Automatic Collision-Free Trajectory Generation and Optimal Base Placement for Collaborative Robotic Car-Painting

Khalil Zbiss

A dissertation proposal submitted for the degree of
Doctor of Philosophy
Electrical, Electronics and Computer Engineering
University of Michigan – Dearborn
2025

Doctoral Committee:
Associate Professor Alireza Mohammadi, **Chair**
Professor Selim Awad
Assistant Professor Zheng Song
Associate Professor Paul Watta

Khalil Zbiss

kzbiss@umich.edu

ORCID iD: 0000-0001-6091-6051

https://orcid.org/0000-0001-6091-6051

© 2025 Khalil Zbiss.

All rights reserved.

No part of this document may be reproduced, stored in a retrieval system,

or transmitted in any form or by any means, without prior written permission.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Professor Alireza Mohammadi, for his unwavering support and guidance throughout my Ph.D. I will always be grateful for his involvement in every step of my academic journey. Without his generosity and mentorship, I would not have achieved this milestone. I am honored to have had him as my mentor.

I would also like to extend my appreciation to the members of my dissertation committee, Dr. Selim Awad, Dr. Paul Watta, and Dr. Zheng Song. Their feedback was invaluable throughout my Ph.D. journey.

Furthermore, I am grateful for the financial support from the University of Michigan-Dearborn's College of Engineering and Computer Science, and the Department of Electrical and Computer Engineering. Their support enabled me to pursue my research without financial burden. I also want to thank Ford Motor Company for providing the challenging project that is the subject of this thesis, and Dr. Mario Santillo for his valuable feedback.

Many thanks to everyone who supported and encouraged me both inside and outside the research environment, including my friends from all over the world and our Tunisian community in Michigan.

I am deeply grateful to my wife, Amal, who has been my rock throughout this incredible journey. As my lab partner and confidant, she has shared every moment of this experience with me. Her support, both emotional and intellectual, has been crucial in reaching this milestone. Working with her has been a wonderful experience,

and our joint efforts made this journey easier. I am grateful for her love and sacrifices. This achievement is as much hers as it is mine, and I couldn't have done it without her.

Finally, I wish to express my deepest love and gratitude to my parents, Adnane and Kaouthar, and my brother Maher, who have supported me financially and emotionally throughout this journey. My parents have always believed in my abilities and have provided me with the foundational values of perseverance and dedication. My brother Maher has been a constant source of encouragement and inspiration. This academic endeavor would not have been possible without them by my side.

PREFACE

This thesis presents research on collision-free trajectory generation and optimal base placement for collaborative industrial robots, with a focus on robotic car painting. The work bridges the gap between robotics, automation, and optimization, offering practical solutions for improving the efficiency and safety of multi-robot systems. Through this work, we sought to develop computationally efficient solutions that integrate optimization algorithms and trajectory planning techniques to improve the performance and safety of industrial automation systems.

The research reflects my academic journey and the insights gained through collaboration with my advisors, fellow researchers, and industry professionals. It is a result of continuous exploration of challenges in flexible manufacturing and multi-robot coordination.

The following section presents a comprehensive list of my publications during my PhD. The first two papers are closely tied to the central theme of this dissertation, which focuses on collision-free trajectory generation and optimal base placement for collaborative robotic car painting. These publications address the challenges of coordinating multiple robots with overlapping workspaces and propose innovative solutions to ensure safe, efficient, and automated operations. Specifically, they explore how geometric modeling, optimization algorithms, and motion planning techniques can be applied to avoid collisions and improve task execution.

The final four publications included in this list represent research I contributed to outside the scope of this dissertation. These papers highlight my broader work within

the field of robotics, covering topics such as the sonification of the motion of robots. These publications delve into the challenges of monitoring the kinematic behavior of robots with many degrees of freedom and propose innovative sonification techniques as an alternative to traditional visual methods. Specifically, they explore how auditory feedback can be effectively applied to hyper-redundant robotic mechanisms, providing intuitive ways to interpret complex motion patterns and improve system monitoring.

List of Publications

1. K Zbiss, A Kacem, M Santillo, A Mohammadi, 2022, *Automatic collision-free trajectory generation for collaborative robotic car-painting*, IEEE Access 10, 9950-9959 [1]. **Discussed in Chapter IV**

2. K Zbiss, A Kacem, M Santillo, A Mohammadi, 2024, *Automatic Optimal Robotic Base Placement for Collaborative Industrial Robotic Car Painting*, Applied Sciences 14 (19), 8614 [2] **Discussed in Chapter V**

3. A Kacem, K Zbiss, A Mohammadi, 2023, *A Numerical Integrator for Forward Dynamics Simulations of Folding Process for Protein Molecules Modeled as Hyper-Redundant Robots*, 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023) [3]

4. A Kacem, K Zbiss, A Mohammadi, 2024, *A Numerical Integrator for Kinetostatic Folding of Protein Molecules Modeled as Robots with Hyper Degrees of Freedom*, Robotics 13 (10), 150 - [4]

5. A Kacem, K Zbiss, A Mohammadi, 2024, *Wave space sonification of the folding pathways of protein molecules modeled as hyper-redundant robotic mechanisms*, Multimedia Tools and Applications 83 (2), 4929-4949 [5]

6. A Kacem, K Zbiss, A Mohammadi, 2024, *Transforming Motion Into Sound: A Novel Sonification Approach for Teams of Mobile Robots*, International Symposium on Flexible Automation 87882, V001T07A005 [6]

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

PREFACE iv

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF ABBREVIATIONS xiv

ABSTRACT xv

CHAPTER

I. Introduction 1

1.1 Collaborative Robots 1
1.2 Coverage Path Planning 2
1.3 Robots' Base Placement 3
1.4 Motivation and Contribution 4
1.5 Dissertation Organization 6

II. Background 8

2.1 Robotic Painting 8
2.2 Collision avoidance in Collaborative Robotic Manipulation . . 9
2.3 Coverage Path Planning 11
2.3.1 Heuristic Algorithms 12
2.3.2 Cellular Decomposition 12
2.3.3 Contact sensor-based coverage 15
2.3.4 Grid-Based Methods 15
2.4 Robot Kinematics 16
2.4.1 The Jacobian Matrix 17
2.4.2 Forward Kinematics 17
2.4.3 Inverse Kinematics 19

2.4.4 Robot Manipulability 22
2.5 Multi-Robot Path Planning 24

III. Problem Statement 26

3.1 Assumptions 27
3.2 Solution Strategy 28

IV. Automatic Collision-Free Trajectory Generation for Collaborative Industrial Robot 30

4.1 Achieving Collision Avoidance 30
4.1.1 Approximating The Reachable Workspace of Robot Forearms Using 3D Point Clouds 30
4.1.2 Finding The Point Clouds of Intersection in-Between The Generated 3D Point Clouds 34
4.1.3 Fitting Ellipsoids To The Volume of Intersection . . 36
4.2 Achieving The Paint Coverage 38
4.2.1 Projection of the Fitted Ellipsoids Onto The Car CAD Model 39
4.2.2 Decomposition of the Vehicle Grid Map Into Cells Using the Morse Critical Points 39
4.3 Task Assignment and Generation of Robotic Joint Trajectories 44
4.3.1 Assigning Robotic Tasks 44
4.3.2 3D Path Generation For the Paint Nozzles and Generation of Joint Trajectories For the Robotic Arms . 46
4.4 Overall Algorithm and Simulation Results 47

V. Optimal Robotic Base Placement for Collaborative Industrial Robot 51

5.1 Related Work 51
5.1.1 Robot Base Placement 52
5.1.2 Robot Workspace Computation 53
5.1.3 Multiobjective Optimization 55
5.2 The Base Placement Problem and High-Level Solution Strategy 57
5.3 Point Cloud Rendering and Ellipsoidal Fitting 58
5.3.1 Robot Workspace Point Cloud Generation 60
5.3.2 Ellipsoidal Fitting to the Robot Workspace Point Clouds 62
5.3.3 Computational Complexity for Point Cloud Generation and Ellipsoid Fitting 64
5.4 Multi-Objective Optimization Problem 65
5.4.1 Cost Function 1: Minimizing the Possibility of Collision 65

5.4.2 Cost Function 2: Maximizing the Possibility of Paint Coverage 69
5.4.3 Multi-Objective Optimization Problem 72
5.5 Results 75
5.5.1 Simulations with a Homogeneous Robotic Team . . 76
5.5.2 Simulations with a Heterogeneous Robotic Team . . 80
5.6 Discussion and comparison with the state of the art 81

VI. Future Work and Conclusion 85

APPENDICES 88
.1 CoppeliaSim Simulation Code 89
.2 Ellipse Overlap Area of Intersection Code 98

BIBLIOGRAPHY 101

LIST OF TABLES

Table

4.1	DH parameters of ABB IRB4600	32
5.1	A sample overview of robotic base placement algorithms proposed in the literature.	54
5.2	Impact of robotic base positions on paint coverage and manipulability with the homogeneous robotic team consisting of ABB IRB 4600 robots. The blue entries demonstrate the results obtained from our proposed optimal base placement algorithm.	75
5.3	Computational times.	77

LIST OF FIGURES

Figure

2.1 Cellular decomposition 12
2.2 Fewer cells mean more efficient path planning. 13
2.3 Difference between trapezoidal decomposition and boustrophedon decomposition 14
2.4 Morse decomposition 15
3.1 Example of the CAD model of a Ford F-150 truck 26
3.2 Mutli-robot car-painting base placement problem. The base positions directly impact the quality of vehicle paint coverage and the possibility of robot collisions during painting. For instance, robots placed on golden diamonds are more prone to collisions. On the other hand, robots placed on red diamonds will not be able to ensure complete vehicle paint coverage. 27
3.3 Example of an RRR articulated arm. The robot arm in the figure is an ABB IRB 4600 industrial manipulator.The unit lengths are in millimeters 28
3.4 Example of the arrangement of two rows of three robotic arms around the vehicle. 29
4.1 Point cloud representing the reachable workspace of ABB IRB 4600 manipulator. 33
4.2 Example of the point clouds representing the reachable workspace by the forearms of six ABB IRB 4600 manipulators. The point clouds in black represent areas of possible collisions between the robots. The axis units are in decimeters (Note: 1 meter = 10 decimeters). 34
4.3 Top view of the point clouds representing the reachable workspace by the forearms of 4 ABB IRB 4600 manipulators. 35
4.4 A 2D top view and a 3D view of the point cloud of intersection in a 4 robot setup. 36
4.5 A 2D top view and a 3D view of the MVEE of the point cloud of intersection in a 4 robot setup. 37
4.6 Grid map representation of the CAD model of a Ford F-150 truck . 38
4.7 Example of projection of the fitted ellipsoids onto the grid map representation of the sides and top of the vehicle CAD model. 39

4.8 Example of a cellular decomposition of the top of the car for a 4-robot setup 40
4.9 Example of assigning back-and-forth (boustrophedon-like) motions on the side and top of the vehicle for the applicator/nozzle footprint on the grid maps. Each colored boustrophedon path belongs to one of the six robots working on one side and top of the car 41
4.10 Example of assigning back-and-forth (boustrophedon-like) motions on the top of the vehicle for the applicator/nozzle footprint of the areas without projection of the ellipsoids (on the left) and those with the projection of volume intersections as voids (on the right) 42
4.11 Example of assigning back-and-forth (boustrophedon-like) motions on the top of the vehicle for the applicator/nozzle footprint for R1 and R2. black-colored applicator/nozzle footprint represents the trajectory that was not yet assigned to any robot 43
4.12 Example of final set of collision-free nozzle/applicator footprint paths on the side and top of the vehicle. black-colored applicator/nozzle footprints represent the trajectory that was not yet assigned to any robot 43
4.13 Example of assignment of painting tasks to the robots, which is done based on computing the Euclidean distances in-between the centroids of the reachable workspace point clouds of the robots and the centroids of the boustrophedon-like motions obtained from Stage 2-Step 2. Each colored collection of paths corresponds to one robot. For instance, the red paths on the side and the top of the car are assigned to a single robot. 45
4.14 Example of generating 3D boustrophedon-like paths of the paint nozzle 3D footprints for four and six robots. 46
4.15 The flowchart associated with the proposed algorithm for CRCP. . . 47
4.16 Snapshots of six ABB IRB 4600 robots painting an F-150 truck. . . 48
4.17 Snapshots of two ABB IRB 4600 and two FANUC R-2000iC/220U robots painting an F-150 truck. 49
5.1 Algorithm 1 generates a 3D point cloud and a minimum volume enclosing ellipsoid (MVEE) for each of the robotic arms. 63
5.2 Intuition behind the first cost function formulation. The ellipsoids in the plots are the MVEEs that closely fit the reachable workspace of the robot forearms. The larger the volume of the intersection of the ellipsoids, the higher the possibility of robot collision during painting. Larger intersection volumes between the 3D ellipsoids, in turn, correspond to larger overlap areas of their projections on the factory floor. 65
5.3 The planar ellipse overlap areas are employed to define the first cost function, which quantifies the possibility of collision in-between the robotic arms during painting. 66

5.4 The algorithm proposed by [7] distinguishes a variety of different ellipse intersection cases. It computes the overlap area of any two general ellipses without resorting to proxy curves. 68
5.5 Boundary detection in the vehicle's CAD model and creation of the grip map representations. 70
5.6 The overall proposed solution to the multi-robot car-painting base placement problem. 72
5.7 Snapshots of multi-robot car painting (a homogeneous team of robots) utilizing the optimal robot base positions obtained from our base placement algorithm. 78
5.8 Snapshots of multi-robot car painting (a homogeneous team of robots) utilizing non-optimal base positions. 79
5.9 The normalized manipulability of the robotic arms during collaborative vehicle painting: (top) optimal; and, (bottom) non-optimal base positions. The percentage values on each bar in the histograms provide the painting durations in which the robot manipulability belonged to a certain interval. In particular, a percentage value of $p_0\%$ on a bar located in the interval $[\sigma_1, \sigma_2]$ for the i^{th} robot, $1 \leq i \leq 3$, indicates that the robot had a normalized manipulability value belonging to the interval $[\sigma_1, \sigma_2]$ during $p_0\%$ of the painting task duration. 80
5.10 Snapshots of multi-robot car painting (a heterogeneous team of robots) utilizing the optimal robot base positions obtained from our base placement algorithm. 81
5.11 Snapshots of multi-robot car painting (a heterogeneous team of robots) utilizing non-optimal base positions. 81
5.12 Elbow singular configurations of (left) ABB IRB 4600 and (right) FANUC R-2000iC robotic arms. 82

LIST OF ABBREVIATIONS

CRCP Collaborative Robotic Car-Painting

CAD Computer-Aided Design

CPP Coverage Path Planning

TSP Traveling Salesmen Problem

mTSP multiple Traveling Salesmen Problem

GA Genetic Algorithm

ACO Ant Colony Optimization

PSO Particle Swarm Optimization

FK Forward Kinematics

IK Inverse Kinematics

MVEE Minimum Volume Enclosing Ellipsoid

DLS Damped Least Squares

SQP Sequential Quadratic Programming

NSGA-II Non-dominated Sorting Genetic Algorithm II

SPEA2 Strength Pareto Evolutionary Algorithm 2

ABSTRACT

by

Khalil Zbiss

This study delves into the challenges associated with collaborative robotic car painting, employing a team of industrial manipulators with potential heterogeneity. The focus lies in two primary areas: efficiently generating joint trajectories for each robot and finding the optimal fixed base positions within the factory space. Using the given CAD model of the car, the objective is to maximize paint coverage while minimizing the risk of collisions during the painting process, using nozzles attached to the robotic arms.

Our proposed solution tackles the computationally demanding task of collaborative coverage path planning by strategically separating collision avoidance from coverage path planning, leveraging the intrinsic two-dimensional structure of the problem. We partition the reachable space of the forearms of the robots, project the resulting volumes of intersection onto the sides and top of the car body, and subsequently conduct the coverage planning on these projected volumes.

Additionally, we construct two types of cost functions that formally capture the notions of paint coverage maximization and collision avoidance minimization. These functions are used to formulate a multi-objective optimization problem, which is solvable using any standard multi-objective optimizer. Our optimal base placement algorithm decouples base placement from motion/trajectory planning, offering a hierarchical solution where the generated results can be utilized within existing robotic painting motion/trajectory planners.

The effectiveness of our approach is demonstrated through simulation results involving several industrial robotic arms collaboratively painting a Ford F-150 truck, validating both the innovative trajectory planning and optimal base placement strategies.

CHAPTER I

Introduction

Over the last few years, multi-robot collaborative systems have made major progress in modern manufacturing. Modern warehouses are using robots to use fleets of robots that can communicate with each other to quickly and efficiently pick, sort, and deliver packages to reduce the time that it takes to execute each task. Robots are also widely used in vehicle assembly. They can pass parts to each other, weld, and paint in a perfectly timed sequence without human intervention. If one robot needs to stop, others can adjust their tasks to keep the assembly line moving smoothly, ensuring there are little to no delays in production.

1.1 Collaborative Robots

Collaborative robotics in flexible manufacturing involves a team of robots working together to handle complex tasks beyond what a single robot can do. A team of robots is crucial for tasks like intricate assembly or dealing with uncertainties in processes such as welding, painting, or coating. In automated planned assemblies, multiple robots collaborate to streamline the manufacturing process. This teamwork not only boosts efficiency but also addresses challenges in modern manufacturing [8; 9; 10; 11; 12; 13].

Collaborative robotics aims to break away from traditional repetitive industrial

processes by focusing on flexible manufacturing systems. Unlike the rigid nature of conventional methods, collaborative robotics targets agile customization of production lines. In scenarios like flexible manufacturing-based car painting, where a robotic arm may malfunction, the goal is to ensure uninterrupted operations. In such cases, replacing the non-functional arms with different types from the inventory becomes a practical solution, allowing the painting process to continue seamlessly. In another scenario, the manufacturer might need to change the make of vehicles in their production line on short notice. In both scenarios, a new collection of trajectories needs to be planned in the least possible amount of time to address the agile manufacturing requirements.

Planning the motion of a team of robots requires the computation of paths for each robot, intending to achieve optimal or nearly optimal paths individually while ensuring the collective path of all robots is optimal. This task is more complex than single-robot motion planning, where coordination among robots is not a concern, and each robot can independently compute its path.

1.2 Coverage Path Planning

Path planning is an important challenge in robotics, which aims to process high-level actions into possible low-level motor instructions. Coverage Path Planning (CPP) consists of identifying a route that traverses every point within a designated area or volume, ensuring avoidance of obstacles. [14] defined the requirements for the coverage of a mobile robot moving in a 2-dimensional environment that are also applicable for robot manipulators:

- The robot must move through every point and cover the entire target area without overlapping or repeating paths.
- The robot must avoid all obstacles.

- The robot preferably has simple motion trajectories for simplicity in control. (e.g. straight lines)
- Under the available conditions, it is desirable to seek an "optimal" path.

The CPP problem is a variant of the well-known traveling salesman problem where a robot needs to visit a set of target points and return to its starting point, all while minimizing the total distance traveled. The main objective of this NP-hard problem is to find the most efficient path that allows the robot to visit each point exactly once. The computational time required to solve this problem increases exponentially when the number of target points increases.

1.3 Robots' Base Placement

Robot base placement refers to the robot's fixed bases within a workspace to optimize its performance in a collaborative task. The positioning of a robot base is an essential component of robotics that has a big impact on the functionality efficacy and performance of robotic systems in various applications. The location of a robot's base is a fundamental factor that dictates its operational range stability and interaction with the environment regardless of the application—industrial automation service robotics or autonomous systems.

Base placement is crucial as it affects the robot's ability to perform tasks correctly and successfully. Finding the optimal base position is key to significantly reducing the time and energy required to complete a task. An efficiently placed base minimizes unnecessary movement, which directly contributes to better energy efficiency and faster task completion.

Despite its importance, the optimal position for robot bases is not a straightforward task and depends on several variables, including the workspace's shape, the tasks to be completed, the kinematic and dynamic capabilities of the robots, and any

environmental restrictions like limited space or barriers. An optimal base placement allows for coordination between the robots and guarantees that every robot can reach its target areas with the least amount of effort and maximum precision.

The optimal base placement must also consider task-specific requirements. For example, robots performing different operations—such as painting or assembly—may need distinct base positions even within the same workspace. Environmental factors, such as floor conditions and the presence of obstacles, must be accounted for to ensure stable and efficient operation.

1.4 Motivation and Contribution

The increasing demand for flexible and efficient manufacturing processes has led to significant advancements in multi-robot collaborative systems. Traditional robotic systems often struggle with time-consuming trajectory generation and inefficient base placement, which can lead to delays and suboptimal performance. Our main interest is to address these challenges by developing computationally efficient algorithms for both optimal base placement and automatic collision-free trajectory generation in multi-robot systems. Our research aims to directly address these limitations by developing computationally efficient algorithms focused on two key areas: optimal base placement and automatic collision-free trajectory generation for multi-robot systems.

One of the motivations behind this research is the ability to dynamically replace robots in the manufacturing setup if a breakdown occurs. In a highly automated industrial environment, the sudden failure of a robotic arm can lead to significant delays and interruptions in the production line. To mitigate this, our approach focuses on developing algorithms that allow for the seamless integration of a new robot—either of the same make or any standard 6-DOF (Degrees of Freedom) robot—without the need to reconfigure the entire system. This flexibility not only ensures that produc-

tion continues with minimal disruption but also enables manufacturers to utilize a diverse range of robotic arms available in their inventory.

We propose a computationally efficient approach to the multi-robot car-painting problem. In the following paragraphs, we summarize the main contributions of our work.

The first contribution is founded on proposing a computationally efficient, hierarchical, and versatile path-planning solution that relies on exploiting the geometrical features of the car-painting problem. Our solution to this computationally intensive problem relies on decoupling the collision avoidance objective in the 3D environment from the coverage path planning objective on the 2D surface of the car while computing the solution in polynomial time complexity. Furthermore, our solution does not require any communication in-between the robotic arms to prevent collision avoidance. The second contribution is developing two types of cost functions specifically designed for multi-robot vehicle painting scenarios. These cost functions can be efficiently computed iteratively. Consequently, they are well suited for being utilized within any standard multi-objective optimization algorithm. A key advantage of our proposed approach is that it primarily involves planar computations, except for a one-time three-dimensional (3D) point cloud rendering and ellipsoidal fitting in its pre-processing stage.

Considering the literature gaps, this study exclusively focuses on the collaborative robotic car-painting application. This specific focus allows us to deviate from generic base placement methods, which try to perform base placement and trajectory planning simultaneously. Specifically, we leverage the unique geometric properties of this industrial automation process to develop a computationally efficient algorithm without the need for simultaneous path planning and base placement. Our hierarchi-

cal approach prioritizes determining optimal base positions before engaging existing robotic painting motion/trajectory planners. The core challenge lies in quantifying paint coverage maximization and collision avoidance minimization for multi-robot vehicle painting without relying on prior knowledge or real-time data from motion/trajectory planners. Furthermore, this quantification requires the design of cost functions tailored to the collaborative robotic car-painting application, which can be efficiently computed iteratively, hence making them well suited for being utilized within any standard multi-objective optimization algorithm.

1.5 Dissertation Organization

The remainder of the dissertation is organized as follows. Chapter II introduces the necessary background on robot kinematics and path planning, setting the foundation for the computational solutions developed for efficient and collaborative robotic car painting. Chapter III defines the central problem, outlines the key assumptions, and describes the step-by-step solution strategy that guides the development of our approach.

Chapter IV focuses on the generation of automatic robot trajectories. We start by Approximating the reachable Workspace of Robot Forearms. Then, we find the point clouds of intersection in between the generated 3D point cloud and decompose the vehicle grid map into cells using the Morse critical points decomposition algorithm and applying the boustrophedon-like paths to these cells. The chapter concludes by presenting how to create the 3D paths for the robots' paint nozzles and assign each painting task to the appropriate robot. We finish this chapter with a summary of the algorithm and present the simulation result for a team of 4 and 6 robots.

Chapter V addresses the optimization of the robots' base placements. We render a point cloud representation of the robot workspace. We define the cost functions, one for minimizing the possibility of collisions and another for maximizing the possibility

of paint coverage. We define the multi-objective optimization problem and present the simulation results for a homogeneous and heterogeneous robotic team. Lastly, we present a discussion and comparison with the state-of-the-art methods.

Chapter VI is devoted to summarizing and discussing our works. After that, the thesis concludes with a discussion about possible future works.

CHAPTER II

Background

2.1 Robotic Painting

Robotic spray painting, which is concerned with covering the object surface by means of varnish/paint/ink, utilizes spray guns that are driven by compressed air. These spray guns atomize the fluid particles and direct them onto the industrial target surfaces [15; 16; 17]. A drawback of the conventional industrial painting automation systems using robots is the time-consuming nature of trajectory generation for the robotic arms [18; 19]. In order to depart from the time-consuming procedure of direct manual tool path operation, which is based on driving the robot arm by a human operator through a complete spraying cycle, CAD-based (Computer-Aided Design) methods have been considered a viable and less time-consuming alternative for generating proper trajectories for the painting robots [20]. In these CAD-based methods, path planning for single industrial robots can be performed using alternatives such as RRT-based algorithms [21], iterative schemes with adaptively changing spray-gun stroke diameters [22], and path-constrained trajectory planning [18], to name a few.

2.2 Collision avoidance in Collaborative Robotic Manipulation

In the context of collaborative robotic manipulation, a plethora of modeling, control, and optimization algorithms have been developed in the literature. Fuzzy genetic algorithms have been employed in [23] or trajectory planning for two collaborative manipulators, where the robots considered each other as moving obstacles with unknown behaviors. After the offline generation of the robot's initial paths using genetic algorithms, an online fuzzy algorithm-based decision maker corrects the robot paths to prevent them from colliding with each other. Similarly, Safeea *et al* [24] achieve collaborative task execution employing generating a collection of offline computed nominal paths tailored to the industrial task at hand and generating on-the-fly modifications by means of artificial potential fields for collision avoidance. Another method was developed by Yoon *et al* [25] using the concept of manipulator shadow space calculations along with modified genetic algorithms for the generation of an optimal collision-free sequence for executing planar tasks, where the manipulators are assumed to only travel linearly from one point to another. Both manipulators start moving at the same time.

To solve the problem of collision avoidance in unstructured environments, where the robots might share the same workspace, the issues of timing for task allocation as well as planning the robotic motion sequences become of immense importance [26; 27; 28]. Furthermore, having to consider other robots as moving obstacles and preventing their collision in an online manner as manifested in [23] and [24] where hypothetical attraction and repulsion vectors field attract the robot towards the goal while repelling it from the obstacles. coupling this concept with a mathematical representation of the robot's kinematics achieved a task level of control with collision avoidance capability.

[25] used the concept of manipulator shadow space which is defined as a region that is swept by the links and the end-effector of a robot as it travels between task positions. [29] requires establishing secure and trustworthy machine-to-machine communication while [30] requires establishing a machine-to-cloud communication protocol where the robots share their status in real-time.

Reactive methods for collision avoidance are based on real-time decision-making and immediate response to environmental changes. These methods do not require a complete map or predefined path, which makes them adaptable in dynamic environments. One widely known technique is the Potential Fields Method [31], where robots are treated as particles influenced by virtual forces that repel them from obstacles or other robots. Fiorini and Shiller [32] proposed the velocity obstacle approach, which computes collision-free velocities for robots by considering the velocity vectors of other robots.

Path planning techniques for collision avoidance focus on generating a complete, predefined path for each robot that is free from collisions with obstacles and other robots. The A* algorithm is one of the most famous search-based planning methods used for static environments. However, for dynamic environments, the D* algorithm [33] was introduced, which allows real-time replanning as robots or obstacles move.

Decentralized methods distribute the decision-making process across multiple robots. Each robot is responsible for maintaining a collision-free trajectory by communicating or sharing limited information with other robots. Jager and Nebel [34] proposed a decentralized approach for coordinating mobile robots to prevent collisions and deadlocks. When two robots come within a certain distance, they exchange trajectory information to assess collision risk. If a collision is possible, they adjust by adding time gaps in their paths. For deadlocks, robots successively replan their trajectories until the issue is resolved. Marcolino and Chaimowicz [35] developed a decentralized algorithm to manage swarm robot traffic and prevent congestion when robots move

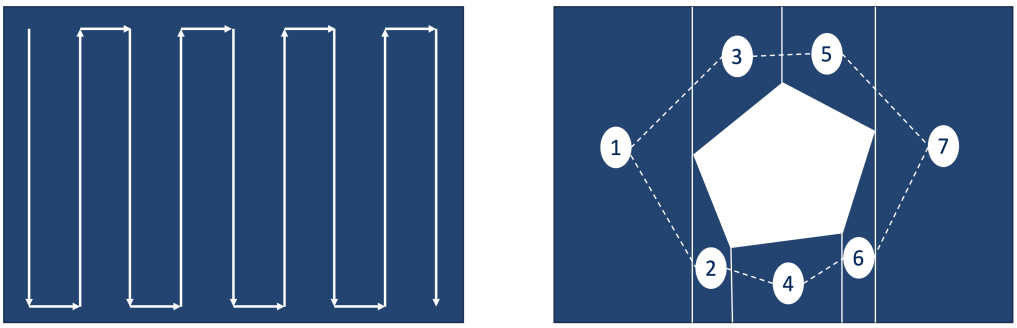
in opposite directions. Robots detect potential collisions through local sensing and communication and then adjust their paths to avoid congestion.

Recent advancements in machine learning have led to new approaches in multi-robot collision avoidance. Long et al [36] demonstrated the use of deep reinforcement learning for decentralized collision avoidance in complex scenarios. Their approach allows robots to learn sophisticated behaviors that can be generalized to new situations. Li et al. [37] proposed a Graph Neural Network (GNN)-based approach for distributed collision avoidance, which can effectively capture the relationships between multiple robots and obstacles in the environment.

Despite significant progress, several challenges remain in multi-robot collision avoidance. One of the foremost challenges is scalability. Most existing methods struggle to handle systems involving large numbers of robots. Additionally, Balancing safety and efficiency in fast-changing environments is an important challenge. Collision avoidance systems need to ensure that robots don't get too close to each other or nearby obstacles. However, this can't come at the expense of slowing down operations or lowering the overall performance of the system, especially in time-sensitive areas like industrial automation or autonomous vehicles, where speed and precision are essential.

2.3 Coverage Path Planning

Coverage algorithms can be categorized as either heuristic or complete, based on whether they reliably ensure complete coverage of the free space. Additionally, they can be classified as either offline or on-line. We borrow this classification from a survey by Choset [38] and by Galceran[39]



(a) Typical back-and-forth motion (b) Adjacency graph

Figure 2.1: Cellular decomposition

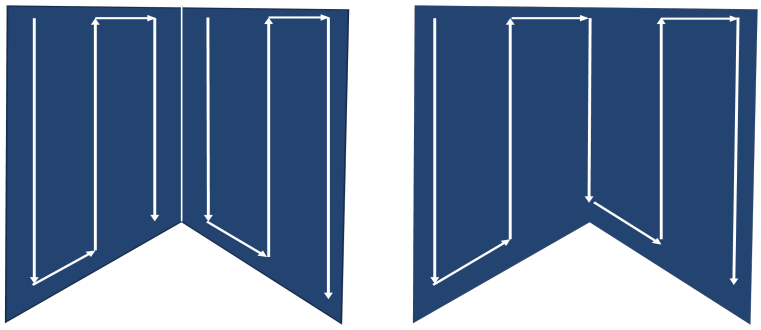
2.3.1 Heuristic Algorithms

Heuristic algorithms play a crucial role in coverage path planning within robotics, offering practical solutions for navigating intricate spatial environments. One notable heuristic, the Greedy Algorithm, exemplified by Karabulut [40], has been effectively applied to solve the Traveling Salesmen Problem (TSP). This algorithm prioritizes selecting the nearest unexplored area at each step, making it particularly efficient for traversing relatively simple environments. Additionally, methods such as Genetic Algorithm (GA) [41], Ant Colony Optimization (ACO) [42], and Particle Swarm Optimization (PSO) [43] are widely employed for finding near-optimal paths by efficiently navigating large search spaces of potential solutions. However, it's essential to note that while these methods can efficiently explore solution spaces, they do not guarantee the discovery of the optimal solution and may converge to local optima.

2.3.2 Cellular Decomposition

Exact cellular decomposition methods involve dividing the obstacle-free space into distinct, non-intersecting regions referred to as cells. The collective union of these cells precisely occupies the entire free space. These cell regions, devoid of obstacles, are considered straightforward to cover. The robot can efficiently cover each cell by employing straightforward back-and-forth motions shown in figure 2.1a. As a result,

the task of coverage path planning is simplified to the planning of motions between individual cells shown in figure 2.1b. After dividing the area into cells, we can create an adjacency graph. Two cells are considered adjacent if they share a common boundary, which is represented as an edge in the graph. Each cell is treated as a separate node in this graph representation. Typically, the exact cellular decomposition methods follow two steps to output the coverage path. It starts by breaking down the free space into cells generating the adjacency graph. Next, it exhausts all the possible walks through the adjacency graph to get the optimal sequence of nodes and not the direct coverage path. Consequently, a specific path for covering each cell needs to be deduced separately.



(a) Two Cells (b) One Cell

Figure 2.2: Fewer cells mean more efficient path planning.

2.3.2.1 Trapezoidal Decomposition

The trapezoidal decomposition [44; 45] is one of the simplest exact cellular decomposition techniques, designed for planar, polygonal spaces exclusively. each cell takes the form of a trapezoid. A back-and-forth motion is used inside each cell. The assurance of complete coverage is achieved by identifying a thorough walk through the adjacency graph associated with the decomposition. One of the limitations of this decomposition method is that it creates too many small cells that could be combined



(a) Trapezoidal cell decomposition (b) Boustrophedon cell decomposition

Figure 2.3: Difference between trapezoidal decomposition and boustrophedon decomposition

to form a larger cell because the trapezoidal method only makes cells that are shaped like simple boxes. This is not ideal because more cells lead to a longer coverage path as shown in figure 2.2.

2.3.2.2 Boustrophedon Decomposition

To address this challenge, Choset and Pignon introduced the boustrophedon decomposition [46]. The key difference between the two methods is in decomposing the free space into cells. The boustrophedon method only considers vertices where a vertical segment can be extended both above and below the vertex which are called critical points. 2.3 shows the difference between the trapezoidal decomposition and the boustrophedon decomposition side by side

2.3.2.3 Morse-based Decomposition

While the boustrophedon decomposition enhanced the results of the trapezoidal decomposition by consolidating unnecessary cells, both approaches face limitations in handling non-polygonal obstacles. To address this, a more advanced technique was devised by extending the boustrophedon method to a novel cellular decomposition approach based on critical points of Morse functions [47; 48]. This approach generates different cell shapes, including spiked, polygonal, and even circular cells shown in 2.4.

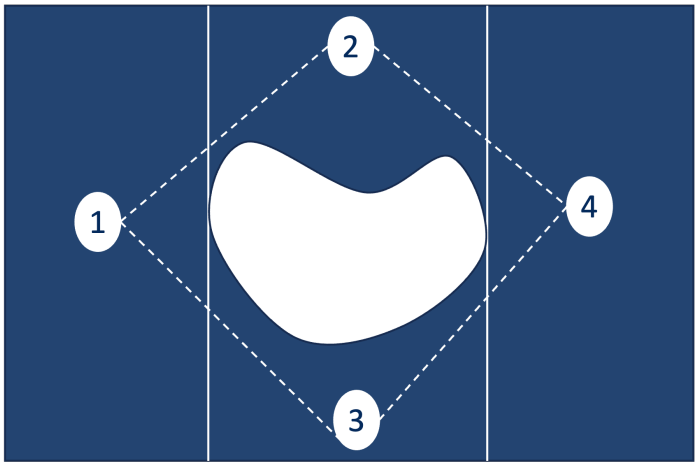


Figure 2.4: Morse decomposition

In theory, Morse decomposition is applicable in any n-dimensional space.

It's important to observe that these exact decomposition methods presuppose knowledge of polygonal obstacles and the terrain in advance. Consequently, it falls under the category of off-line methods.

2.3.3 Contact sensor-based coverage

Contact sensor-based coverage in CPP involves robots exploring an environment while actively making physical contact with surfaces to ensure complete coverage. This approach is useful in cluttered or poorly lit environments where other sensing methods may be unreliable. Butler *et al* [49] proposed an exact cell decomposition to cover an unknown rectilinear environment. algorithm. The robot's path is planned to maximize surface area coverage, adjusting trajectory based on feedback from contact sensors.

2.3.4 Grid-Based Methods

Grid-based methods divide the space into smaller square [50] or triangular shape [51] uniform-size grid cells. This grid contains information on whether an obstacle is

present or not by assigning a binary value or probability to each cell depending on the application [52]. These grid maps can be stored as arrays with each entry containing occupancy information and the covered areas marked in the array.
The downside of this set of methods is the exponential growth of memory usage when the represented environment becomes more complicated.

Different types of CPP were applied to the grid-based methods, Zelinsky *et al* [53] used the wavefront algorithm in an off-line method with a known environment while Shivashankar *et al* [54] generalized the method to an on-line coverage method applied to a mobile robot.

Spanning trees were also used in the work of Gabriel and Rimon []. An online approach was developed by dividing the environment into a grid map and applying a Spriral Spanning Tree Coverage algorithm using the sensors attached to a robot. Neural network was also used in the work of Luo et *et al* [55] and Yang and Luo [56] in the application of robot floor cleaning equipped with sensors to validate this online approach.

2.4 Robot Kinematics

Robot kinematics is a fundamental aspect of robotics. It is primarily divided into forward kinematics (FK) and inverse kinematics (IK). Forward kinematics involves calculating the position and orientation of the robot's end-effector based on given joint parameters, while inverse kinematics determines the necessary joint parameters to achieve a desired position and orientation of the end-effector.

In understanding and controlling robots, two critical concepts come into play: the Jacobian matrix and robot manipulability. These concepts provide the necessary tools to evaluate and optimize the robot's performance in its workspace.

2.4.1 The Jacobian Matrix

The Jacobian matrix represents the first-order partial derivatives of a vector-valued function with respect to another vector. It translates changes in the robot's joint angles into changes in the position and orientation of the robot's end effector.

If $\mathbf{q} = (q_1, q_2, ..., q_n)$ represents the joint coordinates of the robot, and $\mathbf{x} = (x, y, z, \phi, \theta, \psi)$ represents the position and orientation of the robot's end effector in Cartesian space, then the vector-valued function $\mathbf{x} = f(\mathbf{q})$ describes the kinematics of the robot. The Jacobian matrix J of this function is given by:

$$J = \frac{\partial \mathbf{x}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \psi}{\partial q_1} & \cdots & \frac{\partial \psi}{\partial q_n} \end{bmatrix} \tag{2.1}$$

This matrix transforms velocity vectors from joint space (joint velocities $\dot{\mathbf{q}}$) to Cartesian space (end effector velocities), allowing the control system to compute how changes in joint angles affect the position and orientation of the end effector [57].

Each element $J_{i,j}$ represents the instantaneous rate of change of the i-th task space coordinate with respect to the j-th joint variable

2.4.2 Forward Kinematics

Forward kinematics involves calculating the position and orientation (pose) of the robot's end effector based on known joint parameters such as angles and displacements. The typical approach to solving FK is through the use of the Denavit-Hartenberg (D-H) convention, which simplifies the mathematical modeling of joint and link parameters.

For a robotic manipulator with n joints, the position of the end-effector $\mathbf{P}$ in Cartesian coordinates can be derived from the joint angles $\theta_1, \theta_2, \ldots, \theta_n$. The forward kinematics problem can be expressed as $\mathbf{P} = f(\theta)$ where $f(\cdot)$ is the forward kinematics

function

The transformation from one joint to the next can be represented using a homogeneous transformation matrix that combines rotation and translation. The matrix for joint i to $i+1$ is given by:

$$T_{i+1}^{i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where α_i is the link twist and d_i is the link offset. To find the pose of the end effector, multiply the transformation matrices of all joints:

$$T_n^0 = T_1^0 \cdot T_2^1 \cdot \ldots \cdot T_n^{n-1}$$

Where T_n^0 represents the transformation from the base frame to the end effector frame [58; 59]

The basic equation for forward dynamics that describes the velocities of the end effectors can be written as follows:

$$\dot{\vec{\mathbf{f}}} = J(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{2.2}$$

The Jacobian leads to an iterative method for solving $\mathbf{P} = f(\theta)$. To increment the joint angles, θ by $\Delta\theta$, we use

$$\theta := \theta + \Delta\theta \tag{2.3}$$

From 2.2 The change in end effector positions caused by this change in joint angles can be estimated as $\Delta\vec{f} \approx \Delta\theta$

Moreover, the FK problem can be approached using various mathematical tools, including geometric methods, homogeneous transformation matrices, and even quaternions for representing orientations [60]. FK is utilized in robot programming and simulation environments, where the desired end-effector position is computed to ensure that the robot can perform tasks accurately. For instance, in industrial automation [1] it is crucial for path planning and trajectory generation, enabling robots to navigate complex environments while executing precise movements.

2.4.3 Inverse Kinematics

Inverse kinematics is the process of determining the joint parameters that will result in the desired position and orientation of the end effector. This is typically a more complex problem due to the non-linear nature of the equations and the potential for multiple solutions (or none at all).

There exist multiple to solve the inverse kinematic problem. The choice of method depends on factors such as robot configuration, computational requirements, and accuracy needs.

2.4.3.1 Analytical methods

The analytical methods provide exact, closed-form solutions when they exist. These methods are computationally efficient but are only possible for certain robot geometries, particularly those with special kinematic arrangements. They work best for robotic arms with fewer degrees of freedom and specific geometries that simplify the equations. This can be expressed as $\theta = f^{-1}(\mathbf{P})$ where $f(\cdot)$ is a highly non-linear operator which is difficult to invert. However, there are cases where a solution to the IK problem is unattainable, such as when the target position is out of reach of the manipulator or when the optimal solution is not unique. Even under ideal conditions, it's often impossible to derive a closed-form solution. Iterative methods

to approximate a good solution to the problem seem necessary in most cases (see 2.4.3.2).

2.4.3.2 Jacobian-Based Methods

The Jacobian solutions are based on the Jacobian matrix **J** 2.1 and are a linear approximation of the IK problem. These approaches seek to choose $\Delta\theta$ to update the joint angles using equation 2.3 to solve:

$$\vec{e} = J\Delta\theta \tag{2.4}$$

The Jacobian transpose method was first used for inverse kinematics by [61; 62]. It uses the transpose of the Jacobian to solve the IK problem:

$$\Delta\theta = \alpha J^T \vec{e} \tag{2.5}$$

With α being an appropriate scalar. Updating the angles $\Delta\theta$ by a sufficient $\alpha \geq 0$, the end effector positions will be changed by $\alpha J J^T \vec{e}$. the scalar value α can be calculated by reducing the value of the vector $\vec{e}$ after each update. Thus

$$\alpha = \frac{\langle \vec{e}, JJ^T\vec{e} \rangle}{\langle JJ^T\vec{e}, JJ^T\vec{e} \rangle} \tag{2.6}$$

The Jacobian Pseudo-inverse, also known as the Moore-Penrose inverse of the Jacobian [63], sets the value $\Delta\theta$ to

$$\Delta\theta = J^\dagger \vec{e} \tag{2.7}$$

$J^\dagger$ is an $n \times m$ matrix called the pseudo-inverse of J. This pseudo-inverse is defined for all Jacobian matrice J, even ones that are not square nor full rank. This method is widely used in the literature [64; 65] but often performs poorly because of its in-

stability near singularities.

The Damped Least Squares method (DLS) was used by [66; 67] to solve IK problems. It avoids many of the pseudo-inverse method's problems with singularities and can give a numerically stable method of selecting $\Delta\theta$. Instead of directly computing the change in joint angles $\Delta\theta$ that would lead to the desired solution to eq 2.4, The DLS method seeks to minimize the expression

$$\|J\Delta\theta - \vec{e}\|^2 + \lambda^2\|\Delta\theta\|^2 \tag{2.8}$$

Here, $\lambda \in \mathbb{R}$ is a non-zero damping constant. This additional term $\lambda^2\|\Delta\theta\|^2$ helps avoid issues with singularities by regularizing the problem. The damping constant controls how much influence this regularization has on the solution. The DLS solution is derived from the equation:

$$(J^TJ + \lambda^2 I)\Delta\theta = J^T\vec{e} \tag{2.9}$$

It is important to note that $J^TJ + \lambda^2 I$ is non-singular, thus making the system more stable. The DLS solution for $\Delta\theta$ is:

$$\Delta\theta = (J^TJ + \lambda^2 I)^{-1}J^T\vec{e} \tag{2.10}$$

This form is advantageous because J^TJ is a $n \times n$ matrix, where n represents the number of degrees of freedom. By applying this regularization term, the DLS method ensures that the matrix inversion is stable, even when near singular configurations.

2.4.3.3 Newton Methods

The Newton family of methods is based on a second-order Taylor series expansion of the object function f(x):

$$f(x+\sigma) \approx f(x) + [\nabla f(x)]^T \sigma + \frac{1}{2}\sigma^T H_f(x)\sigma \tag{2.11}$$

where $H_f(x)$ is the Hessian matrix. However, calculating the Hessian matrix is a complex task that leads to a high computational cost in each iteration. However, since Newton's methods approach the problem as a minimization task, they produce smooth motion with no abrupt discontinuities. The gradient projection method, introduced by Zhao in [68], is the most straightforward approach for handling constraints. Newton's methods have the added benefit of avoiding singularity issues, like those encountered when computing the Jacobian inverse. However, they tend to be complex, challenging to implement, and have a high computational cost for each iteration.

2.4.4 Robot Manipulability

Robot manipulability is a fundamental concept in robotics that quantifies how effectively a robot can perform tasks in various positions and orientations within its workspace. It describes the ability of a robot's end effector to achieve desired velocities and accelerations in any direction from a given configuration.

The manipulability analysis is based on differential kinematics, which provides the relation between joint space and task space velocities [69].

2.4.4.1 Manipulability Measure

Yoshikawa [70] introduced the manipulability index and indicates the distance to singular configurations or in other words, the ease or difficulty of moving the end

effector in particular directions. This measure is derived from the determinant of the Jacobian matrix 2.1:

$$w = \sqrt{det(JJ^T)} = s_1 s_2 ... s_n$$

Where s_i are the singular values of J. Geometrically, the manipulability measure w can be visualized as the volume of an ellipsoid in the Cartesian velocity space of the end-effector. The result of this equation will be an ellipsoid, whose volume is proportional to the value of w. A bigger volume means the robot end effector can move in more directions.

The concept of manipulability can also be mathematically expressed as:

$$M(\mathbf{q}) = \sqrt{|\mathbf{J}(\mathbf{q})^T \mathbf{J}(\mathbf{q})|} = \sqrt{|\mathbf{J}|^2}$$

where $M(\mathbf{q})$ represents the manipulability measure at a given configuration $\mathbf{q}$ [71; 72]

2.4.4.2 Singularity

Singular configurations of a robot occur when the determinant of the Jacobian matrix is zero ($det(J) = 0$) [73; 71]. At these points, the Jacobian loses rank, and certain directions of motion become impossible, constraining the end effector's ability to move or exert force in those directions. When a robot is near singularity, it loses force capabilities and certain motions become uncontrollable, limiting the robot's operational effectiveness. Small joint movements can lead to large, unpredictable motions of the end effector causing potential instability in the control systems. For instance, in parallel robots, singularities can be classified into various types, including boundary singularities and configuration singularities, which are critical for understanding the robot's workspace and operational limits [74; 75]

2.5 Multi-Robot Path Planning

Several methods for multi-robot path planning were presented in the literature, and most of them extended the ideas applied to single-robot systems to multi-robot systems. Rekleitis *et al* [76] presented a multi-robot boustrophedon decomposition in an unknown environment using a team of mobile robots. The algorithm presented uses the same boustrophedon decomposition method presented in 2.3.2.2 The key distinction lies in how the robots handle covering a single cell, considering the absence of communication among them. The robots assume two roles: explorers responsible for covering the cell boundaries and coverers executing the boustrophedon back-and-forth motion to cover the remaining part of the cell. This method is not straightforward to apply to a team of robot manipulators as it will be very hard to guarantee a collision-free path for the robots as all of them will be inside the same cell at a time.

Zheng *et al* [77] generalized the spanning tree coverage method to a team of robots in a known environment. and Luo and Yang [78] presented a neural network approach inspired by biology where the other robots are considered as moving obstacles and were improved in a more recent work [79] to surpass the limitation of the getting the robots stuck in a deadlock situation.

Path planning using a team of robots can also be looked at as the Multiple Traveling Salesmen Problem (mTSP) which is the generalization of TSP where multiple salesmen, in this case, robots, have to visit a certain number of cities exactly once and return to the initial position with minimum traveling cost. GA was widely used to solve this problem [80; 81; 82]. A random solution is selected to initiate the first generation. Subsequently, a combination of crossover and mutation operations is applied to get two new solutions, which become part of the next generation. This process continues until the desired population size is achieved, ultimately leading to the selection of a 'near-optimal' solution. PSO [83] and ACO [84] were also used to

solve the problem. The drawback of these mTSP implementations is that collision avoidance is not considered in the proposed solution hence the need to combine them with a collision avoidance algorithm.

CHAPTER III

Problem Statement

The **Collaborative Robotic Car-Painting Problem (CRCP)** involves efficiently generating joint trajectories for a collection of heterogeneous articulated robotic arms. The objective is to paint the car body using the nozzles attached to the arms while avoiding collisions between the robots. This task is guided by the CAD model of the car in figure 3.1 to ensure precise and comprehensive coverage during the painting process.

The **Optimal Base Placement in Collaborative Robotic Car Painting Problem** ensures that each robot can reach its assigned areas of the car body efficiently (see figure 3.2). The objective is to Compute the optimal fixed base positions for a collection of heterogeneous articulated robotic arms on the factory floor/ceiling to maximize the possibility of paint coverage and minimize the chance of robot collisions.

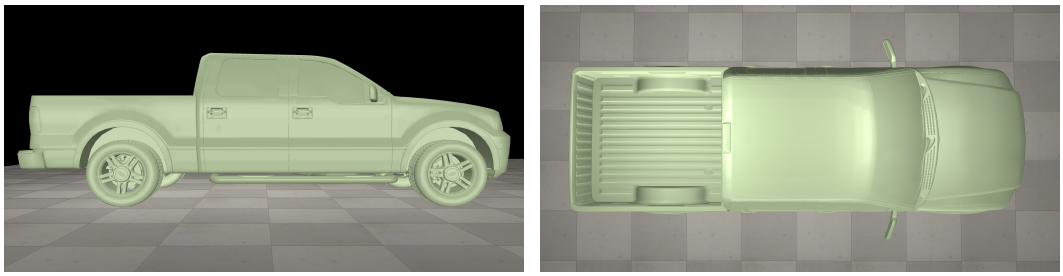


Figure 3.1: Example of the CAD model of a Ford F-150 truck

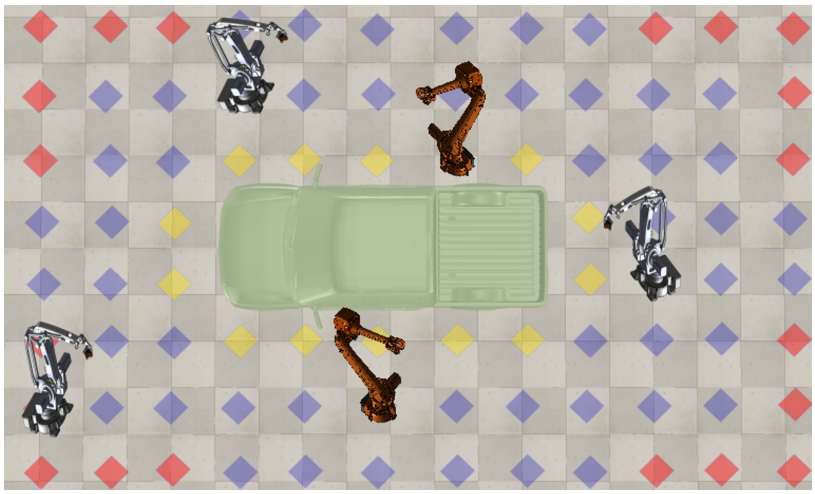


Figure 3.2: Mutli-robot car-painting base placement problem. The base positions directly impact the quality of vehicle paint coverage and the possibility of robot collisions during painting. For instance, robots placed on golden diamonds are more prone to collisions. On the other hand, robots placed on red diamonds will not be able to ensure complete vehicle paint coverage.

3.1 Assumptions

The following assumptions are made to solve the problem:

A1. The robotic arms are all of RRR articulated type (see, e.g., [85], Chapter 1 for classification of robotic manipulators based on their kinematic structures); namely, their first three joints, which are called the body, shoulder, and elbow, are of the revolute kind shown in figure 3.3.

A2. The robotic arms are all either floor-mounted or ceiling-mounted with fixed base positions.

A3. The robotic arms are arranged in two rows on either side of the car, leveraging the vehicle's inherent symmetry during car-painting operations. (see Figure 3.4 for an example).

A4. The robotic arms do not communicate with each other. In other words, the manipulators are not aware of each other's trajectories.

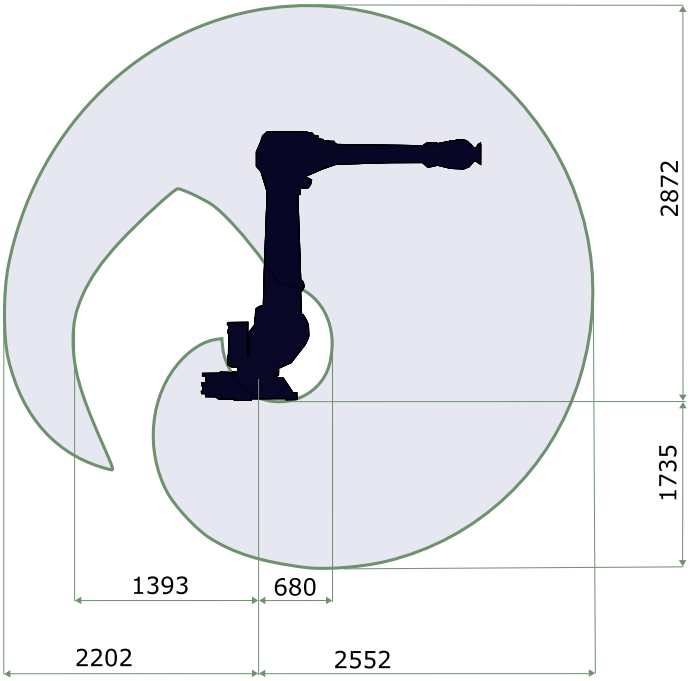


Figure 3.3: Example of an RRR articulated arm. The robot arm in the figure is an ABB IRB 4600 industrial manipulator.The unit lengths are in millimeters

A5. For the given industrial robotic arms, there exist inverse kinematics solvers that can generate the required joint positions from the provided end-effector poses (see, e.g., [86], for the inverse kinematics of sixteen fundamental 6-DOF robot manipulators).

A6. There exists paint deposition modeling for spray painting on automotive surfaces

3.2 Solution Strategy

The proposed solution strategy for CRCP has three stages, which will be discussed in chapter IV.

Stage 1 Achieving the collision avoidance objective (4.1)

Stage 2 Achieving the paint coverage objective using the results from Stage 1 (4.2)

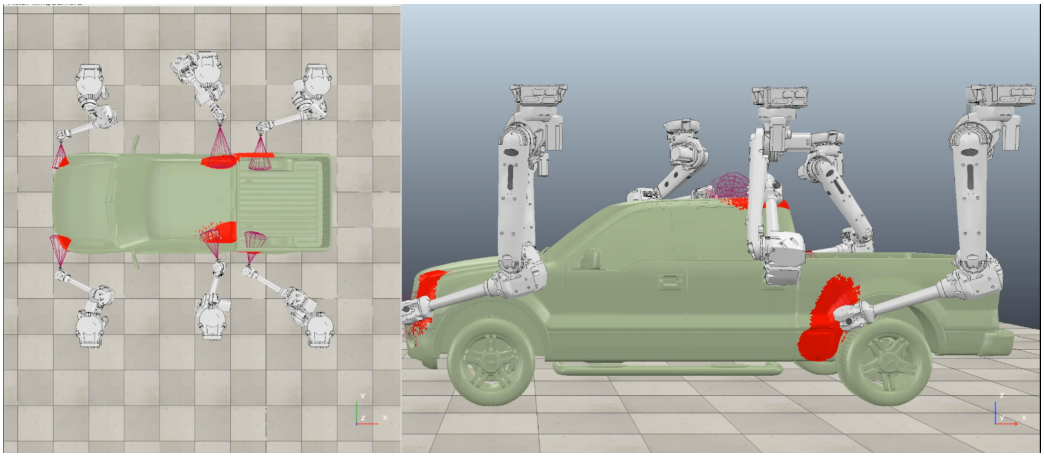


Figure 3.4: Example of the arrangement of two rows of three robotic arms around the vehicle.

Stage 3 Using the results from Stage 1 and Stage 2 to create 3D paths for the paint nozzles attached to the robot end-effectors (4.3)

The proposed solution strategy for the Optimal Base Placement in a Collaborative Robotic Car Painting (chapter V) Problem has four stages.

Stage 1 Devising a scheme for a point cloud rendering of the reachable workspace of the forearms of the robotic arms and fitting ellipsoids to these point clouds. (5.3.3)

Stage 2 projecting the ellipsoids obtained from Stage 1 onto the floor/ceiling to obtain planar ellipses. Using these ellipses, we construct a cost function to quantify the possibility of robot collisions. (5.4.1)

Stage 3 Construct a class of cost functions to quantify the amount of vehicle paint coverage. (5.4.2)

Stage 4 Formulating a proper multi-objective optimization problem using the cost functions obtained from Steps 2 and 3. (5.4.3)

CHAPTER IV

Automatic Collision-Free Trajectory Generation for Collaborative Industrial Robot

4.1 Achieving Collision Avoidance

we will present the 3D workspace calculations for the given robotic arms in order to guarantee non-collision. These calculations unfold in three main steps; namely, approximating the reachable workspace of the robot forearms using 3D point clouds (4.1.1), finding the volumes of intersection in-between the generated 3D point clouds (4.1.2), and fitting proper ellipsoids to the resulting volumes of intersection (4.1.3). The results of this section will later be utilized in the second stage of our algorithm, to achieve the paint coverage objective on the car surface.

4.1.1 Approximating The Reachable Workspace of Robot Forearms Using 3D Point Clouds

To prevent the collision of the painting robots, we compute the reachable space of the endpoint of the forearms of these robots [85]. In the computation of the forearm reachable space, we do not consider the wrist of these painting robots since the volume occupied by the wrists is negligible (see Figure 3.3). Furthermore, As demonstrated later in the thesis, we exclude the volumes of intersection between

the reachable spaces of the forearms through proper geometric calculations based on convex hull computations [87] and Khachiyan's algorithm [88]. The exclusion of these volumes of intersection ensures that the wrist collision, which might happen on the boundaries of the reachable space of the robot forearms, will not happen (see 3.3).

In this study, we are assuming that we are given M painting robots with articulated RRR arms (Assumption A1). We denote the angular position of the j^{th} joint angle of the i^{th} robot by θ^i_j, where $1 \leq i \leq M$ and $1 \leq j \leq 3$. therefore, the vector of joint angles for the j^th robot is given by

$$\theta^i := [\theta^i_1, \theta^i_2, \theta^i_3]^T, \quad \theta^i_{j,min} \leq \theta^i_j \leq \theta^i_{j,max} \tag{4.1}$$

where the constant angles $\theta^i_{j,min}$ and $\theta^i_{j,max}$ determine the limits of the robot joint motions.

We also assume that the base position of these robots are fixed and expressed with respect to the origin of a common coordinate frame that is rigidly attached to the factory floor/ceiling. We denote these base positions by

$$\mathbf{b}^i \in \mathbb{R}^3, \quad 1 \leq i \leq M. \tag{4.2}$$

To investigate the reachable volume by the forearm of each articulated robot, we use the Denavit-Hartenberg parameters (table 4.1) associated with the first three joints of these robots along with the forward kinematics maps that relate the joint configurations to the position of the robot forearm tip. In particular, the forearm tip position $t^i \in \mathbb{R}^3$ of the i^{th} robot its base located at $\mathbf{b}^i \in \mathbb{R}^3$ on the factory floor/ceiling, $1 \leq i \leq M$, and forward kinematics mapping $T^i(\boldsymbol{\theta}^i) \in \mathbb{R}^4$, which maps the angular position of the first three joints to the orientation and position of the forearm tip, can be computed from

Table 4.1: DH parameters of ABB IRB4600

joint	θ_i	l_{i-1}	$d_i(mm)$	$\alpha_{i-1}(deg)$	range
1	θ_i	0	0	0	-180∼180
2	θ_2	$l_1 = 175$	0	-90	-90∼150
3	θ_3	$l_2 = 900$	0	0	-180∼75
4	θ_4	$l_3 = 175$	d4 = 960	-90	-400∼400
5	θ_5	0	0	90	-125∼120
6	θ_6	0	0	-90	-400∼400

$$\begin{bmatrix} \boldsymbol{t^i}(\boldsymbol{\theta^i}) \\ 1 \end{bmatrix} = T^i(\boldsymbol{\theta^i}) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}^i \\ 0 \end{bmatrix}, \qquad 1 \leq i \leq M \tag{4.3}$$

In order to keep the algorithms for computing the reachable space of the forearms of the painting robots applicable to any given type of RRR industrial manipulator, we will use a sampling-based numerical method to approximate these reachable spaces. Using the sampling-based numerical methods such as the ones in [89], as opposed to their analytical or geometric counterparts (see, e.g., [90]), will yield 3D point clouds for which a plethora of numerically efficient algorithms such as the Quickhull [87] and Khachiyan's [88] can be invoked. Although we could have employed β-distribution-based sampling for generating the reachable volume of the robot forearm tips as in [89], we found out that uniform sampling of the joint positions for achieving the car-painting objective in CRCP suffices.

The 3D point cloud $\mathscr{P}_i$, $1 \leq i \leq M$, associated with reachable volume of the i^th robot forearm tip (shown in 4.1) is given by

$$\mathscr{P}_i = \left\{ \boldsymbol{t}^i(\boldsymbol{\theta}^k) \right\}_{N_i}^{k=1} \tag{4.4}$$

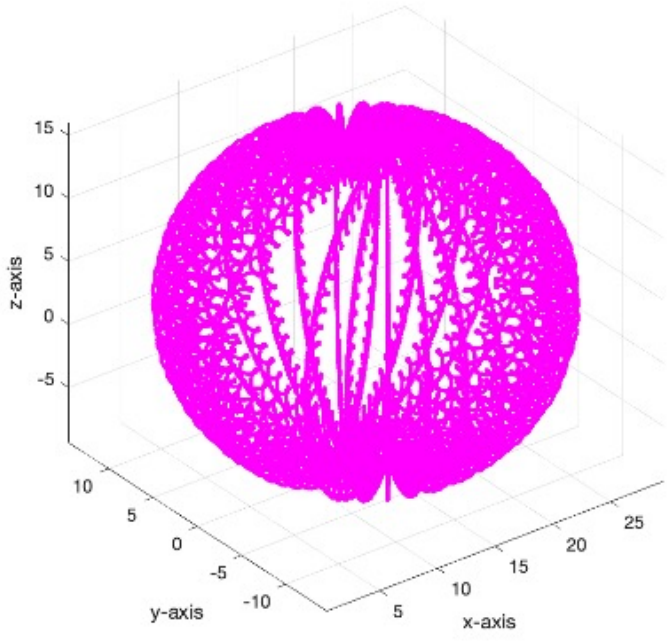


Figure 4.1: Point cloud representing the reachable workspace of ABB IRB 4600 manipulator.

Where $t^i(\theta^k)$ is the position of the forearm tip under configuration vector $\boldsymbol{\theta^k}$ and determined from the forward kinematics mapping in 4.3, $\boldsymbol{\theta^k}$ in the k^th sample picked from the interval $[\theta^i_{1,min}, \theta^i_{1,max}] \times [\theta^i_{2,min}, \theta^i_{2,max}] \times [\theta^i_{3,min}, \theta^i_{3,max}]$, and the number of samples in the point cloud, i.e., N, is given by

$$N_i = \prod_{j=1}^{3} (\lfloor \frac{\theta^i_{j,max} - \theta^i_{j,min}}{\delta_{ij}} \rfloor + 1) \tag{4.5}$$

In equation 4.5, $\lfloor . \rfloor$ denotes the floor function, and the constant δ_{ij} determines the distance between the samples in each angular interval. For instance, if $\delta_{i1} = \frac{\pi}{10}$, we mean that the samples from the interval $[\theta^i_{1,min}, \theta^i_{1,max}]$ are $\frac{\pi}{10}$ apart from each other. The number of samples in the point cloud N_i in equation 4.5 in each of the forearm point clouds determines the computational complexity of the overall CRCP algorithm. Figure 4.2 depicts the point clouds that represent the reachable workspace by the forearms of six ABB IRB 4600 manipulators.

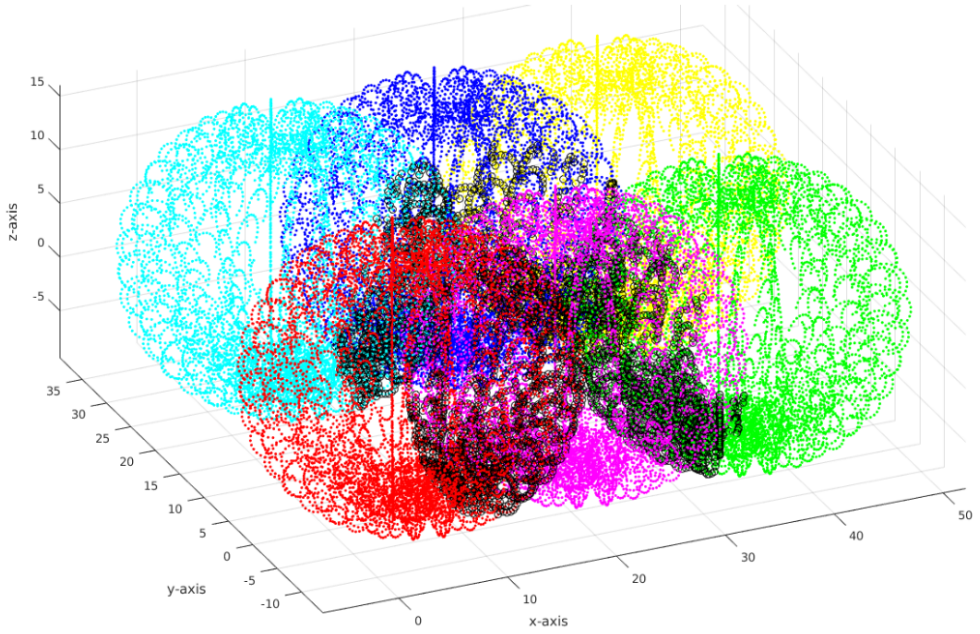


Figure 4.2: Example of the point clouds representing the reachable workspace by the forearms of six ABB IRB 4600 manipulators. The point clouds in black represent areas of possible collisions between the robots. The axis units are in decimeters (Note: 1 meter = 10 decimeters).

4.1.2 Finding The Point Clouds of Intersection in-Between The Generated 3D Point Clouds

In section 4.1.1, we computed the reachable workspace of the robot forearms using 3D point clouds. In this section, we will compute the point clouds resulting from the intersection of the point clouds from the previous step. Our method is based on using the Quickhull algorithm [87], [91].

The Quickhull algorithm is an efficient method for computing the convex hull of a finite set of points in n-dimensional spaces. The worst-case complexity for computing the convex hull of N points in a 3-dimensional space can be shown to be $O(Nlog(N))$. In our case, we need to compute the convex hull $\mathcal{CP}_i \subset \mathbb{R}^3, 1 \leq i \leq M$, associated with the point clouds $\mathcal{P}_i$ given by 4.4. In other words, we have

$$\mathcal{CP}_i = conv(\mathcal{P}_i), \tag{4.6}$$

where $conv(.)$ is the convex hull associated with a given set of points. Therefore, the worst-case complexity of computing $\mathcal{CP}_i$ is given by $O(N_i log(N_i))$, where the number of points N_i in the point cloud $\mathcal{P}_i$ is given by 4.5.

Having computed the convex hulls $\mathcal{CP}_i$, we compute their volumes of intersection using a proper 3D-clipping algorithm [92]. Clipping refers to the process of determining what region of 3D space is contained within another region of 3D space. For instance, one can utilize the extended Sutherland- Hodgman clipping algorithm for computing the intersection in-between the robotic forearm workspace convex hulls to obtain the convex sets

$$\mathcal{CP}_{ij} := \mathcal{CP}_i \cap \mathcal{CP}_j \tag{4.7}$$

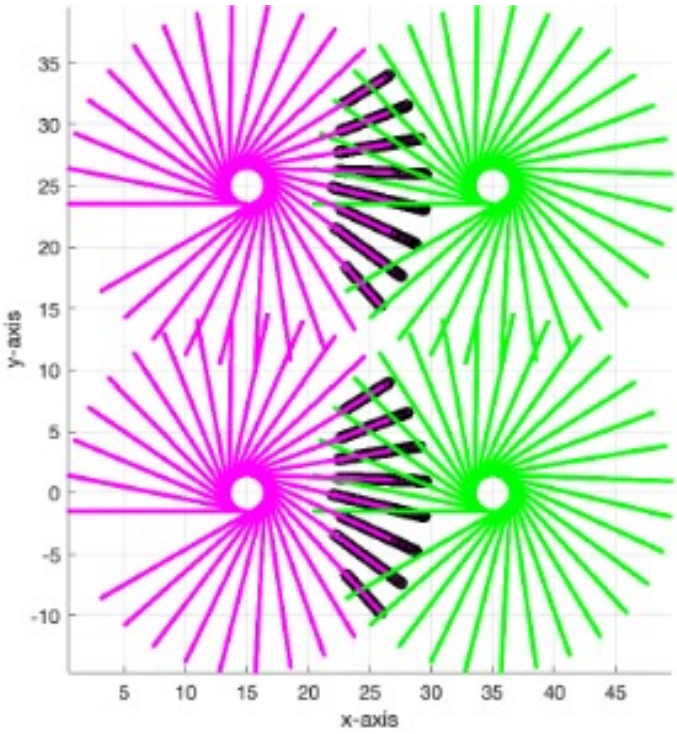


Figure 4.3: Top view of the point clouds representing the reachable workspace by the forearms of 4 ABB IRB 4600 manipulators.

which represent the intersection of 3D point clouds of the robotic forearm workspaces represented with black dots in figure 4.3 for a 4-robot setup. In general, efficient 3D polyhedral intersection algorithms enjoy linear-time complexities

Remark - The computations in this section result in finding collision-free volumes of intersection $\mathcal{CP}_i \bigcup_j \mathcal{CP}_{ij}$, where $\mathcal{CP}_i$ in equation 4.6 is the convex hull of the reachable workspace associated with the i-th robot and $\bigcup_j \mathcal{CP}_{ij}$ in equation 4.7 is the union of the volumes of intersection in-between the i-th robotic forearm and its neighbors.

4.1.3 Fitting Ellipsoids To The Volume of Intersection

Having computed the intersection of 3D point clouds of the robotic forearm workspaces, we fit the ellipsoids to these volumes of intersection. The projection of these ellipsoids on the 2D surface of the car body will later be utilized for achieving the painting coverage objective in the next section.

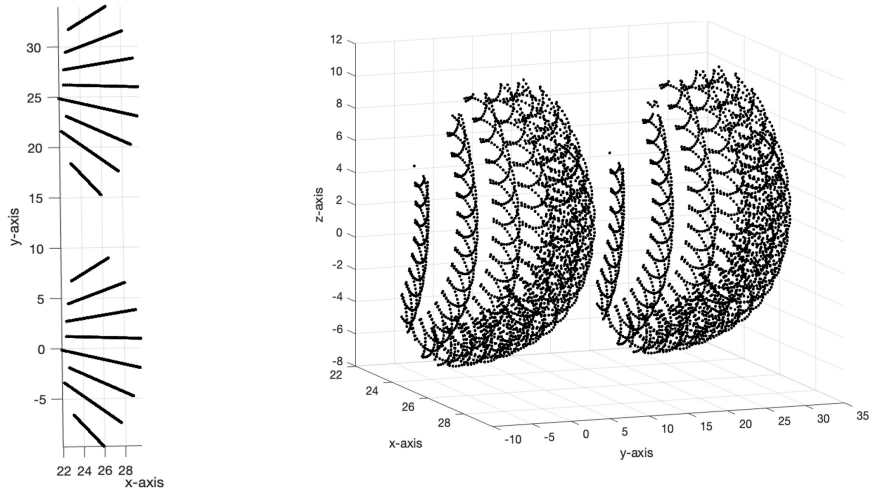


Figure 4.4: A 2D top view and a 3D view of the point cloud of intersection in a 4 robot setup.

Consider the point cloud representing the intersection of the robot workspace (shown in figure 4.4) $\mathscr{P}_{ij} := \{p_k\}_{k=1}^{N_i j}$ of $N_i j$ points in $\mathbb{R}^3$, where $\mathscr{P}_{ij}$ is computed from equation 4.7. In this section, we will use Khachiyan's algorithm, which is widely used in convex optimization [93], for fitting 3D ellipsoids to the point cloud $\mathscr{P}_{ij}$. Khachiyan's algorithm [88] is concerned with computing the Minimum Volume En-

closing Ellipsoid (MVEE) of the point cloud $\mathscr{P}_{ij}$ denoted by MVEE($\mathscr{P}_{ij}$) which is obtained by solving the optimization problem

$$\begin{aligned} \underset{A,c}{\text{Argmin}} \quad & \log(\det(A)), \\ & \text{subject to } (p_k - c)^T A(p_k - c) \leq 1, \text{ for all } p_k \in \mathcal{C}\mathscr{P}_{ij} \end{aligned} \tag{4.8}$$

where c is the center of MVEE($\mathcal{C}\mathscr{P}_{ij}$) and the matrix A provides the shape of the ellipsoid MVEE($\mathcal{C}\mathscr{P}_{ij}$). The MVEE of the point cloud in figure 4.4 is shown in figure 4.5. In particular by computing the singular value decomposition of A as $U\Xi V^T$, the radii of the ellipsoid are given by $r_k = \frac{1}{\sqrt{\xi_k}},\ 1 \leq k \leq 3$, where ξ_k is the k-th component of the diagonal of the matrix Ξ. Furthermore, the orientation of the ellipsoid is given by the rotation matrix V.

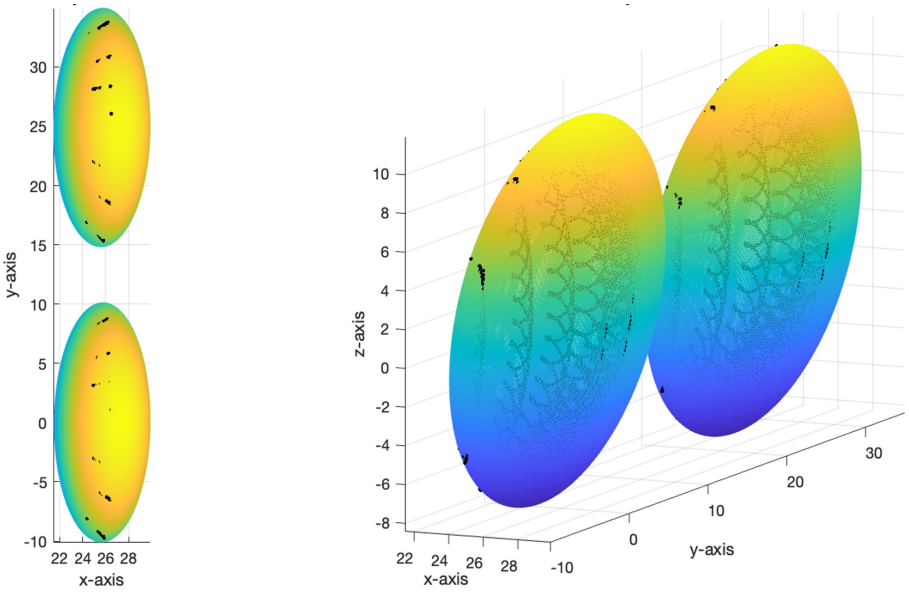


Figure 4.5: A 2D top view and a 3D view of the MVEE of the point cloud of intersection in a 4 robot setup.

Khachiyan's algorithm is based on solving the Lagrangian dual problem of equation 4.8 by using a conditional gradient ascent. As it is demonstrated in [88], the original Khachiyan algorithm for computing MVEE can be solved using $O(N_{i,j}^{3.5} ln(\frac{N_{ij}}{\epsilon}))$

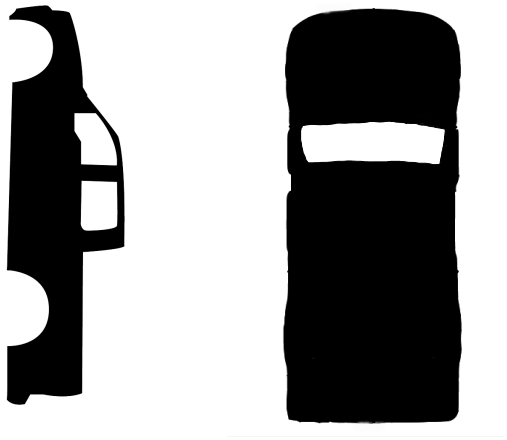


Figure 4.6: Grid map representation of the CAD model of a Ford F-150 truck

arithmetic operations to the relative accuracy of ϵ in the volume. In this study, we are using the MATLAB implementation of Khachiyan's algorithm in [94].

4.2 Achieving The Paint Coverage

In this chapter, we will utilize the results from the previous section in order to generate the nozzle footprint profile for achieving the paint coverage objective. The calculations for achieving the objective unfold in two steps: namely, projection of the fitted ellipsoids from 4.1.3 onto the grid map representation of the sides and top of the vehicle CAD model, and automatic decomposition of the resulting grid maps into cells by utilizing the Morse critical points of the obtained map from the projection step and assigning back-and-forth (boustrophedon-like motions) for covering the cells by the motion of the applicator footprint 4.2.2.

This study was tested on a Ford F-150 truck. Using the CAD file of the vehicle, we were able to generate a side and top view of the vehicle shown in figure 4.6

4.2.1 Projection of the Fitted Ellipsoids Onto The Car CAD Model

If we consider a 6-robot scenario with 3 robots on each side (depicted in figure 4.2) As the first step for achieving the paint coverage objective, we project the fitted ellipsoids from 4.1.3 onto the grid map representation of the sides and top of the vehicle CAD model to obtain images like the one depicted in Figure 4.7. To obtain the grid map of the sides and the top of the vehicle CAD model, one can employ standard image processing algorithms such as the Moore-Neighbor Tracing algorithm for tracing given contours and detecting the boundaries of objects in a given image [95].

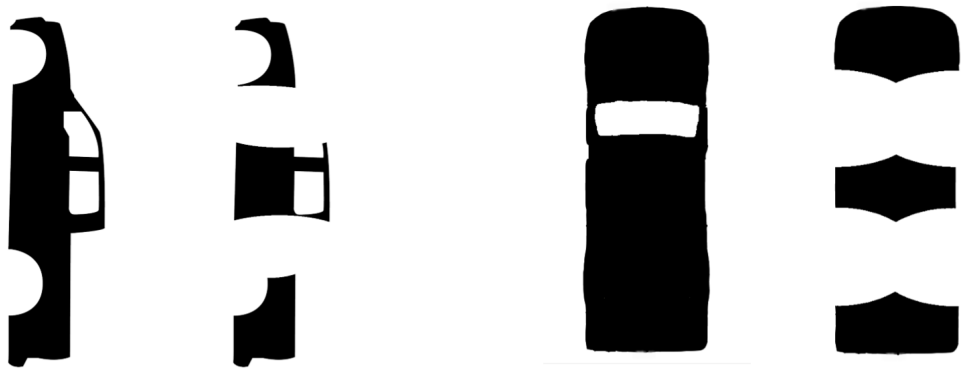


Figure 4.7: Example of projection of the fitted ellipsoids onto the grid map representation of the sides and top of the vehicle CAD model.

4.2.2 Decomposition of the Vehicle Grid Map Into Cells Using the Morse Critical Points

After projecting the fitted ellipsoids from 4.1.3 onto the grid map of the sides and the top of the vehicle CAD model, we use the algorithm due to Acar, Choset, et al. [48] for automatically decomposing the car grid maps, here the volumes of intersection have been projected onto them, by utilizing the Morse critical points of the obtained

map from the previous step. The Morse critical point-based decomposition is an extension of the exact cellular decomposition due to Barraquand and Latombe [44].

The exact cellular decomposition represents a robot's free space (here, the nozzle footprint profile) by dividing it into regions with simple structures such that the union of the regions is equal to the free space. The extension due to [48] relies on using the critical points of Morse functions for indicating the location of cell boundaries. Since we are using the projection of the fitted ellipsoids on the vehicle grid map, these projections have non-degenerate critical points. Having M_c critical points on a planar grid map, the algorithm can decompose the surface into cells with complexity $O(M_c \log(M_c))$. The MATLAB implementation by Horvath [96] was used to achieve cellular decomposition.

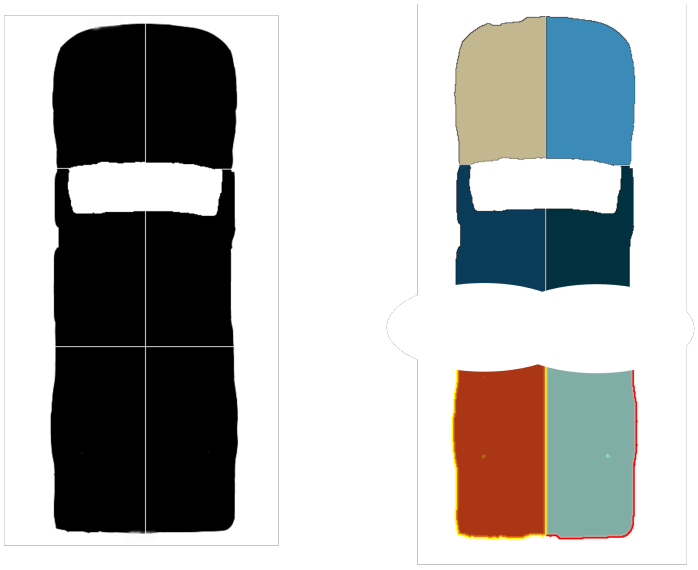


Figure 4.8: Example of a cellular decomposition of the top of the car for a 4-robot setup

After the computations associated with the cellular decomposition shown in figure 4.8, we assign back-and-forth (boustrophedon-like) motions for covering the obtained cells by the motion of the applicator/nozzle footprint using the prominent Boustrophedon Algorithm. The result of generating such boustrophedon-like paths for 6 robots is depicted in Figure 4.9 The two top figures represent the trajectories of

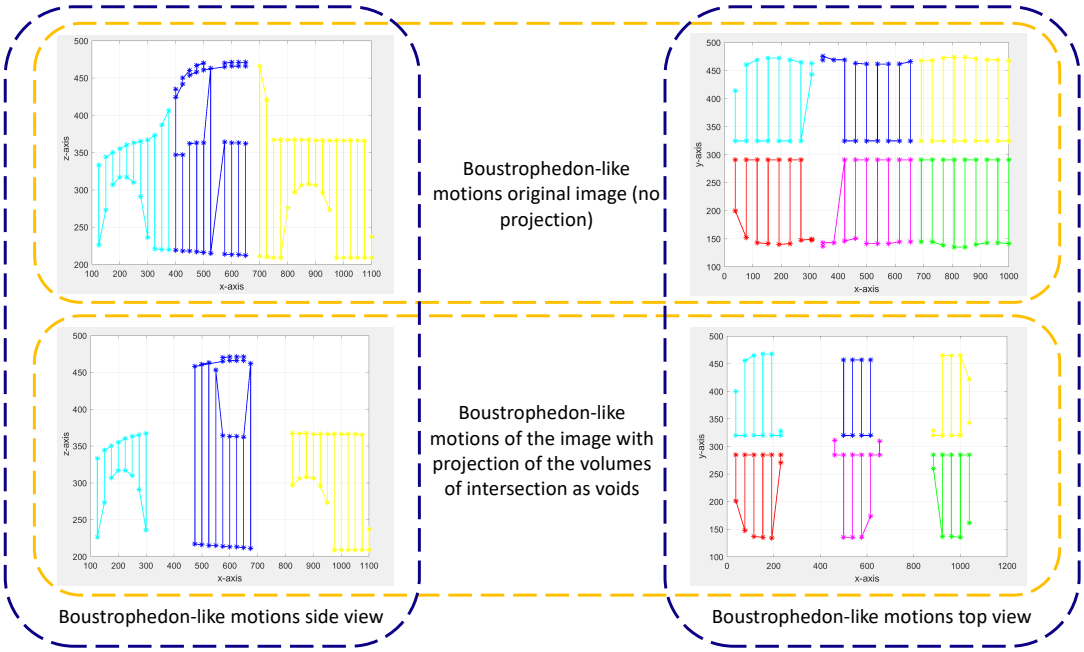


Figure 4.9: Example of assigning back-and-forth (boustrophedon-like) motions on the side and top of the vehicle for the applicator/nozzle footprint on the grid maps. Each colored boustrophedon path belongs to one of the six robots working on one side and top of the car

the applicator/nozzle footprints obtained by assigning boustrophedon-like paths on the sides and top surfaces of the vehicle, without the projection of the ellipsoids. These trajectories comprehensively cover the entire vehicle without accounting for any volume projection.

Additionally, the two bottom figures in Figure 4.9 represent the boustrophedon-like paths determined for the areas on the sides and top of the car that overlap with the projection of the ellipsoid.

Aiming to identify trajectories for the void areas. The process involves taking the union of neighboring cells from both the boustrophedon-like paths of the areas without projection of the ellipsoids and those with the projection of volume intersections as voids. The difference between these two unions represents the desired boustrophedon-like trajectories for the void areas.

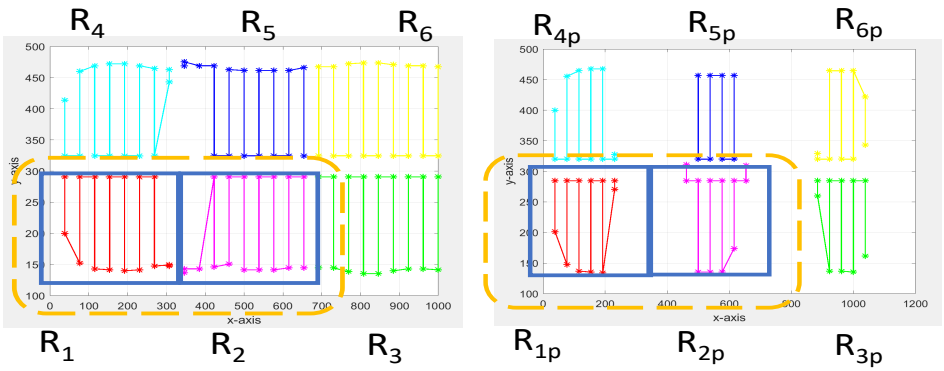


Figure 4.10: Example of assigning back-and-forth (boustrophedon-like) motions on the top of the vehicle for the applicator/nozzle footprint of the areas without projection of the ellipsoids (on the left) and those with the projection of volume intersections as voids (on the right)

Figure 4.10 illustrates the top of the car for both trajectories with and without the void areas. In this figure, robot 1's trajectory is depicted in red (denoted as R_1: without projection and R_{1P}: with projection), while robot 2's trajectory is shown in magenta (denoted as R_2: without projection and R_{2P}: with projection).

To determine the boustrophedon-like trajectories for the void areas between the red-colored path and the magenta-colored path, we define the sets A_1 and B_1 as the unions of trajectories R_1, R_2, R_{1p}, and R_{2p}, respectively. Specifically, $A_1 = R_1 \cup R_2$ represents the combined coverage of the red and magenta paths, while $B_1 = R_{1p} \cup R_{2p}$ encompasses the area covered by the projections of these paths. To isolate the nozzle/applicator footprint paths specifically for covering the void areas (denoted as V_1), we take the set difference between A_1 and B_1, i.e., $V_1 = A_1 - B_1$. This operation yields the trajectories required to ensure complete coverage of the void regions. This iterative process is applied to all neighboring cells along both the sides and the top of the car. By repeating the steps described above, boustrophedon-like paths are generated to cover void areas until complete coverage is achieved.

we obtain nozzle/applicator footprint paths that effectively cover the void areas but have not yet been assigned to any specific robot. These paths are illustrated

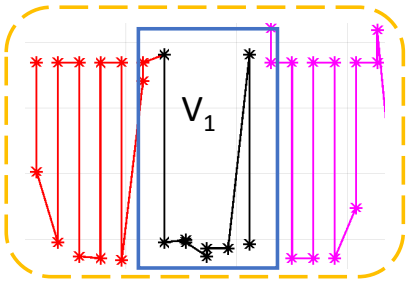


Figure 4.11: Example of assigning back-and-forth (boustrophedon-like) motions on the top of the vehicle for the applicator/nozzle footprint for R1 and R2. black-colored applicator/nozzle footprint represents the trajectory that was not yet assigned to any robot

in Figure 4.11, depicted with black-colored applicator/nozzle footprints. Once we isolate these paths specifically for covering the void areas, we assign these areas to only one of the robots. Figure 4.12 contains the completed nozzle/applicator footprint paths where the boustrophedon-like paths are completed by creating proper boustrophedon-like trajectories for the void areas (depicted with black-colored applicator/nozzle footprints) and assigning them to the appropriate robot.

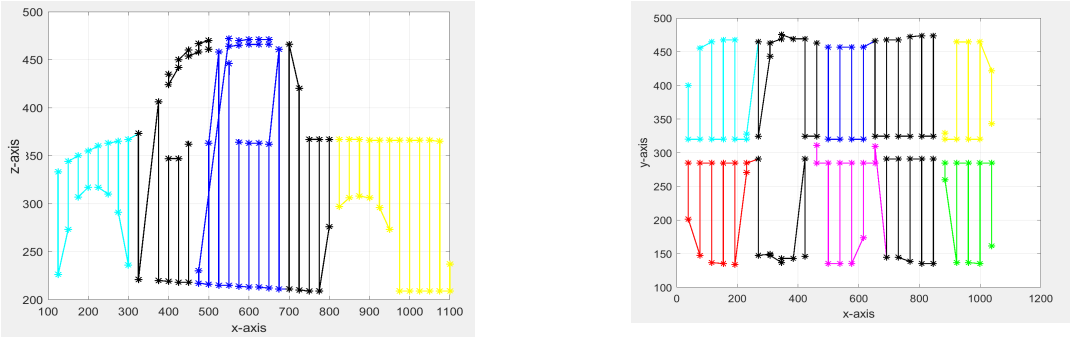


Figure 4.12: Example of final set of collision-free nozzle/applicator footprint paths on the side and top of the vehicle. black-colored applicator/nozzle footprints represent the trajectory that was not yet assigned to any robot

4.3 Task Assignment and Generation of Robotic Joint Trajectories

In this chapter, we will allocate the painting tasks to the robots using the boustrophedon-like motions obtained in 4.2.2. Additionally, we will generate 3D paths from the resulting boustrophedon-like applicator footprints for the paint nozzles attached to the robot end-effectors. These 3D paths will be used to derive the desired joint trajectories for each robot in 4.3.2

4.3.1 Assigning Robotic Tasks

Having obtained the boustrophedon-like paths of the applicator footprints on the sides and the top of the car, we compute the centroid of the obtained paths, which depends on the number of the painting robots. For instance, in the case of six robots, we have six centroids associated with the sides of the car and six centroids associated with the top of the car. We then compute the Euclidean distance in-between the centroids of the boustrophedon-like paths and the centroids of the collision-free volumes of intersection $\mathcal{CP}_i - \bigcup_j \mathcal{CP}_{ij}$, where $\mathcal{CP}_i$ in 4.6 is the convex hull of the reachable workspace associated with the i-th robot and $\bigcup_j \mathcal{CP}_{ij}$ in 4.7 is the union j of the volumes of intersection in-between the i-th robotic forearm and its neighbors.

By utilizing the Euclidean distances in-between the centroid of the collision-free workspace volumes and the centroid of the boustrophedon-like paths (see Figure 4.13), the boustrophedon cells that are the nearest to a given reachable workspace point cloud will be assigned to their respective robotic manipulators. The result of such cell assignment is depicted in Figure 4.13. In Figure 4.13, we are using six different colors to represent the six different nozzle/applicator footprint paths assigned to each of the six robots.

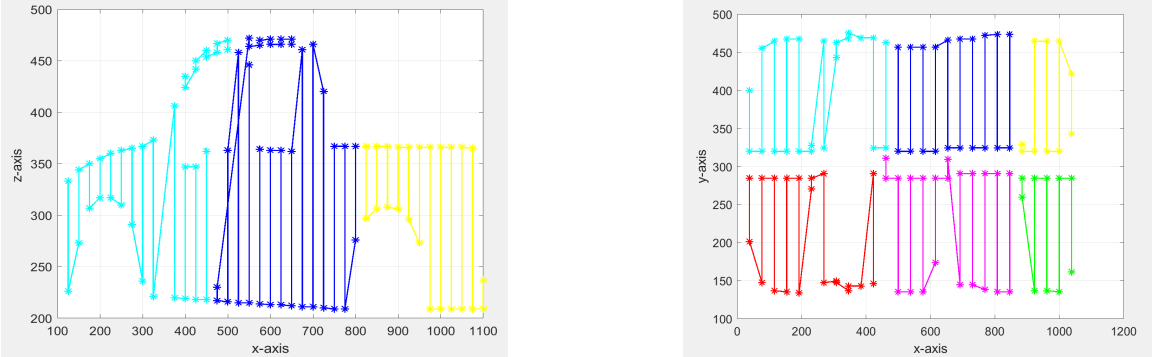


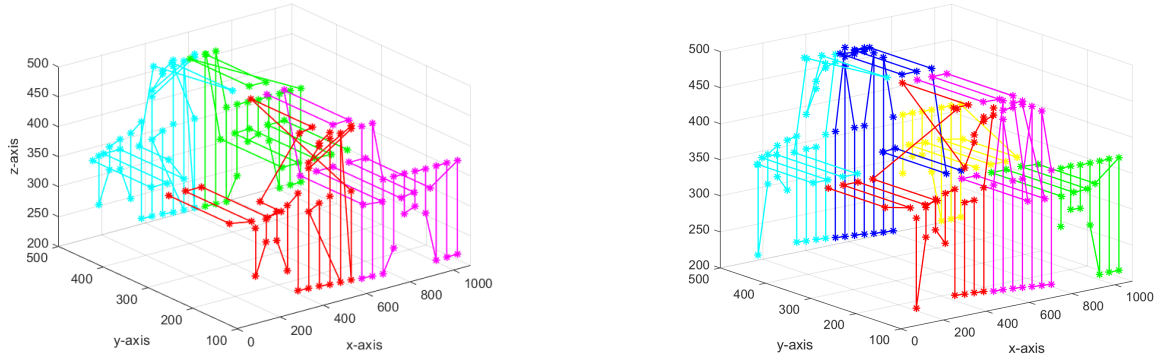
Figure 4.13: Example of assignment of painting tasks to the robots, which is done based on computing the Euclidean distances in-between the centroids of the reachable workspace point clouds of the robots and the centroids of the boustrophedon-like motions obtained from Stage 2-Step 2. Each colored collection of paths corresponds to one robot. For instance, the red paths on the side and the top of the car are assigned to a single robot.

One collection of these footprint paths corresponds to the top of the car. In particular, considering the right plot in Figure 8, these nozzle/applicator footprints on the top of the car are depicted (in clockwise order) by the red, magenta, green, yellow, dark blue, and ceil blue (sky blue) paths, respectively. The other collection of these footprint paths corresponds to the sides (the driver's and the passenger's sides) of the car. In particular, in the left plot in Figure 4.13, the three colored paths correspond to the three nozzle/applicator footprint trajectories of the three robots that are located on the driver's side. In other words, the red, magenta, and green-colored paths represent the nozzle/applicator footprint of the three robots that are painting the vehicle on the driver's side. Due to symmetry, we have omitted the plot containing the three nozzle/applicator footprint paths (yellow, dark blue, and ceil blue) on the passenger's side.

It is remarked that due to the design procedure in 4.1 and 4.2 of our algorithm when the robots are painting the sides and top of the car, they do not collide with each other.

Remark - Our idea of partitioning the 3D workspace of the painting robots resembles the recent work by Kim and Son [11]. In that work, the authors propose using a Voronoi diagram-based workspace partition algorithm for weak collaboration of UGVs in the context of agricultural robotics. However, the algorithm in [11] relies on 2D node clustering calculations and covers a 2D agricultural land. On the other hand, our workspace partitioning takes place in a 3D setting (achieved in 4.1). Subsequently, path planning for the applicator footprint is performed in 4.2. Finally, based on the computations from the previous stages, robotic task assignment is done in 4.3.

4.3.2 3D Path Generation For the Paint Nozzles and Generation of Joint Trajectories For the Robotic Arms



(a) Path for four robots

(b) Path for six robots

Figure 4.14: Example of generating 3D boustrophedon-like paths of the paint nozzle 3D footprints for four and six robots.

In this step, we utilize the 2D boustrophedon-like paths from the previous stage to generate 3D boustrophedon-like paths for the paint nozzle footprints. To illustrate this process, let us consider Robot 1 and its associated cells on the side and the top of the car. Each of these cells are covered by 2D boustrophedon-like paths obtained from 4.2. In order to generate 3D boustrophedon-like paths for the paint nozzle footprint of Robot 1, we simply connect the last point of the boustrophedon-like path on the

side cell to the first point of the boustrophedon-like path on the top cell associated with Robot 1 by means of a straight line segment. The results of such process for 6 and 4 robots are depicted in Figure 4.14.

After obtaining the 3D paths of the footprint of the nozzles, we can use proper on-the-shelf algorithms that can provide us with the paint deposition model and robotic arm joint trajectories obtained from efficient inverse kinematics solvers

4.4 Overall Algorithm and Simulation Results

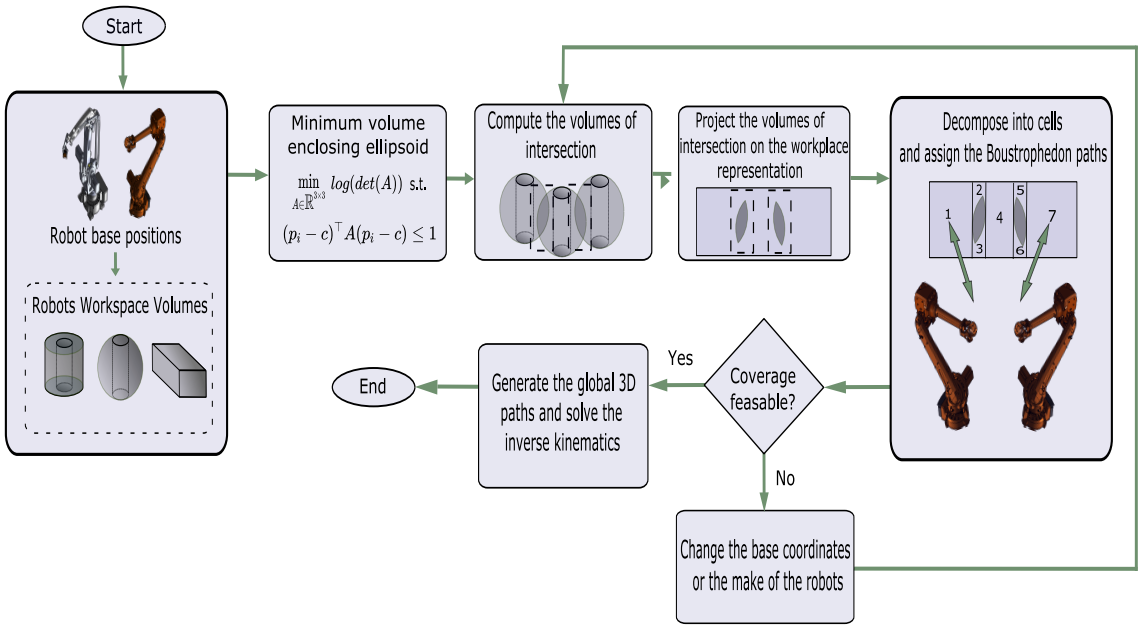


Figure 4.15: The flowchart associated with the proposed algorithm for CRCP.

The flowchart associated with the overall algorithm proposed in this paper is depicted in Figure 4.15. The overall computational complexity of our algorithm is determined by Khachiyan's algorithm in 4.1.3. In particular, we have a polynomial-time complexity of $O(N_0^{3.5} ln(\frac{N_0}{\epsilon}))$, where ϵ is the relative accuracy of the ellipsoids

fitted to the volumes of intersection in-between the reachable workspace of the robotic forearms, and the integer N_0 is proportional to the number of points in point clouds representing the reachable workspace of each robotic forearm, which is given by 4.5. For the particular case of collaborative robotic car painting, the obtained polynomial-time complexity is quite manageable for a modest PC with only one of its cores performing the required computations. For instance, the overall run-time of the algorithm implemented on MATLAB for a group of six ABB IRB 4600 robots on an Intel Core i7 CPU with an N_0 of the order of 10^3 and an ellipsoidal fitting tolerance of $\epsilon = 0.01$ is less than four minutes.

The snapshots of a homogeneous team of six robots painting a Ford Motor Company F-150 truck are depicted in Figure 4.16.

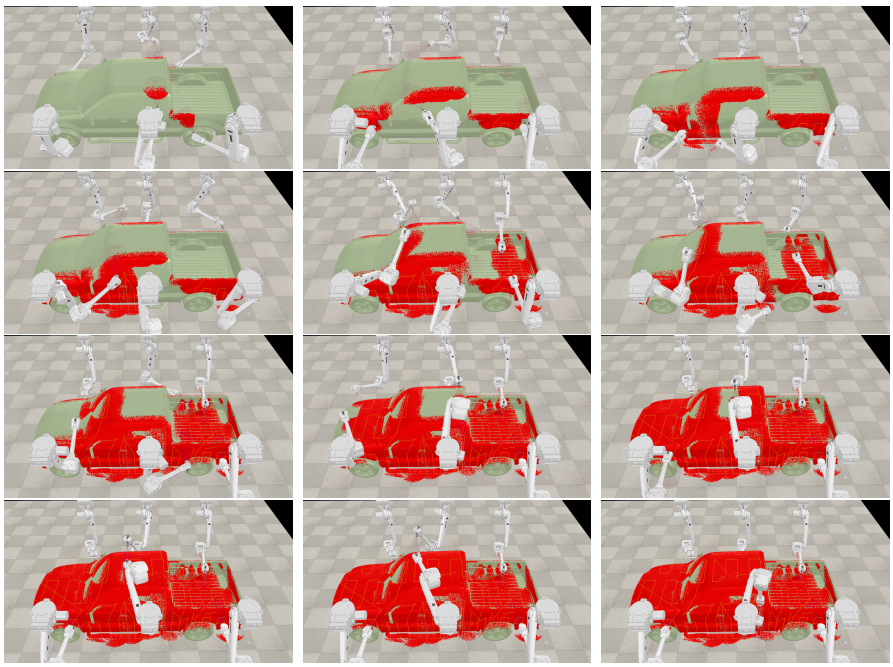


Figure 4.16: Snapshots of six ABB IRB 4600 robots painting an F-150 truck.

Our algorithm can be applied to heterogeneous teams of robots as well. This is particularly useful when one or more robotic arms become dysfunctional and there is a need to continue the painting process without any disruptions, the manufacturer

might need to replace the non-operational arms with robotic manipulators of a different kind from their inventory. The snapshots of a heterogeneous team of four robots painting a Ford Motor Company F-150 truck are depicted in Figure 4.17.

For the interested reader, a movie clip associated with our numerical simulations can be accessed through this link. The movie clip contains three painting scenarios: a team of six homogeneous robots (six ABB IRB 4600 robots), a team of four heterogeneous robots (two ABB IRB 4600 and two FANUC R-2000iC robots), and a team of four homogeneous robots (four ABB IRB 4600 robots).

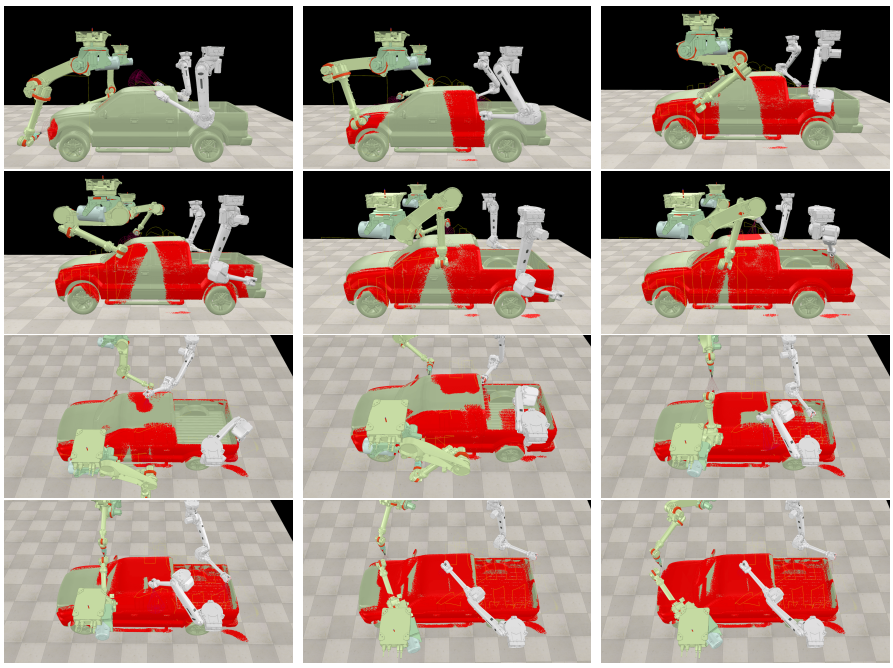


Figure 4.17: Snapshots of two ABB IRB 4600 and two FANUC R-2000iC/220U robots painting an F-150 truck.

Remark - To cope with the existence of multiple inverse kinematic solutions in path planning for serial robotic mechanisms and generating unique continuous paths in the robot joint space from a collection of potential paths, the community has proposed several solutions throughout the years. One of these popular solutions, which is also employed by CoppeliaSim and utilized in our paper, is based on utilizing damped least squares (DLS) numerical inverse kinematics solvers (also known as the

Levenberg-Marquardt solvers), which were originally proposed by Wampler [66] Given a sequence of desired end-effector positions and orientations, which form a continuous path in the end-effector pose space, and an initial desired joint configuration, the DLS-based inverse kinematics solvers are guaranteed to generate a unique and continuous sequence of joint configurations that closely solve for the inverse kinematics problems associated with the provided path in the end-effector pose space.

Another possibility, which is not utilized in this study, is based on the notion of aspects originally proposed by Borrel and Liégeois [97]. In particular, they introduced the notion of aspects to cope with the existence of multiple inverse kinematic solutions in serial manipulators. The aspects are defined as the maximal singularity-free domains in the robot joint space. For conventional industrial serial robotic arms, the aspects are the maximal sets in the robot joint space where there exists only one inverse kinematics solution.

CHAPTER V

Optimal Robotic Base Placement for Collaborative Industrial Robot

5.1 Related Work

The optimal placement of robotic bases is a critical factor in enhancing the performance and efficiency of collaborative industrial robots, particularly in tasks such as car painting. The positioning of a robot's base directly influences its reachability, manipulability, and the overall coverage of its workspace, which are essential for minimizing redundant movements and optimizing energy consumption in flexible production lines. Over the years, various approaches have been proposed to address the challenges associated with robot base placement, ranging from single-robot optimization to multi-robot coordination. These methods often involve complex computational algorithms, including genetic algorithms, particle swarm optimization, and sequential quadratic programming, to determine the best base positions that maximize task efficiency while avoiding collisions and singularities. In the context of multi-robot systems, the problem becomes even more intricate, as it requires balancing the competing objectives of maximizing coverage and minimizing interference among robots. This section reviews the existing literature on robotic base placement, highlighting the key methodologies and their applications in industrial settings, and sets the stage

for the proposed solution in this study.

5.1.1 Robot Base Placement

Base placement plays a critical role in optimizing performance across various robotic application domains, including industrial manufacturing [98; 99]. In particular, the placement of a robot's base affects the robot's reachability, manipulability, and workspace coverage during task execution [100]. These factors can, in turn, minimize redundant movements in flexible production lines with limited workspace and further optimize the robot energy consumption profiles [101; 102].

Optimal base placement on a single robot has been covered in the literature in different environments such as manufacturing environments [103] and underwater environments [104]. The genetic algorithm was used in [103] with a set of critical points that belong to the robot workspace to optimize the base placement. The main issue with finding the optimal base placement for a robot is that it can be computationally expensive. Some techniques that were used in the literature to simplify the problem by limiting the number of discrete end-effector positions [105] considered for base placement optimization. Others used objectives optimization that is specific to the task e.g. Maximizing manipulability for milling applications that require the robot's end-effector to move with precision to perform the task [106]. Fan et al [107] developed a solution approach based on sparse uniform grid decomposition and Sequential Quadratic Programming (SQP) to find the optimal base placement by considering the manipulability of the robot manipulator and multiple constraints (e.g. joint velocity and singularity avoidance...). Doan and Li [108] used the Particle Swarm Optimization algorithm to maximize reachability while avoiding singularities and collision.

Research in the literature has also focused on determining the optimal base posi-

tions for multiple robots. Hassan et al [109] used the Non-dominated Sorting Genetic Algorithm (NSGA-II) by maximizing average and manipulability while minimizing the makespan and joint torque. Kalawoun et al [110] used a Hybrid: Greedy, Genetic algorithm and Simulated annealing to minimize the number of robots while maximizing the coverage. Whale and Ant Colony optimization was also used to solve the problem [111] generating collision-free trajectories.

Table 5.1 provides an overview of various industrial robotic base placement algorithms that have been proposed in the literature. Many of these algorithms have been designed to be general and applicable to a wide range of applications including robotic welding, coating, and painting. Consequently, their generality prevents them from leveraging the inherent geometrical and task-specific features in particular industrial applications such as robotic-based vehicle painting. Furthermore, a majority of these algorithms take a holistic and centralized approach to the problem by combining base placement and motion/trajectory planning. However, centralized approaches to multi-robot path planning is plagued by NP-hardness, where the computational complexity grows exponentially with increasing robot numbers [112; 113]

5.1.2 Robot Workspace Computation

Robot workspace computation is a fundamental aspect of robotics. The workspace of a robot manipulator is defined as the set of points that can be reached by its end-effector. Accurately determining the reachable and usable workspace is of big importance in optimizing robot performance, ensuring task feasibility, and avoiding collisions in dynamic environments. Multiple methods have been proposed over the years to tackle this problem, ranging from geometric and analytical approaches to more sophisticated numerical and simulation-based methods.

Early works in workspace computation primarily focused on geometric and kine-

References	Number of Robots	Algorithm	Base Placement Objective
[106] Vosniakos and Matsas (2010)	Single	Genetic Algorithms	Optimizing manipulability and torque profiles
[114; 109] Hassan et al. (2016, 2018)	Multiple	Hybrid: Genetic Algorithm and simulated annealing	Maximizing the coverage while minimizing the makespan
[108] Doan and Li (2017)	Single	Particle Swarm Optimization	Maximizing reachability while avoiding singularities and collision
[110] Kalawoun and al (2018)	Multiple	Hybrid: greedy, genetic, and simulated annealing	Minimizing the number of robots while maximizing the coverage
[107] Fan et al (2021)	Single	Sequential quadratic programming	Maximizing the stiffness performance index while avoiding singularities and collision
[111] Mutti et al (2021)	Multiple	Whale and Ant Colony Optimization	Ensuring kinematic feasibility while checking for collision-free trajectories
[115] De Caro et al. (2022)	Single	Genetic algorithm	Maximizing the available workspace
[102] Ghungrad and Haghighi (2024)	Multiple	Metaheuristic Search	Energy efficiency in additive manufacturing

Table 5.1: A sample overview of robotic base placement algorithms proposed in the literature.

matic models of robots. Litvin proved in his work [116] that the robot manipulator is in a singular configuration when it reaches the workspace boundaries. His finding was used later in multiple studies [117; 118; 119; 120]. Kumar and Waldron [121] used reciprocal screws to describe the boundary surface of the manipulator's workspace. Freudenstein and Primrose [122] used skew-torus geometry to determine the workspace's boundaries of a three-link mechanism.

Forward kinematics plays a central role in calculating the reachable workspace by determining the robot's end-effector position for different joint configurations[123]. Zhang et al [124] proposed a method based on forward kinematics to generate the robot workspace of a parallel robot.

Numerical methods have been widely used in the literature. Monte Carlo methods, introduced in [125], were used in [126] for solving mathematical problems by means of random sampling to generate samples from a uniform distribution within a specified range. 2D workspace generations were presented in [127; 128] and were extended to 3D workspace generation in [129]. Monte Carlo β-distribution method [89] involves drawing samples from a continuous probability distribution defined on the interval $[0, 1]$ and Gaussian-Growth distribution [130] generates samples from a Gaussian distribution with increasing variance, simulating a "growth" pattern.

5.1.3 Multiobjective Optimization

Multiobjective optimization involves solving problems with multiple objectives, often in conflict with each other, necessitating a trade-off. It differs from single-objective optimization, which aims for a single optimal solution. Multiobjective optimization seeks a set of optimal solutions, known as the Pareto front. These solutions are considered optimal in the sense that no other solutions are superior without being inferior in at least one other objective.

5.1.3.1 The Weighted Sum Method

The Weighted Sum Method introduced by Churchman and Ackoff [131] represents one of the earliest and most fundamental approaches in multi-criteria decision-making. The method is based on assigning weights to reflect the importance of each criterion. Each normalized criterion value is multiplied by its corresponding weight and then these products are summed to obtain a a single composite score for each alternative.

5.1.3.2 Evolutionary Algorithms

Evolutionary Algorithms are a class of metaheuristic optimization algorithms inspired by biological evolution [132]. They apply mechanisms such as selection, mutation, recombination, and inheritance to a population of potential solutions to iteratively produce better approximations of a solution. The main advantage of these algorithms is the ability to explore a large search space and maintain a diverse set of solutions, which is crucial for identifying the Pareto front.

Non-dominated Sorting Genetic Algorithm II (NSGA-II) was introduced by Dab et al [133]. It classifies the population (a subset of solutions) into different fronts based on the level of domination. Then, the algorithm sorts the solutions and assigns crowding distance to each individual, which measures how close an individual is to their neighbors, to finish, the algorithm ensures that the best solutions from the current population are carried over to the next generation, thus preserving the best genes and enhancing convergence stability.

Strength Pareto Evolutionary Algorithm 2 (SPEA2), developed by Zitzler [134], finds a diverse set of Pareto-optimal solutions through a robust mechanism for fitness assignment and diversity preservation. The algorithm uses a technique based on the k-th nearest neighbor to estimate the density of solutions surrounding a particular individual while maintaining an external archive to store the best non-dominated solutions found during the search.

5.1.3.3 Goal Attainment Method

The goal attainment method was introduced by Gembiki [135]. The method uses a set of design goals, $F^* = F_1^*, F_2^*, ..., F_m^*$, associated with a set of objectives, $F(x) = F_1(x), F_2(x), ..., F_m(x)$. The core of the algorithm involves formulating the problem such that the solution seeks to minimize the maximum relative difference between the set of target goals and the actual performance measures. The mathematical formulation of this problem involves introducing a slack variable γ and a set of weights w_i to represent the relative importance of each objective.

The problem can be formulated as:

$$\underset{\gamma,x}{min} \text{ subject to: } F_i(x) - w_i \cdot \gamma \leq F_i^*,\ i = 1, 2, ..., m$$
$$c(x) \leq 0, \qquad c_{eq}(x) = 0, \qquad A \cdot x \leq b, \qquad A_{eq} \cdot x = b_{eq}, \qquad l \leq x \leq u$$

This formulation allows us to express the goals with a degree of flexibility, enabling the objectives to be over- or underachieved depending on the weight assigned.

The Goal Attainment method extends also to dynamic environments where objectives and constraints may evolve. This adaptability is crucial in fields like big data optimization, where real-time data can significantly influence decision-making processes [136].

5.2 The Base Placement Problem and High-Level Solution Strategy

To solve the multi-robot car-painting base placement problem, we begin by devising a scheme in 5.3.3 to render a point cloud representing the reachable workspace of the robotic arms' forearms and fit ellipsoids to these point clouds, a computation performed only once. In 5.4.1, we project these ellipsoids onto the floor or ceiling to derive planar ellipses and construct a cost function quantifying the potential for

robot collisions, parameterized by the Cartesian coordinates of the planar ellipse centers. 5.4.2, we develop a class of cost functions to measure vehicle paint coverage by projecting the point clouds from Section in 5.3.3 onto the vehicle's sides and top. Finally, in 5.4.3, we formulate a multi-objective optimization problem incorporating the cost functions from 5.4.1 and 5.4.2, enabling the use of any standard multi-objective optimizer to determine the optimal fixed base positions.

In this work, we adopt a hierarchical optimization approach to vehicle paint, which contrasts with methods that attempt to address base placement and painting objectives simultaneously. Our approach prioritizes optimal base placement, minimizing collision risk and maximizing the reachability of the robotic arms. This establishes a strong foundation for the subsequent stages, where path-planning algorithms, drawn from the existing literature, are applied to ensure complete paint coverage of the vehicle's surface. This approach offers two key advantages. First, by decoupling base placement from path planning, we enhance both computational efficiency and practical effectiveness. While complete paint coverage remains a critical goal, treating it as a constraint in the later stages of path optimization allows us to fine-tune the robot's trajectory while capitalizing on the earlier optimal base positioning. Second, once base placement is determined, there is flexibility to choose from a wide range of established path-planning algorithms in the literature such as [1; 137]

5.3 Point Cloud Rendering and Ellipsoidal Fitting

In this section, we focus on the 3D modeling of the reachable workspace for the robotic arms involved in the painting process. Specifically, we develop a numerical algorithm to render point clouds representing the space that the forearms of these robots can access. To further refine our model, we fit 3D ellipsoids to these point clouds, providing a more precise mathematical representation of the workspaces. These computations serve as a critical pre-processing step, setting the stage for subsequent path

planning and collision avoidance tasks. The 3D computations in this section can be considered as the pre-processing stage and **need to be run only once**.

We consider a team of M painting robots, each equipped with an articulated RRR arm (See the assumptions in 3.1) We describe the base positions of the robots relative to a common coordinate frame rigidly attached to the factory floor/ceiling. Note that some robotic arm manufacturers, such as FANUC with their R-2000iC/220U model ([138]), offer flip-over capability. This allows for ceiling installation in work cells unsuitable for floor or wall mounting, accommodating narrower cell designs. Without loss of generality, we consider the robots to be floor-mounted and assume that the floor coincides with the plane $z = 0$. Accordingly, we can express the base positions as

$$\mathbf{p}^i = [p_{x,i},\, p_{y,i},\, 0]^\top \in \mathbb{R}^3,\, 1 \leq i \leq M, \tag{5.1}$$

where i is the enumeration index for the robotic arms. In base placement problem, the coordinates $p_{x,i}$ and $p_{y,i}$, $1 \leq i \leq M$, are unknown and will be determined through the optimal base placement algorithm.

Let θ_k^i represent the angular position of the k^{th} joint angle of the i^{th} robot, where $1 \leq i \leq M$ and $1 \leq k \leq 3$. The joint angle vector for the i^{th} robot is then expressed as

$$\boldsymbol{\theta}^i := [\theta_1^i,\, \theta_2^i,\, \theta_3^i]^\top, \tag{5.2}$$

where $\theta_{k,\min}^i \leq \theta_k^i \leq \theta_{k,\max}^i$. Specifically, the constant angles $\theta_{k,\min}^i$ and $\theta_{k,\max}^i$ represent the minimum and maximum joint angle limits, respectively, for the k^{th} joint of the i^{th} robot (e.g., see [139] for limit values of the ABB IRB 4600).

5.3.1 Robot Workspace Point Cloud Generation

To determine the reachable volume of each robot's forearm, we employ Denavit-Hartenberg parameters for the first three joints and forward kinematics mapping of the robot. Specifically, the forearm tip position $\mathbf{t}^i \in \mathbb{R}^3$ of the i^{th} robot with its base at $\mathbf{p}^i \in \mathbb{R}^3$ on the factory floor/ceiling, $1 \leq i \leq M$, can be computed from the forward kinematics mapping $T^i(\boldsymbol{\theta}^i) \in \mathbb{R}^{4\times 4}$. This mapping relates the first three joint angles to the forearm tip's orientation and position. The forearm positions can be expressed as

$$\begin{bmatrix} \mathbf{t}^i(\boldsymbol{\theta}^i, \mathbf{p}^i) \\ 1 \end{bmatrix} = T^i(\boldsymbol{\theta}^i) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{p}^i \\ 0 \end{bmatrix}, \; 1 \leq i \leq M. \tag{5.3}$$

To maintain generality for any RRR industrial manipulator (see the assumptions in Section 3.1), we approximate reachable forearm spaces using a sampling-based numerical approach. Unlike analytical or geometric methods (e.g., [90]), sampling techniques (see, e.g., [89; 140; 141]) generate 3D point clouds amenable to efficient Khachiyan's algorithm [88; 142]. It is remarked that while β-distribution or Gaussian growth-based sampling algorithms (see5.1.2) [89], [130] are also viable options for workspace point cloud generation, uniform joint position sampling proved sufficient for our car-painting base placement application.

The reachable volume of the i^{th}, where $1 \leq i \leq M$, robot forearm tip is represented by a 3D point cloud. This point cloud can be expressed as

$$\mathscr{P}_i\big(\mathbf{p}^i\big) = \Big\{ \mathbf{t}^i(\boldsymbol{\theta}^i_j, \mathbf{p}^i) \Big\}_{j=1}^{N_i}. \tag{5.4}$$

In Equation (5.4), $\mathbf{t}^i(\boldsymbol{\theta}^i_j, \mathbf{p}^i)$ is the position of the forearm tip under base position $\mathbf{p}^i$ and joint configuration vector $\boldsymbol{\theta}^i_j$ that can be computed using the forward kinematics

mapping in (5.3). Each of the configuration vectors $\boldsymbol{\theta}_j^i$, which is associated with the i^{th} robot, is a sample drawn from the cube

$$[\theta_{1,\min}^i,\, \theta_{1,\max}^i] \times [\theta_{2,\min}^i,\, \theta_{2,\max}^i] \times [\theta_{3,\min}^i,\, \theta_{3,\max}^i]. \tag{5.5}$$

Consequently, the number of samples N_i in the point cloud $\mathscr{P}_i\big(\mathbf{p}^i\big)$ is given by:

$$N_i = \Big(\left\lfloor \frac{\theta_{1,\max}^i - \theta_{1,\min}^i}{\delta_{i1}} \right\rfloor + 1 \Big) \Big(\left\lfloor \frac{\theta_{2,\max}^i - \theta_{2,\min}^i}{\delta_{i2}} \right\rfloor + 1 \Big) \Big(\left\lfloor \frac{\theta_{3,\max}^i - \theta_{3,\min}^i}{\delta_{i3}} \right\rfloor + 1 \Big). \tag{5.6}$$

In Equation (5.6), $\lfloor \cdot \rfloor$ denotes the floor function, and δ_{ij} is a constant that specifies the distance between the samples in the j^{th} angular interval. For example, if $\delta_{i1} = \frac{\pi}{10}$, the samples from the interval $[\theta_{1,\min}^i,\, \theta_{1,\max}^i]$ are $\frac{\pi}{10}$ radians apart. As it will be seen in what follows, the number of points N_i in (5.6) in each forearm point cloud directly affects the computational complexity of the 3D ellipsoidal fitting algorithm.

As seen from Equation (5.4), the robot workspace point clouds depend on the base positions. However, the base coordinates are not known beforehand. To address this issue, we can express the relationship in (5.4) as follows:

$$\mathscr{P}_i\big(\mathbf{p}^i\big) = \mathscr{P}_i\big(\mathbf{0}\big) + \big\{\mathbf{p}^i\big\},\ 1 \leq i \leq M. \tag{5.7}$$

This expression indicates that the point cloud $\mathscr{P}_i\big(\mathbf{p}^i\big)$ can be obtained by translating each element of a point cloud based at the origin. In particular, each element in the set $\mathscr{P}_i\big(\mathbf{p}^i\big)$ is the sum of one element from $\mathscr{P}_i\big(\mathbf{0}\big)$ and the base position vector $\mathbf{p}^i$. Accordingly, in the pre-processing stage of the algorithm, we can carry out point cloud rendering of the reachable workspace of the robot forearms at the origin without the need to know the base positions. As it will be seen in Sections 5.4.1 and 5.4.2, the base positions $\mathbf{p}^i$, $1 \leq i \leq M$, can be considered as optimization decision variables.

5.3.2 Ellipsoidal Fitting to the Robot Workspace Point Clouds

Consider the point cloud $\mathscr{P}_i(\mathbf{0})$ associated with the i^{th} robot, which consists of N_i points in $\mathbb{R}^3$. This point cloud $\mathscr{P}_i(\mathbf{0})$ is computed using (5.4) by setting $\mathbf{p}_i = \mathbf{0}$ (also, see 5.3.1).

To fit 3D ellipsoids to the point cloud $\mathscr{P}_i(\mathbf{0})$, we will use Khachiyan's algorithm, a widely used method in convex optimization [93; 88]. Khachiyan's algorithm is designed to compute the *minimum volume enclosing ellipsoid (MVEE)* of the point cloud $\mathscr{P}_i(\mathbf{0})$, denoted by $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$, which is found by solving the following optimization problem:

$$\begin{aligned} &\underset{\mathbf{A}_i, \mathbf{c}_i}{\text{argmin}} \ \log(\det(\mathbf{A}_i)), \\ &\text{subject to} \quad (\mathbf{t}_k - \mathbf{c}_i)^\top \mathbf{A}_i (\mathbf{t}_k - \mathbf{c}_i) \leq 1, \quad \text{for all } \mathbf{t}_k \in \mathscr{P}_i(\mathbf{0}) \end{aligned} \tag{5.8}$$

where $\mathbf{c}_i$ is the center of the ellipsoid $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$ and the matrix $\mathbf{A}_i$ provides the shape of the ellipsoid $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$. In particular, by computing the singular value decomposition of $\mathbf{A}_i$ as $\mathbf{U}_i \mathbf{\Xi}_i \mathbf{V}_i^\top$, the length of the ellipsoid semi-axes are given by $r_{k,i} = 1/\sqrt{\xi_{k,i}}$, $k \in \{x, y, z\}$, where $\xi_{k,i}$'s are the components of the diagonal of the matrix $\mathbf{\Xi}$. Furthermore, the orientation of the ellipsoid is given by the rotation matrix $\mathbf{V}_i$.

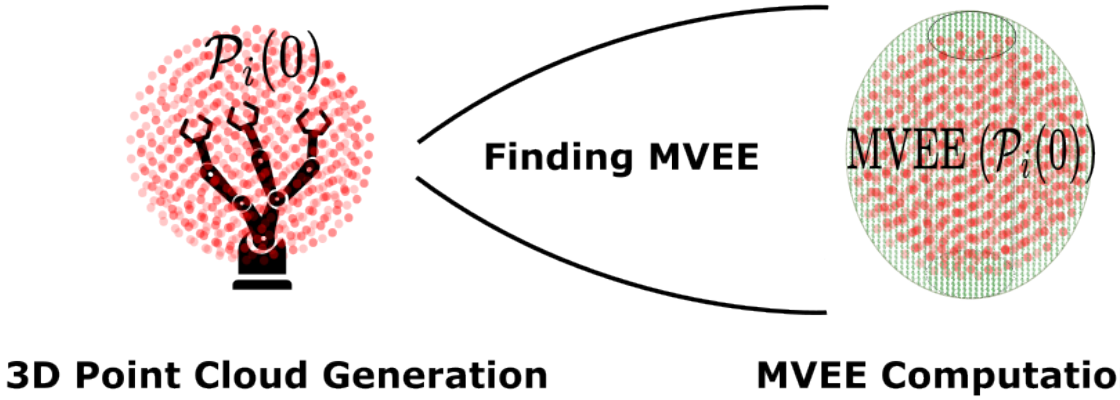


Figure 5.1: Algorithm 1 generates a 3D point cloud and a minimum volume enclosing ellipsoid (MVEE) for each of the robotic arms.

Algorithm 1 The pre-processing stage (run only once)

Input: The make of the M robotic arms and their kinematic parameters

Output: 3D point clouds $\mathscr{P}_i(\mathbf{0})$ and 3D ellipsoids $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$, $1 \leq i \leq M$

for $i \leftarrow 1$ to M **do**

  if Make and kinematic parameters of the i^{th} robot arm are the same as the j^{th} robot arm, where $j < i$ **then**

    $\mathscr{P}_i(\mathbf{0}) \leftarrow \mathscr{P}_j(\mathbf{0})$

    $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big) \leftarrow \text{MVEE}\big(\mathscr{P}_j(\mathbf{0})\big)$

    continue ▷ Terminate the current iteration and go to the next one.

  end if

  Compute the 3D point cloud $\mathscr{P}_i(\mathbf{0})$ associated with the reachable volume of the i^{th} robot forearm tip. ▷ Utilize 5.7 with $\mathbf{p}_i = \mathbf{0}$ and 5.3

  Fit an ellipsoid $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$ to the point cloud $\mathscr{P}_i(\mathbf{0})$. ▷ Solve 5.8 using Khachiyan's algorithm

end for

Return $\mathscr{P}_i(\mathbf{0})$, $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$, $1 \leq i \leq M$.

Khachiyan's algorithm solves the Lagrangian dual problem of (5.8) using a con-

ditional gradient ascent method [88] (see also [142]). In this paper, we utilize the implementation of Khachiyan's algorithm provided in [143].

Point cloud rendering of the robotic forearm reachable spaces and Khachiyan's algorithm result in M ellipsoids given by

$$\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big) = \left\{(x,\, y,\, z)\middle|\, \frac{(x-c_{x,i})^2}{r_{x,i}^2} + \frac{(y-c_{y,i})^2}{r_{y,i}^2} + \frac{(z-c_{z,i})^2}{r_{z,i}^2} = 1\right\},\ 1 \le i \le M. \tag{5.9}$$

Algorithm **??** summarizes the computations needed for the pre-processing stage. As depicted by Figure 5.1, Algorithm **??** generates a 3D point cloud and an MVEE for each of the robotic arms. To prevent unnecessary computations, the if-conditional in Algorithm **??** ensures that the point cloud rendering and ellipsoid fitting computations are done only for robotic arms having different kinematic parameters. For instance, if there are six ABB IRB 4600 robots, since their point clouds $\mathscr{P}_i(\mathbf{0})$, $1 \le i \le 6$ and corresponding fitted ellipsoids $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$, $1 \le i \le 6$ are the same, the rendering/fitting computations in the for-loop are only run for the first robot.

5.3.3 Computational Complexity for Point Cloud Generation and Ellipsoid Fitting

Computational Complexity for Point Cloud Generation. The forward kinematics for a RRR arm typically involves calculations using trigonometric operations (like sine and cosine) and matrix multiplications. These calculations are generally constant time operations, i.e., $O(1)$ for each configuration. Since we are computing forward kinematics for each configuration, the total computational complexity is the number of configurations multiplied by the complexity of forward kinematics. Thus, the total computational complexity for point cloud generation is $O(N_i)$.

Computational Complexity for Ellipsoid Fitting. As demonstrated in [88], the original algorithm for computing MVEEs can be solved in $O(N_i^{3.5} \ln(\frac{N_i}{\varepsilon}))$ arithmetic

operations, achieving a relative accuracy of ε in the volume.

5.4 Multi-Objective Optimization Problem

5.4.1 Cost Function 1: Minimizing the Possibility of Collision

In this section, we construct a cost function that quantifies the notion of collision possibility in the context of multi-robot car painting, which is based on computing the overlap area between planar ellipses.

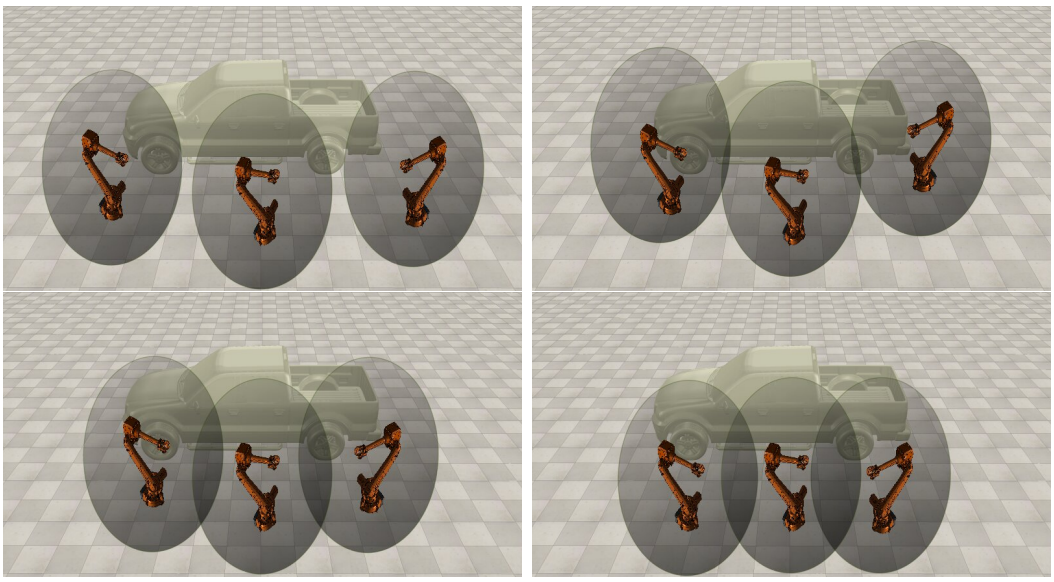


Figure 5.2: Intuition behind the first cost function formulation. The ellipsoids in the plots are the MVEEs that closely fit the reachable workspace of the robot forearms. The larger the volume of the intersection of the ellipsoids, the higher the possibility of robot collision during painting. Larger intersection volumes between the 3D ellipsoids, in turn, correspond to larger overlap areas of their projections on the factory floor.

To provide the intuition behind our proposed cost function in this section, let us consider the multi-robot arrangements depicted in Figure 5.2. The ellipses are the MVEEs that closely fit the reachable workspace of the forearms of the robotic arms. As it can be seen from these arrangements, larger volumes of intersection in between the ellipsoids amount to a higher possibility of collision in-between the robots

while they are painting the vehicle. Additionally, larger volumes of intersection in between the 3D ellipsoids correspond to larger areas of intersection in between their projections on the factory floor. Remarkably, the projection of these 3D ellipses on the factory floor results in planar ellipses. Hence, the overlap areas of planar ellipses can quantify the possibility of collision in-between the arms during painting (see Figure 5.3). Additionally, there are very efficient computational algorithms in the literature for computing the overlap area between ellipses as elaborated in what follows.

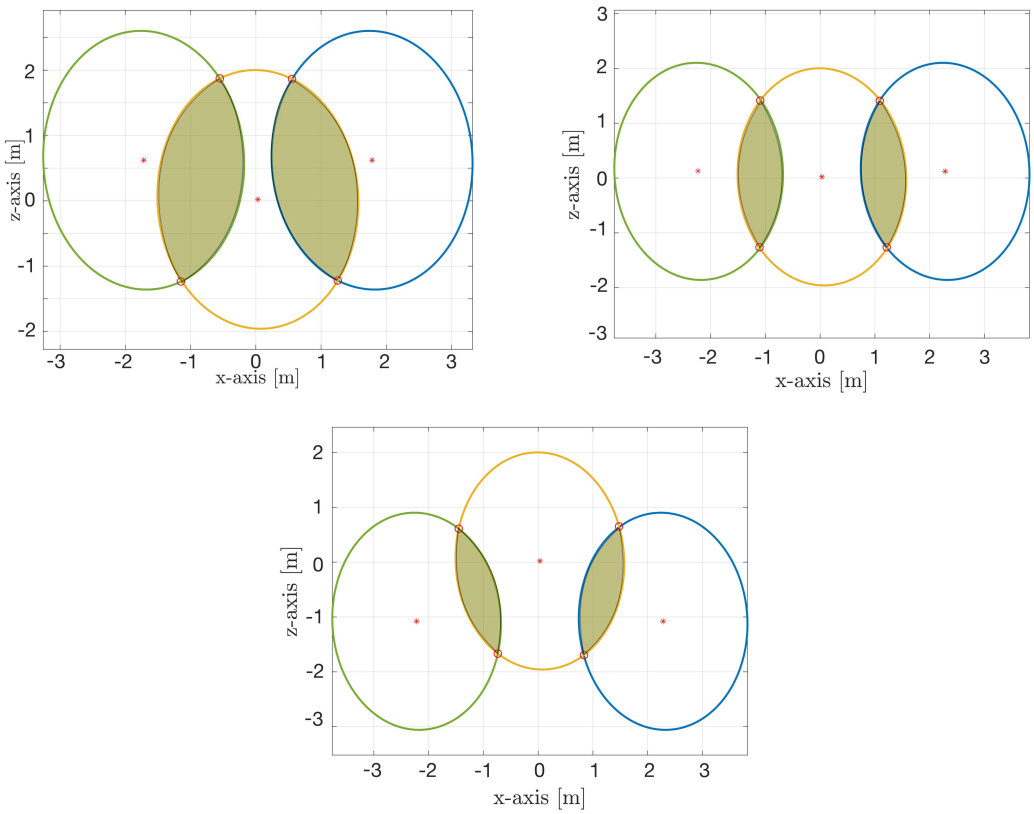


Figure 5.3: The planar ellipse overlap areas are employed to define the first cost function, which quantifies the possibility of collision in-between the robotic arms during painting.

To formalize this intuition, we first consider each of the ellipsoids $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$, $1 \leq i \leq M$, which are generated by the pre-processing stage summarized in Algorithm **??**. According to Remark 5.3.1, each computed ellipsoid $\text{MVEE}\big(\mathscr{P}_i(\mathbf{0})\big)$ is

obtained based on the assumption that its corresponding robot base position is located at the origin. Considering the variable unknown base position vector $\mathbf{p}_i$ given by (5.1), it is possible to translate the ellipsoid using this vector. In particular, the ellipsoid associated with the i^{th} arm with base position $\mathbf{p}_i$, $1 \leq i \leq M$, is given by

$$\text{MVEE}\big(\mathscr{P}_i(\mathbf{p}_i)\big) = \left\{(x,\, y,\, z)\middle|\, \frac{(x - c_{x,i} - p_{x,i})^2}{r_{x,i}^2} + \frac{(y - c_{y,i} - p_{y,i})^2}{r_{y,i}^2} + \frac{(z - c_{z,i})^2}{r_{z,i}^2} = 1\right\}. \tag{5.10}$$

We project the translated ellipsoids given by (5.10) onto the floor/ceiling to obtain planar ellipses, which are parameterized by the Cartesian coordinates of the robot base positions. Specifically, we project the ellipsoid $\text{MVEE}\big(\mathscr{P}_i(\mathbf{p}_i)\big)$, $1 \leq i \leq M$, given by (5.10) to obtain the planar ellipse

$$\mathscr{E}\big(\mathscr{P}_i(\mathbf{p}_i)\big) := \left\{(x,\, y)\middle|\, \frac{(x - c_{x,i} - p_{x,i})^2}{r_{x,i}^2} + \frac{(y - c_{y,i} - p_{y,i})^2}{r_{y,i}^2} = 1 - \frac{c_{z,i}^2}{r_{z,i}^2}\right\}. \tag{5.11}$$

Having obtained the planar ellipses, we denote the overlap area between the i^{th} ellipse $\mathscr{E}\big(\mathscr{P}_i(\mathbf{p}_i)\big)$ and the j^{th} ellipse $\mathscr{E}\big(\mathscr{P}_j(\mathbf{p}_j)\big)$, where $1 \leq i < j \leq M$, by $\mathscr{A}_{i,j}(\mathbf{p}_i, \mathbf{p}_j)$. It can be easily seen that $\mathscr{A}_{i,j}(\mathbf{p}_i, \mathbf{p}_j) \geq 0$ for all $\mathbf{p}_i$, $\mathbf{p}_j$, and $1 \leq i < j \leq M$ (i.e., overlap areas are positive or zero). Moreover, $\mathscr{A}_{i,j}(\mathbf{p}_i, \mathbf{p}_j) = \mathscr{A}_{j,i}(\mathbf{p}_j, \mathbf{p}_i)$ due to symmetry.

It is now possible to define the cost function quantifying the possibility of collision between the M robots during painting. In particular, using the ellipse overlap areas, we define the cost function $\mathscr{F}_1(\cdot)$ according to

$$\mathscr{F}_1\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big) := \sum_{1 \leq i < j \leq M} \mathscr{A}_{i,j}(\mathbf{p}_i, \mathbf{p}_j). \tag{5.12}$$

There are a variety of ellipse overlap area computation algorithms in the literature such as computationally intensive Monte Carlo methods [144], approximations based on polyarcs [145], and polygon-based methods [146]. Since we will feed the cost

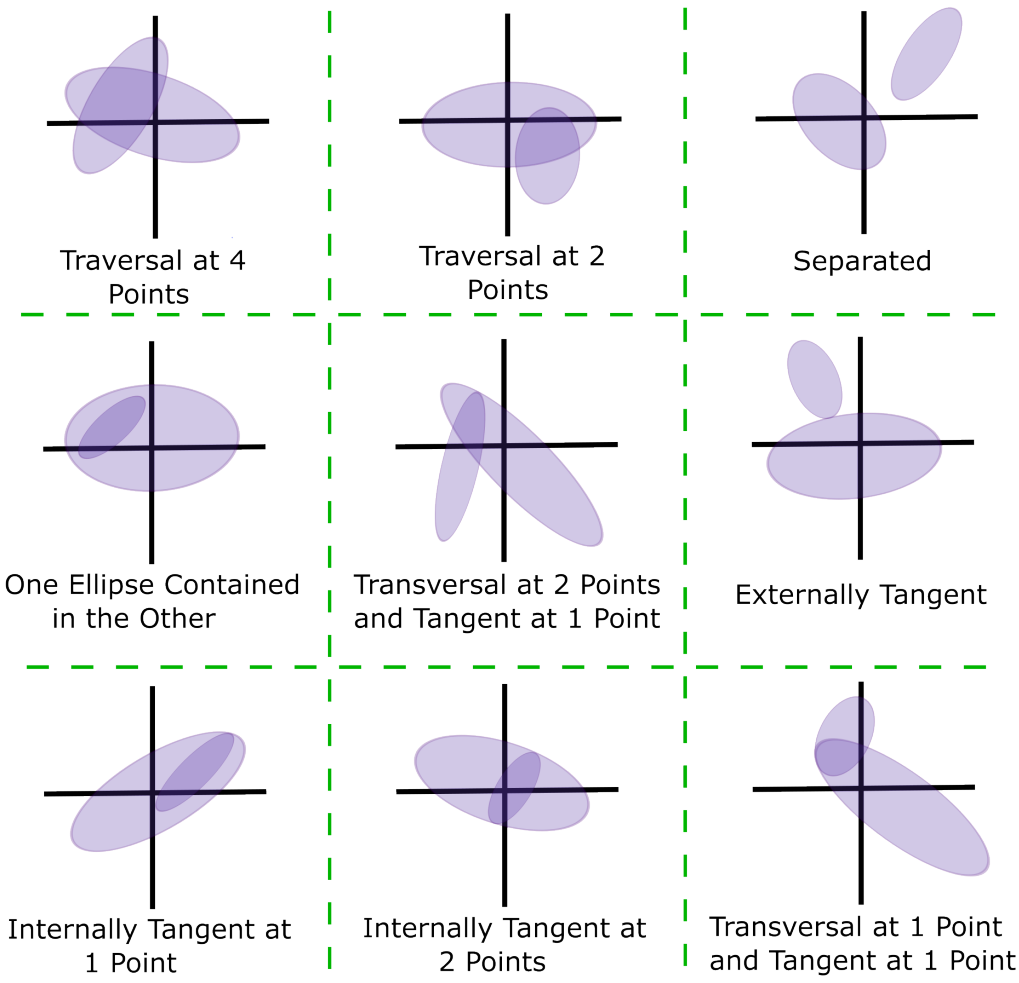


Figure 5.4: The algorithm proposed by [7] distinguishes a variety of different ellipse intersection cases. It computes the overlap area of any two general ellipses without resorting to proxy curves.

function $F_1(\cdot)$ given by (5.12) to a multi-objective optimizer, we need a fast and accurate algorithm for computing ellipse overlap areas. A very appealing algorithm suited for our needs is the one due to Hughes and Chraibi [7], which avoids the use of approximating proxy curves. Rather, this algorithm, which is based on a strategy that combines segment and polygon areas to compute the overlap, is applicable to any two general ellipses. The algorithm distinguishes a variety of different ellipse intersection configurations (see Figure 5.4) and utilizes the Gauss-Green theorem to compute segment areas (see [7] for further details).

Remark: While the overlap of robotic arm workspaces provides a useful heuristic for evaluating collision potential, it is important to recognize that the actual likelihood of collisions is primarily influenced by the motion planning and control strategies applied after base placement. Workspace overlap alone does not directly correlate with collision risk, as modern algorithms are capable of dynamically adjusting the robot trajectories to avoid collisions, even in scenarios where significant overlap exists. However, poor base placement can negatively impact the performance of subsequent path planning algorithms, potentially leading to suboptimal coverage or even failure in certain areas. To address this, advanced collision avoidance strategies that operate in densely populated robotic environments are applied after base placement. These strategies mitigate the risks associated with workspace overlap while maximizing the effective utilization of robotic arms. This hierarchical approach ensures that while base placement reduces collision potential, the subsequent dynamic adjustments during motion planning handle remaining collision concerns, thus optimizing both safety and efficiency.

5.4.2 Cost Function 2: Maximizing the Possibility of Paint Coverage

In this section we construct a group of cost functions to quantify the amount of vehicle paint coverage. These cost functions are obtained by projecting the point clouds, which are computed in the pre-processing stage, onto the sides and top of the vehicle.

We first create grid map representations of the top and sides of the car. These grid maps are computed using standard image processing algorithms like the Moore-Neighbor Tracing algorithm (see, e.g., [95]), which traces given contours and detects object boundaries in the vehicle's CAD model (see Figure 5.5). It is remarked that the grid map computation is needed to be *run only once*.

After creating the grid maps from the vehicle's CAD model, we consider the point

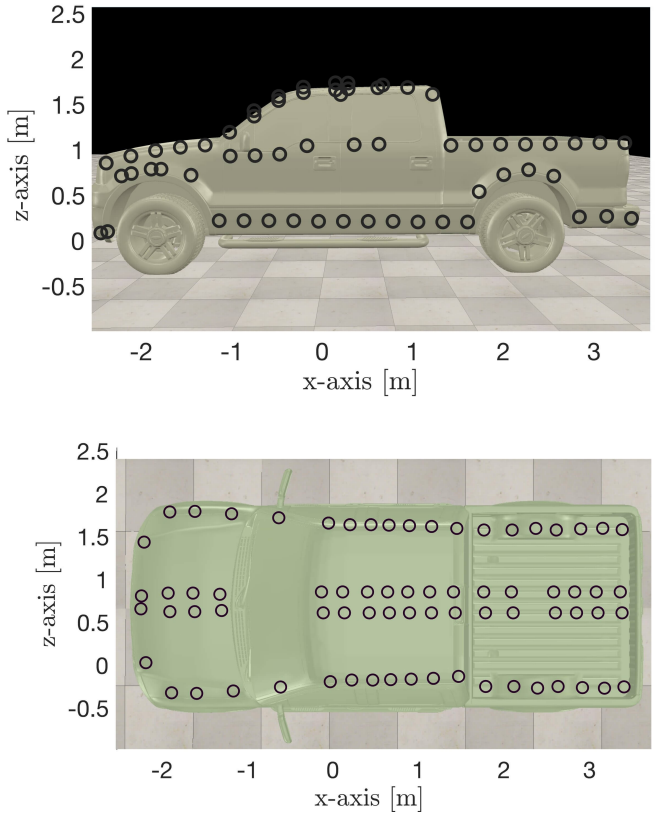


Figure 5.5: Boundary detection in the vehicle's CAD model and creation of the grip map representations.

cloud representation of the reachable workspace of the robotic forearms. In particular, we consider the point clouds $\mathscr{P}_i(\mathbf{0})$, $1 \leq i \leq M$, obtained from the pre-processing stage (see Section 5.3.3). Given a collection of base positions $\mathbf{p}_i$, $1 \leq i \leq M$, we translate each of the given point clouds $\mathscr{P}_i(\mathbf{0})$ to obtain their translated versions $\mathscr{P}_i(\mathbf{p}_i)$, $1 \leq i \leq M$, by utilizing Equation (5.7). Subsequently, we obtain a planar point cloud by projecting $\mathscr{P}_i(\mathbf{p}_i)$, $1 \leq i \leq M$, onto the sides and top of the vehicle. This is equivalent to the projection of $\bigcup_{i=1}^{M} \mathscr{P}_i(\mathbf{p}_i)$ onto the car sides and top.

Focusing on the vehicle top for brevity, we can assess its coverage using a computationally efficient technique like ray-casting (see, e.g., [147]). This method, which can determine if a point lies within a traced boundary from the Moore-Neighbor algorithm, can be used to provide a quantitative measure of the top's (resp. sides') paint

Algorithm 2 Computation of the second cost function (the car's top paint coverage ratio)

Remark: The algorithm structure remains the same for the sides of the car.

1: **Input:** Base positions $\mathbf{p}_i$, 3D point clouds $\mathscr{P}_i(\mathbf{0})$, $1 \leq i \leq M$, and grid map of the top of the vehicle CAD model of size $M_G^{\text{top}} \times N_G^{\text{top}}$.
2: **Output:** The paint coverage ratio of the vehicle top given by $\mathscr{F}_2^{\text{top}}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big)$.
3: Compute $\mathscr{P}_i(\mathbf{p}_i)$, $1 \leq i \leq M$, using (5.7).
4: Initialize `covered_cells` as an empty set.
5: Consider the vehicle top grid map of size $M_G^{\text{top}} \times N_G^{\text{top}}$.
6: Project $\bigcup_{i=1}^{M} \mathscr{P}_i(\mathbf{p}_i)$ onto the top of the vehicle to obtain a 2D point cloud.
7: Run a ray-casting computation to add cells to the `covered_cells`.
8: Compute the cardinality $N_{\text{covered}}^{\text{top}}$ of the set `covered_cells`.
9: Calculate the paint coverage ratio as:

$$\mathscr{F}_2^{\text{top}}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big) = \frac{N_{\text{covered}}^{\text{top}}}{M_G^{\text{top}} \times N_G^{\text{top}}}$$

10: **Return** $\mathscr{F}_2^{\text{top}}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big)$.

coverage. Algorithm 2 provides the procedure for computing the top paint coverage ratio. The same algorithm can be utilized for computing the paint coverage ratio of the vehicle sides, which we will skip the details for brevity.

In summary, the second cost function $\mathscr{F}_2^{\text{top}}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big)$, which is calculated according to Algorithm 2, can be expressed as

$$\mathscr{F}_2^{\text{top}}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big) = \frac{N_{\text{covered}}^{\text{top}}}{M_G^{\text{top}} \times N_G^{\text{top}}}, \tag{5.13}$$

where the resulting value of $\mathscr{F}_2^{\text{top}}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big)$ lies in the interval [0, 1]. We can repeat the same process (details omitted for brevity) for obtaining the coverage ratio of the sides of the car to obtain the cost functions $\mathscr{F}_2^{\text{side}_1}(\cdot)$ and $\mathscr{F}_2^{\text{side}_2}(\cdot)$ according to

$$\mathscr{F}_2^{\text{side}_1}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big) = \frac{N_{\text{covered}}^{\text{side}_1}}{M_G^{\text{side}_1} \times N_G^{\text{side}_1}}, \tag{5.14}$$

$$\mathscr{F}_2^{\text{side}_2}\big(\mathbf{p}_1, \cdots, \mathbf{p}_M\big) = \frac{N_{\text{covered}}^{\text{side}_2}}{M_G^{\text{side}_2} \times N_G^{\text{side}_2}}. \tag{5.15}$$

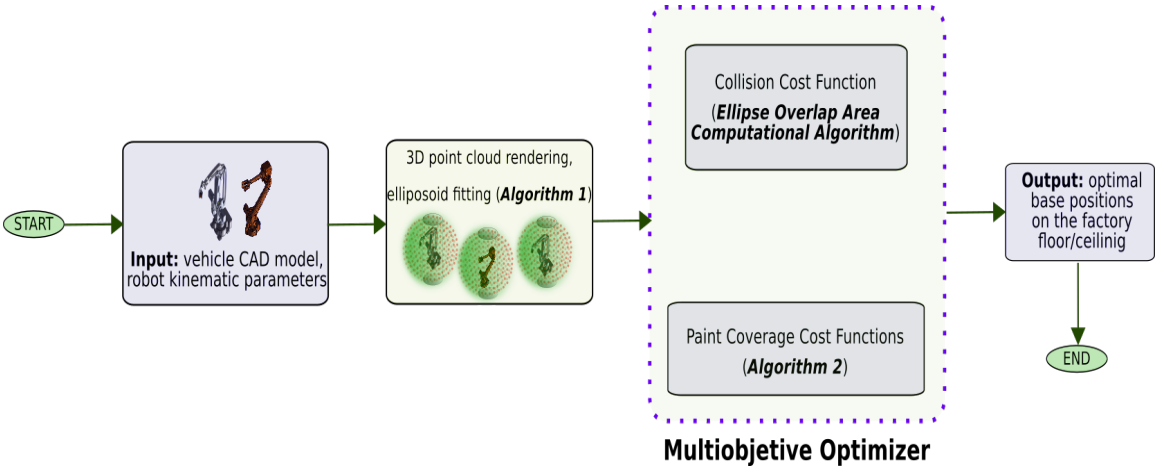


Figure 5.6: The overall proposed solution to the multi-robot car-painting base placement problem.

Computational Complexity for Computing the Paint Coverage Ratios. The overall computational complexity for testing the inclusion of N_i points from a point cloud in a boundary from a grid map of size $M_G^{\text{top}} \times N_G^{\text{top}}$ is $O(N_i \cdot M_G^{\text{top}} \cdot N_G^{\text{top}})$. Similar computational complexity expressions can also be stated for computing $\mathcal{F}_2^{\text{side}_1}(\cdot)$ and $\mathcal{F}_2^{\text{side}_2}(\cdot)$.

5.4.3 Multi-Objective Optimization Problem

In this section we formulate a proper multi-objective optimization problem using the cost functions developed in Sections 5.4.1 and 5.4.2.

Considering the base positions $\mathbf{p}_i$, $1 \leq i \leq M$, as the decision variables, the cost functions $\mathcal{F}_1(\cdot), \mathcal{F}_2^{\text{top}}(\cdot), \mathcal{F}_2^{\text{side}_1}(\cdot), \mathcal{F}_2^{\text{side}_2}(\cdot)$ constitute a proper objective function vector. We recall that $\mathcal{F}_1(\cdot)$ given by (5.12) quantifies the possibility of collision between the robots during painting. On the other hand, $\mathcal{F}_2^{\text{top}}(\cdot)$, $\mathcal{F}_2^{\text{side}_1}(\cdot)$, and $\mathcal{F}_2^{\text{side}_2}(\cdot)$ given by (5.13), (5.14), and (5.15), respectively, quantify the vehicle's top and sides paint coverage ratios.

We are interested in finding optimal decision variables $\mathbf{p}_i$, $1 \leq i \leq M$, such that

$\mathcal{F}_1(\cdot)$ is minimized while $\mathcal{F}_2^{\text{top}}(\cdot)$ (resp. $\mathcal{F}_2^{\text{side}_1}(\cdot), \mathcal{F}_2^{\text{side}_2}(\cdot)$) is maximized. Equivalently, we can seek to minimize $-\mathcal{F}_2^{\text{top}}(\cdot)$ (resp. $-\mathcal{F}_2^{\text{side}_1}(\cdot), -\mathcal{F}_2^{\text{side}_2}(\cdot)$). Using the cost functions $\mathcal{F}_1(\cdot)$, $-\mathcal{F}_2^{\text{top}}(\cdot)$, $-\mathcal{F}_2^{\text{side}_1}(\cdot)$, and $-\mathcal{F}_2^{\text{side}_2}(\cdot)$ we formulate the following multi-objective optimization problem

$$\min_{\mathbf{p}_1,\cdots,\mathbf{p}_M} \mathcal{F}_{\text{BP}}(\mathbf{p}_1,\cdots,\mathbf{p}_M) \tag{5.16}$$

where the objective function vector $\mathcal{F}_{\text{BP}}(\cdot)$ is defined by

$$\mathcal{F}_{\text{BP}}(\mathbf{p}_1,\cdots,\mathbf{p}_M) := \begin{bmatrix} \mathcal{F}_1(\mathbf{p}_1,\cdots,\mathbf{p}_M) \\ -\mathcal{F}_2^{\text{top}}(\mathbf{p}_1,\cdots,\mathbf{p}_M) \\ -\mathcal{F}_2^{\text{side}_1}(\mathbf{p}_1,\cdots,\mathbf{p}_M) \\ -\mathcal{F}_2^{\text{side}_2}(\mathbf{p}_1,\cdots,\mathbf{p}_M) \end{bmatrix}. \tag{5.17}$$

Having formulated the multi-objective optimization problem as in (5.16) and (5.17), we can invoke any standard multi-objective optimizer for computing the optimal fixed base positions. In this paper, we adopt the goal attainment algorithm [148], which has an efficient implementation through the function `fgoalattain` in MATLAB. To determine the goals in the goal attainment problem, we set the goal of the first cost function equal to 0. This indicates a goal value of zero for the overlap area between the projection of the robot workspace ellipsoids. Since we are minimizing the cost functions $-\mathcal{F}_2^{\text{top}}(\cdot)$, $-\mathcal{F}_2^{\text{side}_1}(\cdot)$, and $-\mathcal{F}_2^{\text{side}_2}(\cdot)$ in the objective function vector given by (5.17), we set their associated goal values equal to -1.

Remark: As ensuring 100% paint coverage of the vehicle is a critical requirement in any vehicle painting operation, our algorithm is designed to prioritize this objective. To achieve this, we employ the goal attainment optimization algorithm in MATLAB,

which uses a **Relative Attainment Factor** (weight) to balance multiple objectives. Specifically, the algorithm finds the smallest multiplier γ that satisfies the inequality:

$$F_i(x) - \text{goal}_i \leq \text{weight}_i \cdot \gamma \tag{5.18}$$

which allows for the flexible setting of weights on different objectives. In our case, one of the key objectives is the maximization of paint coverage.

The weight used in the fgoalattain function differs from traditional weights that simply prioritize objectives. Instead, the weight's sign determines how the algorithm treats the objective relative to its goal. When the weight is positive, the function attempts to reduce the objective function's value below the goal. Conversely, with a negative weight, the algorithm works to make the objective function meet or exceed the goal.

For our application, we set the weight in a way that ensures paint coverage is treated as a constraint, driving the algorithm to achieve 100% coverage. This ensures that the vehicle's entire surface is painted while allowing us to optimize other factors, such as minimizing the possibility of robot collisions.

Figure 5.6 illustrates the overall proposed solution to the multi-robot car-painting base placement problem. The pre-processing stage (see Algorithm **??**) performs point cloud rendering of the reachable workspace of the forearms of the robotic arms and fits ellipsoids to these point clouds. Using the constructed cost function in (5.12), we quantify the possibility of collision between the robotic arms without any need for planning their motion trajectories a priori. This cost function can be computed using the projection of the ellipsoids on the factory floor/ceiling and summing the resulting ellipse overlap areas. Similarly, using the constructed cost functions in (5.13), (5.14), and (5.15) computed using Algorithm 2, we quantify the possibility of the vehicle paint coverage. These cost functions can then be utilized within any standard multi-objective optimization routine to compute the optimal robot base positions.

Table 5.2: Impact of robotic base positions on paint coverage and manipulability with the homogeneous robotic team consisting of ABB IRB 4600 robots. The blue entries demonstrate the results obtained from our proposed optimal base placement algorithm.

Base positions (p_x, p_y) for $R_1\|R_2\|R_3$ (units: [m])	Overlap Areas (units: [m^2])	Paint Coverage (%)	Mean Normalized Manipulability (NM)	NM Standard Deviation
$(-4.5, 1)\|(0, 0)\|(4.5, 1)$	0	56.1	0.2476	0.1902
$(-3.5, 0)\|(0, 0)\|(3.5, 0)$	0	66.7	0.2669	0.2283
$(-3.0, 0)\|(0, 0)\|(3.0, 0)$	0.4929	80.3	0.2831	0.2340
$(-2.5, 1)\|(0, 0)\|(2.5, 1)$	2.05	98.5	0.6932	0.0974
$(-2, 1.3)\|(0, 0)\|(2, 1.3)$	4.12	100	0.7717	0.0919
$(-1.5, 0)\|(0, 0)\|(1.5, 0)$	9.13	90.9	0.4053	0.2428
$(-1.0, 1)\|(0, 0)\|(1.0, 1)$	10.88	92.4	0.7211	0.1317
$(-1.0, 0)\|(0, 0)\|(1.0, 0)$	12.96	83.3	0.4886	0.2719

5.5 Results

In this section, we present the simulation results associated with our optimal base placement algorithm. Our simulations are conducted using two different kinds of robotic arms, namely, ABB IRB 4600 and FANUC R-2000iC robots, for painting an F-150 truck produced by Ford Motor Company (see Figure 3.1). To demonstrate the versatility of our proposed base placement solution, we apply our algorithm to homogeneous as well as heterogeneous teams of robots. This is especially beneficial when one or more robotic arms malfunction, where the manufacturer can replace the faulty arms with alternative robotic manipulators from their inventory to avoid disruptions in the painting process.

To further assess the impact of base positions on the performance of the robotic arms during painting, we also compute the manipulability index (manipulability for brief) of the robots in our simulations. Given the Jacobian matrix $J(\boldsymbol{\theta})$ of a robotic arm, where $\boldsymbol{\theta}$ represents the configuration vector of the arm, the manipulability at $\boldsymbol{\theta}$ can be computed according to $w(\boldsymbol{\theta}) = \sqrt{\det(J(\boldsymbol{\theta})J(\boldsymbol{\theta})^\top)}$ (see [85, Section 4.12] for

further details). The normalized manipulability at configuration $\boldsymbol{\theta}$ during a certain task such as car painting is then given by $w_{\text{normalized}}(\boldsymbol{\theta}) := \frac{w(\boldsymbol{\theta})}{w_{\max}}$, where $w_{\max}$ is the maximum value of arm manipulability over all possible robot configurations.

Figure 5.6 illustrates the overall algorithm for optimal multi-robot car-painting base placement. The most computationally expensive part of the algorithm is the pre-processing stage (Algorithm **??**), which is needed to be run only once. Even on a single core of a modest PC, this polynomial-time algorithm can efficiently handle point cloud rendering and ellipsoid fitting. For instance, a MATLAB implementation of Algorithm **??** was run on six ABB IRB 4600 robots on an Intel Core i7 CPU. With point cloud sizes around 1000 and an ellipsoidal fitting tolerance of 0.01, the pre-processing computations were completed in less than one minute.

5.5.1 Simulations with a Homogeneous Robotic Team

We first consider six ABB IRB 4600 robotic arms and compute their optimal base positions using our proposed algorithm. Table 5.2 provides an overview of various base positions, planar ellipse overlap areas, percentages of vehicle paint coverage, and the mean and standard deviation values of the normalized manipulability of the robotic arms during painting. Since the robot arms are positioned in a symmetric manner on the sides of the F-150 vehicle, we are only including the information associated with one side.

The blue entries demonstrate the results obtained from our proposed optimal base placement algorithm. As shown in Table 5.2, the optimal base positions (the blue entries in the table) not only achieved complete paint coverage and minimal ellipse overlap areas but also exhibited the highest mean normalized manipulability with a low standard deviation. After base placement calculations, we utilize the multi-robot path planning algorithm devised in our previous work [1] to determine the joint trajectories of the robotic arms for painting the vehicle. In our previous work [1], we

Table 5.3: Computational times.

Base Positions (p_x, p_y) for $R_1\|R_2\|R_3$ (Units: [m])	**Overlap Area Computation Time (Units: [s])**	**Paint Coverage Computation Time (Units: [s])**	**Base Placement Algorithm Computation Time (Units: [s])**
$(-4.5, 1)\|(0, 0)\|(4.5, 1)$	0.2404	0.3550	-
$(-3.5, 0)\|(0, 0)\|(3.5, 0)$	0.2491	0.3538	-
$(-3.0, 0)\|(0, 0)\|(3.0, 0)$	0.2588	0.3566	-
$(-2.5, 1)\|(0, 0)\|(2.5, 1)$	0.2499	0.3664	-
$(-2, 1.3)\|(0, 0)\|(2, 1.3)$	0.2942	0.3664	197.24
$(-1.5, 0)\|(0, 0)\|(1.5, 0)$	0.2363	0.3705	-
$(-1.0, 1)\|(0, 0)\|(1.0, 1)$	0.2975	0.3897	-
$(-1.0, 0)\|(0, 0)\|(1.0, 0)$	0.2825	0.3671	-

assumed that the manufacturer would provide a suitable collection of base positions, usually from experience and/or heuristics. However, as it will be seen in the simulations, a poor choice of base positions for painting robots results in failure of the motion/trajectory planning.

Table 5.3 presents the computational times (in seconds) required to compute the overlap areas and paint coverage ratios for various base positions, using an Intel Core i7 CPU. Additionally, the multi-objective optimizer fgoalattain [135] took approximately 3.33 min to determine the optimal base positions, highlighted in bold in the table.

Figure 5.7 depicts the snapshots of the homogeneous team of six ABB IRB 4600 robots painting a Ford Motor Company F-150 truck where the robot base positions were obtained from our proposed optimal base placement algorithm. Furthermore, Figure 5.8 depicts the snapshots of the same team placed at non-optimal positions. The figures demonstrate that in the non-optimal case the robots failed to paint the top of the vehicle completely, despite the fact that both painting scenarios were utilizing the same motion planning algorithm. Furthermore, suboptimal base positions forced the robots to traverse configurations close to elbow and shoulder singularities

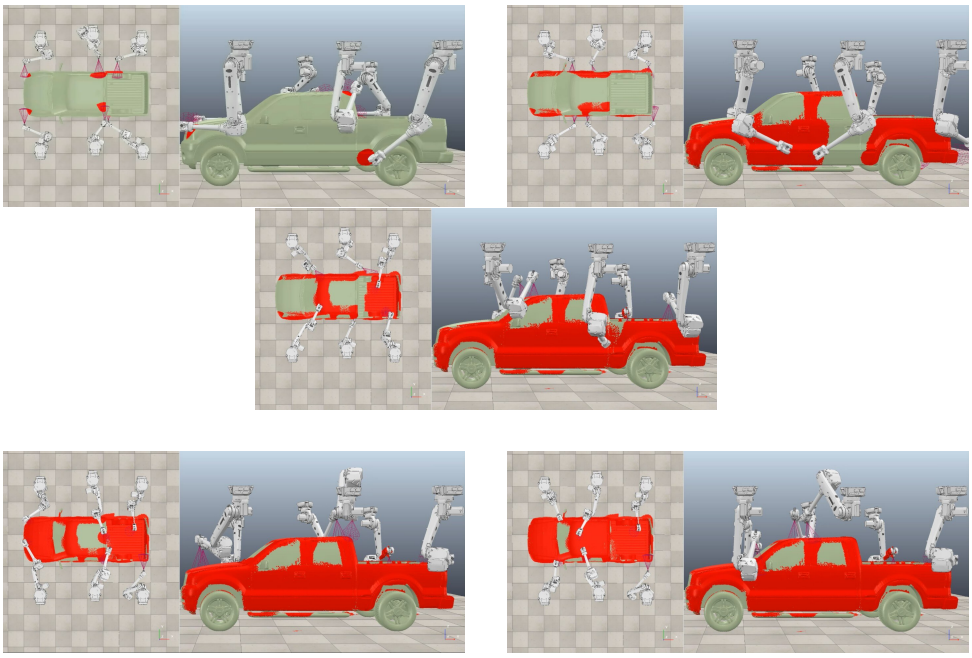


Figure 5.7: Snapshots of multi-robot car painting (a homogeneous team of robots) utilizing the optimal robot base positions obtained from our base placement algorithm.

during painting, which can be clearly seen when the arms have to overstretch to perform their painting task. Finally, the histograms in Figure 5.12 depict the normalized manipulability values of the robots during painting. The percentage values on each bar in the histograms provide the painting durations in which the robot manipulability belonged to a certain interval. For instance, a 50% value on the bar in $[0.85, 0.9]$ for Robot 1 in histograms on the left, indicate that Robot 1 had a normalized manipulability value belonging to the interval $[0.85, 0.9]$ during half of the painting task duration.

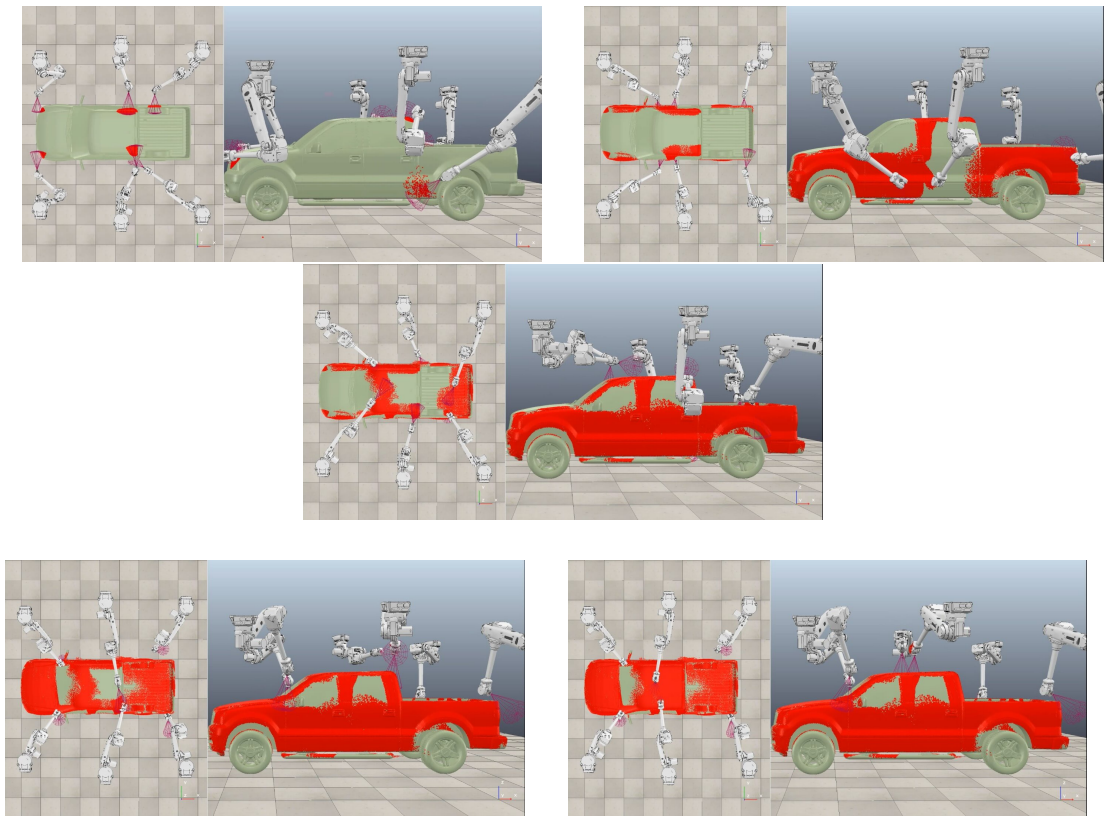


Figure 5.8: Snapshots of multi-robot car painting (a homogeneous team of robots) utilizing non-optimal base positions.

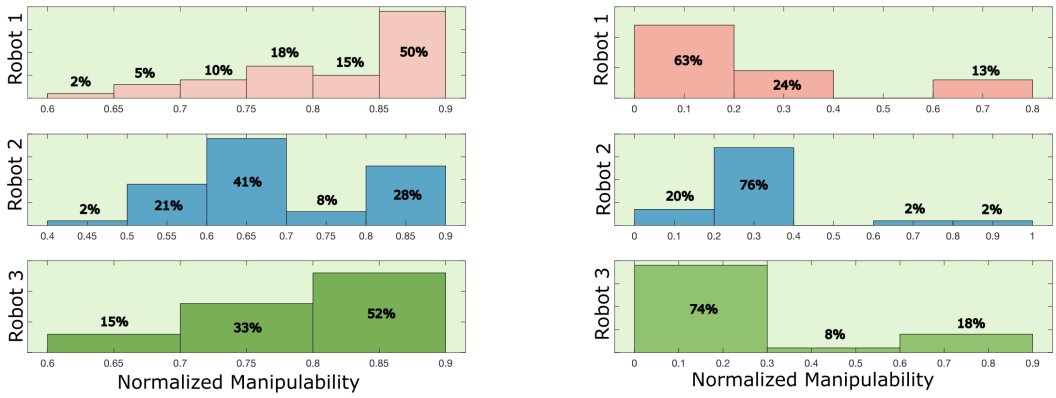


Figure 5.9: The normalized manipulability of the robotic arms during collaborative vehicle painting: (top) optimal; and, (bottom) non-optimal base positions. The percentage values on each bar in the histograms provide the painting durations in which the robot manipulability belonged to a certain interval. In particular, a percentage value of $p_0\%$ on a bar located in the interval $[\sigma_1, \sigma_2]$ for the i^{th} robot, $1 \leq i \leq 3$, indicates that the robot had a normalized manipulability value belonging to the interval $[\sigma_1, \sigma_2]$ during $p_0\%$ of the painting task duration.

5.5.2 Simulations with a Heterogeneous Robotic Team

collection of simulation results consider six robotic arms of different types, namely, four ABB IRB 4600 and two Fanuc R2000iC robots. Figure 5.10 depicts the snapshots of the heterogeneous team of robots painting the Ford Motor Company F-150 truck where the robot base positions were obtained from our proposed optimal base placement algorithm. Furthermore, Figure 5.11 depicts the snapshots of the same team placed at non-optimal positions. The figures demonstrate that in the non-optimal case the robots failed to paint the top of the vehicle completely, despite the fact that both painting scenarios were utilizing the same motion planning algorithm. Furthermore, suboptimal base positions forced the robots to traverse configurations close to elbow and shoulder singularities during painting, which can be clearly seen when the arms have to overstretch to perform their painting task.

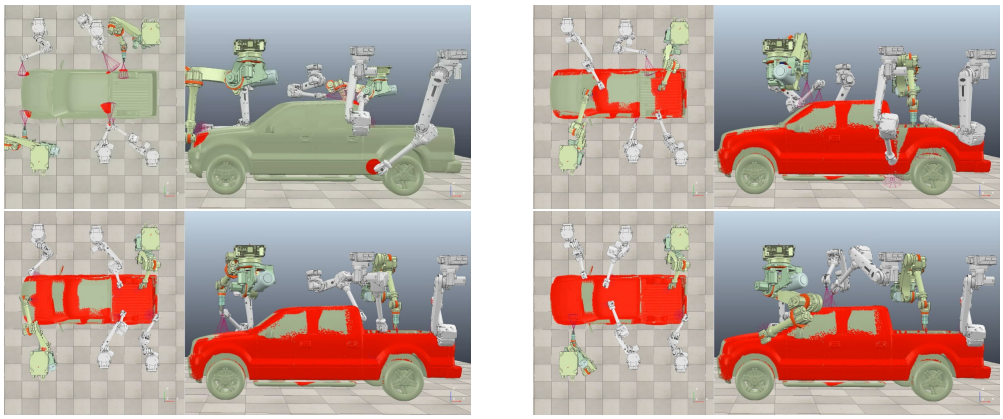


Figure 5.10: Snapshots of multi-robot car painting (a heterogeneous team of robots) utilizing the optimal robot base positions obtained from our base placement algorithm.

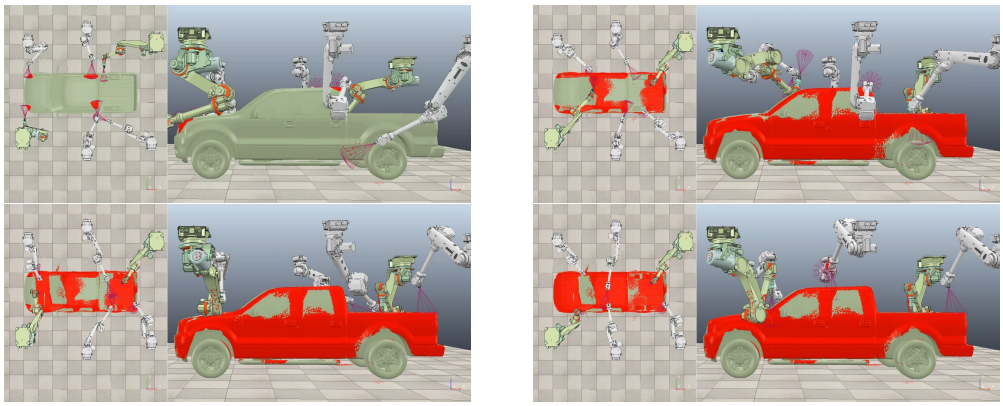


Figure 5.11: Snapshots of multi-robot car painting (a heterogeneous team of robots) utilizing non-optimal base positions.

5.6 Discussion and comparison with the state of the art

In developing our base placement algorithm for industrial painting robots, we initially utilized ellipsoidal fitting and point cloud rendering techniques (Algorithm **??**)during the pre-processing phase. This stage was critical for modeling the reachable workspace of the robots, ensuring that we had an accurate geometric representation to base further calculations on. Although this process involved polynomial time complexity, it was executed only once, thereby minimizing its impact on the overall computational load. Following the pre-processing, we implemented a multi-objective optimization strategy by utilizing custom-designed cost functions specifically aimed

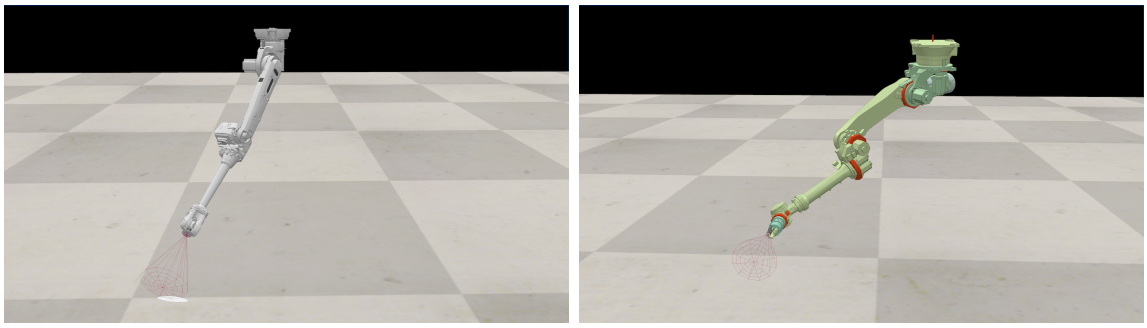


Figure 5.12: Elbow singular configurations of (left) ABB IRB 4600 and (right) FANUC R-2000iC robotic arms.

at minimizing collision risk (see section 5.4.1) and maximizing paint coverage (see Section 5.4.2). As it can be seen from Table 5.3, the computation of each of these cost functions took less than 0.4 seconds on a modest Intel Core i7 CPU. The primary advantage of our approach is decoupling the base placement from trajectory planning. Additionally, after base placement, one can choose from a plethora of robotic painting path-planning algorithms available in the literature.

The method of base placement for single robot machining operation proposed by [107] incorporates two main algorithms: sparse uniform grid decomposition (SUGD) and sequential quadratic programming (SQP). Initially, the SUGD is employed to identify feasible base positions within a defined search space. The SUGD requires significant computational resources, largely due to its dependency on the resolution of the grid and the extent of the search area. This approach serves as a preliminary filter to establish a starting point for further optimization. Following this, SQP takes over to find the optimal base positions using these initial guesses. SQP is well-suited for handling the nonlinearities and constraints typical in such optimization scenarios, offering an effective means to navigate through complex decision spaces. As for SQP, while it is capable of achieving superlinear convergence rates, the complexity, and the computational expense can escalate quickly depending on the size of the problem and the intricacies of the constraints involved. In the base position optimization proposed by [115], a GA was employed to iteratively evaluate and refine multiple solutions,

aiming to find the optimal configuration. While a GA provides flexibility in exploring various solutions, its iterative nature often results in higher computational demands, particularly when multiple constraints like collision avoidance and workspace coverage must be considered. Complementing the optimization, the Object Collision Algorithm was used to detect potential collisions between robot arms and their environment. This algorithm evaluated bounding boxes around objects with a quadratic computational cost. To further ensure efficient and stable robot movements, Singularity Detection through the Manipulability Index assessed the robot's Jacobian matrix for potential kinematic singularities. Additionally, Inverse Kinematics, often solved using Powell's method, calculated joint angles for specific end-effector positions. By integrating these algorithms, especially within a GA framework, the overall complexity increased, but the approach ensured a simultaneous optimization of robot base placement while addressing collisions, singularities, and precise kinematic configurations.

The authors in [102] developed a comprehensive framework for optimizing robot placement in large-scale manufacturing environments, utilizing an intricate blend of energy consumption models and quality metrics like dimensional accuracy. Central to this framework was the application of a simulated annealing algorithm—a heuristic method celebrated for its ability to approximate global optima efficiently. Complementing the simulated annealing was the Recursive Newton–Euler Algorithm (RNEA), which calculated the energy requirements of robotic movements. Moreover, the framework integrated a multi-objective optimization model that simultaneously addressed energy conservation and printing quality, navigating the complex trade-offs between these objectives. Additionally, the system incorporated task decomposition and scheduling algorithms to manage the assignment and coordination of tasks among multiple robots, ensuring collision-free operation and optimizing production timelines. The computational methods employed, like simulated annealing, can be

resource-intensive and might not scale well with larger numbers of robots or more complex tasks. Additionally, the effectiveness of these algorithms can depend heavily on initial conditions and parameter settings, which may complicate their implementation and optimization in practical scenarios.

The proposed algorithm is designed to work with a variety of industrial robot teams and vehicle models. Separating the task of base placement from motion and trajectory planning offers a more efficient solution that doesn't rely on motion planning data. This structured approach ensures that the results can be easily integrated into current robotic painting and trajectory planning systems.

CHAPTER VI

Future Work and Conclusion

This study has delved into collaborative robotic car painting, addressing the challenges posed by employing heterogeneous teams of industrial manipulators. By focusing on the efficient generation of joint trajectories for each robot, considering the CAD model of the car, the diverse robotic arms, and their fixed base positions within the factory, significant progress has been made toward optimizing the painting process while actively avoiding collisions.

A novel algorithm with polynomial-time complexity was introduced, tailored for collaborative robotic car painting, leveraging computational geometry and convex optimization techniques. It facilitates the partitioning of reachable forearm space and path planning for nozzle/applicator footprints on the car's grid map. Classical Morse cellular decomposition and boustrophedon algorithms, combined with off-the-shelf inverse kinematics solvers, enable the efficient generation of joint trajectories. This algorithm offers versatility by accommodating heterogeneous teams of industrial arms and diverse vehicle models. Furthermore, its practicality is enhanced by the absence of communication requirements among robotic arms during operation.

However, the algorithm also has certain limitations. For applications requiring finer surface coating, such as stealth coating of fighter jets, alternative non-boustrophedon-like paths are necessary to generate smoother applicator footprints.

Additionally, exploring faster implementations of Khachiyan's algorithm beyond the standard one employed could enhance computational efficiency. Addressing the challenge of planning paths for interacting with moving targets is another important future direction to expand the applicability of our approach.

This study also presented a multi-objective optimization framework for optimal fixed base placement of articulated robotic arms, aiming to maximize vehicle paint coverage and minimize the risk of collisions. Taking advantage of the two-dimensional geometric properties of car painting, we developed cost functions for paint coverage and collision avoidance. These were integrated into a multi-objective optimization problem solvable using standard optimization techniques. Our hierarchical approach decouples base positioning from motion and trajectory planning, enabling seamless integration with existing robotic motion planning systems.

Future research will focus on extending our framework to mobile manipulators, applying it to aircraft painting, and incorporating energy consumption optimization. Although the current study concentrates on kinematic manipulability for noncontact tasks like painting, we acknowledge the importance of force ellipsoid analysis for contact-intensive applications, such as welding and assembly. The alignment of the force ellipsoid with gravity can enhance the operational performance of the robot by minimizing torque requirements and improving stability. As such, incorporating force ellipsoid analysis into base placement and path planning algorithms is a promising direction for future research, which we aim to explore in subsequent studies. In certain industries, such as aerospace, special-shaped surfaces present unique challenges for robotic painting due to their non-uniform and complex geometries. While our current approach is designed for commercial vehicles, which generally have more regular shapes, multi-plane base placements may offer advantages for these specialized applications. Specifically, the ability to position robotic arms in different planes would allow greater flexibility in reaching difficult contours and ensuring full coverage.

In summary, future investigations will address the parallelization of the algorithm and the enhancement of manipulability to improve efficiency and precision in automotive surface coating applications. These efforts will pave the way for a wider adoption of robotic painting technologies across diverse industrial settings.

APPENDICES

.1 CoppeliaSim Simulation Code

```
% in a child script of a CoppeliaSim scene, add following command
% to be executed just once, at simulation start:
%
% simRemoteApi.start(19999)
%
% then start simulation, and run this program.
%
% IMPORTANT: for each successful call to simxStart, there
% should be a corresponding call to simxFinish at the end!

vrep=remApi('remoteApi'); % using the prototype file (rem oteApiProto.m)
vrep.simxFinish(-1); % just in case, close all opened connections


load(robotTrajectories.mat)

route1 = zeros(length(robot1));
for i = 1:length(robot1)
    route1(i) = i;
end
route2 = zeros(length(robot2));
for i = 1:length(robot2)
    route2(i) = i;
end
route3 = zeros(length(robot3));
for i = 1:length(robot3)
    route3(i) = i;
end

route4 = zeros(length(robot4));
```

```
for i = 1:length(robot4)
    route4(i) = i;
end

route5 = zeros(length(robot5));
for i = 1:length(robot5)
    route5(i) = i;
end

route6 = zeros(length(robot6));
for i = 1:length(robot6)
    route6(i) = i;
end
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
isReachPoint = false;


if (clientID>-1)
    disp('Connected to remote API server');

    time = 0;
    dt = 0;
    position_x = 0.2750;
    len_route1 = length(robot1);
    len_route2 = length(robot2);
    len_route3 = length(robot3);
    len_route4 = length(robot4);
    len_route5 = length(robot5);
    len_route6 = length(robot6);
    len = max([len_route1, len_route2, len_route3,len_route4, len_route5, len_route6]);
    isFirstTime = true;

     for i = 2:len
```

```
64    % calcultate the figure eight path
65     if (i > len_route1)
66             start_position_x1 = robot1(route1(len_route1),1);
67             start_position_y1 = robot1(route1(len_route1),2);
68             start_position_z1 = robot1(route1(len_route1),3);
69            end_position_x1 = robot1(route1(len_route1),1);
70            end_position_y1 = robot1(route1(len_route1),2);
71            end_position_z1 = robot1(route1(len_route1),3);
72         else
73             start_position_x1 = robot1(route1(i-1),1);
74             start_position_y1 = robot1(route1(i-1),2);
75             start_position_z1 = robot1(route1(i-1),3);
76            end_position_x1 = robot1(route1(i),1);
77            end_position_y1 = robot1(route1(i),2);
78            end_position_z1 = robot1(route1(i),3);
79     end
80
81         if (i > len_route2)
82             start_position_x2 = robot2(route2(len_route2),1);
83             start_position_y2 = robot2(route2(len_route2),2);
84             start_position_z2 = robot2(route2(len_route2),3);
85            end_position_x2 = robot2(route2(len_route2),1);
86            end_position_y2 = robot2(route2(len_route2),2);
87            end_position_z2 = robot2(route2(len_route2),3);
88         else
89              start_position_x2 = robot2(route2(i-1),1);
90             start_position_y2 = robot2(route2(i-1),2);
91             start_position_z2 = robot2(route2(i-1),3);
92            end_position_x2 = robot2(route2(i),1);
93            end_position_y2 = robot2(route2(i),2);
94            end_position_z2 = robot2(route2(i),3);
95         end
96
```

```
97          if (i > len_route3)
98              start_position_x3 = robot3(route3(len_route3),1);
99              start_position_y3 = robot3(route3(len_route3),2);
100             start_position_z3 = robot3(route3(len_route3),3);
101            end_position_x3 = robot3(route3(len_route3),1);
102            end_position_y3 = robot3(route3(len_route3),2);
103            end_position_z3 = robot3(route3(len_route3),3);
104
105         else
106              start_position_x3 = robot3(route3(i-1),1);
107             start_position_y3 = robot3(route3(i-1),2);
108             start_position_z3 = robot3(route3(i-1),3);
109            end_position_x3 = robot3(route3(i),1);
110            end_position_y3 = robot3(route3(i),2);
111            end_position_z3 = robot3(route3(i),3);
112         end
113
114          if (i > len_route4)
115             start_position_x4 = robot4(route4(len_route4),1);
116             start_position_y4 = robot4(route4(len_route4),2);
117             start_position_z4 = robot4(route4(len_route4),3);
118            end_position_x4 = robot4(route4(len_route4),1);
119            end_position_y4 = robot4(route4(len_route4),2);
120            end_position_z4 = robot4(route4(len_route4),3);
121         else
122              start_position_x4 = robot4(route4(i-1),1);
123             start_position_y4 = robot4(route4(i-1),2);
124             start_position_z4 = robot4(route4(i-1),3);
125            end_position_x4 = robot4(route4(i),1);
126            end_position_y4 = robot4(route4(i),2);
127            end_position_z4 = robot4(route4(i),3);
128          end
129
```

```
130             if (i > len_route5)
131               start_position_x5 = robot5(route5(len_route5),1);
132              start_position_y5 = robot5(route5(len_route5),2);
133              start_position_z5 = robot5(route5(len_route5),3);
134             end_position_x5 = robot5(route5(len_route5),1);
135             end_position_y5 = robot5(route5(len_route5),2);
136             end_position_z5 = robot5(route5(len_route5),3);
137           else
138               start_position_x5 = robot5(route5(i-1),1);
139              start_position_y5 = robot5(route5(i-1),2);
140              start_position_z5 = robot5(route5(i-1),3);
141             end_position_x5 = robot5(route5(i),1);
142             end_position_y5 = robot5(route5(i),2);
143             end_position_z5 = robot5(route5(i),3);
144           end
145
146             if (i > len_route6)
147              start_position_x6 = robot6(route6(len_route6),1);
148              start_position_y6 = robot6(route6(len_route6),2);
149              start_position_z6 = robot6(route6(len_route6),3);
150             end_position_x6 = robot6(route6(len_route6),1);
151             end_position_y6 = robot6(route6(len_route6),2);
152             end_position_z6 = robot6(route6(len_route6),3);
153           else
154               start_position_x6 = robot6(route6(i-1),1);
155              start_position_y6 = robot6(route6(i-1),2);
156              start_position_z6 = robot6(route6(i-1),3);
157             end_position_x6 = robot6(route6(i),1);
158             end_position_y6 = robot6(route6(i),2);
159             end_position_z6 = robot6(route6(i),3);
160             end
161
162   a = 1;
```

```
163     sample = 50;
164     position_x1=linspace(start_position_x1,end_position_x1,sample);
165     position_y1=linspace(start_position_y1,end_position_y1,sample);
166     position_z1=linspace(start_position_z1,end_position_z1,sample);
167
168     position_x2=linspace(start_position_x2,end_position_x2,sample);
169     position_y2=linspace(start_position_y2,end_position_y2,sample);
170     position_z2=linspace(start_position_z2,end_position_z2,sample);
171
172     position_x3=linspace(start_position_x3,end_position_x3,sample);
173     position_y3=linspace(start_position_y3,end_position_y3,sample);
174     position_z3=linspace(start_position_z3,end_position_z3,sample);
175
176     position_x4=linspace(start_position_x4,end_position_x4,sample);
177     position_y4=linspace(start_position_y4,end_position_y4,sample);
178     position_z4=linspace(start_position_z4,end_position_z4,sample);
179
180     position_x5=linspace(start_position_x5,end_position_x5,sample);
181     position_y5=linspace(start_position_y5,end_position_y5,sample);
182     position_z5=linspace(start_position_z5,end_position_z5,sample);
183
184     position_x6=linspace(start_position_x6,end_position_x6,sample);
185     position_y6=linspace(start_position_y6,end_position_y6,sample);
186     position_z6=linspace(start_position_z6,end_position_z6,sample);
187
188      if  position_y1 < -0.9
189          eul1 = [-pi/2 0 0];
190          quat1 = eul2quat(eul1);
191     else
192          eul1 = [0 pi  0];
193          quat1 = eul2quat(eul1);
194     end
195
```

```
196         if  position_y2 < -0.8
197             eul2 = [-pi/2 0 0];
198             quat2 = eul2quat(eul2);
199         else
200             eul2 = [0 pi  0];
201             quat2 = eul2quat(eul2);
202         end
203
204         if  position_y3 < -0.9
205             eul3 = [-pi/2 0 0];
206             quat3 = eul2quat(eul3);
207         else
208             eul3 = [0 pi  0];
209             quat3 = eul2quat(eul3);
210         end
211
212
213 if  position_y4 > 2.21
214             eul4 = [pi/2 0  0];
215             quat4 = eul2quat(eul1);
216
217 else
218             eul4 = [0 pi  0];
219             quat4 = eul2quat(eul4);
220 end
221
222 if  position_y5 > 2.21
223             eul5 = [pi/2 0  0];
224             quat5 = eul2quat(eul5);
225 else
226             eul5 = [0 pi  0];
227             quat5 = eul2quat(eul2);
228 end
```

```
229
230     if  position_y6  > 2.21
231         eul6  = [pi/2 0  0];
232         quat6 = eul2quat(eul6);
233
234     else
235         eul6  = [0 pi  0];
236         quat6 = eul2quat(eul6);
237
238     end
239         isReachPoint = false;
240
241     % R2000iC: [0 pi/2 0]   Top of the car
242     % R2000iC: [0 0 -pi/2] Side of the car
243
244     while ~isReachPoint
245         % calcultate the figure eight path
246 vrep.simxCallScriptFunction(clientID, 'Arm_3',vrep.sim_scripttype_childscript
      ,'SetPos' ,[1],[ position_x1(a) , position_y1(a), position_z1(a),
      quat1,eul1], '' ,[], vrep.simx_opmode_oneshot);
247
248 vrep.simxCallScriptFunction(clientID,'Arm_1',vrep.sim_scripttype_childscript ,'SetPos',
      [1],[ position_x2(a) , position_y2(a), position_z2(a),quat2,eul2], '' ,[]
      ,vrep.simx_opmode_oneshot);
249
250 vrep.simxCallScriptFunction(clientID,'Arm_5',vrep.sim_scripttype_childscript
      ,'SetPos' ,[1],[ position_x4(a) , position_y4(a), position_z4(a), quat4,
      eul4], '' ,[], vrep.simx_opmode_oneshot);
251
252 vrep.simxCallScriptFunction(clientID,'Arm_4',vrep.sim_scripttype_childscript
      ,'SetPos' ,[1],[ position_x5(a) , position_y5(a), position_z5(a), quat5,
      eul5], '' ,[], vrep.simx_opmode_oneshot);
253
```

```
254 vrep.simxCallScriptFunction(clientID,'Arm_6',vrep.sim_scripttype_childscript
      ,'SetPos' ,[1],[ position_x6(a) , position_y6(a), position_z6(a), quat6,
      eul6], '' ,[], vrep.simx_opmode_oneshot);
255
256 [returnCode,outInts,outFloats,outStrings,outBuffer] = vrep.simxCallScriptFunction
      (clientID,'Arm_2',vrep.sim_scripttype_childscript ,'SetPos' ,[1],[ position_x3(a),
      position_y3(a), position_z3(a), quat3, eul3], '' ,[], vrep.simx_opmode_blocking);
257
258           if isFirstTime
259               disp("Press Enter to continue !!!") ;
260               pause();
261               isFirstTime = false;
262           end
263
264           a = a + 1;
265           if a > sample
266               isReachPoint = true;
267               a = 1;
268           end
269 %            vrep.simxSynchronousTrigger(clientID);
270       end
271     % send the deired position to Vrep
272
273     % read the Vrep simulation timestamp
274 %           vrep.simxSynchronousTrigger(clientID);
275     end
276
277     vrep.simxSynchronousTrigger(clientID);
278
279     vrep.simxGetPingTime(clientID)
280     % close the connection to CoppeliaSim:
281     vrep.simxFinish(clientID);
282 end
```

```
283 %%% end coding
284 % Before closing the connection to CoppeliaSim, make sure that the last command sent out
      had time to arrive. You can guarantee this with (for example):
285 vrep.simxGetPingTime(clientID)
286 % Now close the connection to CoppeliaSim:
287 vrep.simxFinish(clientID);
288 vrep.delete(); % call the destructor!
289
290 disp('Program ended');
```

.2 Ellipse Overlap Area of Intersection Code

```
1  function OverlapArea = EllipseOverlap(a, b, phi, cx1, cy1, cx2, cy2, intersect1, intersect2)
2
3  % Find the overlap area between the center robot and the robot on the right:
4  intersectX1=intersect1(1);
5  intersectX2=intersect2(1);
6  intersectY1=intersect1(2);
7  intersectY2=intersect2(2);
8
9  E11_x1 = intersectX1;
10 E11_x2 = intersectX2;
11 E11_y1 = intersectY1;
12 E11_y2 = intersectY2;
13
14 E21_x1 = intersectX2;
15 E21_x2 = intersectX1;
16 E21_y1 = intersectY2;
17 E21_y2 = intersectY1;
18
19 E11_theta11 = FindTheta(E11_x1, E11_y1, cx1, cy1, a, b, phi);
```

```
E11_theta12 = FindTheta(E11_x2, E11_y2, cx1, cy1, a, b, phi);
E21_theta11 = FindTheta(E21_x1, E21_y1, cx2, cy2, a, b, phi);
E21_theta12 = FindTheta(E21_x2, E21_y2, cx2, cy2, a, b, phi);

E11_SegmentArea = SegmentArea(E11_theta11, E11_theta12, a, b, E11_x1,E11_x2,cx1,
    E11_y1,E11_y2,cy1);
E21_SegmentArea = SegmentArea(E21_theta11, E21_theta12, a, b, E21_x1,E21_x2,cx2,
    E21_y1,E21_y2,cy2);

OverlapArea = real(E11_SegmentArea + E21_SegmentArea);
```

```
function theta = FindTheta(x, y, cx, cy, A, B, phi)

xx = cos(phi)*(x-cx) + sin(phi)*(y-cy);
yy = -sin(phi)*(x-cx) + cos(phi)*(y-cy);

if (xx<0 && yy>=0)
    theta = acos((xx)/A);
% %     theta = pi - asin(yy/B);a
elseif (xx>=0 && yy>=0)
    theta = acos((xx)/A);
%     theta = asin(yy/B);
elseif (xx<0 && yy<0)
    theta = 2*pi - acos((xx)/A);
%     theta = pi - asin(yy/B);
else
    theta= 2*pi - acos((xx)/A);
%     theta = 2*pi + asin(yy/B);
end
```

```
function SectorArea = SectorArea(theta1, theta2, A, B)
```

```
2
3  if  (theta1 > theta2)
4      SectorArea = ((theta2 - (theta1 - (2*pi)))*A*B)/2;
5  else
6      SectorArea = ((theta2 - theta1)*A*B)/2;
7  end
```

```
1  function SegmentArea = SegmentArea(theta1, theta2, A, B, x1, x2,cx, y1, y2,cy)
2
3  TriangleArea = 0.5 * abs(x1*(y2- cy) -x2*(y1-cy) + cx*(y1-y2));
4
5  if  abs(theta1 - theta2) <= pi || abs((theta1-(2*pi)) - theta2) <= pi
6      SegmentArea = SectorArea(theta1, theta2, A, B) - TriangleArea ;
7  else
8      SegmentArea = SectorArea(theta1, theta2, A, B) + TriangleArea ;
9  end
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] K. Zbiss, A. Kacem, M. Santillo, and A. Mohammadi, "Automatic collision-free trajectory generation for collaborative robotic car-painting," *IEEE Access*, vol. 10, pp. 9950–9959, 2022.

[2] K. Zbiss, A. Kacem, M. Santillo, and A. Mohammadi, "Automatic optimal robotic base placement for collaborative industrial robotic car painting," *Applied Sciences*, vol. 14, no. 19, p. 8614, 2024.

[3] A. Kacem, K. Zbiss, and A. Mohammadi, "A numerical integrator for forward dynamics simulations of folding process for protein molecules modeled as hyper-redundant robots," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023)*, 2023.

[4] A. Kacem, K. Zbiss, and A. Mohammadi, "A numerical integrator for kinetostatic folding of protein molecules modeled as robots with hyper degrees of freedom," *Robotics*, vol. 13, no. 10, p. 150, 2024.

[5] A. Kacem, K. Zbiss, P. Watta, and A. Mohammadi, "Wave space sonification of the folding pathways of protein molecules modeled as hyper-redundant robotic mechanisms," *Multimedia Tools and Applications*, vol. 83, no. 2, pp. 4929–4949, 2024.

[6] A. Kacem, K. Zbiss, and A. Mohammadi, "Transforming motion into sound: A novel sonification approach for teams of mobile robots," in *International Symposium on Flexible Automation*, vol. 87882, p. V001T07A005, American Society of Mechanical Engineers, 2024.

[7] G. B. Hughes and M. Chraibi, "Calculating ellipse overlap areas," *Computing and Visualization in Science*, vol. 15, no. 5, pp. 291–301, 2012.

[8] F. Vicentini, "Collaborative robotics: a survey," *Journal of Mechanical Design*, vol. 143, no. 4, p. 040802, 2021.

[9] M. P. Polverini, A. M. Zanchettin, and P. Rocco, "A computationally efficient safety assessment for collaborative robotics applications," *Robotics and Computer-Integrated Manufacturing*, vol. 46, pp. 25–37, 2017.

[10] Z. Feng, G. Hu, Y. Sun, and J. Soon, "An overview of collaborative robotic manipulation in multi-robot systems," *Annual Reviews in Control*, vol. 49, pp. 113–127, 2020.

[11] J. Kim and H. I. Son, "A voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard," *IEEE Access*, vol. 8, pp. 20676–20686, 2020.

[12] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen, *et al.*, "Smerobotics: Smart robots for flexible manufacturing," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 78–90, 2019.

[13] F. Zeng, J. Xiao, and H. Liu, "Force/torque sensorless compliant control strategy for assembly tasks using a 6-dof collaborative robot," *Ieee Access*, vol. 7, pp. 108795–108805, 2019.

[14] Z. L. Cao, Y. Huang, and E. L. Hall, "Region filling operations with random obstacle avoidance for mobile robots," *Journal of Robotic systems*, vol. 5, no. 2, pp. 87–102, 1988.

[15] S. Moe, J. T. Gravdahl, and K. Y. Pettersen, "Set-based control for autonomous spray painting," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1785–1796, 2018.

[16] A. Gasparetto, R. Vidoni, D. Pillan, and E. Saccavini, "Automatic path and trajectory planning for robotic spray painting," in *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 1–6, VDE, 2012.

[17] G. Biegelbauer, A. Pichler, M. Vincze, C. L. Nielsen, H. J. Andersen, and K. Häusler, "The inverse approach of flexpaint [robotic spray painting]," *IEEE Robotics & Automation Magazine*, vol. 12, no. 3, pp. 24–34, 2005.

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

[19] S. C. Akkaladevi, M. Propst, M. Hofmann, L. Hiesmair, M. Ikeda, N. C. Chitturi, and A. Pichler, "Programming-free approaches for human–robot collaboration in assembly tasks," in *Advanced Human-Robot Collaboration in Manufacturing*, pp. 283–317, Springer, 2021.

[20] H. Chen and W. Sheng, "Transformative cad based industrial robot program generation," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 5, pp. 942–948, 2011.

[21] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path planning of industrial robot based on improved rrt algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018.

[22] S. Seriani, A. Cortellessa, S. Belfio, M. Sortino, G. Totis, and P. Gallina, "Automatic path-planning algorithm for realistic decorative robotic painting," *Automation in Construction*, vol. 56, pp. 67–75, 2015.

[23] E. Merchan-Cruz and A. Morris, "Fuzzy-ga-based trajectory planner for robot manipulators sharing a common workspace," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 613–624, 2006.

[24] M. Safeea, P. Neto, and R. Bearee, "On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case," *Robotics and Autonomous Systems*, vol. 119, pp. 278–288, 2019.

[25] H. J. Yoon, S. Y. Chung, and M. J. Hwang, "Shadow space modeling and task planning for collision-free cooperation of dual manipulators for planar task," *International Journal of Control, Automation and Systems*, vol. 17, pp. 995–1006, 2019.

[26] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[27] D.-H. Lee, "Resource-based task allocation for multi-robot systems," *Robotics and Autonomous Systems*, vol. 103, pp. 151–161, 2018.

[28] A. Montaño and R. Suárez, "An on-line coordination algorithm for multi-robot systems," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–7, IEEE, 2013.

[29] G. Dumonteil, G. Manfredi, M. Devy, A. Confetti, and D. Sidobre, "Reactive planning on a collaborative robot for industrial applications," in *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, vol. 2, pp. 450–457, IEEE, 2015.

[30] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE network*, vol. 26, no. 3, pp. 21–28, 2012.

[31] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[32] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.

[33] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE international conference on robotics and automation*, pp. 3310–3317, IEEE, 1994.

[34] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 3, pp. 1213–1219, IEEE, 2001.

[35] L. S. Marcolino and L. Chaimowicz, "Traffic control for a swarm of robots: Avoiding group conflicts," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1949–1954, IEEE, 2009.

[36] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6252–6259, IEEE, 2018.

[37] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 856–892, 2020.

[38] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, pp. 113–126, 2001.

[39] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[40] K. Karabulut and M. F. Tasgetiren, "A variable iterated greedy algorithm for the traveling salesman problem with time windows," *Information Sciences*, vol. 279, pp. 383–395, 2014.

[41] Z. Wang and Z. Bo, "Coverage path planning for mobile robot based on genetic algorithm," in *2014 IEEE Workshop on Electronics, Computer and Applications*, pp. 732–735, IEEE, 2014.

[42] Z. Chibin, W. Xingsong, and D. Yong, "Complete coverage path planning based on ant colony algorithm," in *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, pp. 357–361, IEEE, 2008.

[43] S. Thabit and A. Mohades, "Multi-robot path planning based on multi-objective particle swarm optimization," *IEEE Access*, vol. 7, pp. 2138–2147, 2018.

[44] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[45] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.

[46] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and service robotics*, pp. 203–209, Springer, 1998.

[47] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, "Exact cellular decompositions in terms of critical points of morse functions," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3, pp. 2270–2277, IEEE, 2000.

[48] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *The international journal of robotics research*, vol. 21, no. 4, pp. 331–344, 2002.

[49] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, "Contact sensor-based coverage of rectilinear environments," pp. 266–271, 1999.

[50] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings. 1985 IEEE international conference on robotics and automation*, vol. 2, pp. 116–121, IEEE, 1985.

[51] 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, no. 3, pp. 718–726, 2004.

[52] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[53] A. Zelinsky, R. A. Jarvis, J. Byrne, S. Yuta, *et al.*, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proceedings of international conference on advanced robotics*, vol. 13, pp. 533–538, 1993.

[54] V. Shivashankar, R. Jain, U. Kuter, and D. Nau, "Real-time planning for covering an initially-unknown spatial environment," in *Twenty-Fourth International FLAIRS Conference*, 2011.

[55] C. Luo, S. X. Yang, D. A. Stacey, and J. C. Jofriet, "A solution to vicinity problem of obstacles in complete coverage path planning," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1, pp. 612–617, IEEE, 2002.

[56] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 718–724, 2004.

[57] D. E. Orin and W. W. Schrader, "Efficient computation of the jacobian for robot manipulators," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 66–75, 1984.

[58] F. Merat, "Introduction to robotics: Mechanics and control," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 166–166, 1987.

[59] R. L. Ciuccio[1], V. J. da Silva, and R. A. P. Vitalli, "Comparison of modified dh notation with standard dh for and all of direct kinematics of industrial robotic manipulators," 2022.

[60] N. Zivkovic, J. Vidakovic, S. Mitrovic, and M. Lazarevic, "Implementation of dual quaternion-based robot forward kinematics algorithm in ros," in *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–4, IEEE, 2022.

[61] A. Balestrino, G. De Maria, and L. Sciavicco, "Robust control of robotic manipulators," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 2435–2440, 1984.

[62] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *The 23rd IEEE Conference on Decision and Control*, pp. 1359–1363, IEEE, 1984.

[63] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on man-machine systems*, vol. 10, no. 2, pp. 47–53, 1969.

[64] A. Liegeois *et al.*, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE transactions on systems, man, and cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[65] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *The international journal of robotics research*, vol. 4, no. 3, pp. 109–117, 1985.

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

[67] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," 1986.

[68] J. Zhao and N. I. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 4, pp. 313–336, 1994.

[69] M. A. Arteaga, A. Gutiérrez-Giles, and J. Pliego-Jiménez, *A General Overview of Robot Manipulators*, pp. 3–14. Cham: Springer International Publishing, 2022.

[70] T. Yoshikawa, "Manipulability of robotic mechanisms," *The international journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[71] Y. Zhao, "Singularity, isotropy, and velocity transmission evaluation of a three translational degrees-of-freedom parallel robot," *Robotica*, vol. 31, no. 2, pp. 193–202, 2013.

[72] J. Hubert and J.-P. Merlet, "Static of parallel manipulators and closeness to singularity," 2009.

[73] Y. Shi, H. X. Yue, Y. Lu, and L. H. Guo, "Singularity analysis of a plane-symmetry 3-rps parallel robot based on translational/rotational jacobian matrices," *Applied Mechanics and Materials*, vol. 121, pp. 1590–1594, 2012.

[74] M. B. Hong, "On the robot singularity: A novel geometric approach," *International Journal of Advanced Robotic Systems*, vol. 9, no. 5, p. 197, 2012.

[75] S. Zou, H. Liu, Y. Liu, J. Yao, and H. Wu, "Singularity analysis and representation of 6dof parallel robot using natural coordinates," *Journal of Robotics*, vol. 2021, no. 1, p. 9935794, 2021.

[76] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: an algorithmic approach," *Annals of Mathematics and Artificial Intelligence*, vol. 52, pp. 109–142, 2008.

[77] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3852–3857, IEEE, 2005.

[78] C. Luo and S. X. Yang, "A real-time cooperative sweeping strategy for multiple cleaning robots," in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*, pp. 660–665, IEEE, 2002.

[79] C. Luo, S. X. Yang, and D. A. Stacey, "Real-time path planning with deadlock avoidance of multiple cleaning robots," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3, pp. 4080–4085, IEEE, 2003.

[80] S. Yuan, B. Skinner, S. Huang, and D. Liu, "A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms," *European journal of operational research*, vol. 228, no. 1, pp. 72–82, 2013.

[81] H. Zhou, M. Song, and W. Pedrycz, "A comparative study of improved ga and pso in solving multiple traveling salesmen problem," *Applied Soft Computing*, vol. 64, pp. 564–580, 2018.

[82] M. A. Al-Omeer and Z. H. Ahmed, "Comparative study of crossover operators for the mtsp," pp. 1–6, 2019.

[83] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: A multi-objective approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2530–2537, 2020.

[84] Z. Xu, Y. Li, and X. Feng, "Constrained multi-objective task assignment for uuvs using multiple ant colonies system," in *2008 ISECS international colloquium on computing, communication, control, and management*, vol. 1, pp. 462–466, IEEE, 2008.

[85] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.

[86] S. Kucuck and Z. Bingul, "The inverse kinematics solutions of fundamental robot manipulators with offset wrist," in *IEEE International Conference on Mechatronics, 2005. ICM'05.*, pp. 197–202, IEEE, 2005.

[87] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[88] L. G. Khachiyan, "Rounding of polytopes in the real number model of computation," *Mathematics of Operations Research*, vol. 21, no. 2, pp. 307–320, 1996.

[89] Y. Cao, K. Lu, X. Li, and Y. Zang, "Accurate numerical methods for computing 2d and 3d robot workspace," *International Journal of Advanced Robotic Systems*, vol. 8, no. 6, p. 76, 2011.

[90] Z. Vafa and S. Dubowsky, "The kinematics and dynamics of space manipulators: The virtual manipulator approach," *The International Journal of Robotics Research*, vol. 9, no. 4, pp. 3–21, 1990.

[91] T. M. Chan, "Optimal output-sensitive convex hull algorithms in two and three dimensions," *Discrete & Computational Geometry*, vol. 16, no. 4, pp. 361–368, 1996.

[92] J. D. Foley, A. Van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips, *Introduction to computer graphics*, vol. 55. Addison-Wesley Reading, 1994.

[93] Y. Nesterov, "Rounding of convex sets and efficient gradient methods for linear programming problems," *Optimisation Methods and Software*, vol. 23, no. 1, pp. 109–128, 2008.

[94] N. Moshtagh, "Minimum volume enclosing ellipsoid." `https://www.mathworks.com/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid`, February 2024. MATLAB Central File Exchange.

[95] S. Biswas and R. Hazra, "Robust edge detection based on modified moore-neighbor," *Optik*, vol. 168, pp. 931–943, 2018.

[96] E. Horvath, "Coverage path planning algorithms (cpp) on grid maps," 2017. Accessed: 2021-08-27.

[97] P. Borrel and A. Liégeois, "A study of multiple manipulator inverse kinematic solutions with applications to trajectory planning and workspace determination," in *Proceedings. 1986 ieee international conference on robotics and automation*, vol. 3, pp. 1180–1185, IEEE, 1986.

[98] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Optimal TCP and robot base placement for a set of complex continuous paths," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9659–9665, IEEE, 2021.

[99] B. Balci, J. Donovan, J. Roberts, and P. Corke, "Optimal workpiece placement based on robot reach, manipulability and joint torques," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12302–12308, IEEE, 2023.

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

[101] A. Mohammed, B. Schmidt, and L. Wang, "Energy-efficient robot configuration for assembly," *Journal of Manufacturing Science and Engineering*, vol. 139, no. 5, p. 051007, 2017.

[102] S. Ghungrad and A. Haghighi, "Three-dimensional spatial energy-quality map construction for optimal robot placement in multi-robot additive manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 88, p. 102735, 2024.

[103] M. Aly, A. Abbas, and S. M. Megahed, "Robot workspace estimation and base placement optimisation techniques for the conversion of conventional work cells into autonomous flexible manufacturing systems," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 12, pp. 1133–1148, 2010.

[104] P. Sotiropoulos, N. Aspragathos, F. Andritsos, *et al.*, "Optimum docking of an unmanned underwater vehicle for high dexterity manipulation," *IAENG International Journal of Computer Science*, vol. 38, no. 1, pp. 48–56, 2011.

[105] S. Mitsi, K.-D. Bouzakis, D. Sagris, and G. Mansour, "Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 1, pp. 50–59, 2008.

[106] G.-C. Vosniakos and E. Matsas, "Improving feasibility of robotic milling through robot placement optimisation," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 5, pp. 517–525, 2010.

[107] Q. Fan, Z. Gong, B. Tao, Y. Gao, Z. Yin, and H. Ding, "Base position optimization of mobile manipulators for machining large complex components," *Robotics and Computer-Integrated Manufacturing*, vol. 70, p. 102138, 2021.

[108] N. C. N. Doan and W. Lin, "Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6r articulated robots," *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 233–242, 2017.

[109] M. Hassan, D. Liu, and G. Paul, "Collaboration of multiple autonomous industrial robots through optimal base placements," *Journal of Intelligent & Robotic Systems*, vol. 90, pp. 113–132, 2018.

[110] R. Kalawoun, S. Lengagne, and Y. Mezouar, "Optimal robot base placements for coverage tasks," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 235–240, IEEE, 2018.

[111] S. Mutti, G. Nicola, M. Beschi, N. Pedrocchi, and L. M. Tosatti, "Towards optimal task positioning in multi-robot cells, using nested meta-heuristic swarm algorithms," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102131, 2021.

[112] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, pp. 624–631, 2002.

[113] S. S. Chiddarwar and N. R. Babu, "Conflict free coordinated path planning for multiple robots using a dynamic path modification sequence," *Robotics and Autonomous Systems*, vol. 59, no. 7-8, pp. 508–518, 2011.

[114] M. Hassan, D. Liu, and G. Paul, "Modeling and stochastic optimization of complete coverage under uncertainties in multi-robot base placements," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2978–2984, IEEE, 2016.

[115] J. D. S. De Caro, M. S. H. Sunny, E. J. M. Montenegro, H. U. Ahmed, and M. H. Rahman, "Optimal base placement of a 6-dofs robot to cover essential activities of daily living," *IEEE Access*, vol. 10, pp. 134536–134548, 2022.

[116] F. Litvin, "Application of theorem of implicit function system existence for analysis and synthesis of linkages," *Mechanism and Machine Theory*, vol. 15, no. 2, pp. 115–125, 1980.

[117] D. Kohli and J. Spanos, "Workspace analysis of mechanical manipulators using polynomial discriminants," 1985.

[118] E. J. Haug, C.-M. Luh, F. A. Adkins, and J.-Y. Wang, "Numerical algorithms for mapping boundaries of manipulator workspaces," 1996.

[119] K. Abdel-Malek, F. Adkins, H.-J. Yeh, and E. Haug, "On the determination of boundaries to manipulator workspaces," *Robotics and Computer-Integrated Manufacturing*, vol. 13, no. 1, pp. 63–72, 1997.

[120] J. Yang, K. Abdel-Malek, and Y. Zhang, "On the workspace boundary determination of serial manipulators with non-unilateral constraints," *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 1, pp. 60–76, 2008.

[121] A. Kumar and K. Waldron, "The workspaces of a mechanical manipulator," 1981.

[122] F. Freudenstein and E. Primrose, "On the analysis and synthesis of the workspace of a three-link, turning-pair connected robot arm," 1984.

[123] C. Liu, G.-H. Cao, and Y.-Y. Qu, "Workspace analysis of delta robot based on forward kinematics solution," in *2019 3rd International Conference on Robotics and Automation Sciences (ICRAS)*, pp. 1–5, IEEE, 2019.

[124] H. Zhang, H. Fang, D. Zhang, X. Luo, and Q. Zou, "Forward kinematics and workspace determination of a novel redundantly actuated parallel manipulator," *International Journal of Aerospace Engineering*, vol. 2019, no. 1, p. 4769174, 2019.

[125] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association*, pp. 335–341, 1949.

[126] J. Rastegar and D. Perel, "Generation of manipulator workspace boundary geometry using the monte carlo method and interactive computer graphics," *Journal of Mechanical Design*, vol. 112, no. 3, pp. 452–454, 1990.

[127] D. G. Alciatore and C.-C. D. Ng, "Determining manipulator workspace boundaries using the monte carlo method and least squares segmentation," in *International design engineering technical conferences and computers and information in engineering conference*, vol. 12860, pp. 141–146, American Society of Mechanical Engineers, 1994.

[128] Y. Cao, S. Qi, K. Lu, Y. Zang, and G. Yang, "Shape and size computation of planar robot workspace," in *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 2, pp. 126–130, IEEE, 2009.

[129] Y. Cao, H. Zang, L. Wu, and T. Lu, "An engineering oriented method for the three dimensional workspace generation of robot manipulator," *Journal of Information and Computational Science*, vol. 8, no. 1, pp. 51–61, 2011.

[130] A. Peidró, Ó. Reinoso, A. Gil, J. M. Marín, and L. Payá, "An improved monte carlo method based on gaussian growth to calculate the workspace of robots," *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 197–207, 2017.

[131] C. W. Churchman and R. L. Ackoff, "An approximate measure of value," *Journal of the Operations Research Society of America*, vol. 2, no. 2, pp. 172–187, 1954.

[132] K. Deb, *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*, pp. 3–34. London: Springer London, 2011.

[133] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[134] E. Zitzler, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[135] F. Gembicki, "Vector optimization for control with performance and parameter sensitivity indices," *Ph. D. thesis, Case Western Reserve Univ.*, 1974.

[136] C. Barba-González, J. García-Nieto, A. J. Nebro, J. A. Cordero, J. J. Durillo, I. Navas-Delgado, and J. F. Aldana-Montes, "jmetalsp: a framework for dynamic multi-objective big data optimization," *Applied Soft Computing*, vol. 69, pp. 737–748, 2018.

[137] D. Gleeson, S. Jakobsson, R. Salman, F. Ekstedt, N. Sandgren, F. Edelvik, J. S. Carlson, and B. Lennartson, "Generating optimized trajectories for robotic spray painting," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1380–1391, 2022.

[138] F. A. Corporation, "FANUC R-2000iC/220U." https://www.fanucamerica.com/products/robots/series/r-2000/r-2000ic-220u, 2021. Accessed: 2021-08-28.

[139] ABB Company, "Robotics: Product specification: IRB 4600," tech. rep., 2020. Document ID: 3HAC032885-001, Revision: AB.

[140] O. Bohigas, M. Manubens, and L. Ros, "A complete method for workspace boundary determination on general structure manipulators," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 993–1006, 2012.

[141] Y. Guan, K. Yokoi, and X. Zhang, "Numerical methods for reachable space generation of humanoid robots," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 935–950, 2008.

[142] M. J. Todd and E. A. Yıldırım, "On khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids," *Discrete Applied Mathematics*, vol. 155, no. 13, pp. 1731–1744, 2007.

[143] N. Moshtagh, *Minimum Volume Enclosing Ellipsoid*. MATLAB Central File Exchange, 2021. url: https://www.mathworks.com/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid. Retrieved July 20, 2021.

[144] N. Rabiei and E. G. Saleeby, "On intersection volumes of confidence hyper-ellipsoids and two geometric Monte Carlo methods," *Monte Carlo Methods and Applications*, vol. 27, no. 2, pp. 153–167, 2021.

[145] P. Fu, O. R. Walton, and J. T. Harvey, "Polyarc discrete element for efficiently simulating arbitrarily shaped 2D particles," *International Journal for Numerical Methods in Engineering*, vol. 89, no. 5, pp. 599–617, 2012.

[146] K. Han, Y. Feng, and D. Owen, "Polygon-based contact resolution for superquadrics," *International Journal for Numerical Methods in Engineering*, vol. 66, no. 3, pp. 485–501, 2006.

[147] I.-S. Kim, W.-K. Lee, and Y.-D. Hong, "Simple global path planning algorithm using a ray-casting and tracking method," *Journal of Intelligent & Robotic Systems*, vol. 90, pp. 101–111, 2018.

[148] F. Gembicki and Y. Haimes, "Approach to performance and sensitivity multiobjective optimization: The goal attainment method," *IEEE Transactions on Automatic control*, vol. 20, no. 6, pp. 769–771, 1975.