin and L. Tang, "Coverage path planning on three-dimensional terrain for arable farming," *Journal of Field Robotics*, vol. 28, pp. 424–440, 2011.

[38] L. C. Santos, F. N. Santos, E. J. S. Pires, A. Valente, P. Costa, and S. Magalhaes, "Path planning for ground robots in agriculture: A short review," *2020 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2020*, pp. 61–66, 2020.

[39] Y. Li, T. Wu, Y. Xiao, L. Gong, and C. Liu, "Path planning in continuous adjacent farmlands and robust path-tracking control of a rice-seeding robot in paddy field," *Computers and Electronics in Agriculture*, vol. 210, 2023.

[40] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, pp. 651–668, 2009.

[41] R. V. Nanavati, Y. Meng, M. Coombes, and C. Liu, "Generalized data-driven optimal path planning framework for uniform coverage missions using crop spraying uavs," *Precision Agriculture*, pp. 1–29, 2023.

[42] E. Lu, L. Xu, Y. Li, Z. Tang, and Z. Ma, "Modeling of working environment and coverage path planning method of combine harvesters," *International Journal of Agricultural and Biological Engineering*, vol. 13, pp. 132–137, 2020.

[43] I. A. Hameed, A. L. Cour-Harbo, and O. L. Osen, "Side-to-side 3d coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths," *Robotics and Autonomous Systems*, vol. 76, pp. 36–45, 2016.

[44] G. Shang, G. Liu, P. Zhu, and J. Han, "Complete coverage path planning for horticultural electric tractors based on an improved genetic algorithm," *Journal of Applied Science and Engineering (Taiwan)*, vol. 24, pp. 447–456, 2021.

[45] M. Taix, P. Soueres, H. Frayssinet, and L. Cordesses, "Path planning for complete coverage with agricultural machines," *Field and Service Robotics*, vol. 24, pp. 549–558, 2006.

[46] Y. Han, M. Shao, Y. Wu, and X. Zhang, "An improved complete coverage path planning method for intelligent agricultural machinery based on backtracking method," *Information (Switzerland)*, vol. 13, p. 313, 2022.

[47] Z. Wang, B. Zhao, P. Zhang, L. Yao, Q. Wang, B. Li, M. Q. Meng, and Y. Hu, "Full-coverage path planning and stable interaction control for automated robotic

breast ultrasound scanning," *IEEE Transactions on Industrial Electronics*, vol. 70, pp. 7051–7061, 2023.

[48] C. Graumann, B. Fuerst, C. Hennersperger, F. Bork, and N. Navab, "Robotic ultrasound trajectory planning for volume of interest coverage," *In 2016 IEEE international conference on robotics and automation (ICRA)*, pp. 736–741, 2016.

[49] A. Antoniou, A. Georgiou, N. Evripidou, and C. Damianou, "Full coverage path planning algorithm for mrgfus therapy," *International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 18, 2022.

[50] K. Felsner, K. Schlachter, and S. Zambal, "Robotic coverage path planning for ultrasonic inspection," *Applied Sciences (Switzerland)*, vol. 11, 2021.

[51] R. Jahanmahin, S. Masoud, J. Rickli, and A. Djuric, "Human-robot interactions in manufacturing: A survey of human behavior modeling," *Robotics and Computer-Integrated Manufacturing*, vol. 78, 2022.

[52] A. Hentout, M. Aouache, A. Maoudj, and I. Akli, "Human–robot interaction in industrial collaborative robotics: a literature review of the decade," *Advanced Robotics*, vol. 33, pp. 764–799, 2019.

[53] A. B. Moniz and B. J. Krings, "Robots working with humans or humans working with robots? searching for social dimensions in new human-robot interaction in industry," *Societies*, vol. 6, 2016.

[54] P. Rouanet, P. Y. Oudeyer, F. Danieau, and D. Filliat, "The impact of human-robot interfaces on the learning of visual objects," *IEEE Transactions on Robotics*, vol. 29, pp. 525–541, 2013.

[55] P. Akella, M. Peshkin, E. Colgate, W. Wannasuphoprasit, N. Nagesh, J. Wells, S. Holland, T. Pearson, and B. Peacock, "Cobots for the automobile assembly line," *In IEEE International Conference on Robotics and Automation*, vol. 1, pp. 728–733, 1999.

[56] H. Bo, D. M. Mohan, M. Azhar, K. Sreekanth, and D. Campolo, "Human-robot collaboration for tooling path guidance," *Proceedings of the IEEE RAS and EMBS*

International Conference on Biomedical Robotics and Biomechatronics, vol. 2016-July, pp. 1340–1345, 2016.

[57] C. Gaz, E. Magrini, and A. D. Luca, "A model-based residual approach for human-robot collaboration during manual polishing operations," *Mechatronics*, vol. 55, pp. 234–247, 2018.

[58] L. Gammieri, M. Schumann, L. Pelliccia, G. D. Gironimo, and P. Klimant, "Coupling of a redundant manipulator with a virtual reality environment to enhance human-robot cooperation," *Procedia CIRP*, vol. 62, pp. 618–623, 2017.

[59] A. Khan, I. Noreen, and Z. Habib, "On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges," *Journal of Information Science and Engineering*, 2017.

[60] H. Choset, "Coverage for robotics-a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[61] D. Applegate, R. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press,, 2006.

[62] D. Applegate, W. Cook, and A. Rohe, "Chained lin-kernighan for large traveling salesman problems," *Informs journal on computing*, vol. 15, pp. 82–92, 2003.

[63] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, vol. 21. 1987.

[64] M. Fischetti, J. Gonzalez, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, pp. 378–394, 1997.

[65] J. R. Current and D. A. Schilling, "The covering salesman problem," *Source: Transportation Science*, vol. 23, pp. 208–213, 1989.

[66] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 1, p. 53, 1997.

[67] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *New Series*, vol. 220, pp. 671–680, 1983.

[68] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Source: Operations Research*, vol. 21, pp. 498–516, 1973.

[69] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," *Operations Research Forum*, vol. 3, 3 2022.

[70] K. Savla, F. Bullo, and E. Frazzoli, "Traveling salesperson problems for a double integrator," *IEEE Transactions on Automatic Control*, vol. 54, pp. 788–793, 2009.

[71] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, pp. 1378–1391, 2008.

[72] S. N. Spitz and A. A. G. Requicha', "Multiple-goals path planning for coordinate measuring machines," *In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2322–2327, 2000.

[73] M. Saha and J.-C. Latombe, "Planning multi-goal tours for robot arms," *In 2003 IEEE International Conference on Robotics and Automation*, vol. 3, 2003.

[74] C. Wurll and D. Henrich, "Point-to-point and multi-goal path planning for industrial robots," *Journal of Robotic Systems*, vol. 18, pp. 445–461, 2001.

[75] A. Biswas, S. P. Tripathy, and T. Pal, "On multi-objective covering salesman problem," *Neural Computing and Applications*, vol. 34, pp. 22127–22140, 2022.

[76] V. Pandiri, A. Singh, and A. Rossi, "Two hybrid metaheuristic approaches for the covering salesman problem," *Neural Computing and Applications*, vol. 32, pp. 15643–15663, 2020.

[77] R. Bahnemann, N. Lawrance, J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," *In Field and Service Robotics: Results of the 12th International Conference*, pp. 277–290, 2021.

[78] V. Pandiri and A. Singh, "An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem," *Applied Soft Computing Journal*, vol. 78, pp. 481–495, 2019.

[79] J. Xie, L. R. G. Carrillo, and L. Jin, "An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems," *IEEE Control Systems Letters*, vol. 3, pp. 67–72, 2019.

[80] P. T. Kyaw, A. Paing, T. T. Thu, R. E. Mohan, A. V. Le, and P. Veerajagadheswar, "Coverage path planning for decomposition reconfigurable grid-maps using deep reinforcement learning based travelling salesman problem," *IEEE Access*, vol. 8, pp. 225945–225956, 2020.

[81] P. S. Blaer and P. K. Allen, "View planning and automated data acquisition for three-dimensional modeling of complex sites," *Journal of Field Robotics*, vol. 26, pp. 865–891, 2009.

[82] H. Banos and J. Latombe, "A randomized art-gallery algorithm for sensor placement," *In Proceedings of the seventeenth annual symposium on Computational geometry*, pp. 232–240, 2001.

[83] W. R. Scott, G. Roth, and J.-F. O. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Computing Surveys*, vol. 35, pp. 64–96, 2003.

[84] R. Zeng, Y. Wen, W. Zhao, and Y. J. Liu, "View planning in robot active vision: A survey of systems, algorithms, and applications," *Computational Visual Media*, vol. 6, pp. 225–245, 2020.

[85] L. Zacchini, M. Franchi, and A. Ridolfi, "Sensor-driven autonomous underwater inspections: A receding-horizon rrt-based view planning solution for auvs," *Journal of Field Robotics*, vol. 39, pp. 499–527, 2022.

[86] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai, "A survey on inspecting structures using robotic systems," *International Journal of Advanced Robotic Systems*, vol. 13, pp. 1–18, 2016.

[87] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, pp. 1363–1388, 2020.

[88] W.-P. Chin and S. Ntafos, "Shortest watchman routes in simple polygons," *Discrete Computational Geometry*, vol. 6, pp. 9–31, 1991.

[89] W. pang Chin and S. NTAFOS, "Optimum watchman routes," *In Proceedings of the second annual symposium on Computational geometry*, pp. 24–33, 1988.

[90] S. Ntafos, "Watchman routes under limited visibility," *Computational Geometry: Theory and Applications*, vol. 1, pp. 149–170, 1992.

[91] K. A. Tarabanis, "A survev of sensor planning in commter vision," *IEEE Transactions on Robotics and Automation*, vol. 11, 1995.

[92] K. A. Tarabanis, R. Y. Tsai, and P. K. Allen, "The mvp sensor planning system for robotic vision tasks," *IEEE Transactions on Robotics and Automation*, 1996.

[93] G. Olague and R. Mohr, "Optimal camera placement for accurate reconstruction," *Pattern Recognition*, vol. 35, pp. 927–944, 2002.

[94] G. Tarbox and S. Gottschlich, "Planning for complete sensor coverage in inspection," *Computer Vision and Image Understanding*, vol. 61, pp. 84–111, 1995.

[95] W. R. Scott, "Model-based view planning," *Machine Vision and Applications*, vol. 20, pp. 47–69, 2009.

[96] T. Danner and L. E. Kavraki, "Randomized planning for short inspection paths," *In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 971–976, 2000.

[97] J. Faigl, M. Kulich, and L. Přeučil, "A sensor placement algorithm for a mobile robot inspection planning," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 62, pp. 329–353, 2011.

[98] G. D. Kazazakis and A. A. Argyras, "Fast positioning of limited-visibility guards for the inspection of 2d workspaces," *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002.

[99] P. Fazli, A. Davoodi, P. Pasquier, and A. Mackworth, "Complete and robust cooperative robot area coverage with limited range," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5577–5582, 2010.

[100] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, pp. 247–253, 2000.

[101] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," *In Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 1, pp. 27–32, 2001.

[102] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," *In Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 1, pp. 27–32, 2010.

[103] J. S. Oh, Y. H. Choi, J. B. Park, and Y. F. Zheng, "Complete coverage navigation of cleaning robots using triangular-cell-based map," *IEEE Transactions on Industrial Electronics*, vol. 51, pp. 718–726, 2004.

[104] G. Tang, C. Tang, H. Zhou, C. Claramunt, and S. Men, "R-dfs: A coverage path planning approach based on region optimal decomposition," *Remote Sensing*, vol. 13, 2021.

[105] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," *Field and Service Robotics*, pp. 203–209, 1998.

[106] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of mathematics and artificial intelligence*, vol. 31, pp. 77–98, 2001.

[107] D. Latimer, S. Srinivasa, V. Lee-Shue, S. Sonne, H. Choset, and A. Hurst, "Towards sensor based coverage with robot teams," *In Proceedings 2002 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 961–967, 2002.

[108] K. Easton and J. Burdick, "A coverage algorithm for multi-robot boundary inspection," *In Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 727–734, 2005.

[109] P. N. Atkar, H. Choset, A. A. Rizzi, and E. U. Acar, "Exact cellular decomposition of closed orientable surfaces embedded in 3," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, pp. 699–704, 2001.

[110] P. Urhal, A. Weightman, C. Diver, and P. Bartolo, "Robot assisted additive manufacturing: A review," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 335–345, 2019.

[111] M. Özcan and U. Yaman, "A continuous path planning approach on voronoi diagrams for robotics and manufacturing applications," *Procedia Manufacturing*, vol. 38, pp. 1–8, 2019.

[112] H. Zheng, M. Cong, H. Dong, Y. Liu, and D. Liu, "Cad-based automatic path generation and optimization for laser cladding robot in additive manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 92, pp. 3605–3614, 2017.

[113] A. Alhijaily, Z. M. Kilic, and A. N. Bartolo, "Teams of robots in additive manufacturing: a review," *Virtual and Physical Prototyping*, vol. 18, 2023.

[114] M. Beschi, S. Mutti, G. Nicola, M. Faroni, P. Magnoni, E. Villagrossi, and N. Pedrocchi, "Optimal robot motion planning of redundant robots in machining and additive manufacturing applications," *Electronics (Switzerland)*, vol. 8, 2019.

[115] T. Felsch, F. Silze, and M. Schnick, "Process control for robot based additive manufacturing," *International Conference on Emerging Technologies and Factory Automation*, p. 1840, 2019.

[116] J. N. Pires and A. S. Azar, "Advances in robotics for additive/hybrid manufacturing: robot control, speech interface and path planning," *Industrial Robot*, vol. 45, pp. 311–327, 7 2018.

[117] A. D. Marzi, M. Vibrante, M. Bottin, and G. Franchin, "Development of robot assisted hybrid additive manufacturing technology for the freeform fabrication of lattice structures," *Additive Manufacturing*, vol. 66, 2023.

[118] L. Evjemo, S. Moe, J. Gravdahl, and D. Olivier, "Additive manufacturing by robot manipulator: Anoverview of the state-of-the-art andproof-of-concept results," *International Conference on Emerging Technologies and Factory Automation*, 2017.

[119] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

[120] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, pp. 846–894, 6 2011.

[121] M. Weghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," *IEEE-RAS International Conference on Humanoid Robots*, 2007.

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

[123] M. Shimizu, H. Kakuya, W. K. Yoon, K. Kitagaki, and K. Kosuge, "Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution," *IEEE Transactions on Robotics*, vol. 24, pp. 1131–1142, 2008.

[124] Z. Kingston, M. Moll, and L. E. Kavraki, "Decoupling constraints from sampling-based planners appeared in int . symp . robotics research 2017," *International Symposium of Robotics Research*, pp. 1–16, 2017.

[125] M. Stilman, "Task constrained motion planning in robot joint space," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3074–3081, 2007.

[126] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. i, pp. 625–632, 2009.

[127] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Transactions on Robotics*, vol. 29, pp. 105–117, 2013.

[128] T. McMahon, S. Thomas, and N. M. Amato, "Sampling-based motion planning with reachable volumes: Theoretical foundations," *International Journal of Robotics Automation*, pp. 6514–6521, 2014.

[129] A. H. Qureshi, J. Dong, A. Baig, M. C. Yip, and S. Member, "Constrained motion planning networks x," vol. 38, pp. 868–886, 2022.

[130] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[131] Z. Yao and K. Gupta, "Path planning with general end-effector constraints: Using task space to guide configuration space search," *International Conference on Intelligent Robots and Systems*, 2005.

[132] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, pp. 576–584, 2010.

[133] J. J. Kuffner, "Rrt-connect : An efficient approach to single-query path planning," *In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, vol. 2, pp. 995–1001, 2000.

[134] J. Gammell, S. Srinivasa, and T. Barfoot, "Informed rrt*:optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[135] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, vol. 68, pp. 1–11, 2015.

[136] L. Jaillet and J. M. Porta, "Path planning with loop closure constraints using an atlas-based rrt," *Robotics Research*, vol. 100, 2017.

[137] S. LaVelle, "Rapidly-exploring random trees: A new tool for path planning," *Mathematics*, 1998.

[138] L. Jaillet and J. M. Porta, "Efficient asymptotically-optimal path planning on manifolds," *Robotics and Autonomous Systems*, vol. 61, pp. 797–807, 2013.

[139] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, pp. 48–66, 2021.

[140] M. Tarini, "Volume-encoded uv-maps," *ACM Transactions on Graphics*, vol. 35, pp. 1–13, 7 2016.

[141] Pixar, "Generating uv maps for modified meshes," 2019.

[142] R. Poranne, M. Tarini, S. Huber, D. Panozzo, and O. Sorkine-Hornung, "Autocuts: Simultaneous distortion and cut optimization for uv mapping," *ACM Transactions on Graphics*, vol. 36, 2017.

[143] B. Purnomo, J. D. Cohen, and S. Kumar, "Seamless texture atlases," *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 65–74, 2004.

[144] N. Aigerman, R. Poranne, and Y. Lipman, "Seamless surface mappings," *ACM Transactions on Graphics*, vol. 34, pp. 1–13, 2015.

[145] S. Liu, Z. Ferguson, A. Jacobson, and Y. Gingold, "Seamless: seam erasure and seam-aware decoupling of shape from mesh resolution," *ACM Transactions on Graphics*, 2017.

[146] R. Toth, "Avoiding texture seams by discarding filter taps," *Journal of Computer Graphics Techniques*, vol. 2, pp. 91–103, 2013.

[147] N. Ray, V. Nivoliers, S. Lefebvre, and B. Lévy, "Invisible seams," *Computer Graphics Forum*, vol. 29, pp. 1489–1496, 2010.

[148] G. Bruno and D. Antonelli, "Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells," *International Journal of Advanced Manufacturing Technology*, vol. 98, pp. 2415–2427, 2018.

[149] H. Ding, M. Schipper, and B. Matthias, "Optimized task distribution for industrial assembly in mixed human-robot environments –case study on io module assembly," *In 2014 IEEE international conference on automation science and engineering*, pp. 19–24, 2014.

[150] T. Koltai, I. Dimény, V. Gallina, A. Gaal, and C. Sepe, "An analysis of task assignment and cycle times when robots are added to human-operated assembly lines, using mathematical programming models," *International Journal of Production Economics*, vol. 242, 2021.

[151] A. Nourmohammadi, M. Fathi, and A. H. Ng, "Balancing and scheduling assembly lines with human-robot collaboration tasks," *Computers and Operations Research*, vol. 140, 2022.

[152] T. Kopp, M. Baumgartner, and S. Kinkel, "Success factors for introducing industrial human-robot interaction in practice: an empirically driven framework," *The International Journal of Advanced Manufacturing Technology*, vol. 112, pp. 685–704, 2021.

[153] M. Klumpp, M. Hesenius, O. Meyer, C. Ruiner, and V. Gruhn, "Production logistics and human-computer interaction—state-of-the-art, challenges and requirements for the future," *International Journal of Advanced Manufacturing Technology*, vol. 105, pp. 3691–3709, 2019.

[154] F. Ore, B. R. Vemula, L. Hanson, and M. Wiktorsson, "Human - industrial robot collaboration: Application of simulation software for workstation optimisation," *Procedia CIRP*, vol. 44, pp. 181–186, 2016.

[155] J. Kofman, X. Wu, T. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, vol. 52, pp. 1206–1219, 2005.

[156] M. C. Bingol and O. Aydogmus, "Practical application of a safe human-robot interaction software," *Industrial Robot*, vol. 47, pp. 359–368, 2020.

[157] M. Hasanuzzaman, T. Zhang, V. Ampornaramveth, H. Gotoda, Y. Shirai, and H. Ueno, "Adaptive visual gesture recognition for human-robot interaction using a knowledge-based software platform," *Robotics and Autonomous Systems*, vol. 55, pp. 643–657, 2007.

[158] R. Schiavi, A. Bicchi, and F. Flacco, "Integration of active and passive compliance control for safe human-robot coexistence," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 259–264, 2009.

[159] Y. Zhu, C. Yang, Q. Wei, X. Wu, and W. Yang, "Human–robot shared control for humanoid manipulator trajectory planning," *Industrial Robot*, vol. 47, pp. 395–407, 2020.

[160] S. Haddadin, A. A.-S. Signffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *International Journal of Robotics Research*, vol. 28, pp. 1507–1527, 2009.

[161] A. Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas, "Safety bounds in human robot interaction: A survey," *Safety Science*, vol. 127, p. 104667, 2020.

[162] N. Wang, Y. Zeng, and J. Geng, "A brief review on safety strategies of physical human-robot interaction," *In ITM Web of Conferences*, vol. 25, 2019.

[163] C. Zhou, F. Yuan, T. Huang, Y. Zhang, and J. Kaner, "The impact of interface design element features on task performance in older adults: Evidence from eye-tracking and eeg signals," *International Journal of Environmental Research and Public Health*, vol. 19, 2022.

[164] J. Zhang, Z. Wang, M. Zhou, Y. Xu, L. Zhong, H. Du, L. Wang, and Y. Wu, "A study of the impact of changes in software interface design elements on visual fatigue," vol. 1419, pp. 182–188, Springer Science and Business Media Deutschland GmbH, 2021.

[165] T. Nomura, T. Kanda, T. Suzuki, and K. Kato, "Prediction of human behavior in human - robot interaction using psychological scales for anxiety and negative attitudes toward robots," *IEEE Transactions on Robotics*, vol. 24, pp. 442–451, 2008.

[166] S. Haddadin, A. Albu-Schäffer, A. D. Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 3356–3363, 2008.

[167] A. Fenucci, M. Indri, and F. Romanelli, "A real time distributed approach to collision avoidance for industrial manipulators," *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, pp. 3–10, 2014.

[168] F. Flacco, T. Kröger, A. D. Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 338–345, 2012.

[169] S. McGovern and J. Xiao, "Uv grid generation on 3d freeform surfaces for constrained robotic coverage path planning," *IEEE International Conference on Automation Science and Engineering*, pp. 1503–1509, 2022.

[170] S. McGovern and J. Xiao, "Efficient feasibility checking on continuous coverage motion for constrained manipulation," *IEEE International Conference on Automation Science and Engineering*, pp. 189–195, 2021.

[171] S. Mcgovern and J. Xiao, "A general approach for constrained robotic coverage path planning on 3d freeform surfaces," *Conditionally Accepted to Transactions on Automation Science and Engineering*, 2023.

[172] S. McGovern, H. Mao, and J. Xiao, "Learning to estimate centers of mass of arbitrary objects," *In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1848–1853, 2019.

[173] S. McGovern and J. Xiao, "Learning and predicting center of mass through manipulation and torque sensing," *In 2022 8th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pp. 60–66, 2022.

[174] P. N. Atkar, H. Choset, and A. A. Rizzi, "Towards optimal coverage of 2-dimensional surfaces embedded in r3: Choice of start curve," *In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 4, 2003.

[175] T. Yang, J. V. Miro, Q. Lai, Y. Wang, and R. Xiong, "Cellular decomposition for non-repetitive coverage task with minimum discontinuities," *IEEE/ASME Transactions on Mechatronics*, vol. 25, pp. 1698–1708, 2020.

[176] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Hierarchical segmentation of surfaces embedded in r for auto-body painting," *Proceedings of the 2005 IEEEInternational Conference on Robotics and Automation Barcelona, Spain, April 20050*, 2005.

[177] C. Cao, J. Zhang, M. Travers, and H. Choset, "Hierarchical coverage path planning in complex 3d environments," *In 2020 IEEE International Conference on Robotics and Automation*, 2020.

[178] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Pearson Education, Inc., 3rd ed., 2005.

[179] L. D. Nielsen, I. Sung, and P. Nielsen, "Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges," *Sensors (Switzerland)*, vol. 19, pp. 1–11, 2019.

[180] M. Ramesh, F. Imeson, B. Fidan, and S. L. Smith, "Optimal partitioning of non-convex environments for minimum turn coverage planning," *IEEE Robotics and Automation Letters*, vol. 7, pp. 9731–9738, 2022.

[181] A. Saric, J. Xiao, and J. Shi, "Robotic surface assembly via contact state transitions," *In 2013 IEEE International Conference on Automation Science and Engineering*, pp. 954–959, 2013.