Decentralized Motion Planning for Autonomous Multi-Agent Systems:

Multi-Segment Manipulators and Mobile Robot Collectives

by

Amir Salimi Lafmejani

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved June 2022 by the
Graduate Supervisory Committee:

Spring Berman, Co-Chair
Konstantinos Tsakalis, Co-Chair
Antonia Papandreou-Suppappola
Hamid Marvi

ARIZONA STATE UNIVERSITY

August 2022

ABSTRACT

Multi-segment manipulators and mobile robot collectives are examples of multi-agent robotic systems, in which each segment or robot can be considered an agent. Fundamental motion control problems for such systems include the stabilization of one or more agents to target configurations or trajectories while preventing inter-agent collisions, agent collisions with obstacles, and deadlocks. Despite extensive research on these control problems, there are still challenges in designing controllers that (1) are scalable with the number of agents; (2) have theoretical guarantees on collision-free agent navigation; and (3) can be used when the states of the agents and the environment are only partially observable. Existing centralized and distributed control architectures have limited scalability due to their computational complexity and communication requirements, while decentralized control architectures are often effective only under impractical assumptions that do not hold in real-world implementations.

The main objective of this dissertation is to develop and evaluate decentralized approaches for multi-agent motion control that enable agents to use their onboard sensors and computational resources to decide how to move through their environment, with limited or absent inter-agent communication and external supervision. Specifically, control approaches are designed for multi-segment manipulators and mobile robot collectives to achieve position and pose (position and orientation) stabilization, trajectory tracking, and collision and deadlock avoidance. These control approaches are validated in both simulations and physical experiments to show that they can be implemented in real-time while remaining computationally tractable.

First, kinematic controllers are proposed for position stabilization and trajectory tracking control of two- or three-dimensional hyper-redundant multi-segment manipulators. Next, robust and gradient-based feedback controllers are presented for individual holonomic and nonholonomic mobile robots that achieve position stabilization, trajectory track-

ing control, and obstacle avoidance. Then, nonlinear Model Predictive Control methods are developed for collision-free, deadlock-free pose stabilization and trajectory tracking control of multiple nonholonomic mobile robots in known and unknown environments with obstacles, both static and dynamic. Finally, a feedforward proportional-derivative controller is defined for collision-free velocity tracking of a moving ground target by multiple unmanned aerial vehicles.

DEDICATION

To My Dear Family!

ACKNOWLEDGMENTS

I feel blessed to be able to follow my dreams and I would like to thank God for always being there for me. I am grateful for receiving a lot of support from my advisor, colleagues, family, and friends. First and foremost, I am thankful to my advisor, Professor Spring M. Berman, who provided a peaceful environment for doing high-quality research. She was not only an academic advisor but also an understanding friend. None of my achievements during graduate school, including this dissertation, would have been possible without her help. I am also grateful for receiving support and helpful advice from Prof. Georgios Fainekos. I also would like to acknowledge the supporting grants from the Office of Naval Research (ONR) Award N00014-17-1-2117, Arizona State University Global Security Initiative (GSI), DARPA AMP N6600120C4020, NSF CNS 1932068, NSF IIP-1361926, and the NSF I/UCRC Center for Embedded Systems. I would like to express my appreciation to my committee members, Prof. Konstantinos Tsakalis, Prof. Antonia Papandreou-Suppappola and Prof. Hamid Marvi for their valuable guidance and comments. I would like to acknowledge my colleagues, specifically, Hamed Farivarnejad and Azadeh Doroudchi, from whom I received assistance during my research. In addition, I would like to thank my mother and my siblings for all their sacrifices and supports. You have been always there for me and I love you so much. I dedicate this dissertation and all of my endeavor during my academic career to my beloved father who was inspirational for many people and dedicated his life to support his family and to carry over discipline and hardworking.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1 INTRODUCTION 1

1.1 Problem Definition 1

1.2 Background 3

1.2.1 Multi-Segment Manipulators 4

1.2.2 Mobile Robot Collectives 8

1.2.3 Motivation 19

1.3 Research Contributions and Organization of Dissertation 20

2 POSITION STABILIZATION AND TRAJECTORY TRACKING CONTROL OF MULTI-SEGMENT MANIPULATORS 23

2.1 Position Stabilization and Trajectory Tracking Control of 2D Multi-Segment Manipulators 23

2.1.1 Rigid-Link Model of Soft Continuum Robot 24

2.1.2 Controller Design 25

2.1.3 Stability and Convergence Analysis 29

2.1.4 Simulation Results 35

2.1.5 Discussion 40

2.2 Trajectory Tracking Control of 3D Multi-Segment Manipulators 41

2.2.1 Kinematic Model of Hyper-Redundant Robot 42

2.2.2 Kinematic Control of Robot Motion 48

2.2.3 Simulation Results 52

2.2.4 Discussion 59

CHAPTER Page

3 COLLISION-FREE POSE STABILIZATION AND TRAJECTORY TRACKING CONTROL OF AUTONOMOUS MOBILE ROBOTS 60
3.1 Pose Stabilization and Trajectory Tracking Control of a Holonomic Robot 60
3.1.1 Formulation and Problem Setup 61
3.1.2 Control Approaches 64
3.1.3 Simulation Results 68
3.1.4 Experimental Implementation and Results 72
3.1.5 Discussion 79
3.2 Position Stabilization of a Nonholonomic Robot with Static Obstacle Avoidance 80
3.2.1 Problem Formulations 80
3.2.2 Simulation and Experimental Results 91
3.2.3 Discussion 96
3.3 Pose Stabilization of Multiple Nonholonomic Robots with Collision and Deadlock Avoidance 98
3.3.1 Problem Formulation 99
3.3.2 Control Approaches 101
3.3.3 MPC Design for Multiple Nonholonomic Robots 104
3.3.4 Simulation and Experimental Results 113
3.3.5 Analysis and Deadlock 118
3.3.6 Discussion 130
3.4 Trajectory Tracking Control of Multiple Nonholonomic Robots with Collision and Deadlock Avoidance 131
3.4.1 Problem Formulation 131

CHAPTER Page

3.4.2 Safe NMPC Control Design and Analysis 135
3.4.3 Simulation Results 142
3.4.4 Discussion 143
3.5 Pose Stabilization of Multiple Nonholonomic Robots with Static and Dynamic Obstacle Avoidance 144
3.5.1 Control Problem Formulations 145
3.5.2 Decentralized NMPC-LBF Method 148
3.5.3 Simulation and Experimental Results 155
3.5.4 Analysis of Proposed Control Approach 157
3.5.5 Discussion 162
3.6 Velocity Tracking of a Ground Target by Multiple Unmanned Aerial Vehicles with Collision Avoidance 162
3.6.1 Problem Formulation 163
3.6.2 Control Approaches 164
3.6.3 Simulation and Experimental Results 167
3.6.4 Discussion 175
4 CONCLUSION AND FUTURE WORK 178
REFERENCES 182

LIST OF TABLES

Table Page

2.1 Calculation Time of the Proposed Controller in Milliseconds (ms) for the Simulated Multi-Segment Robot with Different Numbers of Segments. 58

3.1 Sampling Time (s) and Prediction Horizon (Time Steps) for Each Scenario in the Simulations and Experiments. 116

LIST OF FIGURES

Figure Page

2.1 Schematic of a Continuum Robot Composed of Soft Segments, Along with its Equivalent Model as a Serial configuration of multiple rigid-link RPR mechanisms. Local coordinate frames (Frenet-Serret frames (Webster III and Jones, 2010)) are defined at the two ends of each soft segment. 24

2.2 Schematic of Information Propagation Between Segments of the Continuum Robot. (a) Each Segment $i \in \{1, ..., N\}$ Communicates with its Adjacent Segment(s) in Order to Share its Measurements of the Position Vector ${}^{i}\vec{p}_i$ and Rotation Matrix ${}^{i-1}\vec{R}_i$, which are Represented in its Local Coordinate Frame. (b) Graph of the Robot's Communication Network, where Each Blue Node Represents a Segment and Each Red Edge Represents a Bidirectional Communication Channel. 26

2.3 Illustration of an Equivalent N-RPR Rigid-Link Model for the Soft Continuum Robot in Fig. 2.2. The x-Axis of Each Local Coordinate Frame is Aligned with the Corresponding Prismatic Joint. 27

2.4 Length of Prismatic Joints Over Time for a Simulated 5-Link Serial Robot, Equivalent to a 5-Segment Soft Continuum Robot, that is Controlled by the Position Regulator (2.3) with Gain $K = 0.0382$. 36

2.5 Length of Prismatic Joints Over Time for a Simulated 5-Link Serial Robot, Equivalent to a 5-Segment Soft Continuum Robot, that is Controlled by the Position Regulator (2.3) with Gain $K = 0.764$. 36

2.6 Plots of Tracking Error $||\vec{E}(t)||_2$ Over Time During the Position Regulator Simulations. The Gain K = 0.764 Results in Faster Convergence of the Robot's Tip to the Target Point than the Gain K = 0.0382. 37

Figure Page

2.7 Configuration Over Time of the Simulated 5-Link Robot when Controlled by the Position Regulator (2.3) with Gain $K = 0.0382$. The Robot Reconfigures from its Initial Configuration to Approach the Target Point with its Tip. Intermediate Configurations are Shown in Gray. 38

2.8 Configuration Over Time of the Simulated 5-Link Robot when Controlled by the Position Regulator (2.3) with Gain $K = 0.764$. The Robot Reconfigures from its Initial Configuration to Approach the Target Point with its Tip. Intermediate Configurations are Shown in Gray. 39

2.9 Configuration Over Time of a Simulated 15-Link Robot, Equivalent to a 15-Segment Soft Continuum Robot, when Controlled by the Trajectory Tracking Controller (2.8) with Gain $K = 0.25$. The Robot Reconfigures such that its Tip Moves Between a Series of Points Defined Along a Trajectory that Spells ASU. Intermediate Configurations and Their Corresponding Times Are Shown in Gray. The Average RMSE of Tracking is 0.024 cm. 40

2.10 Model of a Single Segment of the Hyper-Redundant Robot Based on a UPS 6-DoF Gough-Stewart (GS) Platform. 43

2.11 Multi-Segment Model of an Octopus-Inspired Hyper-Redundant Robot Composed of a Series of 6-DoF GS Platforms. 44

2.12 (a) Snapshots of the Simulated Hyper-Redundant Robot Tracking the Linear Trajectory Eq. (2.63). Average Tracking RMSE = 0.032 cm. (b) Time Evolution of the Desired Position (x,y,z) and Tracked Position $(x_{\text{tip}},y_{\text{tip}},z_{\text{tip}})$ of the Robot's Tip. 53

Figure Page

2.13 (a) Snapshots of the Simulated Hyper-Redundant Robot Tracking the Elliptical Trajectory (2.64). Average Tracking RMSE = 0.044 cm. (b) Time Evolution of the Desired Position (x,y,z) and Tracked Position (x_{tip},y_{tip},z_{tip}) of the Robot's Tip. 54

2.14 (a) Snapshots of the Simulated Robot Tracking the Sinusoidal Trajectory Eq. (2.65). Average Tracking RMSE = 0.049 cm. (b) Time Evolution of the Desired Position (x,y,z) and Tracked Position (x_{tip},y_{tip},z_{tip}) of the Robot's Tip. 55

2.15 (a) Snapshots of the Robot Tracking the Reaching and Fetching Reference Trajectories, Eq. (2.66) and Eq. (2.67). Average Tracking RMSEs for Reaching and Fetching Are 0.051 cm and 0.057 cm, Respectively. (b), (c) Time Evolution of the Desired and Tracked Positions of the Robot's Tip. 56

3.1 Schematic of a Three-Wheeled Omnidirectional WMR, Illustrating the Driving and Free-Sliding Directions of the Robot's Wheels. 63

3.2 Block Diagram of the Tracking Controller, Including Disturbances and Sensor Noise as Input Signals. 65

3.3 Trajectory of a Simulated Three-Wheeled Omnidirectional Robot that Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller. 67

3.4 Norm of the Pose and Twist Errors Over 50 s of the Simulations in which the Robot Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller. 68

Figure Page

3.5 Trajectory of a Simulated Three-Wheeled Omnidirectional Robot that Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller. 70

3.6 Norm of the Pose and Twist Errors Over 50 s of the Simulations in which the Robot Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller. 71

3.7 (a) Isometric and (b) Overhead Views of the Three-Wheeled Omnidirectional WMR Used in the Experiments. The Global Coordinate Frame is Defined as in Fig. 3.1. 73

3.8 Snapshots of the Trajectory of a Three-wheeled Omnidirectional Robot During an Experiment in which it Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller. 75

3.9 Norm of the Pose and Twist Errors Over 25 s of an Experiment in which the Robot Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller. 76

3.10 Snapshots of the Robot's Trajectory During an Experiment in which it Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) the Full-State Feedback H_∞-Optimal Controller... 77

Figure Page

3.11 Norm of the Pose and Twist Errors Over 10 s of an Experiment in which the Robot Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller. 78

3.12 (a) 3-D View of the Turtlebot3 Burger Robot (Robotis, 2020), a Nonholonomic WMR that We Used in Our Experiments, and (b) Definitions of the State Variables ρ, α, and θ in an Overhead View of the Robot. 81

3.13 The Phase Portrait of the Closed-Loop System when the (a) Negative and (b) Positive Controller Gains are Used. 84

3.14 Stabilization of a Simulated Nonholonomic WMR to the Origin from Different Initial Configurations, Using Negative Controller Gains $k_v = -0.1$ and $k_\omega = -0.5$ (1 to 6), and Positive Controller Gains $k_v = 0.1$ and $k_\omega = 0.5$ (7 to 12). The Robot's Heading at Each Initial Configuration is Indicated by a Green Arrow. 91

3.15 Stabilization of a simulated nonholonomic WMR to the origin from different initial configurations, while avoiding collisions with obstacles, using our proposed controller with the potential field φ defined as **(a)** a navigation function (Koditschek, 1987); **(b)** an attractive-repulsive potential field (Khatib, 1985). 92

3.16 Experimental Validation of Our Position Controller with **(a)** Negative Controller Gains, Producing Backward Motion, and **(b)** Positive Controller Gains, Producing Forward Motion. The Robot's Heading at its Initial Configuration is Indicated by a Green Arrow, and its Trajectory is Plotted as a Yellow Dashed Line. 93

3.17 Time Evolution of ρ, the Robot's Distance to the Target Position, During the Experimental Tests of Position Stabilization in Fig. 3.16. 94

3.18 Experimental validation of our obstacle avoidance controller, with the potential field φ defined as **(a)** a navigation function (Koditschek, 1987); **(b)** an attractive-repulsive potential field (Khatib, 1985). The robot's initial heading is indicated by a green arrow, and its trajectory is plotted as a yellow dashed line. 97

3.19 Time Evolution of φ and $||\nabla\varphi||$, where $\varphi = \psi$ or $\varphi = U$, During the Experimental Tests of Obstacle Avoidance in Fig. 3.18. 98

3.20 (a) 3-D View (Robotis, 2020) of the Turtlebot3 Burger Robot, and (b) Overhead View with Body-Fixed Coordinate Frame. 99

3.21 Illustration of the Execution of an MPC Method Over two Consecutive Time Steps, k (*left*) and $k+1$ (*right*). The Robot States $\mathbf{x}(k)$ (*Upper Plots*) and Control Inputs $\mathbf{u}(k)$ (*Lower Plots*) are Computed Over the Prediction Horizon N_P and the Control Horizon N_C, Respectively. 101

3.22 Examples of Potential Collisions, Indicated by Lightning Bolts, Between Pairs of Robots. 106

3.23 Framework for Implementing the Proposed MPC Method for Collision-Free Navigation by Multiple WMRs. 113

3.24 Snapshots of the Gazebo Simulation of *Scenario 1*. The robots' Goal Poses Are: $\mathbf{x}_{g_1} = [1 \;\; -1 \;\; -0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [-1 \;\; -1 \;\; -2.356]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [1 \;\; 1 \;\; 0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [-1 \;\; 1 \;\; 2.356]^{\mathrm{T}}$. 117

3.25 Snapshots of the Experimental Run of *Scenario 1.* The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [0.71\ -0.71\ -0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0.71\ 0.71\ -2.356]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [-0.71\ -0.71\ 0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.71\ -0.71\ 2.356]^{\mathrm{T}}$. 118

3.26 Snapshots of the Gazebo Simulation of *Scenario 2.* The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [1\ \ 0.5\ \ 0]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0\ \ 0.75\ \ -1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [-0.5\ \ -0.5\ \ 3.14]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [-0.5\ -0.75\ \ 0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0.75\ -0.75\ -0.785]^{\mathrm{T}}$. 119

3.27 Snapshots of the Experimental Run of *Scenario 2.* The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [0.56\ -0.71\ 0]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0.84\ -0.36\ 0]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [1.18\ 0\ 0]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.84\ \ 0.36\ \ 0]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0.56\ \ 0.71\ \ 0]^{\mathrm{T}}$. 120

3.28 Snapshots of the Gazebo Simulation of *Scenario 3.* The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [-0.866\ -0.5\ -2.618]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0\ -1\ -1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [0.866\ -0.5\ -0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.866\ 0.5\ 0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0\ 1\ 1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_6} = [-0.866\ \ 0.5\ \ 2.618]^{\mathrm{T}}$. 121

3.29 Snapshots of the Experimental Eun of *Scenario 3.* The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [-0.7\ -0.4\ -2.618]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0\ -0.8\ -1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [0.7\ -0.4\ -0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.7\ 0.4\ 0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0\ 0.8\ 1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_6} = [-0.7\ \ 0.4\ \ 2.618]^{\mathrm{T}}$. 122

3.30 Optimal Robot Linear Velocities Over Time in the Simulation of *Scenario 1* 122

3.31 Optimal Robot Linear Velocities Over Time During the Experimental Run of *Scenario 1.* 123

3.32 Optimal Robot Angular Velocities Over Time in the Simulation of *Scenario 1.* 123

3.33 Optimal Robot Angular Velocities Over Time During the Experimental Run of *Scenario 1.* 124

3.34 Optimal Robot Linear Velocities Over Time in the Simulation of *Scenario 2.* 124

Figure Page

3.35 Optimal Robot Linear Velocities Over Time During the Experimental Run of *Scenario 2.* 125
3.36 Optimal Robot Angular Velocities Over Time in the Simulation of *Scenario 2.* 125
3.37 Optimal Robot Angular Velocities Over Time During the Experimental Run of *Scenario 2.* 126
3.38 Optimal Robot Linear Velocities Over Time in the Simulation of *Scenario 3.* 126
3.39 Caption for LOF 127
3.40 Optimal Robot Angular Velocities Over Time in the Simulation of *Scenario 3.* 127
3.41 Optimal Robot Angular Velocities Over Time During the Experimental Run of *Scenario 3.* 128
3.42 Three Types of Deadlocks that Can Occur in Our Multi-Robot Navigation Scenarios. Illustration is Based on Fig. 8 in (Wang *et al.*, 2017). 129
3.43 (a) Isometric View (Robotis, 2020) of the Turtlebot3 Burger Robot, and (b) Overhead View with Body-Fixed and Global Coordinate Frames. 132
3.44 Illustration of Three Robots Following their Desired Trajectories, with Potential Collisions Between Two Robots (Collisions of R $\#1$ and R $\#3$, R $\#2$ and R $\#3$) and Between a Robot and an Obstacle (Collisions of R $\#1$ and R $\#2$ with Obstacles). 136
3.45 Overhead View of the Initial Configurations and Desired Trajectories of the Robots in an *Intersection Cross* Scenario with Obstacles (gray circles). 143
3.46 Overhead View of the Initial Configurations and Desired Trajectories of the Robots in a *Lane Merging* Scenario with Obstacles (Gray Circles). 144
3.47 Overhead View of the Initial Configurations and Desired Trajectories of the Robots in a *Lane Changing* Scenario with Obstacles (Gray Circles). 144

Figure

Page

3.48 LiDAR Sensor Rays and Point Sampling in Safe and Unsafe Regions within the Sensing Range of the LiDAR. The Unsafe Samples are Taken from the Red Shaded Regions and the Safe Samples are in Areas Excluding the Unsafe Regions. $d^s_{r,s}$, $d_{r,\text{lidar}}$, $d^u_{r,s}$ Show the Distances of Robot to the Safe, Boundary, and Unsafe Samples on a Ray of LiDAR. 148

3.49 An Illustration of Ground-Truth BF, $h(\bar{\mathbf{x}})$, and Approximation of BF by the DeepNN, $\hat{h}(\bar{\mathbf{x}})$, on a Single Ray of LiDAR Along with a 2D Visualization of Safe and Unsafe Samples and a Sample on the Boundary of Obstacle. We Note that the Sample Points are Defined in $x - y$ Plane with the Height of Zero. 149

3.50 Block Diagram of the NMPC-LBF Method Described in Eq. (3.105). \Scan and \Odom are ROS Topics Through Which the States of the Robot and the LiDAR Measurements Would be Available. 154

3.51 A Snapshot of the Simulation of Scenario 1 in Gazebo. The Robot Should Stabilize to the Goal Pose at the Origin of the Global Frame. 157

3.52 A Snapshot for Simulated Scenario 2 in Gazebo. All Robots Should Stabilize to Their Goal Poses while Avoiding Collisions with Other Robots and Static Obstacles. 158

3.53 Plots of Distance to Goal and Optimal Control Inputs of the Robot in Scenario 1. 158

3.54 The Distance of Robots to Their Goal Poses in Scenario 2 Over Time. 159

3.55 Schematic of Our Experimental Setup for the Target Tracking Problem, in which Four Crazyflie Quadrotors Track a Turtlebot3 Burger Robot Moving on the Ground. 164

Figure Page

3.56 Robot Trajectories During a MATLAB Simulation of *Scenario 1*. 168

3.57 Robot Trajectories During a MATLAB Simulation of *Scenario 2*. The Position at which One UAV Fails is Indicated by a Small Black Circle. 168

3.58 Scenario 1: One UAV Tracking the Ground Mobile Target. Total Duration: 20 seconds. 170

3.59 Robot Trajectories During a Gazebo Simulation of *Scenario 1*. 171

3.60 Robot Trajectories During a Gazebo Simulation of *Scenario 2*. The Position at which One UAV Fails is Indicated by a Small Black Circle. 172

3.61 Scenario 1: One UAV Tracking the Ground Mobile Target. Total Duration: 20 Seconds. 173

3.62 Robot Trajectories During an Experimental Trial (Duration: 30 s) of *Scenario 1*. 174

3.63 Robot trajectories during an experimental trial (duration: 40 s) of *Scenario 2*. The position at which one UAV fails is indicated by a small black circle. 175

Chapter 1

INTRODUCTION

1.1 Problem Definition

The research in this dissertation is motivated by the need for scalable approaches to motion control of multi-agent robotic systems with potentially large numbers of agents. In particular, we consider hyper-redundant multi-segment manipulators, in which each segment can be considered an agent, and mobile robot collectives, in which each agent is a mobile robot. A multi-segment manipulator is composed of multiple links, which may be rigid or soft, that are connected in a serial kinematic chain. Multi-segment manipulators have been developed for a variety of uses, including medical procedures (e.g., steerable needles), remote inspection, search-and-rescue, and space applications (Walker *et al.*, 2016). Mobile robots are robotic systems that are capable of moving in their environment, such as Unmanned Aerial Vehicles (UAVs) (Mahony *et al.*, 2012), Autonomous Underwater Vehicles (AUVs) (Khatib *et al.*, 2016), legged mobile robots (Radford *et al.*, 2015), and Wheeled Mobile Robots (WMRs) (Lynch and Park, 2017). Motion control of mobile robots is required in a wide variety of applications in robotics such as autonomous vehicle navigation, automated warehouse operations, and package delivery.

An appropriate control architecture for a given application of a multi-agent system is determined by the number of agents and their capabilities and the available computational resources. In *centralized* control approaches for multi-agent systems, a central computational unit designs and transmits control commands for all the robots. One way to reduce the computational complexity of the control problem for multi-agent systems is to employ a *distributed* control approach. In these approaches, the control problem is decomposed

into sub-problems, which are assigned to each agent to solve using its on-board computational resources, and neighboring agents communicate their solutions to each other to accomplish their assigned tasks. To eliminate the requirement for a central computational unit or inter-agent communication, *decentralized* control approaches have been developed for multi-agent systems, in which each agent computes its own control inputs using only local measurements, without communicating with other agents. These types of approaches can be scaled to large numbers of agents that lack communication.

The specific multi-agent control problems that we address in this dissertation are position and pose stabilization, trajectory tracking, and collision and deadlock avoidance, all of which are fundamental to instantiating motion plans for multi-agent systems. Below, we define each of these control problems for both multi-segment manipulators and mobile robot collectives.

Position and Pose Stabilization:

The problem of position stabilization for a multi-segment manipulator is to design control inputs for the segments (agents) that drive the tip of the manipulator to a goal position via mechanical interactions and communication between the segments. The problem of position stabilization for a group of mobile robots is to design control inputs for each robot that drive it from an initial position to a goal position. Position stabilization is a special case of pose stabilization, in which the control inputs are designed to drive a manipulator tip to a goal position and orientation or a group of mobile robots to goal positions and orientations.

Trajectory Tracking:

The problem of trajectory tracking control for a single mobile robot is to design control inputs that cause the robot to converge to a time-parameterized reference trajectory, which in general consists of desired poses (positions and orientations) and twists (linear and angular

velocities) that vary with time. This problem can also be defined for multiple mobile robots. Examples of trajectory tracking controllers for such multi-agent systems include the problems of controlling the robots to follow reference trajectories of time-varying poses or to converge to a time-varying reference velocity, e.g., that of a moving target. The trajectory tracking problem for a multi-segment manipulator that we consider in this dissertation is to control the segments such that the tip of the manipulator follows a reference trajectory of time-varying positions.

Collision and Deadlock Avoidance:

In position and pose stabilization and trajectory tracking control problems for multi-agent systems, it is often important to design the agent control inputs to prevent collisions with one another and obstacles in the environment, in addition to achieving the stabilization and tracking control objectives. The obstacles may be static or dynamic, and information about their locations, geometry, and motion may be known or unknown *a priori*. Deadlock is another challenging problem that occurs when an agent either stops moving or exhibits oscillatory behavior during an attempt to complete its task while avoiding collisions, as a consequence of becoming trapped in an equilibrium point of the system dynamics. Deadlocks inevitably occur in multi-agent systems that use decentralized control approaches, since the agents can only respond to local information within their sensing or communication range. In this dissertation, we address collision avoidance and deadlock avoidance problems specifically in the context of navigation by multiple mobile robots.

1.2 Background

As described, this dissertation addresses various control problems for multi-agent robotic systems, including position and pose stabilization, trajectory tracking, and collision and deadlock avoidance. In this section, we provide an overview of existing approaches to

these problems for multi-segment manipulators and mobile robot collectives and their limitations.

1.2.1 Multi-Segment Manipulators

The high-dimensional configuration space of a multi-segment robot enables it to navigate and perform dexterous manipulations in unstructured environments. However, this high degree of reconfigurability poses challenges for the modeling and control of such robots. Dynamic control of a hyper-redundant manipulator requires an accurate dynamic model of the robot, demanding the use of complicated models from continuum mechanics. Therefore, kinematic control methods, which are based on simpler kinematic models of hyper-redundant manipulators, are often used to control these manipulators.

Kinematic Modeling of Multi-Segment Manipulators

Kinematic models of hyper-redundant robots differ significantly from models of conventional rigid-link robots due to their intrinsic continuous, complex, and highly compliant deformations. Whereas rigid-link robots can change their configuration only at a finite set of locations along their structure, hyper-redundant robots can change their configuration at any location; this property must be reflected in the kinematic modeling procedure (Walker, 2013).

Due to their versatile motions, the continuously deformable arms of the octopus provide a source of inspiration for the design of extensible hyper-redundant robotic manipulators (Grissom *et al.*, 2006; Walker, 2013). Existing kinematic modeling approaches for octopus-inspired hyper-redundant manipulators fall into three main categories (Burgner-Kahrs *et al.*, 2015): (1) Constant Curvature (CC), (2) Non-Constant Curvature (NCC) (also called Variable-Curvature (VC)) approaches, and (3) discrete approaches. In the first method, the robot is represented as a continuous curve in 2D or 3D space that overlaps

the arcs of a set of circles. This method is based on the CC assumption, which permits the use of integral representations and direct continuum methods in the kinematic modeling (Webster III and Jones, 2010). Therefore, the transformation matrices along the hyper-redundant robot backbone can be obtained from conventional techniques such as the Denavit-Hartenberg (DH) method, homogeneous transformations, exponential coordinates (screw theory), Frenet-Serret frames, and integral representations (Webster III and Jones, 2010). The main disadvantage of this method for modeling a 3D multi-segment manipulator is that it does not model the torsion that is contributed by each of the robot's segments.

The second approach compensates for limitations of the first approach; specifically, it lacks representation singularities and does not require the simplifying CC assumption. The NCC property of octopus-inspired hyper-redundant robots is more suited to the use of modal approaches (Godage *et al.*, 2011) for kinematic modeling of hyper-redundant mechanisms. In these approaches, a sliding frame is defined at each point along the backbone of the hyper-redundant robot, and the position and orientation of the frame are represented as functions of curvature vectors. Although soft continuum robots generally can deform into shapes with variable curvature, many existing soft robots are designed to conform to the CC assumption (Xu *et al.*, 2008; Neppalli and Jones, 2007), which enables the derivation of closed-form kinematics and Jacobian formulations (Webster III and Jones, 2010).

In the third approach, which was introduced in (Walker, 2013; Burgner-Kahrs *et al.*, 2015), curved sections of the hyper-redundant robot are approximated as rigid links, enabling the use of existing kinematic models of conventional rigid robots (Kang *et al.*, 2012). The robot is divided into multiple segments that each deform in the shape of a circular arc (the CC assumption). Each soft curved segment is modeled as an equivalent rigid-link RPR mechanism, which has two pairs of revolute joints located on its base and end-effector. Each pair is connected by a prismatic joint.

Given the complexity of modeling torsional and shearing movements and limitations on the mechanical design of hyper-redundant robots, most prior kinematic models of octopus-inspired hyper-redundant robots do not reproduce the controlled twisting and shearing motions of octopus arms (Kang *et al.*, 2012; Zheng *et al.*, 2012). A model of a multi-segment hyper-redundant robot based on the anatomy of octopus arm musculature is proposed in (Zheng *et al.*, 2011), although it does not include the effects of oblique muscles that produce torsion in the arm. A kinematic model of a hyper-redundant robot backbone that reproduces torsional motions is presented in (Walker, 2014), which demonstrates that uncontrolled torsion due to gravity and other external forces can produce large tip-tracking errors, even for relatively small values of torsion. Moreover, the kinematic models in (Godage *et al.*, 2011, 2012) do not include the constraint that the volume of an octopus arm always remains constant, called the *isovolumetric* property (Kang *et al.*, 2012; Kier, 2016).

Kinematic Control of Multi-Segment Manipulators

Various approaches have been developed for kinematic control of multi-segment manipulators. One widely studied approach is the *fitting algorithm* (Hatton and Choset, 2010; Andersson, 2006), in which the configuration of a hyper-redundant robot is fit onto a continuously-curved backbone. This method is sufficient for robots with only universal joints, but it is computationally intensive. In (Nho Cho *et al.*, 2015), a modular Jacobian-based control scheme is proposed to drive a hyper-redundant robot to a target backbone configuration, aiming at reducing the computational load of the fitting algorithm by dividing the robot into multiple modules. However, this work assumed that the desired backbone curve configuration is given, which addresses the kinematic control of the robot in a centralized fashion. A modal-based approach is used in (Yu *et al.*, 2018) for a path planning algorithm for a multi-segment hyper-redundant robot. Furthermore, learning-based methods

have been proposed to tackle this control problem. For example, an online reinforcement learning algorithm is used to control an octopus-like hyper-redundant robot in (Engel *et al.*, 2006). This approach is implemented in simulation for a 2D model of the robot and would not be efficient in real-time applications due to its computational complexity.

A more computationally efficient approach for kinematic control of hyper-redundant robots is the widely-used pseudo-inverse Jacobian method (Buss, 2004; Hannan and Walker, 2003). This method utilizes the Moore-Penrose pseudoinverse of the Jacobian matrix of a hyper-redundant robot to compute the joint velocities with minimum magnitudes that achieve the control objective. Thus, this approach can be employed to generate local solutions to the differential kinematics for each segment of a hyper-redundant robot. Moreover, this method is compatible with conventional methods for controlling hyper-redundant rigid robots (Soylu *et al.*, 2007). The main drawback of this centralized method is that the hyper-redundancy of the robot results in a large rectangular Jacobian matrix whose pseudo-inverse is time-consuming to compute, preventing the method from being efficient in practice.

Decentralized control methods can be used to avoid the aforementioned limitations of centralized control approaches for kinematic control (Ishimura *et al.*, 2002), cooperative control (Liu and Arimoto, 1998), and fault-tolerant control (Kimura *et al.*, 1995) of multi-segment continuum robots. Moreover, decentralized control approaches can significantly reduce the computational complexity of the control strategy. Recently, decentralized approaches have been employed for the control of multi-segment continuum robots with soft segments. In (Kano *et al.*, 2017), a decentralized control mechanism with local feedback was developed for a multi-segment millipede-like robot. In (McEvoy and Correll, 2018), a soft multi-segment shape-changing robot was developed with integrated sensing, actuation, and process modules, and a distributed controller was designed for shape control of the robot. A reinforcement learning-based approach was proposed in (Sartoretti *et al.*, 2019) to control serpenoid locomotion of a snake-like continuum robot. Furthermore, a de-

centralized method for shape control of a rigid-link continuum manipulator was proposed in (Mochiyama *et al.*, 1996) and verified in simulations and experiments. In (Ishimura *et al.*, 2002), the authors were inspired by the analogy between the heat and wave equations to propose a decentralized control approach for hyper-redundant robots. The work (Vittor and Willgoss, 2005) proposed a modular decentralized control approach for a general N-segment single-DoF continuum robot that exploited its stable configurations. In (Doroudchi *et al.*, 2018), a decentralized control approach was presented for a 1D soft robot arm composed of segments with local sensing, actuation, and control, and was validated in simulation. Novel morphological observation and decentralized control approaches were presented in (Sadati *et al.*, 2018) for passive shape adaptation, geometrical disturbance rejection, and task space anisotropic stiffness regulation of a 3D-printable thermoactive helical interface on a continuum manipulator.

1.2.2 Mobile Robot Collectives

Position and Pose Stabilization and Trajectory Tracking Control of Mobile Robots

We focus here on motion control problems for Wheeled Mobile Robots (WMRs), which may be designed with different types of wheeled drive systems such as two-wheeled differential-drive, car-like, and omnidirectional. The kinematics of two-wheeled and car-like robots are described by the unicycle model (Lynch and Park, 2017). In this model, there exists a constraint along the direction of wheels' axis, which is called a nonholonomic constraint and does not allow the robot to produce controlled motions along this direction. Due to this nonholonomic constraint, WMRs whose model conforms with the unicycle model are not linearly controllable and cannot be stabilized to the origin by a continuous smooth time-invariant feedback control law according to the well-known Brockett's condition (Brockett *et al.*, 1983). Given the complexity of designing controllers for nonholonomic WMRs,

there has been considerable attention to the use of holonomic WMRs such as three- or four-wheeled omnidirectional mobile robots. The design of omnidirectional robots' wheels (omni and Mecanum wheels) provide these type of WMRs with the capability of moving from any arbitrary configuration to any other arbitrary configuration (Bräunl, 2008). This in turn enables the design of linear time-invariant control laws for omnidirectional WMRs.

A linear time-invariant state-feedback control law could be a promising control strategy for an omnidirectional WMR only when an exact dynamical model of the robot is available and no disturbances or sensor noise are present. However, disturbances and noise, known as undesired exogenous inputs, can significantly affect the controlled performance of the robot in practice. These exogenous inputs can originate from different sources, such as the friction force between the robot's wheels and the ground, actuator inaccuracy, and noise in sensor measurements. Therefore, a proper control design for real-world applications should be robust enough to reject or at least attenuate the undesired effects of exogenous inputs. Optimal control approaches, such as H_2 and H_∞ controllers, employ properties of Linear Matrix Inequalities (LMIs) in order to produce control laws with these characteristics. Moreover, these optimal controllers guarantee asymptotic stability of the closed-loop system (Duan and Yu, 2013), with a robustness margin that depends on the type and magnitude of parametric uncertainties. By minimizing the H_2-norm and/or H_∞-norm of the closed-loop system, these controllers reduce and potentially minimize the effects of disturbances and noise on the robot's performance (Boyd *et al.*, 1994; Caverly and Forbes, 2019). Several control approaches that employ the properties of LMIs have been developed for the control of WMRs, e.g. (Araújo *et al.*, 2011; Rigatos and Siano, 2014).

In order to achieve a desired closed-loop behavior, an exact dynamic model of the omnidirectional WMR and the environment in which it is moving is of significant importance (Vázquez and Velasco-Villa, 2008; Ren *et al.*, 2019). Although it is possible to derive an inexact dynamic model of the robot and environment, the resulting controller often fails

to produce the desired performance in real-world implementations due to uncertainties in the model. These uncertainties include inaccuracies in identification of the aforementioned friction forces, parameters of the driving motors, the robot's inertia matrix, and the position of the robot's center of mass, all of which are necessary to account for in dynamic modeling of the robot (Ren and Ma, 2016; Williams *et al.*, 2002). Despite extensive research on these control problems, there remain challenges in designing controllers that exhibit robust performance in the presence of sensor noise and disturbances, are guaranteed to prevent collisions with both static and dynamic obstacles, and readily scale up to multiple robots while accommodating robot failures.

Motion control of nonholonomic WMRs is required in a wide variety of applications in robotics. Despite substantial work on designing motion controllers for nonholonomic WMRs, challenges still arise due to the nonholonomic constraints in the robot's kinematic model (Tzafestas, 2013; Brockett *et al.*, 1983). Various controllers have been designed to stabilize a nonholonomic WMR to a target position. Discontinuous controllers were developed in (Tanner and Kyriakopoulos, 2002; Astolfi, 1999) for exponential position stabilization. Since these controllers are based on a switching framework, noise in the robot's sensor measurements can result in undesired chattering along the boundaries of the switching conditions (Oriolo *et al.*, 2002). A time-varying feedback stabilization controller was proposed in (Samson, 1993), although it can produce transient oscillations and undesired cusps in the trajectory of the robot as it converges to the target position (Oriolo *et al.*, 2002). Control approaches based on Lyapunov stability theory were developed in (Shim and Sung, 2003; Cui *et al.*, 2019); however, these approaches cannot produce smooth continuous time-invariant feedback laws (Oriolo *et al.*, 2002). A feedback linearization method was proposed in (De Luca *et al.*, 2000) for smooth position stabilization, but this method produces a singularity in the control law when the robot's linear velocity is zero.

Other types of control methods have been proposed in the literature for position/pose

stabilization of nonholonomic WMRs, including smooth time-varying controllers (Gu and Hu, 2005), differential kinematic-based approaches (Dixon *et al.*, 2000), and guiding vector field (GVF) controllers (Kapitanyuk *et al.*, 2017). However, these control methods do not incorporate constraints on the robot's control inputs, which are enforced by the limitations of the robot's actuators, and constraints on the robot's states, which are determined by the boundaries of the free space in which the robot can move.

Other navigation strategies for nonholonomic WMRs that do incorporate these constraints have been developed using Model Predictive Control (MPC) (Saska *et al.*, 2016; Zarei *et al.*, 2020), which is also called Receding Horizon Control (RHC). MPC is a powerful control method that can handle Multiple-Input Multiple-Output (MIMO) constrained control problems (Rawlings *et al.*, 1994; Camacho and Alba, 2013). Nonlinear MPC (NMPC) methods for position stabilization of nonholonomic robots with nonlinear kinematic models have also been investigated (Worthmann *et al.*, 2015a; Mehrez *et al.*, 2020). However, online implementations of these methods are computationally intensive, and the transient portion of the robot's trajectory will not be smooth if the prediction horizon is not large enough. Geometric control techniques developed for holonomic systems, e.g. the approaches in (Bullo and Murray, 1995; Maithripala *et al.*, 2006), cannot be applied to nonholonomic robots, given their no-slip velocity constraint. To address this issue, a geometric control approach for position stabilization of noholonomic WMRs was designed in (Tayefi and Geng, 2019) by exploiting properties of exponential coordinates and the special Euclidean group $SE(2)$. However, in implementation, the controller's dependence on direct measurements of the robot's heading angle can produce chattering when this angle is near π or $-\pi$ rad. Moreover, discontinuities arise in the control law due to the use of the $atan2$ function.

Collision and Deadlock Avoidance by Mobile Robots

Although collision and deadlock avoidance in multi-robot systems can be guaranteed in centralized control approaches, these approaches have limited scalability since their computational complexity increases with the number of the robots (Salimi Lafmejani and Berman, 2021; Wang *et al.*, 2017). In distributed control approaches, inter-robot collision avoidance can be established using inter-robot communication (Wang *et al.*, 2017; Mehrez *et al.*, 2017b). However, inter-robot communication might be subject to communication delays or losses. Thus, decentralized control approaches have been developed to eliminate the aforementioned limitations of centralized and distributed approaches. However, existing decentralized methods are not fully decentralized due to one or more of the following simplifying assumptions: **(1)** the reliance of robots on inter-robot communication; **(2)** a priori knowledge of the robots about the positions of obstacles in the environment, or the absence of obstacles; and **(3)** dependency of the approach on a global localization system such as a camera or motion capture system.

Obstacle Avoidance

Obstacle avoidance is a well-studied challenging problem in the control of nonholonomic WMRs. Many existing controllers for obstacle avoidance by WMRs are based on artificial potential fields (Khatib, 1985; Rimon and Koditschek, 1992). In these approaches, the potential field is constructed to produce virtual attraction forces on the robot toward the target position and virtual repulsion forces away from the obstacles (Pandey *et al.*, 2017). These methods do not ensure global convergence to the target point while avoiding obstacles, since the robot can become trapped in local minima (Arslan and Koditschek, 2019). When applied to a nonholonomic WMR, potential field-based controllers that have been developed for holonomic robots may result in infeasible robot trajectories, since the robot cannot necessarily move along the prescribed gradient due to its nonholonomic constraint.

In (Lamiraux *et al.*, 2004; Papadopoulos *et al.*, 2002; Qu *et al.*, 2004), methods are proposed for mapping the infeasible trajectories obtained by the holonomic planner to feasible ones for nonholonomic robots; however, the potential functions in these methods are not strictly decreasing. To overcome this drawback, approaches that use differential flatness (Rigatos, 2015) and iterative calculations (Lamiraux *et al.*, 2004) have been proposed to derive feasible collision-free trajectories for nonholonomic WMRs.

A feedback controller for obstacle avoidance by nonholonomic WMRs was designed in (Urakubo, 2015) using a time-varying potential function with no local minima or saddle points. This approach is subject to the aforementioned issues associated with discontinuous time-varying controllers. In traditional control methods for collision-free navigation of mobile robots, collision avoidance has been achieved by incorporating the gradient of an artificial potential field into the controller design, e.g., navigation functions or attractive-repulsive potential fields (Salimi Lafmejani *et al.*, 2020b). A navigation function (Rimon and Koditschek, 1992) guarantees the convergence of a robot to a target position, defined as the global minimum of the function, and ensures no collisions with obstacles. The design of navigation function-based controllers for nonholonomic WMRs was studied in (Tamba *et al.*, 2009; Widyotriatmo and Hong, 2011; Kowalczyk, 2019). These methods rely on prior knowledge about the environment, including the geometry of the obstacles and the domain. In addition to potential forces, gyroscopic forces were proposed in (Chang and Marsden, 2003) to implement obstacle avoidance while ensuring that the robot converges to the target position. In (Ataka *et al.*, 2018), a reactive control method inspired by magnetic fields was proposed for obstacle avoidance by nonholonomic WMRs in environments with convex obstacles. This method and the sliding mode controller in (Savkin *et al.*, 2015) guarantee almost global convergence to the target position, can be applied in unbounded convex domains, and require the robot to have only local sensing capabilities, with no prior information about the environment such as the locations and shapes of the obsta-

cles. Other control approaches have been proposed based on the Velocity Obstacle (VO) method (Fiorini and Shiller, 1998) and Optimal Reciprocal Collision Avoidance (ORCA) method (Alonso-Mora *et al.*, 2018a). Recently, several methods have been developed to incorporate collision avoidance into the constraints of an optimization problem in order to enforce the kinematics or dynamics of the robot, constraints on the robot's states and actuation, and the dynamics of the environment in which robots are navigating (Salimi Lafmejani and Berman, 2021; Wang *et al.*, 2017; Zeng *et al.*, 2020).

Inter-Robot Collision and Deadlock Avoidance

Due to their ability to incorporate constraints on robots' states and control inputs, MPC-based methods have been employed for collision-free multi-robot navigation, in which collision avoidance between pairs of robots is encoded as constraints on the robots' states. Existing MPC-based methods have been developed for a single robot (Park *et al.*, 2012; Rösmann *et al.*, 2020) or for multiple robots in environments without obstacles (Wang *et al.*, 2017). Moreover, while centralized approaches can guarantee collision-free and deadlock-free (Wang *et al.*, 2017; Grover *et al.*, 2019) multi-robot navigation, it is challenging to design an MPC-based method that is computationally tractable and has these theoretical guarantees.

The main limitation of *online* implementations of MPC methods, especially for multi-robot systems with large numbers of robots, is their high computational cost, arising from the fact that the controller runs a constrained optimization problem over a specified horizon at each time step (Borrelli *et al.*, 2017). If sufficient computational resources are in fact available, online MPC methods can be used to compute controllers for large-scale systems and systems with fast dynamics (Di Carlo *et al.*, 2018; Rosolia and Ames, 2020). The stability of an MPC method can be ensured by introducing terminal constraints or terminal costs, e.g. (Keerthi and Gilbert, 1988; Azizi and Keighobadi, 2017). However, this

increases the computational complexity of the associated optimization problem.

Additional MPC-based approaches for multi-robot navigation have been developed which can be implemented with lower computational resources than online methods. Some of these approaches calculate the reachable set of the robots' states or the optimal control solutions *offline*, prior to the robots' deployment. For example, the MPC method in (Huang *et al.*, 2016) incorporates collision avoidance constraints in the optimization problem, and the feasibility and stability of the method are established by computing the reachable set of the robots' states. However, such offline computations scale poorly with the number of robots, and they cannot be used for real-time implementations in dynamic or uncertain environments. Another approach to reducing the computational effort is to simplify the control problem by formulating it as a linear MPC method (Kuwata *et al.*, 2007; Richards and How, 2003). This can be done by using linear models to describe the robots' motion, such as linear point-mass models or linearizations of the unicycle model (Mao *et al.*, 2020; Tallamraju *et al.*, 2018; Wang *et al.*, 2017). For instance, in (Mao *et al.*, 2020), a linear MPC is combined with the ORCA approach to synthesize navigation controllers for nonholonomic robots by computing holonomic reference trajectories for the robots, using the approximation that they are holonomic. However, many of these methods have been implemented only in simulation, e.g. (Huang *et al.*, 2016; Richards and How, 2003), not in real-world experiments. Moreover, applying these methods to nonholonomic robots, which are described by nonlinear kinematic models, will inevitably result in a tracking error between the reference trajectories and the robots' actual trajectories.

Various software tools and techniques have been developed for speeding up the solution of the optimization problem in an MPC method. In (Wang and Boyd, 2009), the authors propose approaches to implementing fast MPC using online optimization with interior-point methods. The work (Stellato *et al.*, 2020) presents a solver for convex quadratic programs, implemented in the Operator Splitting Solver for Quadratic Programming (OSQP),

which is typically ten times faster than the interior-point methods. However, these approaches are restricted to linear MPC problems (Mao *et al.*, 2020; Wang *et al.*, 2018) or a linearized form of the original nonlinear problem (Kuhne *et al.*, 2004; Lages and Alves, 2006). MPC methods for collision-free navigation by multiple nonholonomic WMRs involve the solution of a nonlinear optimal control problem. The software packages NLopt (Johnson, 2020) and APOPT (Hedengren *et al.*, 2012) can be used to solve this type of problem. Moreover, the open-source software tool CasADi (Andersson *et al.*, 2012) provides a symbolic programming framework for formulating the optimization problem, which further reduces the computational effort required to solve the problem and facilitates efficient implementation of NMPCs. Another way to accelerate the solution of the optimization problem is to define its decision variables as both the control inputs and the robot states, an approach called the *multiple-shooting method* (Bock and Plitt, 1984), instead of just the control inputs, as is done in the *single-shooting method* (Stella *et al.*, 2017). The single-shooting method can exhibit slow convergence to the optimal solution and numerical instabilities (Hussein *et al.*, 2019). The multiple-shooting method produces faster convergence to optimal solutions than the single-shooting method (Tamimi and Li, 2009), since it lifts the optimization problem to a higher-dimensional space. The multiple-shooting method is employed for MPC in (Mehrez, 2017) to accelerate the convergence of the associated optimization problem.

The need for inter-robot collision and deadlock avoidance also arises in the specific application of tracking of a moving target, e.g. a ground WMR, by multiple unmanned aerial vehicles (UAVs). Various control approaches have been proposed for this problem (Khan *et al.*, 2016), which occurs in scenarios such as **(1)** localization of a rover via multiple UAVs in space applications and new planet exploration (Agha-Mohammadi and Ebadi, 2018; Cristofalo *et al.*, 2016), **(2)** food and medicine delivery in response to disaster situations and search-and-rescue operations (Zohdi, 2018), **(3)** capturing aerial footage for films

and sporting events (Mademlis *et al.*, 2018), and **(4)** police chase of a moving suspect (Dille, 2013). In centralized control approaches, a central computational unit calculates and sends control commands to all the UAVs. However, the required computational resources of the central unit increase with the number of UAVs. In distributed approaches, the control problem is decomposed into multiple sub-problems, which are assigned to each UAV to solve using its on-board computational resources. In this setting, neighboring UAVs communicate their solutions to each other to plan collision-free paths. However, control methods that require inter-robot communication can suffer from communication losses or delays in the transmission of information between robots (Reis *et al.*, 2013; Marcotte, 2019).

To eliminate the requirement for a central computational unit or inter-robot communication, decentralized approaches have been developed for multi-robot systems, in which each robot computes its own control inputs using only local measurements, without communicating with other robots (Antonelli *et al.*, 2014; Omidshafiei *et al.*, 2017). However, these approaches require UAVs to be equipped with on-board 3D LiDAR sensors or cameras to detect other robots during tracking, and they necessitate a complicated controller design to guarantee both collision-free and deadlock-free navigation of the UAVs. Furthermore, most of the existing control methods only focus on tracking and collision avoidance, without addressing the controller's robustness to failure of UAVs during tracking.

Safe Multi-Robot Navigation

In general, collision avoidance in multi-robot systems entails the avoidance of both inter-robot and robot-obstacle collisions. *Safe navigation* of multiple mobile robots is achieved when all the robots move through the environment while avoiding collisions with one another and with any obstacles that are present. Inter-robot collision avoidance can be ensured in NMPC control methods by including distance functions in the constraints of the NMPC optimization problem, forcing each pair of robots to maintain a minimum

distance during navigation (Salimi Lafmejani and Berman, 2021). Collision avoidance can only be guaranteed by distance functions in a centralized framework, as described in (Salimi Lafmejani and Berman, 2021). However, since safety conditions based on distance functions do not account for obstacle dynamics when planning collision-free paths, this method fails to stabilize the robots to target trajectories while guaranteeing safety. Alternatively, Barrier Functions (BFs) and their derivatives, along with the vector field that defines each robot's kinematics, can be used to construct safety constraints that ensure collision avoidance (Majd *et al.*, 2021; Wang *et al.*, 2017). BFs separate safe and unsafe regions in the environment and can be included in a constrained minimization problem whose solution minimally deviates from a nominal controller with guaranteed stability (Ames *et al.*, 2014). Although safe navigation of the robots is guaranteed, control methods that employ BFs do not always guarantee deadlock avoidance (Alonso-Mora *et al.*, 2018b). For instance, the approach in (Wang *et al.*, 2017) for collision-free navigation of multi-robot systems, which uses safety barrier certificates, is not able to resolve all possible types of deadlocks since the method is decentralized.

There are several studies that incorporate BFs into constraints or directly into the objective function of the MPC optimization to ensure collision-free navigation of mobile robots. For instance, a safety-critical MPC method with discrete-time BF has been presented in (Zeng *et al.*, 2020) for collision-free navigation of mobile robots in known, static environments. The existing BF-based MPC methods analytically construct the BFs and incorporate them into MPC design to ensure collision-free navigation, which is impractical in real-world applications. Synthesizing BFs is straightforward in a centralized control architecture where the robots have information about the positions and geometry of the obstacles. Given this information, a BF, representing the boundary of the smallest circle that encloses an obstacle, could be defined for each obstacle. On the other hand, it would be challenging for each robot to individually construct BFs in a decentralized manner using

only its own sensor measurements, without such prior information. To address this challenge, several recent studies have investigated the use of neural networks (NNs) to learn BFs offline or online in priori both known and unknown environments (Yaghoubi *et al.*, 2020; Long *et al.*, 2021; Zhao *et al.*, 2020; Saveriano and Lee, 2019; Srinivasan *et al.*, 2020; Li *et al.*, 2021).

1.2.3 Motivation

Here, we highlight some major challenges associated with controller design for multi-segment manipulators and mobile robot collectives that have motivated our research.

Multi-Segment Manipulators

The large number of degrees of freedom of hyper-redundant multi-segment manipulators creates many challenges for conventional centralized control strategies, e.g., the large number of solutions for the Inverse Kinematic (IK) problem, which motivates the use of alternative control approaches. Existing alternative methods that use centralized control have employed numerical methods (Ananthanarayanan and Ordóñez, 2015) and the pseudo-inverse Jacobian method (Siciliano, 1990; Salimi Lafmejani *et al.*, 2020a). However, these approaches are limited by their high computational complexity, which make them inefficient for real-time applications. Moreover, centralized control of a multi-segment manipulator requires the measurement of the robot's joint variables represented in the global coordinate frame, which makes them unsuitable for autonomous control applications. The computational effort of implementing such controllers is likely to rise as the number of joints in the robot increases. Moreover, existing methods for kinematic control of hyper-redundant multi-segment robots are constrained by their computational complexity and limited to robot configurations in 1D or 2D space.

Mobile Robot Collectives

Many applications of multi-robot systems require mobile robots to navigate through an environment while avoiding collisions with one another and with obstacles. Despite extensive research on this topic, there are still challenges to designing control strategies for multi-robot navigation that are computationally efficient and have theoretical guarantees on collision avoidance and absence of deadlocks. Addressing these challenges motivates the design of collision-free and deadlock-free multi-robot navigation strategies that are based on an online, computationally efficient Nonlinear Model Predictive Control (NMPC) method for pose stabilization and trajectory tracking of multiple mobile robots, which may be nonholonomic (e.g., differential-drive WMRs). In addition, existing control approaches for stabilizing the position of a nonholonomic robot suffer from various limitations, including (1) chattering in the robot's motion that results from the use of discontinuous functions in the control law, e.g. *sgn* (Tanner and Kyriakopoulos, 2002), *arctan* (Astolfi, 1999), and *atan2*; and (2) erratic or oscillatory transient robot motions, which are intrinsic characteristics of time-varying control laws (Worthmann *et al.*, 2015a) and pure geometric techniques (Laumond *et al.*, 1998). Furthermore, many existing control approaches for obstacle avoidance (1) have been developed using a holonomic motion planner, which may introduce infeasible collision-free paths and cannot be implemented on nonholonomic robots (De Luca and Oriolo, 1994); (2) do not have mathematical guarantees on performance; and (3) can result in the robot becoming trapped in a local minimum.

1.3 Research Contributions and Organization of Dissertation

The contributions of this dissertation are the following, described in the sections listed below.

- Section 2.1: A kinematic model of a 2D soft multi-segment manipulator and a de-

centralized, consensus-based kinematic control method for position stabilization and trajectory tracking control of the manipulator's tip. The consensus protocol is defined by considering the segments' local measurements of their own configurations as the shared information states, which are communicated between adjacent segments.

- Section 2.2: A kinematic model of a 3D soft multi-segment manipulator that reproduces all fundamental motions of an octopus arm, including elongation, shortening, bending, and twisting, while enforcing the constant-volume property of the octopus arm, and a pseudo-inverse Jacobian method for position stabilization and trajectory tracking control of the manipulator's tip.

- Section 3.1: A full-state feedback H_∞-optimal controller for pose stabilization and trajectory tracking control of a 3-wheeled omnidirectional mobile robot that is subject to frictional disturbances and sensor noise. The control approach does not require a rigid-body dynamical model of the robot and is robust to uncertainties originating from undesired exogenous inputs on the robot.

- Section 3.2: An approach to converting any gradient-based feedback controller designed for a holonomic robot into a controller that can be implemented on a nonholonomic robot for position stabilization and obstacle avoidance. The controller produces a smooth robot trajectory for either forward or backward driving to the target position.

- Section 3.3: A nonlinear MPC method for collision-free and deadlock-free pose stabilization of multiple nonholonomic mobile robots. The method is executable online, which enables real-time implementation, and the computational complexity of the nonlinear MPC is significantly reduced by designing the associated optimization problem without stabilizing terminal constraints or terminal costs.

- Section 3.4: A nonlinear MPC-based method for collision-free, deadlock-free trajectory tracking control of multiple nonholonomic mobile robots in environments with known obstacles, using barrier functions corresponding to the obstacles to ensure safe navigation. The method is executable online and is designed without stabilizing terminal constraints or terminal costs in the associated optimization problem.

- Section 3.5: A decentralized nonlinear MPC-based control method for collision-free pose stabilization of multiple nonholonomic mobile robots, which lack inter-robot communication, in unknown environments with static and/or non-aggressive dynamic obstacles. To ensure safe navigation, the method employs barrier functions that each robot learns in real-time from its on-board LiDAR and odometry measurements using a Deep Neural Network.

- Section 3.6: A feedforward proportional-derivative (PD) controller for collision-free velocity tracking of a moving ground target by multiple unmanned aerial vehicles (UAVs), which lack inter-robot communication and on-board sensing for collision avoidance. The controller drives the UAVs to a fixed formation above the target and is robust to UAV failures.

Chapter 4 concludes with a review of our contributions and possible directions for future research.

Chapter 2

POSITION STABILIZATION AND TRAJECTORY TRACKING CONTROL OF MULTI-SEGMENT MANIPULATORS

This chapter describes research published in Salimi Lafmejani *et al.* (2020e) (Section 2.1) and Salimi Lafmejani *et al.* (2020a) (Section 2.2).

2.1 Position Stabilization and Trajectory Tracking Control of 2D Multi-Segment Manipulators

In this section, we propose a novel decentralized approach to kinematic control of soft segmented continuum robots based on a consensus strategy. The robots under consideration deform in a plane according to a multi-segment Piecewise Constant Curvature (PCC) kinematic model in which each segment is represented as an equivalent rigid-link Revolute-Prismatic-Revolute (RPR) mechanism. In our approach, we assume that each segment of the robot is equipped with sensors to measure joint variables in its local coordinate frame and can communicate with its two adjacent segments. Our consensus-based decentralized control strategy provides an alternative to conventional control methods, which solve the inverse kinematic problem by using computationally intensive numerical methods to calculate the robot's Jacobian matrix at each time instant. We investigate the stability and convergence properties of proposed controllers for position regulation and trajectory tracking tasks and provide theoretical guarantees on the controllers' performance. We evaluate the controllers in simulation for scenarios in which the robot's tip must reach a certain position or follow a specified trajectory. We compare the performance of the position regulator for different controller gains, and we find that a simulated 15-link robot can track a complex reference trajectory with an average root-mean-square error of only 0.16% of the

robot's initial length.

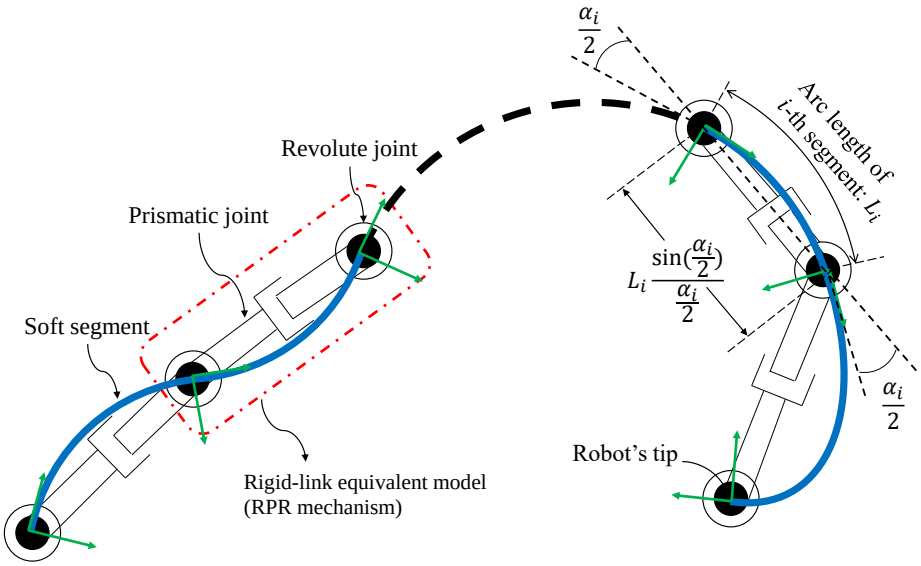


Figure 2.1: Schematic of a Continuum Robot Composed of Soft Segments, Along with its Equivalent Model as a Serial configuration of multiple rigid-link RPR mechanisms. Local coordinate frames (Frenet-Serret frames (Webster III and Jones, 2010)) are defined at the two ends of each soft segment.

2.1.1 Rigid-Link Model of Soft Continuum Robot

The kinematics of the soft continuum robot are discussed in this section. The robot is composed of a set of segments connected to each other in a series configuration. The soft segments of the robot are replaced by equivalent rigid-link RPR mechanisms (Della Santina *et al.*, 2018).

Figure 2.1 depicts a soft continuum robot with soft segments, which is equivalently modeled with multiple rigid-link RPR mechanisms connected in a series. To clarify the details of the model, Figs. 2.2 and 2.3 show the soft segmented robot and its equivalent rigid-link robot, respectively. Figure 2.2 illustrates a planar segmented soft continuum robot with N bending segments, each conforming to the constant-curvature (CC) assump-

tion. The kinematic model of the continuum robot is defined by the kinematic equations of the multi-segment N-RPR rigid-link robot in Fig. 2.3. Since we assume that each segment of the robot is equipped with local sensors and actuators, it is able to measure the position of its prismatic joint and relative rotations of its revolute joints in its local coordinate frame. Furthermore, the i-th segment can communicate these local measurements to the adjacent $(i-1)$-th and $(i+1)$-th segments, as shown in Fig. 2.2.

As shown in Fig. 2.1, the angular difference between the tangential local coordinate frames attached to the base and end-effector of the i-th segment and the orientation of the equivalent RPR rigid-link mechanism is defined as $\frac{\alpha_i}{2}$. Furthermore, we denote the arc length of the i-th soft segment by L_i. Accordingly, as in (Della Santina *et al.*, 2018), the position vector that is aligned with the prismatic joint of the i-th segment can be represented in the segment's local coordinate frame as the following vector ${}^i\vec{p}_i$:

$$ {}^i\vec{p}_i = \begin{bmatrix} L_i \dfrac{\sin(\alpha_i)}{\alpha_i} & L_i \dfrac{1-\cos(\alpha_i)}{\alpha_i} & 0 \end{bmatrix}^{\mathrm{T}}. \tag{2.1} $$

Thus, given the arc parameters L_i and α_i of each soft segment, the position vector ${}^i\vec{p}_i$ of the equivalent RPR mechanism is readily obtained.

2.1.2 Controller Design

In this section, we propose a novel decentralized method for kinematic control of soft segmented continuum robots. This control method utilizes a consensus strategy among the segments of the robot, which can communicate measurements to each other according to a chain network topology.

Position Regulation Controller

Here, we define a control law that drives the tip of the robot to a target 3D position in the global frame G, denoted by ${}^G\vec{p}_d$. We assume that the end-effector segment of the robot,

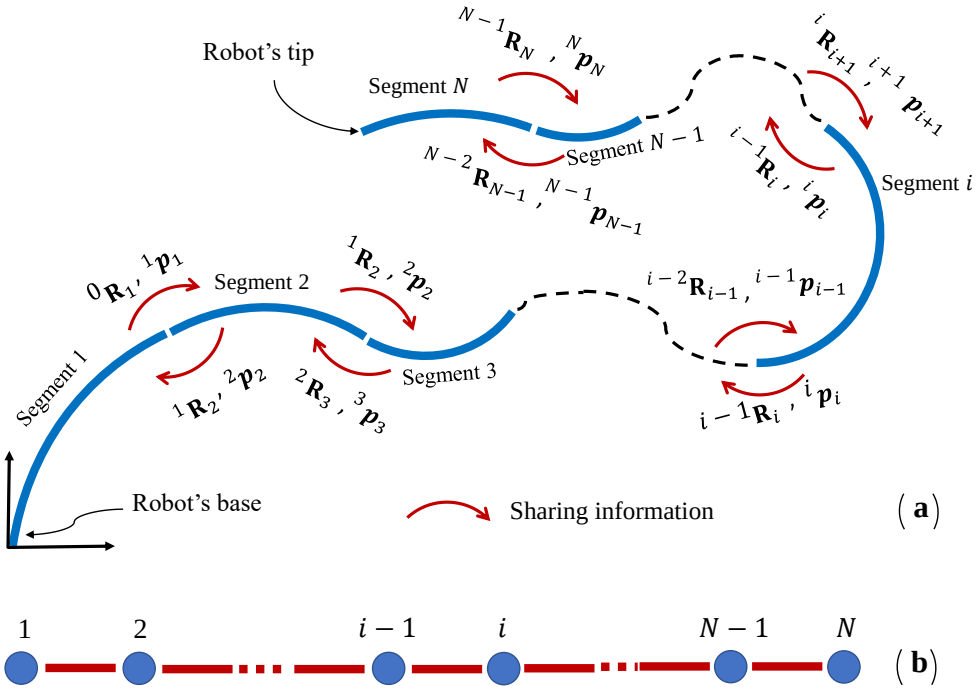


Figure 2.2: Schematic of Information Propagation Between Segments of the Continuum Robot. (a) Each Segment $i \in \{1, ..., N\}$ Communicates with its Adjacent Segment(s) in Order to Share its Measurements of the Position Vector ${}^{i}\vec{p}_i$ and Rotation Matrix ${}^{i-1}\vec{R}_i$, which are Represented in its Local Coordinate Frame. (b) Graph of the Robot's Communication Network, where Each Blue Node Represents a Segment and Each Red Edge Represents a Bidirectional Communication Channel.

segment N, knows the target position ${}^{G}\vec{p}_d$ and is equipped with a localization sensor that can measure ${}^{G}\vec{p}_{tip}$, the 3D position of the tip of the segment in the global frame. The other segments do not have information about ${}^{G}\vec{p}_d$. Segment N can be considered a "leader" agent in that it knows ${}^{G}\vec{p}_d$ and moves toward this position, whereas the other segments are "follower" agents that reconfigure themselves in a coordinated fashion such that the tip of the robot is regulated to ${}^{G}\vec{p}_d$.

The configuration of each rigid segment can be characterized by the joint variables that describe its linear and angular displacements, which are denoted by $p_i \in \mathbb{R}$ and $\theta_i \in \mathbb{R}$,

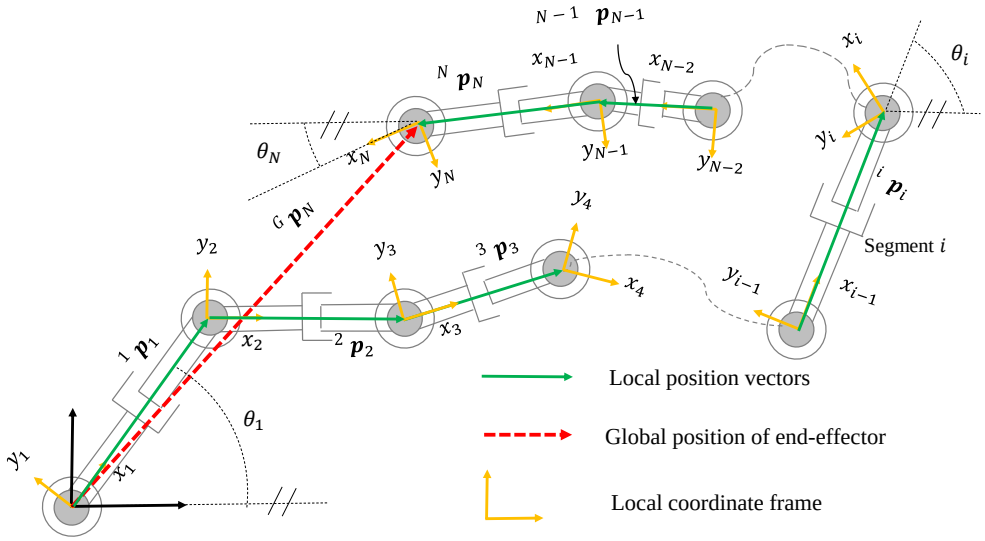


Figure 2.3: Illustration of an Equivalent N-RPR Rigid-Link Model for the Soft Continuum Robot in Fig. 2.2. The x-Axis of Each Local Coordinate Frame is Aligned with the Corresponding Prismatic Joint.

respectively. Given the joint variables, the local position vector of the prismatic joint is written as ${}^i\vec{p}_i = p_i {}^i\vec{e}_i$, where ${}^i\vec{e}_i \in \mathbb{R}^3$ is the unit vector along the prismatic joint expressed in the segment's local frame. The angular velocity of the segment in the global frame can be written as $\vec{\omega}_i = [0 \;\; 0 \;\; \dot{\theta}_i]^T \in \mathbb{R}^3$. We denote the vector of generalized coordinates of the i-th segment as $\vec{q}_i \in \mathbb{R}^6$, which is given by:

$$\vec{q}_i = [\, {}^i\vec{p}_i^{\,T} \;\; 0 \;\; 0 \;\; \theta_i \,]^T . \tag{2.2}$$

The position regulation controller is defined as:

$$\dot{\vec{q}}_i = \vec{G}_i^{\dagger} \left(- \sum_{j \in \mathcal{N}_i} ({}^i\vec{p}_i - {}^i\vec{R}_j {}^j\vec{p}_j) + \vec{B}_i \vec{E} \right). \tag{2.3}$$

The components of the controller are defined as follows. The matrix $\vec{G}_i^{\dagger}$ denotes the Moore-Penrose inverse of $\vec{G}_i \in \mathbb{R}^{3\times 6}$, which is given by:

$$\vec{G}_i = [\, \vec{I}_{3\times 3} \;\; - {}^i\hat{\vec{p}}_i \,], \tag{2.4}$$

where ${}^i\hat{\vec{p}}_i \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix representation of ${}^i\vec{p}_i$. Defining the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V} = \{1, ..., N\}$ contains the segment identities and the edge set $\mathcal{E}$ contains pairs of segments that can communicate with each other, also referred to as *neighboring* segments, the set $\mathcal{N}_i$ is defined as the neighbors of the i-th segment. The matrix ${}^i\vec{R}_j \in \mathbb{R}^{3\times3}$ is the rotation matrix from the local coordinate system of segment j to that of segment i. The matrix $\vec{B}_i \in \mathbb{R}^{3\times3}$ is defined as:

$$\vec{B}_i = \begin{cases} \vec{0}_{3\times3}, & i = 1, 2, \ldots, N-1, \\ -{}^G\vec{R}_N^{-1}K, & i = N, \end{cases} \tag{2.5}$$

where K is a tunable positive gain and ${}^G\vec{R}_N$ is the rotation matrix from the local coordinate system of segment N to the global frame. The vector $\vec{E} \in \mathbb{R}^3$ is defined as the error between the position of the tip of the robot and its target position, both in the global frame:

$$\vec{E} = {}^G\vec{p}_{tip} - {}^G\vec{p}_d. \tag{2.6}$$

Remark 2.1.1. *The controller in Eq. (2.3) is completely decentralized, in the sense that each segment only requires measurements of its own configuration and the configurations of its neighboring segments. Note that the controller requires only segment N to measure a position error in the global frame (the error vector $\vec{E}$).*

Trajectory Tracking Controller

We define a reference trajectory for the robot's tip in the global frame and discretize it into a set of m points. Adopting a switching strategy, we use the decentralized position regulator in Eq. (2.3) to drive the tip of the robot to a position within a radius $\gamma \in \mathbb{R}_{>0}$ of each point, which represents the acceptable tracking error.

The set of target points along the reference trajectory is denoted by $\mathcal{P}$ and is defined as

$$\mathcal{P} = \left\{ {}^G\vec{p}_d^{(1)}, {}^G\vec{p}_d^{(2)}, \ldots, {}^G\vec{p}_d^{(m)} \right\}, \tag{2.7}$$

where ${}^G\vec{p}_d^{(1)}$ is the position of the start point on the trajectory, and ${}^G\vec{p}_d^{(m)}$ is the position of the end point. The other elements of $\mathcal{P}$ are intermediate points along the trajectory. A γ-neighborhood of point ${}^G\vec{p}_d^{(l)}$ is defined as a ball of radius γ centered at this point and is denoted by $\mathcal{B}_\gamma^{(l)}$. The controller first drives the position of the robot's tip toward the start point ${}^G\vec{p}_d^{(1)}$. Once the tip enters the ball $\mathcal{B}_\gamma^{(1)}$, the controller redefines the target point as the second point, ${}^G\vec{p}_d^{(2)}$, and drives the tip toward this point until it enters the ball $\mathcal{B}_\gamma^{(2)}$. This procedure is repeated for each successive point in $\mathcal{P}$ until the robot's tip enters the γ-neighborhood of ${}^G\vec{p}_d^{(m)}$. This switching control strategy can be written as:

$$\dot{\vec{q}}_i = \begin{cases} \vec{G}_i^{\dagger}\big(-\sum_{j\in\mathcal{N}_i}({}^i\vec{p}_i - {}^i\vec{R}_j\,{}^j\vec{p}_j) + \vec{B}_i\vec{E}^{(l)}\big), & {}^G\vec{p}_{\text{tip}} \notin \mathcal{B}_\gamma^{(l)} \\ \vec{G}_i^{\dagger}\big(-\sum_{j\in\mathcal{N}_i}({}^i\vec{p}_i - {}^i\vec{R}_j\,{}^j\vec{p}_j) + \vec{B}_i\vec{E}^{(l+1)}\big), & {}^G\vec{p}_{\text{tip}} \in \mathcal{B}_\gamma^{(l)} \end{cases} \tag{2.8}$$

where $l \in \{1, 2. \ldots, m\}$, and $\vec{E}^{(l)}$ is the vector of the error between the robot's tip and the l-th point on the reference trajectory:

$$\vec{E}^{(l)} = {}^G\vec{p}_{tip} - {}^G\vec{p}_d^{(l)}. \tag{2.9}$$

Note that from Eq. (2.5), segment N is the only segment that requires $\vec{E}^{(l)}$ in its controller, and the term $\vec{B}_i\vec{E}^{(l)}$ is the zero vector for the other segments.

2.1.3 Stability and Convergence Analysis

In this section, we analyze the motion of the robot with the proposed controller. We first study the reconfiguration of each individual segment, and then investigate the stability and convergence properties of the closed-loop system that describes the kinematics of the robot's tip.

Lemma 2.1.2. *Let ${}^G\vec{p}_i$ denote the local position vector ${}^i\vec{p}_i$ of segment i in the global frame G. The decentralized control law in Eq. (2.3) establishes a consensus protocol for the vectors ${}^G\vec{p}_i$, with an input term defined by the error $\vec{E}$ between the robot's tip and the target position.*

Proof. The vector ${}^{G}\vec{p}_i$ can be written as

$$ {}^{G}\vec{p}_i = {}^{G}\vec{R}_i {}^{i}\vec{p}_i, \tag{2.10} $$

and its time derivative is given by

$$ {}^{G}\dot{\vec{p}}_i = {}^{G}\vec{R}_i {}^{i}\dot{\vec{p}}_i + {}^{G}\dot{\vec{R}}_i {}^{i}\vec{p}_i. \tag{2.11} $$

The time derivative of the rotation matrix is ${}^{G}\dot{\vec{R}}_i = {}^{G}\vec{R}_i {}^{i}\hat{\vec{\omega}}_i$, where ${}^{i}\hat{\vec{\omega}}_i \in SO(3)$ is the skew-symmetric matrix representation of the i-th segment's angular velocity expressed in its local coordinate system (Murray, 2017). Hence, the second term in the right-hand side of Eq. (2.11) can be written as ${}^{G}\dot{\vec{R}}_i {}^{i}\vec{p}_i = {}^{G}\vec{R}_i {}^{i}\hat{\vec{\omega}}_i {}^{i}\vec{p}_i$. Moreover, we know that

$$ {}^{i}\hat{\vec{\omega}}_i {}^{i}\vec{p}_i = {}^{i}\vec{\omega}_i \times {}^{i}\vec{p}_i = -{}^{i}\vec{p}_i \times {}^{i}\vec{\omega}_i = -{}^{i}\hat{\vec{p}}_i {}^{i}\vec{\omega}_i. \tag{2.12} $$

Thus, Eq. (2.11) can be rewritten as

$$ {}^{G}\dot{\vec{p}}_i = {}^{G}\vec{R}_i \left({}^{i}\dot{\vec{p}}_i - {}^{i}\hat{\vec{p}}_i {}^{i}\vec{\omega}_i \right), \tag{2.13} $$

and from Eq. (2.2) and Eq. (2.4), it can be simplified to

$$ {}^{G}\dot{\vec{p}}_i = {}^{G}\vec{R}_i \vec{G}_i \dot{\vec{q}}_i. \tag{2.14} $$

Furthermore, substituting the control law in Eq. (2.3) for $\dot{\vec{q}}_i$, we obtain

$$ {}^{G}\dot{\vec{p}}_i = {}^{G}\vec{R}_i \vec{G}_i \vec{G}_i^{\dagger} \left(- \sum_{j \in \mathcal{N}_i} ({}^{i}\vec{p}_i - {}^{i}\vec{R}_j {}^{j}\vec{p}_j) + \vec{B}_i \vec{E} \right). \tag{2.15} $$

Using the identity $\vec{G}_i \vec{G}_i^{\dagger} = \vec{I}$ and the fact that ${}^{G}\vec{R}_i {}^{i}\vec{p}_i = {}^{G}\vec{p}_i$, Eq. (2.15) is reduced to

$$ {}^{G}\dot{\vec{p}}_i = - \sum_{j \in \mathcal{N}_i} ({}^{G}\vec{p}_i - {}^{G}\vec{p}_j) + {}^{G}\vec{R}_i \vec{B}_i \vec{E}. \tag{2.16} $$

Equation (2.16) is in the form of a consensus protocol on ${}^{G}\vec{p}_i$ with an input term ${}^{G}\vec{R}_i \vec{B}_i \vec{E}$ (Olfati-Saber *et al.*, 2007). If we also define $\vec{\vec{P}} \in \mathbb{R}^{3N}$ and $\vec{\vec{B}} \in \mathbb{R}^{3N \times 3}$ as

$$ \vec{\vec{P}} = [{}^{G}\vec{p}_1^{T} \; {}^{G}\vec{p}_2^{T} \; \dots \; {}^{G}\vec{p}_N^{T}]^{T}, \tag{2.17} $$

$$ \vec{\vec{B}} = [({}^{G}\vec{R}_1 \vec{B}_1)^{T} \; ({}^{G}\vec{R}_2 \vec{B}_2)^{T} \; \dots \; ({}^{G}\vec{R}_N \vec{B}_N)^{T}]^{T}, \tag{2.18} $$

we can write the concatenated representation of Eq. (2.16) for all segments as

$$\dot{\vec{\vec{P}}} = -\vec{\mathcal{L}}\vec{\vec{P}} + \vec{\vec{B}}\vec{E}, \tag{2.19}$$

where $\vec{\mathcal{L}} \in \mathbb{R}^{3N\times 3N}$ is defined as

$$\vec{\mathcal{L}} = \vec{L} \otimes \vec{I}_{3\times 3}, \tag{2.20}$$

in which $\vec{L} \in \mathbb{R}^{N\times N}$ is the Laplacian matrix of the graph associated with the communication network of the robot's segments, illustrated in Fig. 2.2b, and $\otimes$ represents the Kronecker product. Hence, Eq. (2.19) is a linear consensus system (Saber and Murray, 2003) driven by the signal $\vec{E}$. □

The next theorem characterizes the stability of the closed-loop system.

Theorem 2.1.3. *The decentralized control law in Eq. (2.3) ensures that the position of the robot's tip, ${}^{G}\vec{p}_{tip}$, is globally exponentially stable to the target position, ${}^{G}\vec{p}_{d}$. Moreover, the magnitudes of all the position vectors ${}^{G}\vec{p}_{i}$ converge to a common value, and the directions of these vectors converge to the direction of ${}^{G}\vec{p}_{d}$.*

Proof. The position of the robot's tip in the global frame can be written as the vector sum of all position vectors ${}^{G}\vec{p}_{i}$ (see Fig. 2.3):

$${}^{G}\vec{p}_{tip} = \sum_{i=1}^{N} {}^{G}\vec{p}_{i}. \tag{2.21}$$

Taking the time derivative of this equation and following the same procedure that was used to obtain Eq. (2.10)–(2.16), the velocity of the robot's tip is derived as:

$${}^{G}\dot{\vec{p}}_{tip} = \sum_{i=1}^{N}\left(-\sum_{j\in\mathcal{N}_i}({}^{G}\vec{p}_{i} - {}^{G}\vec{p}_{j})\right) + \sum_{i=1}^{N} {}^{G}\vec{R}_i\vec{B}_i\vec{E}. \tag{2.22}$$

We can confirm that the double summation on the right-hand side of Eq. (2.22) is equal to the sum of the rows of the product $-\vec{\mathcal{L}}\vec{\vec{P}}$. This, in turn, can be written as the product of the row sum of $-\vec{\mathcal{L}}$ and the matrix $\vec{\vec{P}}$. We know that the sum of the rows of a Laplacian matrix

is a zero row vector (Olfati-Saber *et al.*, 2007). Invoking Lemma 2.1.2 and considering Eq. (2.20), we can conclude that the sum of the rows of $\vec{\mathcal{L}}$ is a zero row vector, and consequently, the double summation in Eq. (2.22) is zero. Also, from the definition of the matrix $\vec{B}_i$ in Eq. (2.5), the second summation is equal to $\vec{I}$. Therefore, Eq. (2.22) is reduced to

$$^{G}\dot{\vec{p}}_{tip} = -K\vec{E}. \tag{2.23}$$

Using the fact that the target position is fixed, *i.e.* $^{G}\dot{\vec{p}}_{d} = \vec{0}$, Eq. (2.23) can be rewritten as

$$\dot{\vec{E}} + K\vec{E} = \vec{0}, \tag{2.24}$$

which is globally exponentially stable to the equilibrium, *i.e.* $\vec{E} = \vec{0}$, for any positive K.

Furthermore, the system in Eq. (2.19) is a linear time-invariant system and can therefore be solved for $\vec{\vec{P}}$ as follows (Williams and Lawrence, 2007):

$$\vec{\vec{P}}(t) = \vec{\vec{P}}_0 e^{-\vec{\mathcal{L}}t} + \int_0^t e^{-\vec{\mathcal{L}}(t-\tau)}\vec{\vec{B}}\vec{E}(\tau)d\tau, \tag{2.25}$$

where $\vec{P}_0$ is the matrix $\vec{P}$ at the initial time, $t = 0$. We can also solve Eq. (2.24) for $\vec{E}$ as

$$\vec{E}(t) = \vec{E}_0 e^{-Kt}, \tag{2.26}$$

where $\vec{E}_0$ is the initial error at time $t = 0$. Substituting the solution for $\vec{E}(t)$ from Eq. (2.26) into Eq. (2.25), and using spectral factorization of the matrix $\vec{\mathcal{L}}$ as in (Mesbahi and Egerstedt, 2010), Eq. (2.25) can be rewritten as:

$$\begin{aligned}\vec{P}(t) = &\sum_{j=1}^{3N} e^{-\lambda_j t}(\vec{u}_j^T\vec{P}_0)\vec{u}_j \\ &+ \int_0^t \left(\sum_{j=1}^{3N} e^{-\lambda_j(t-\tau)}(\vec{u}_j^T\vec{B}\vec{E}_0 e^{-K\tau})\vec{u}_j\right) d\tau,\end{aligned} \tag{2.27}$$

where $\lambda_j \in \mathbb{R}_{\geq 0}$ for $j \in \{1, 2, \ldots, 3N\}$ are the eigenvalues of $\vec{\mathcal{L}}$ ordered from smallest (λ_1) to largest (λ_{3N}), and $\vec{u}_j \in \mathbb{R}^{3N}$ are their corresponding normalized eigenvectors.

Calculating the integral term, we obtain the following expression[1]

$$\vec{P}(t) = \sum_{j=1}^{3N} e^{-\lambda_j t} \left(\vec{u}_j^T \vec{P}_0\right) \vec{u}_j + \sum_{j=1}^{3N} \frac{\left(e^{-Kt} - e^{-\lambda_j t}\right)}{\lambda_j - K} \left(\vec{u}_j^T \vec{B} \vec{E}_0\right) \vec{u}_j. \tag{2.28}$$

We know that the first eigenvalue of the Laplacian matrix $\vec{L}$ is $\lambda_1(\vec{L}) = 0$, and the other eigenvalues are strictly positive (Olfati-Saber *et al.*, 2007). Consequently, the first three eigenvalues of $\vec{\mathcal{L}}$ are $\lambda_1(\vec{\mathcal{L}}), \lambda_2(\vec{\mathcal{L}}), \lambda_3(\vec{\mathcal{L}}) = 0$, and the $3N - 3$ remaining eigenvalues are strictly positive. Thus, we can write

$$\lim_{t\to\infty} \vec{P}(t) = \sum_{j=1}^{3} \left(\vec{u}_j^T \vec{P}_0\right) \vec{u}_j + \frac{1}{K} \sum_{j=1}^{3} \left(\vec{u}_j^T \vec{B} \vec{E}_0\right) \vec{u}_j, \tag{2.29}$$

since the exponential terms associated with the positive eigenvalues converge to zero as $t \to \infty$. Also, we can confirm that the three normalized eigenvectors associated with the three zero eigenvalues of $\vec{\mathcal{L}}$ are:

$$\begin{aligned} \vec{u}_1 &= \frac{1}{\sqrt{N}}[1\,0\,0\,1\,0\,0\ \ldots\ 1\,0\,0]^T, \\ \vec{u}_2 &= \frac{1}{\sqrt{N}}[0\,1\,0\,0\,1\,0\ \ldots\ 0\,1\,0]^T, \\ \vec{u}_3 &= \frac{1}{\sqrt{N}}[0\,0\,1\,0\,0\,1\ \ldots\ 0\,0\,1]^T. \end{aligned} \tag{2.30}$$

Substituting these vectors into Eq. (2.29), and using the definitions in Eq. (2.17) and Eq. (2.18), Eq. (2.29) can be rewritten as

$$\lim_{t\to\infty} \vec{P}(t) = \vec{1}_N \otimes \left(\frac{1}{N} \sum_{i=1}^{N} {}^G\vec{p}_i(0)\right) + \vec{1}_N \otimes \left(-\frac{1}{N} \vec{E}_0\right), \tag{2.31}$$

[1] A special case is when K is set equal to a positive eigenvalue λ_j of $\vec{\mathcal{L}}$ ($j > 3$). Then the term $e^{-\lambda_j t} e^{(\lambda_j - K)\tau}$ in Eq. (2.27) equals $e^{-\lambda_j t} e^0 = e^{-\lambda_j t}$ and integrates to $\int_0^t e^{-\lambda_j t} d\tau = t e^{-\lambda_j t}$ in the second summation of Eq. (2.28), with no denominator $(\lambda_j - K)$ in this summation. The term $t e^{-\lambda_j t}$ converges to zero as $t \to \infty$, and thus Eq. (2.29) still holds true.

where $\vec{1}_N \in \mathbb{R}^N$ is the vector of all ones, and ${}^G\vec{p}_i(0)$ denotes the initial value of ${}^G\vec{p}_i$ at time $t = 0$. Finally, using Eq. (2.21) and the definition of the error, $\vec{E} = {}^G\vec{p}_{tip} - {}^G\vec{p}_d$, we obtain:

$$\lim_{t\to\infty} \vec{P}(t) = \frac{1}{N}\left(\vec{1}_N \otimes {}^G\vec{p}_d\right), \tag{2.32}$$

which means that

$$\lim_{t\to\infty} {}^G\vec{p}_i(t) = \frac{1}{N}\left({}^G\vec{p}_d\right), \quad \forall\, i \in \{1, 2, \ldots, N\}. \tag{2.33}$$

This shows that the ${}^G\vec{p}_i$ vectors all converge to the same magnitude and direction, and this direction is that of the vector ${}^G\vec{p}_d$. □

Note that the final configuration of the robot, which is a straight line, is not singular, since the prismatic joints can still move the end-effector along that line, and the revolute joints can still rotate the segments of the robot.

This section concludes with the following corollary, which characterizes the stability of the switching controller proposed for trajectory tracking. The corollary can be derived from the result that the closed-loop system described by Eq. (2.24) is globally exponentially stable.

Corollary 2.1.4. *The decentralized switching controller in Eq.* (2.8) *drives the tip of the robot to a γ-neighborhood of each point in the set $\mathcal{P}$, defined in Eq.* (2.7)*, in finite time if γ is chosen sufficiently small.*

Using the control law in Eq. (2.8), the trajectory tracking task is performed as m position regulation tasks, which are indexed by l and executed sequentially from $l = 1$ to $l = m$. We proved that the equation for the closed-loop system in a position regulation task is given by Eq. (2.24). Therefore, the closed-loop system for the trajectory tracking task behaves like a switching system, in which each subsystem is:

$$\dot{\vec{E}}^{(l)} + K\vec{E}^{(l)} = \vec{0}, \qquad l = 1, 2, \ldots, m \tag{2.34}$$

with $\vec{E}^{(l)}$ defined in Eq. (2.9). Equation (2.34) is linear and so can be solved for $\vec{E}^{(l)}$ as:

$$\vec{E}^{(l)}(t) = \vec{E}^{(l)}_{t_l} e^{-K(t-t_l)}, \quad \forall t \in [t_l, t_{l+1}), \tag{2.35}$$

where t_l is the time at which subsystem l becomes active, and $\vec{E}^{(l)}_{t_l}$ denotes the value of $\vec{E}^{(l)}(t)$ at time $t = t_l$. This solution converges exponentially to the target equilibrium $\vec{E}^{(l)} = \vec{0}$, which implies that the robot's tip will converge in finite time to a neighborhood of the desired point ${}^G\vec{p}^{(l)}_d$ and that the trajectories of the tip are bounded. Equation (2.35) also holds for the norm of the error:

$$||\vec{E}^{(l)}(t)|| = ||\vec{E}^{(l)}_{t_l}|| e^{-K(t-t_l)}, \quad \forall t \in [t_l, t_{l+1}). \tag{2.36}$$

The *dwell time* for the l-th subsystem to reach $\mathcal{B}^{(l)}_\gamma$ is defined as $T_l := t_l - t_{l-1}$ (Liberzon, 2003), which can be computed by setting $||\vec{E}^{(l)}(t)|| = \gamma$ in Eq. (2.36) and solving this equation for T_l:

$$T_l = -\frac{1}{K} \log \left(\frac{\gamma}{||\vec{E}^{(l)}_{t_l}||} \right), \quad l = 1, ..., m. \tag{2.37}$$

This equation shows that the parameter γ and the intermediate points of the reference trajectory must be chosen such that $\gamma \leq ||\vec{E}^{(l)}_{t_l}||$ for each subsystem. Otherwise, $t_l < t_{l-1}$, which implies the stability of the system backward in time, and consequently, its instability forward in time.

2.1.4 Simulation Results

In this section, we present and discuss the performance of the proposed controllers in simulation. We implemented the position regulation controller in Eq. (2.3) for the rigid-link equivalent of a 5-segment continuum robot, and we implemented the trajectory tracking controller in Eq. (2.8) for the rigid-link equivalent of a 15-segment continuum robot. We set $\gamma = 0.01$ cm in all simulations.

The position regulation control problem was simulated for two different values of the controller gain K. From Eq. (2.26), it is evident that the gain K determines the rate

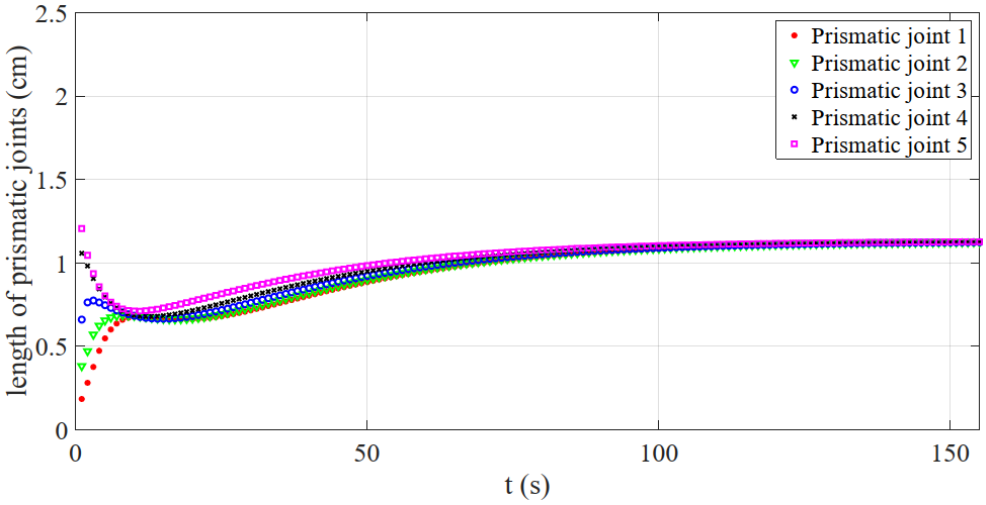


Figure 2.4: Length of Prismatic Joints Over Time for a Simulated 5-Link Serial Robot, Equivalent to a 5-Segment Soft Continuum Robot, that is Controlled by the Position Regulator (2.3) with Gain $K = 0.0382$.

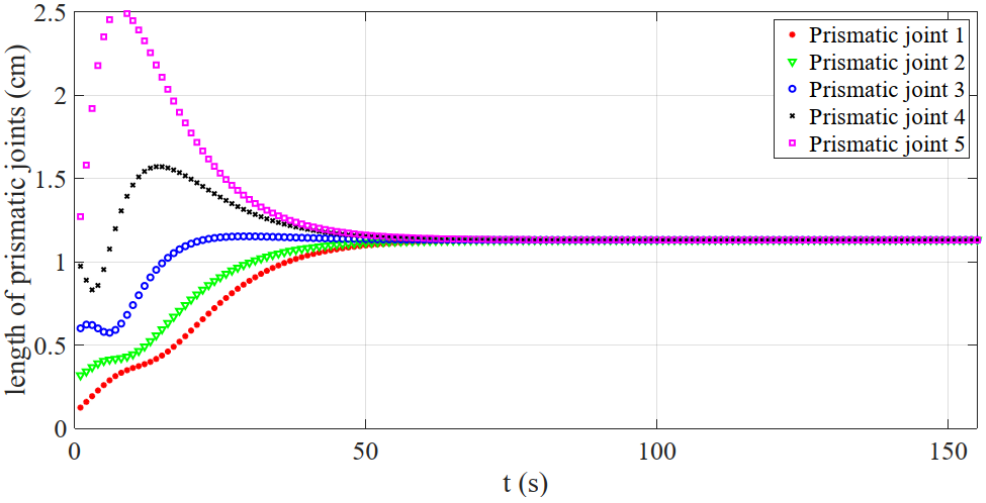


Figure 2.5: Length of Prismatic Joints Over Time for a Simulated 5-Link Serial Robot, Equivalent to a 5-Segment Soft Continuum Robot, that is Controlled by the Position Regulator (2.3) with Gain $K = 0.764$.

at which the position of the robot's tip converges to the target position. Equation (2.28) shows that this gain also affects the convergence rate of consensus among the ${}^{G}\vec{p}_i$ vectors.

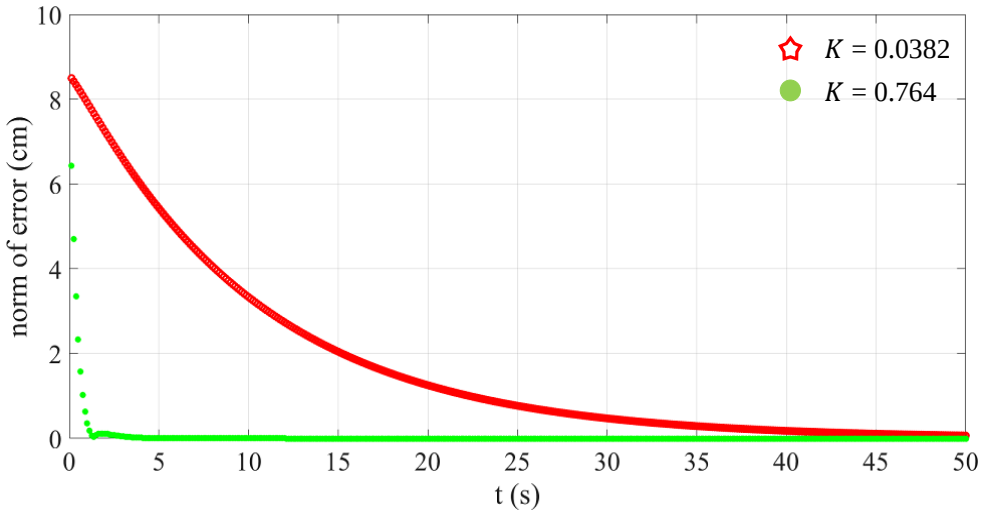


Figure 2.6: Plots of Tracking Error $||\vec{E}(t)||_2$ Over Time During the Position Regulator Simulations. The Gain K = 0.764 Results in Faster Convergence of the Robot's Tip to the Target Point than the Gain K = 0.0382.

Moreover, we know that the smallest positive eigenvalue of the Laplacian matrix, $\lambda_2(\vec{L})$, determines the convergence rate of a consensus system (Olfati-Saber *et al.*, 2007). We consider a 5-segment continuum robot for which $\lambda_2(\vec{L}) = 0.382$, and we simulate its motion under the position regulation controller for cases where $K > \lambda_2(\vec{L})$ and $K < \lambda_2(\vec{L})$.

In the first simulation of the position regulator, we set $K = 0.0382$. Figure 2.4 shows that the convergence rate of consensus among the lengths of the five prismatic joints to their steady-state value is relatively slow. Since $K < \lambda_2(\vec{L})$ in this case, K governs the asymptotic convergence rate of the consensus dynamics. The rate of convergence of the tracking error to zero is also determined by the value of K, and the red plot in Fig. 2.6 confirms that the robot's tip converges slowly to the target position. Figure 2.5 plots snapshots of the robot's configuration over time and illustrates that the links of the robot undergo similar changes in their lengths and orientations as the robot's tip gradually approaches the target point.

In the second simulation of the position regulator, we set $K = 0.764$. The green plot

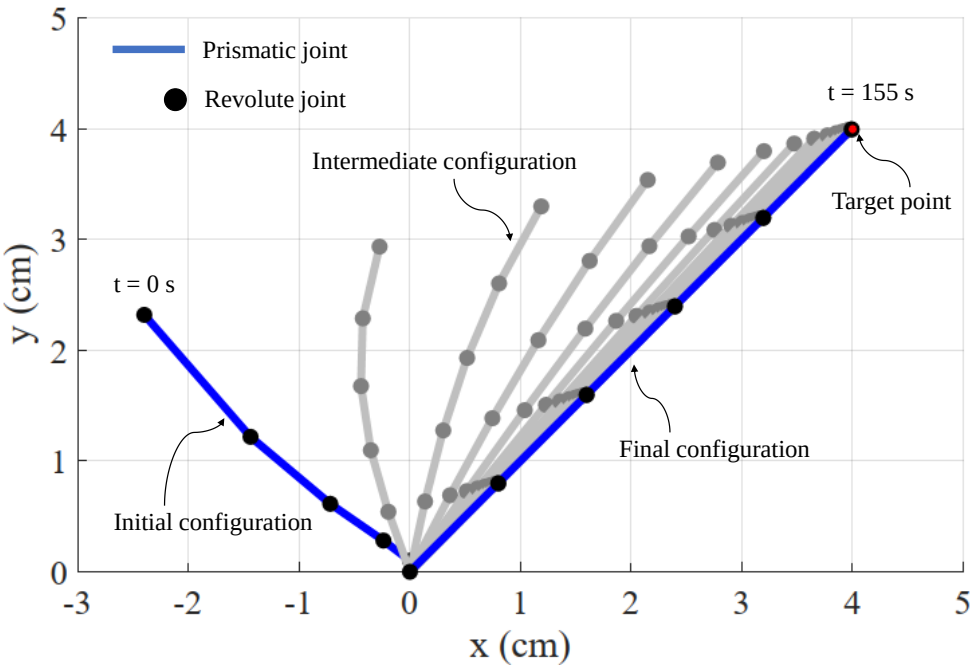


Figure 2.7: Configuration Over Time of the Simulated 5-Link Robot when Controlled by the Position Regulator (2.3) with Gain $K = 0.0382$. The Robot Reconfigures from its Initial Configuration to Approach the Target Point with its Tip. Intermediate Configurations are Shown in Gray.

in Fig. 2.6 shows that the robot's tip converges very quickly to the target position. Since $\lambda_2(\vec{L}) < K$ in this case, $\lambda_2(\vec{L}) = 0.382$ governs the asymptotic convergence rate of the consensus dynamics. As Fig. 2.5 shows, the fact that $\lambda_2(\vec{L}) > 0.0382$ causes the prismatic joints to converge to their common steady-state value more quickly than in the first simulation, in which $K = 0.0382$. However, since $K = 0.764$ governs the convergence rate of the tracking error and $\lambda_2(\vec{L}) < 0.764$, the tracking error in Fig. 2.6 converges at a faster rate than the consensus dynamics in Fig. 2.5. As a result, the lengths of the prismatic joints are still changing significantly, two with large overshoots (see Fig. 2.5), even after the robot's tip enters the γ-neighborhood of the target position. The snapshots in Fig. 2.8 illustrate the disparities in the links' lengths and orientations during the robot's quick reconfiguration to

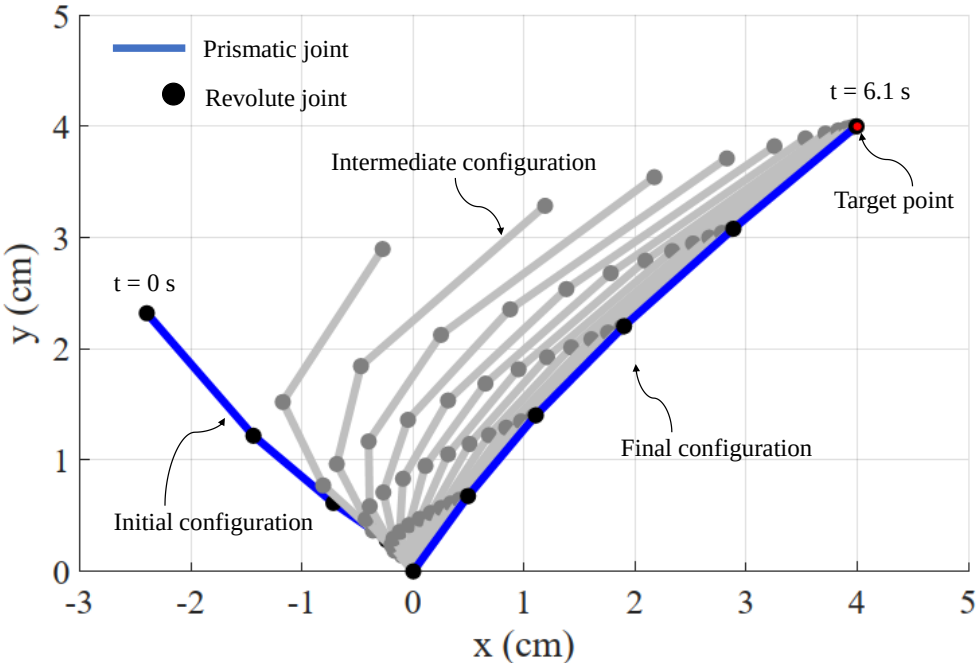


Figure 2.8: Configuration Over Time of the Simulated 5-Link Robot when Controlled by the Position Regulator (2.3) with Gain $K = 0.764$. The Robot Reconfigures from its Initial Configuration to Approach the Target Point with its Tip. Intermediate Configurations are Shown in Gray.

reach the target point. The fifth link undergoes a large elongation, which could exceed its prismatic joint limit in practice.

The simulation results for the position regulator suggest qualitative guidelines for selecting the gain K in the controller. K should be sufficiently large to drive the robot's tip quickly to the target point, but not much larger than $\lambda_2(\vec{L})$ if it is important to maintain fairly consistent changes in all link lengths and orientations throughout the robot's reconfiguration. For the trajectory tracking problem, we simulated a 15-link robot and defined the reference trajectory as the sequence of letters "ASU." Figure 2.9 shows snapshots of the robot's configuration over time as its tip tracks a sequence of points defined along the reference trajectory. We evaluated the trajectory-tracking performance of the robot by

computing the average root-mean-square error (RMSE) between the reference trajectory and the trajectory of the robot's tip. This value is 0.024 cm, which is 0.16% of the initial length of the robot (15 cm). This very low error demonstrates the effectiveness of the control strategy.

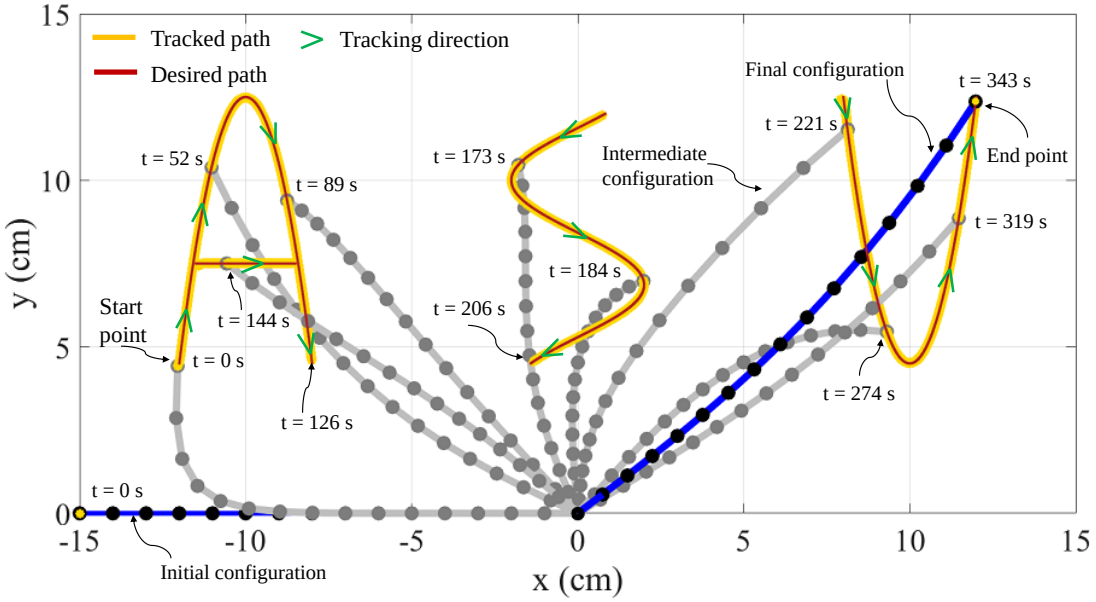


Figure 2.9: Configuration Over Time of a Simulated 15-Link Robot, Equivalent to a 15-Segment Soft Continuum Robot, when Controlled by the Trajectory Tracking Controller (2.8) with Gain $K = 0.25$. The Robot Reconfigures such that its Tip Moves Between a Series of Points Defined Along a Trajectory that Spells ASU. Intermediate Configurations and Their Corresponding Times Are Shown in Gray. The Average RMSE of Tracking is 0.024 cm.

2.1.5 Discussion

We proposed a decentralized approach to kinematic control of soft segmented continuum robots that deform within a plane. The kinematics of the soft segments comply with the CC condition assumption, which enables us to model each segment as an equivalent

rigid-link RPR mechanism. Decentralized controllers were defined for position regulation and trajectory tracking objectives, utilizing a consensus protocol in which adjacent segments share local measurements of changes in the length and orientation of their equivalent rigid link. The controllers were validated with simulations of 5-link and 15-link robots, and the effect of the controller gain was investigated.

2.2 Trajectory Tracking Control of 3D Multi-Segment Manipulators

In this section, we address the kinematic modeling and control of hyper-redundant robots inspired by the octopus arm. We propose a discrete multi-segment model in which each segment is a 6-DoF Gough-Stewart parallel platform. Our model is novel in that it can reproduce all generic motions of an octopus arm, including elongation, shortening, bending, and particularly twisting, which is usually not included in such models, while enforcing the constant-volume property of the octopus arm. We use an approach that is inspired by the unique decentralized nervous system of the octopus arm to overcome challenges in solving the Inverse Kinematics (IK) problem, including the large number of solutions for this problem and the impracticality of numerical methods for real-time applications. We apply the pseudo-inverse Jacobian method to design a kinematic controller that drives the tip of the hyper-redundant robot to track a reference trajectory. We evaluate our proposed model and controller in simulation for a variety of 3D reference trajectories: a straight line, an ellipse, a sinusoidal path, and trajectories that emulate octopus-like reaching and fetching movements. The tip of the simulated hyper-redundant robot tracks the reference trajectories with average root-mean-square errors that are less than 0.3% of the robot's initial length, demonstrating the effectiveness of our modeling and control approaches.

2.2.1 Kinematic Model of Hyper-Redundant Robot

In this section, we derive the kinematic model of an octopus-inspired multi-segment hyper-redundant robot in which each segment is a 6-DoF Gough-Stewart (GS) platform as described in (Salimi Lafmejani *et al.*, 2018). We first derive the kinematic model of a single-segment 6-DoF GS platform without considering the active prismatic joints on its base and end-effector. Then, we include a model of the radial prismatic joints as kinematic constraints that maintain the isovolumetric property. As in (Salimi Lafmejani *et al.*, 2018), we represent the kinematics of the entire hyper-redundant robot by generalizing the kinematic equations of a single-segment 6-DoF GS platform to a multi-segment hyper-redundant robot. Moreover, we describe the FK and IK problems for our proposed model of the multi-segment hyper-redundant robot.

Gough-Stewart Platform Model of Robot

We model each segment of the hyper-redundant robot as a 6-DoF U<u>P</u>S Gough-Stewart (GS) platform (Salimi Lafmejani *et al.*, 2018), shown in Fig. 2.10. This 6-DoF parallel platform consists of twelve active prismatic joints, six passive universal joints, and six passive spherical joints. Six longitudinal prismatic joints connect the universal joints on the base platform to the spherical joints on the end-effector. The radial prismatic joints at the base and the end-effector of each segment are rotated 120° with respect to the local coordinate frame, as shown in Fig. 2.10.

The 6-DoF GS platform in Fig. 2.10 can generate all possible reconfigurations of each segment of the hyper-redundant robot (Salimi Lafmejani *et al.*, 2017; Sharifzadeh *et al.*, 2018); i.e., it can produce three rotations around the segment's X_a, Y_a, and Z_a axes and three translations along these axes. Pure elongations and contractions of the segment along the Z_a axis are generated when all six longitudinal prismatic joints lengthen or shorten

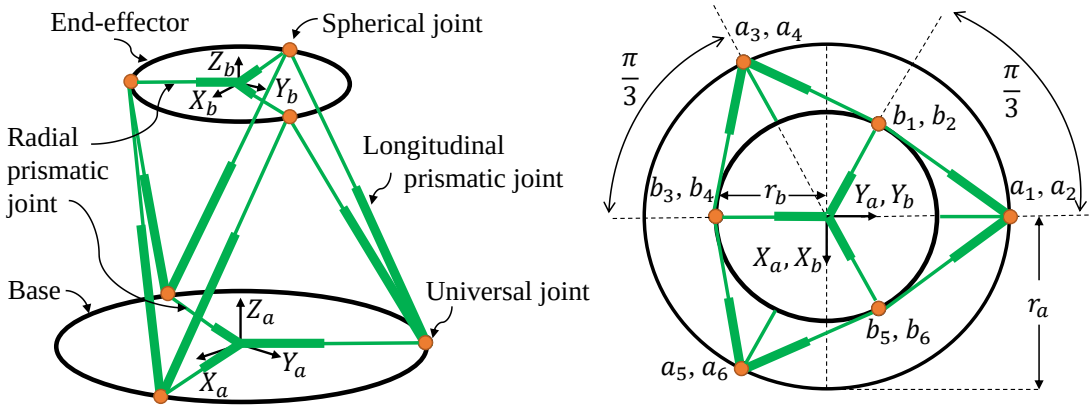


Figure 2.10: Model of a Single Segment of the Hyper-Redundant Robot Based on a UPS 6-DoF Gough-Stewart (GS) Platform.

simultaneously by the same amount. If these lengths change by different amounts, then the segment's end-effector rotates, causing the segment to bend. As a special case, torsional motion about the segment's Z_a axis is produced when every other longitudinal prismatic joint lengthens by the same amount, while the other longitudinal prismatic joints shorten by this amount.

Since the single-segment model captures all possible rotations and translations of an individual segment, the model of the entire multi-segment robot can reproduce four fundamental motions of an octopus arm: bending, elongation, shortening, and twisting (Kier, 2016). Figure 2.11 illustrates a model of the hyper-redundant robot as a series of identical interconnected segments, each modeled as the 6-DoF GS platform shown in Fig. 2.10. The base platform of the i-th segment is the end-effector of the $(i-1)$-th segment. We make the following assumptions about the capabilities of the segments: **(1)** Each segment can measure the 3D pose of its end-effector with respect to its local coordinate frame, which is fixed to its base. **(2)** Each segment can transmit these measurements to its two adjacent segments, similar to the propagation of sensorimotor information through the decentralized nervous system of an octopus arm (Grasso, 2014). **(3)** The 3D pose of the robot's tip with

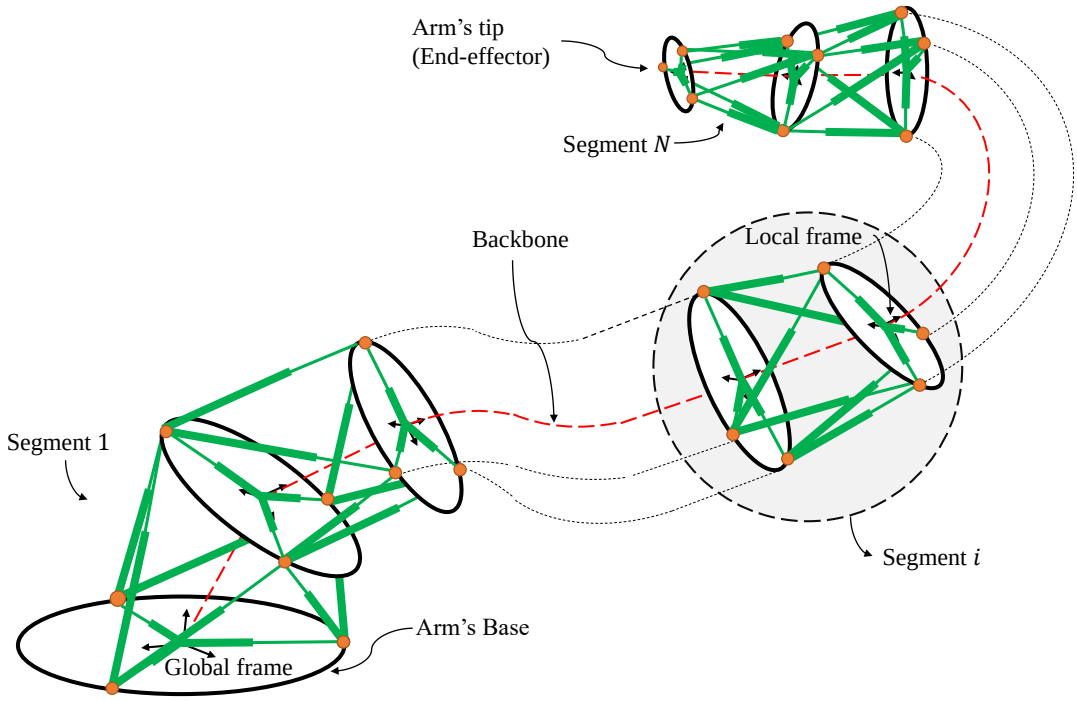


Figure 2.11: Multi-Segment Model of an Octopus-Inspired Hyper-Redundant Robot Composed of a Series of 6-DoF GS Platforms.

respect to the global coordinate frame can be measured, analogous to the use of vision feedback in the octopus (Levy and Hochner, 2017).

Let k denote the index of the joints and $i = 1, 2, ..., N$ be the indices of the segments, where the end-effector of the N-th segment is the distal tip of the robot. We define ${}^{G}\vec{p}_i \in \mathbb{R}^3$ as the position of the end-effector of the i-th segment in the global coordinate frame G. The rotation matrix ${}^{G}\mathbf{R}_i \in \mathbb{R}^{3\times3}$ determines the orientation of the i-th segment's end-effector in the global frame. The vector ${}^{G}\vec{a}_{k,i} \in \mathbb{R}^3$ denotes the position of the k-th universal joint on the base of i-th segment in the global frame, and ${}^{i}\vec{b}_{k,i} \in \mathbb{R}^3$ denotes the position of the k-th spherical joint on the end-effector of the i-th segment in the local coordinate frame of the i-th segment. The vector ${}^{G}\vec{l}_{k,i} \in \mathbb{R}^3$ is the position of the k-th longitudinal prismatic

joint of the i-th segment in the global frame. The position vector ${}^{G}\vec{p}_i$ is calculated as:

$$ {}^{G}\vec{p}_i = {}^{G}\vec{a}_{k,i} + {}^{G}\vec{l}_{k,i} - {}^{G}\mathbf{R}_i {}^{i}\vec{b}_{k,i}. \tag{2.38} $$

The pose of each segment in the global frame can be determined by the corresponding position vector ${}^{G}\vec{p}_i$ and rotation matrix ${}^{G}\mathbf{R}_i$. The position vector ${}^{i-1}\vec{p}_i$ and rotation matrix ${}^{i-1}\mathbf{R}_i$ for the i-th segment in the coordinate frame attached to the end-effector of segment $i-1$ are defined as:

$$ \begin{aligned} {}^{i-1}\vec{p}_i &= {}^{G}\mathbf{R}_{i-1}^{-1} {}^{G}\vec{p}_i \\ {}^{i-1}\mathbf{R}_i &= {}^{G}\mathbf{R}_{i-1}^{-1} {}^{G}\mathbf{R}_i, \end{aligned} \tag{2.39} $$

where ${}^{G}\mathbf{R}_{i-1}^{-1}$ is the inverse of the rotation matrix ${}^{G}\mathbf{R}_{i-1}$. As in (Salimi Lafmejani *et al.*, 2018), by rearranging Eq. (2.38) to solve for ${}^{G}\vec{l}_{k,i}$, using Eq. (2.39) to substitute in the position vector and rotation matrix that are defined in local coordinate frames, and right-multiplying the left and right sides of the resulting equation by their respective transposes, we can obtain the lengths of the longitudinal prismatic joints, $l_{k,i}$, from the following equation:

$$ \begin{aligned} &{}^{G}\vec{l}_{k,i}^{\mathrm{T}} {}^{G}\vec{l}_{k,i} = l_{k,i}^2 \\ &= ({}^{i-1}\vec{p}_i + {}^{i-1}\vec{b}_{k,i}^{\mathrm{T}} {}^{i-1}\vec{b}_{k,i} + {}^{i-1}\vec{a}_{k,i}^{\mathrm{T}} {}^{i-1}\vec{a}_{k,i} - 2^{i-1}\vec{p}_i^{\mathrm{T}} \\ &\quad {}^{i-1}\vec{a}_{k,i} + 2^{i-1}\vec{p}_i^{\mathrm{T}} {}^{i-1}\vec{b}_{k,i} - 2^{i-1}\vec{b}_{k,i}^{\mathrm{T}} {}^{i-1}\vec{a}_{k,i})^{1/2}. \end{aligned} \tag{2.40} $$

Let r_{ai} and r_{bi} denote the radii of the base and end-effector, respectively, of the i-th segment. Let ${}^{i}\mathbf{A}$ and ${}^{i}\mathbf{B}$ denote matrices whose columns are the position vectors ${}^{i-1}\vec{a}_{k,i}$ and ${}^{i-1}\vec{b}_{k,i}$ represented in the frame attached to the end-effector of the $(i-1)$-th segment.

These matrices are defined as:

$$
\begin{aligned}
{}^{i}\mathbf{A} = {}^{i-1}\mathbf{B} &= r_{ai} \begin{bmatrix} -\frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 & 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & -1 & -1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
{}^{i}\mathbf{B} = {}^{i+1}\mathbf{A} &= r_{bi} \begin{bmatrix} 0 & 0 & -\frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & 1 & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.
\end{aligned}
\tag{2.41}
$$

We define α as the ratio between the radius of the i-th segment's base, r_{ai}, and the radius of its end-effector, r_{bi}:

$$
\alpha = \frac{r_{bi}}{r_{ai}}, \quad \alpha \in (0\ 1]. \tag{2.42}
$$

In order to enforce the isovolumetric property in the kinematic model of the robot, the lengths of the radial prismatic joints are determined according to the lengths of the longitudinal prismatic joints so as to keep the volume of each segment constant. Since the total volume of the robot is the sum of all the segment volumes, the total robot volume remains constant as well. The volume of the i-th segment, denoted by s_i, can be computed as:

$$
s_i \ = \ \frac{\pi}{3} || {}^{i-1}\vec{p}_i || r_{ia}^2 (1 + \alpha + \alpha^2). \tag{2.43}
$$

To keep the volume s_i of the i-th segment constant, the radius of its base should be:

$$
r_{ia} = \frac{1}{\sqrt{\pi}} \left[\frac{3 s_i}{|| {}^{i-1}\vec{p}_i || (1 + \alpha + \alpha^2)} \right]^{\frac{1}{2}}. \tag{2.44}
$$

Forward and Inverse Kinematic Problems

Here, we discuss possible solutions to the FK and IK problems for the proposed multi-segment hyper-redundant robot. The transformation matrix of the robot's tip, i.e., the N-th segment's end-effector, represented in the global frame, ${}^{G}\mathbf{T}_N \in \mathbb{R}^{4\times4}$, can be calculated

by consecutive post-multiplication of the transformation matrix of each segment's end-effector, ${}^{i-1}\mathbf{T}_i$, represented in the local frame attached to the end-effector of the segment $i-1$:

$$ {}^{G}\mathbf{T}_N = {}^{G}\mathbf{T}_1 \cdot {}^{1}\mathbf{T}_2 \cdot \ \ldots \ \cdot {}^{N-1}\mathbf{T}_N. \tag{2.45} $$

The solution of the FK problem is straightforward to compute if each segment i is able to measure its local transformation matrix, ${}^{i-1}\mathbf{T}_i$, as we specified in assumption (**1**). In other words, the pose of the robot's tip can be computed from the pose of each segment's end-effector, expressed in the segment's local coordinate frame.

The IK problem for the robot is defined as finding the configuration of the entire robot, i.e., all transformation matrices on the right-hand side of Eq. (2.45), given the pose of the robot's tip, ${}^{G}\mathbf{T}_N$. This problem has a large number of solutions due to the kinematic hyper-redundancy of the robot. Different methods of solving the IK problem for a multi-segment hyper-redundant robot are discussed in (Godage *et al.*, 2011). These approaches utilize computationally intensive numerical methods and can converge to physically impossible solutions. Moreover, they require simplifying assumptions and approximations that preclude the reproduction of some important features of the octopus arm. We adopt the Distributed Inverse Kinematics (DIK) approach presented in (McEvoy and Correll, 2018, 2016) in order to deal with the challenge of solving the IK problem and compute the configuration of the robot, given the pose of the robot's tip. In (McEvoy and Correll, 2018), the authors introduce this approach for a multi-segment planar robot composed of segments that can autonomously change their curvature using local computation, sensing, actuation, and communication. Using this method, the solution to the IK problem is determined by utilizing the communication between the segments to share the transformation matrices and curvatures between neighboring segments of the robot.

2.2.2 Kinematic Control of Robot Motion

In this section, we describe the calculation of the Jacobian matrix of our multi-segment hyper-redundant robot and propose a differential kinematic control approach for the robot based on the pseudo-inverse Jacobian method (Hannan and Walker, 2003). The objective is to design controllers for regulation (position control) and trajectory tracking problems. While we do not impose limits on the joint torques here, such constraints could be included in the robot control strategy as a second prioritized task (Dietrich and Ott, 2020).

Jacobian Matrix of the Robot

From assumptions **(1)** and **(2)** in Section 2.2.1, the i-th segment can measure its position vector ${}^{i-1}\vec{p}_i$ and rotation matrix ${}^{i-1}\mathbf{R}_i$, both represented in its local coordinate frame, and can communicate these measurements to its neighboring segments, segments $i-1$ and $i+1$. We define the pose of the robot's N-th segment as $\vec{X} := [{}^{G}\vec{p}_{tip}^{T} \;\; \vec{\delta}^{T}]^T \in \mathbb{R}^6$, where ${}^{G}\vec{p}_{tip} \in \mathbb{R}^3$ is the position of the robot's tip in the global frame, and $\vec{\delta} = [\psi \;\; \theta \;\; \phi]^T \in \mathbb{R}^3$ is the vector of Euler angles representing the orientation of the N-th segment. We also denote the joint space of the robot by $\vec{Q} \in \mathbb{R}^{6N}$, which is defined as

$$\vec{Q} = [l_{1,1} \;\ldots\; l_{6,1}\, l_{1,2} \;\ldots\; l_{6,2} \;\ldots\; \ldots\; l_{1,N} \;\ldots\; l_{6,N}]^T. \tag{2.46}$$

The twist of the robot's tip, $\dot{\vec{X}}_{\text{tip}}$, is mapped to the velocities of all active prismatic joints, $\dot{\vec{Q}}$, by the Jacobian matrix of the entire robot, $\mathbf{J}_{\text{robot}} \in \mathbb{R}^{6\times 6N}$:

$$\dot{\vec{X}}_{\text{tip}} = \mathbf{J}_{\text{robot}}\dot{\vec{Q}}. \tag{2.47}$$

The Jacobian matrix $\mathbf{J}_{\text{robot}}$ can be calculated as follows. Let $\mathbf{J}_i \in \mathbb{R}^{6\times 6}$, $i = 1, 2, ..., N$, denote the Jacobian matrix and ${}^{i-1}\vec{s}_{k,i}$ denote the unit vector that indicates the direction of the prismatic joints of the i-th segment in its local coordinate frame. As in (Salimi Lafmejani

et al., 2018), this matrix is computed as

$$\mathbf{J}_i = \begin{bmatrix} {}^{i-1}\vec{s}_{k,i} & {}^{i-1}\vec{b}_{k,i} \times {}^{i-1}\vec{s}_{k,i} \end{bmatrix}^{-\mathrm{T}}. \tag{2.48}$$

We define ${}^{G}\widehat{\vec{p}_i} \in \mathbb{R}^{3\times3}$ as the cross product matrix of the position vector ${}^{G}\vec{p}_i$. The adjoint matrix for the i-th segment, $\mathbf{Ad}_{\mathrm{T}_i} \in \mathbb{R}^{6\times6}$, is then given by (Murray, 2017)

$$\mathbf{Ad}_{\mathrm{T}_i} = \begin{bmatrix} {}^{G}\mathbf{R}_i & {}^{G}\widehat{\vec{p}_i}\,{}^{G}\mathbf{R}_i \\ \mathbf{0}_{3\times3} & {}^{G}\mathbf{R}_i \end{bmatrix}. \tag{2.49}$$

Given these matrices, the Jacobian matrix of the robot can be written as

$$\mathbf{J}_{\text{robot}} = \begin{bmatrix} \mathbf{J}_1 & \mathbf{Ad}_{\mathrm{T}_1}\mathbf{J}_2 & \ldots & \mathbf{Ad}_{\mathrm{T}_{N-1}}\mathbf{J}_N \end{bmatrix}. \tag{2.50}$$

As Eq. (2.49) shows, ${}^{G}\mathbf{R}_i$ and ${}^{G}\vec{p}_i$ are required to calculate the corresponding adjoint matrix. This information is contained in the transformation matrix of each segment i in the global coordinate frame, defined as

$${}^{G}\mathbf{T}_i = \begin{bmatrix} {}^{G}\mathbf{R}_i & {}^{G}\vec{p}_i \\ \vec{0}_{1\times3} & 1 \end{bmatrix}. \tag{2.51}$$

From assumption **(3)** in Section 2.2.1, the transformation matrix of the robot's tip in the global frame, ${}^{G}\mathbf{T}_{\text{tip}}$, is known. Since each segment i can communicate its measurements of ${}^{i-1}\vec{p}_i$ and ${}^{i-1}\mathbf{R}_i$ to its neighboring segments, as ensured by assumption (**2**), the transformation matrix ${}^{i-1}\mathbf{T}_i$ can be computed by each segment i. Thus, the robot can calculate the transformation matrix of each segment in the global frame as follows:

$${}^{G}\mathbf{T}_{i-1} = {}^{G}\mathbf{T}_i \cdot {}^{i-1}\mathbf{T}_i^{-1}. \tag{2.52}$$

Therefore, the transformation matrix corresponding to each segment in the global coordinate frame can be found by first solving Eq. (2.52) for segment $N-1$, then for segment $N-2$, and so on until the transformation matrix of each segment from the tip to the base of the robot is computed. Consequently, we can readily calculate the Jacobian matrix of the robot in Eq. (2.50).

Position Control

Here, we design a regulator that stabilizes the position of the tip of the robot to a fixed point in the global frame G. We define the error between the position of the tip, ${}^{G}\vec{p}_{\text{tip}}$, and the desired point, ${}^{G}\vec{p}_{\text{d}}$, as

$$\vec{E} = {}^{G}\vec{p}_{\text{tip}} - {}^{G}\vec{p}_{\text{d}}. \tag{2.53}$$

We decompose $\mathbf{J}_{\text{robot}}$ into upper and lower halves, which are denoted by $\mathbf{J}_v \in \mathbb{R}^{3\times 6N}$ and $\mathbf{J}_\omega \in \mathbb{R}^{3\times 6N}$, respectively. The former matrix is associated with the linear velocity of the robot's tip (${}^{G}\dot{\vec{p}}_{\text{tip}} = \mathbf{J}_v\dot{\vec{Q}}$), and the latter matrix is associated with the angular velocity of the N-th segment ($\dot{\vec{\delta}} = \mathbf{J}_\omega\dot{\vec{Q}}$). Defining $K \in \mathbb{R}_{>0}$ as a positive gain and $\mathbf{J}_v^\dagger \in \mathbb{R}^{6N\times 3}$ as the Moore-Penrose inverse of $\mathbf{J}_v$, we can design the control law for the position regulation problem as:

$$\dot{\vec{Q}} = -K\mathbf{J}_v^\dagger\vec{E}. \tag{2.54}$$

Substituting the control law in Eq. (2.54) into the kinematics of the robot in Eq. (2.47), and considering the fact that ${}^{G}\dot{\vec{p}}_{\text{d}} = \vec{0}$, the equation of the closed-loop system is obtained as

$$\dot{\vec{E}} = -K\mathbf{J}_v\mathbf{J}_v^\dagger\vec{E}. \tag{2.55}$$

We can confirm that the null space of $\mathbf{J}_v^\dagger$ is empty. Thus, the only solution for the equation $\mathbf{J}_v^\dagger\vec{E} = \vec{0}$ is $\vec{E} = \vec{0}$, which shows that the origin is the only equilibrium point for Eq. (2.55). We can now use the identity $\mathbf{J}_v\mathbf{J}_v^\dagger = \mathbf{I}$ and rewrite the closed-loop system in Eq. (2.55) as

$$\dot{\vec{E}} + K\vec{E} = \vec{0}, \tag{2.56}$$

which is a globally exponentially stable system for any positive K. This shows that the tip of the robot exponentially converges to the desired point from any initial position with the proposed controller. The controller gain K governs the rate of convergence of the system's trajectories to the desired position. The proposed controller drives the tip of the robot to the target point faster with larger values of K.

Trajectory Tracking

We use the position regulator (2.54) in a switching strategy that controls the robot to track a predefined reference trajectory with its tip. The reference trajectory is discretized into a set of m points. The objective is to drive the tip of the robot to a position within a radius $\gamma \in \mathbb{R}_{>0}$ of each point, which represents the acceptable tracking error. The set of the desired points is denoted by $\mathcal{P}$ and is defined as

$$\mathcal{P} = \left\{ {}^{G}\vec{p}_d^{(1)}, {}^{G}\vec{p}_d^{(2)}, \ldots, {}^{G}\vec{p}_d^{(m)} \right\}, \tag{2.57}$$

where ${}^{G}\vec{p}_d^{(1)}$ and ${}^{G}\vec{p}_d^{(m)}$ represent the start and the end points of the reference trajectory, respectively, and the others are intermediate points along the trajectory. A γ-neighborhood of point ${}^{G}\vec{p}_d^{(i)}$ is defined as a ball of radius γ centered at this point and is denoted by $\mathcal{B}({}^{G}\vec{p}_d^{(i)}, \gamma)$. The controller first stabilizes the position of the robot's tip to the start point of the trajectory. Once the tip enters the ball $\mathcal{B}({}^{G}\vec{p}_d^{(1)}, \gamma)$, the controller redefines the desired point as the second point, ${}^{G}\vec{p}_d^{(2)}$, and drives the tip toward this point until it enters the ball $\mathcal{B}({}^{G}\vec{p}_d^{(2)}, \gamma)$. This procedure is repeated for each successive point in $\mathcal{P}$ until the robot's tip enters the γ-neighborhood of the end point, ${}^{G}\vec{p}_d^{(m)}$. This switching control strategy can be written as:

$$\dot{\vec{Q}} = \begin{cases} -K\mathbf{J}_v^{\dagger}\left({}^{G}\vec{p}_{\text{tip}} - {}^{G}\vec{p}_d^{(i)}\right), & {}^{G}\vec{p}_{\text{tip}} \notin \mathcal{B}({}^{G}\vec{p}_d^{(i)}, \gamma) \\ -K\mathbf{J}_v^{\dagger}\left({}^{G}\vec{p}_{\text{tip}} - {}^{G}\vec{p}_d^{(i+1)}\right), & {}^{G}\vec{p}_{\text{tip}} \in \mathcal{B}({}^{G}\vec{p}_d^{(i)}, \gamma). \end{cases} \tag{2.58}$$

Thus, the closed-loop system behaves like a switching system in which each subsystem is given by

$$\dot{\vec{E}}^{(i)} + K\vec{E}^{(i)} = \vec{0}, \tag{2.59}$$

where $\vec{E}^{(i)} = {}^{G}\vec{p}_{\text{tip}} - {}^{G}\vec{p}_d^{(i)}$ is the position error associated with the i-th subsystem. Equation (2.59) is linear and can be solved for $\vec{E}^{(i)}$ as

$$\vec{E}^{(i)}(t) = \vec{E}_{t_i}^{(i)} e^{-K(t-t_i)}, \quad \forall t \in [t_i, t_{i+1}), \tag{2.60}$$

where t_i is the time at which subsystem i becomes active, and $\vec{E}_{t_i}^{(i)}$ denotes the value of $\vec{E}^{(i)}(t)$ at time $t = t_i$. This solution converges exponentially to the desired equilibrium ($\vec{E}^{(i)} = \vec{0}$), which consequently shows that the robot's tip will converge in finite time to a neighborhood around the desired point ${}^G\vec{p}_d^{(i)}$ and that the trajectories of the tip are bounded. Equation (2.60) also holds for the norm of the error:

$$||\vec{E}^{(i)}(t)|| = ||\vec{E}_{t_i}^{(i)}||e^{-K(t-t_i)}, \quad \forall t \in [t_i, t_{i+1}). \tag{2.61}$$

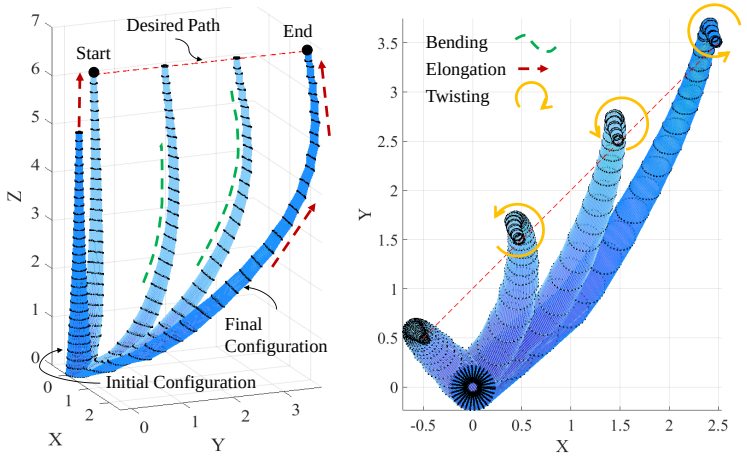
We define the *dwell time* for the i-th subsystem to reach $\mathcal{B}({}^G\vec{p}_d^{(i)}, \gamma)$ as $T_i := t_i - t_{i-1}$ (Liberzon, 2003). The dwell time can be calculated by setting $||\vec{E}^{(i)}(t)||$ equal to γ in Eq. (2.61) and solving this equation for T_i:

$$T_i = -\frac{1}{K}\log\left(\frac{\gamma}{||\vec{E}_{t_i}^{(i)}||}\right), \quad i = 1, ..., m. \tag{2.62}$$

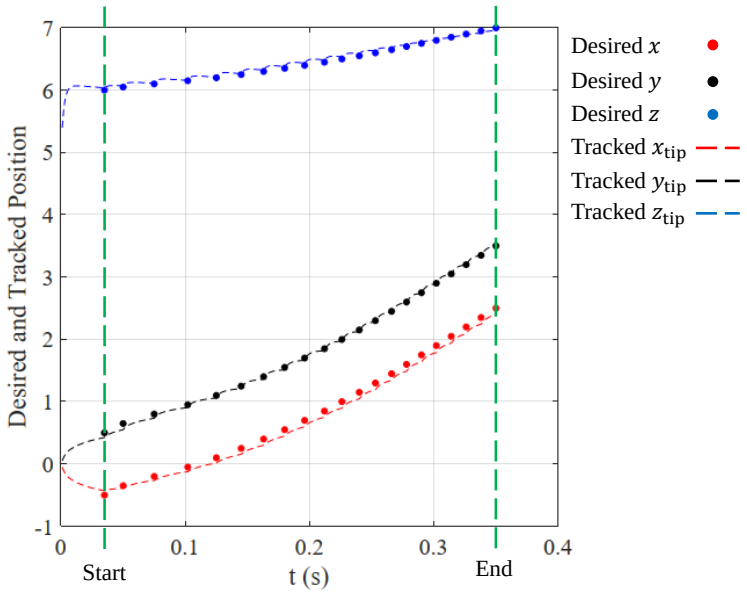
This equation shows that the parameter γ and the intermediate points of the reference trajectory must be chosen such that $\gamma \leq ||\vec{E}_{t_i}^{(i)}||$ for each subsystem. Otherwise, t_i would be smaller than t_{i-1}, which would imply the stability of the system backward in time, and consequently, its instability forward in time. As in the position control task, the controller gain K determines the rate of the error convergence to zero for each subsystem. Moreover, since $\sum_{i=1}^{N} T_i$ is equal to the task completion time, and each time T_i is inversely proportional to K, the gain K can be selected to ensure that the robot's tip tracks a desired trajectory within a specified amount of time. An analytical procedure could be developed to select a value of K that guarantees particular time response characteristics for a given desired trajectory based on the parameters N, m, α, and γ, although this is beyond the scope of this research.

2.2.3 Simulation Results

We validated our kinematic model and controller through MATLAB simulations of a multi-segment hyper-redundant robot that tracks five 3D reference trajectories with the tip



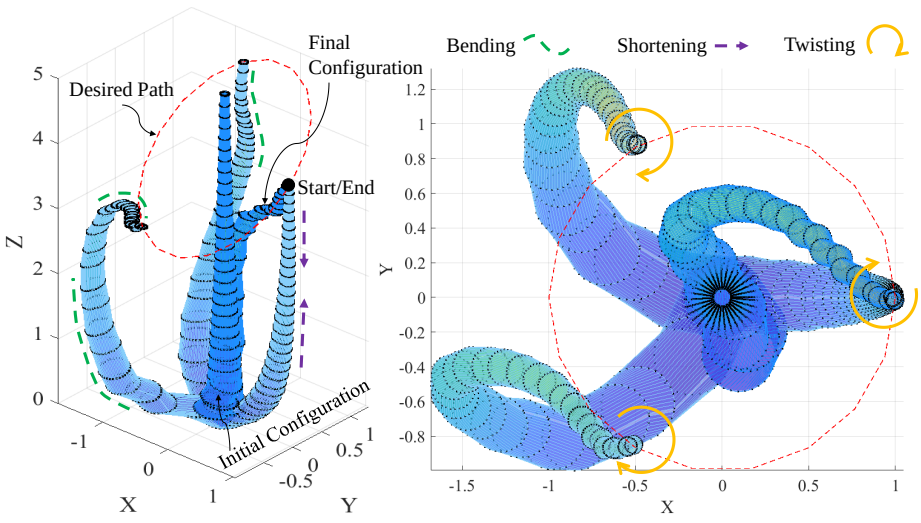
(a)



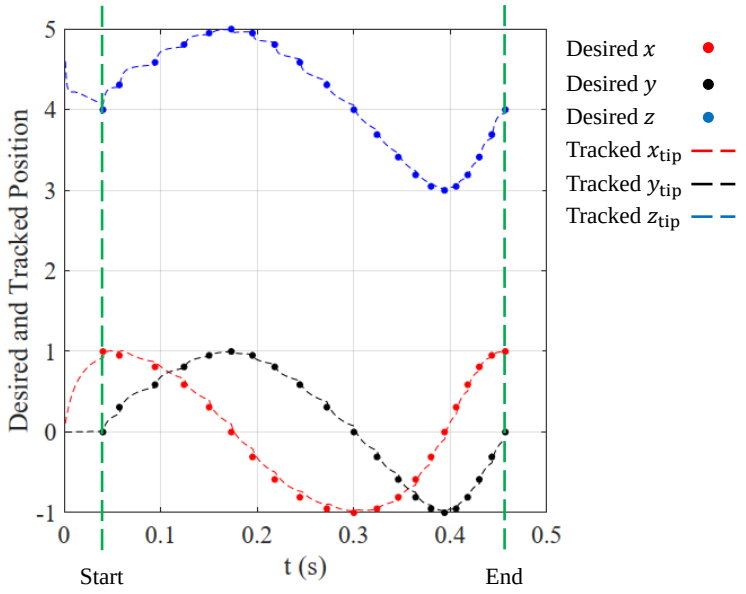
(b)

Figure 2.12: (a) Snapshots of the Simulated Hyper-Redundant Robot Tracking the Linear Trajectory Eq. (2.63). Average Tracking RMSE = 0.032 cm. (b) Time Evolution of the Desired Position (x,y,z) and Tracked Position $(x_{\text{tip}},y_{\text{tip}},z_{\text{tip}})$ of the Robot's Tip.

of its arm. The first three reference trajectories are defined as a straight line, an ellipse, and a sinusoidal. These three trajectories are given by the following three parametric equations,



(a)

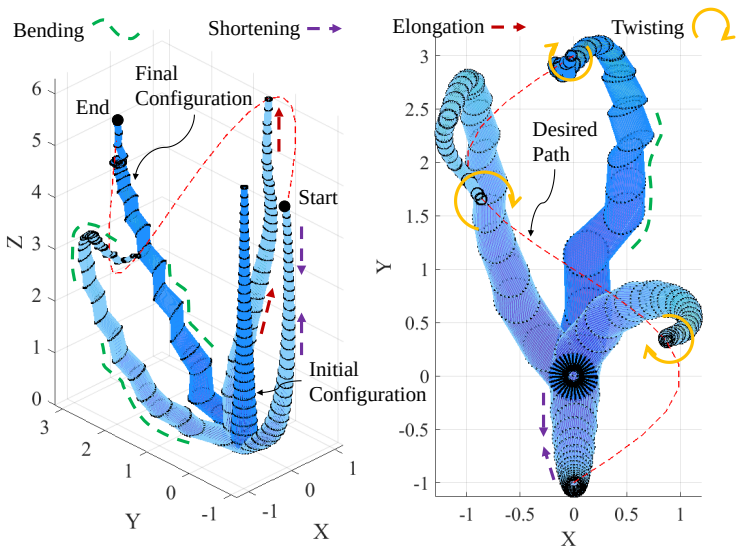


(b)

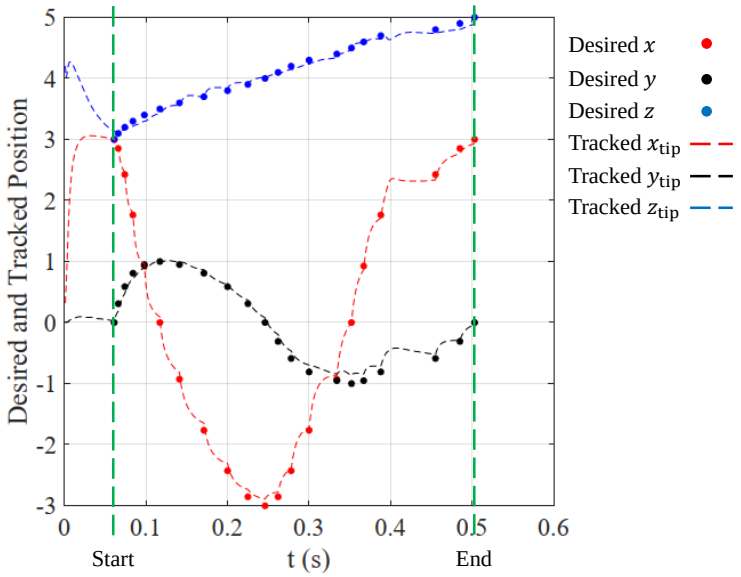
Figure 2.13: (a) Snapshots of the Simulated Hyper-Redundant Robot Tracking the Elliptical Trajectory (2.64). Average Tracking RMSE $= 0.044$ cm. (b) Time Evolution of the Desired Position (x,y,z) and Tracked Position $(x_{\text{tip}},y_{\text{tip}},z_{\text{tip}})$ of the Robot's Tip.

respectively, in terms of time $t \in [0,1]$:

$$^{G}\vec{p}_{\text{d,line}} = [3t - 0.5 \quad 3t + 0.5 \quad t + 6]^{T} \tag{2.63}$$



(a)



(b)

Figure 2.14: (a) Snapshots of the Simulated Robot Tracking the Sinusoidal Trajectory Eq. (2.65). Average Tracking RMSE = 0.049 cm. (b) Time Evolution of the Desired Position (x,y,z) and Tracked Position (x_{tip},y_{tip},z_{tip}) of the Robot's Tip.

$$^{G}\vec{p}_{\text{d,ellipse}} = [\cos(2\pi t) \quad \sin(2\pi t) \quad 4 + \sin(2\pi t)]^T \tag{2.64}$$

$$^{G}\vec{p}_{\text{d,sinusoid}} = [\sin(2\pi t) \quad -1 + 4t \quad 5 + e^t \sin(2\pi t)]^T \tag{2.65}$$

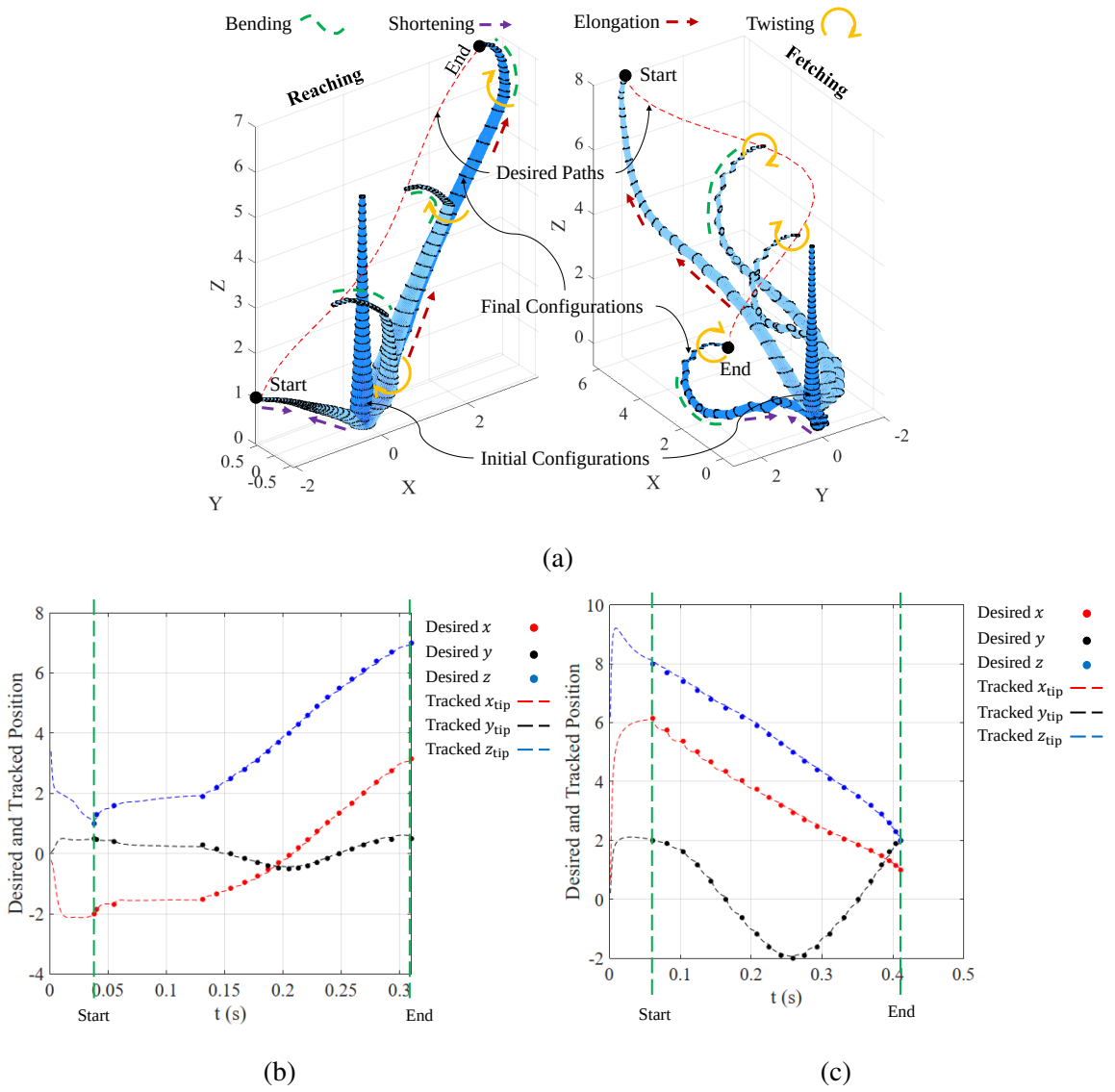


Figure 2.15: (a) Snapshots of the Robot Tracking the Reaching and Fetching Reference Trajectories, Eq. (2.66) and Eq. (2.67). Average Tracking RMSEs for Reaching and Fetching Are 0.051 cm and 0.057 cm, Respectively. (b), (c) Time Evolution of the Desired and Tracked Positions of the Robot's Tip.

We also define two reference trajectories that qualitatively reproduce the reaching and fetching motions of an octopus arm. These two trajectories are given by the following

two parametric equations, respectively, for $t \in [0, 1]$:

$$^{G}\vec{p}_{\text{d,reach}} = [3e^{t} - 5 \quad 0.5\cos(2\pi t) \quad 1 + 6t]^{T} \tag{2.66}$$

$$^{G}\vec{p}_{\text{d,fetch}} = [3e^{1-t} - 2 \quad 2\cos(2\pi(1-t)) \quad 2 + 6(1-t)]^{T} \tag{2.67}$$

In order to track the reaching and fetching trajectories, the robot must perform a combination of elongation, contraction, bending, and twisting motions, requiring each segment to reconfigure along a combination of its six possible DoF. Thus, these two reference trajectories provide challenging test cases for validating our tracking controller.

The number of segments significantly affects the tracking performance of the simulated robot and the calculation time of the controller. In Table 2.1, we list this calculation time on a computer with 64-bit Intel(R) Core(TM) i7-8750H CPU for different numbers of segments. Since the robot did not exhibit effective performance at position and tracking control when the number of segments was lower than 15, we chose 20 segments for the robot in our simulations. The following other parameters and conditions were applied to all simulations. We defined $m = 21$ as the number of target positions for the robot's tip, including the start and the end points, along each reference trajectory. We set $\gamma = 0.05$ in Eq. (2.62). The parameter α in Eq. (2.42), the ratio between the radii of each segment's base and end-effector, was initially set to 0.095. The radius of the base of the robot and the initial length of the robot were 0.25 m and 5 m, respectively. Given the isovolumetric property described in Section 2.2.1, all segments of the robot must maintain the same volume throughout each simulation. Since the total volume of the robot was distributed equally among its segments, the length of each segment was determined by Eq. (2.43). As a result, the lengths of the segments increase from the base of the robot (1^{st} segment) to its tip (N-th segment), since the radii of the segments decrease from base to tip in order to maintain the same volume for all segments and to enforce the isovolumetric property described in Section 2.2.1. We empirically set the controller gain to $K = 20$, which produced accurate

Table 2.1: Calculation Time of the Proposed Controller in Milliseconds (ms) for the Simulated Multi-Segment Robot with Different Numbers of Segments.

Number of segments	10	15	20	25	30
Calculation time (ms)	32.14	77.48	91.17	135.84	192.66

tracking in all simulations.

Figures 2.12a, 2.13a, 2.14a, and 2.15a plot the initial and final configurations of the simulated robot and several of its intermediate configurations, shown in a paler shade, as it tracks each reference trajectory in Eqs. (2.63)–(2.67). In all cases, the robot starts in a vertical initial configuration. Each desired reference trajectory is depicted as a red dashed line, and its start and end points are labeled. As shown in Fig. 2.12a, the robot exhibits bending, elongation, and twisting motions along its length in order to track the desired linear trajectory. Figure 2.13a shows the shortening, twisting, and bending of the robot that enables it to track the desired elliptical trajectory. Tracking the desired sinusoidal trajectory and performing reaching and fetching tasks require the robot to exhibit all four fundamental octopus-like motions, as depicted in Figs. 2.14a and 2.15a. We evaluate the performance of the robot at tracking each reference trajectory by computing the average root-mean-square error (RMSE) between the reference trajectory and the trajectory of the robot's tip, which are plotted over time in Figs. 2.12b, 2.13b, 2.14b, 2.15b, and 2.15c. In each figure, the colored dots indicate the sequence of 21 target positions along the reference trajectory, and the colored dashed lines show the simulated position of the robot's tip. The average RMSE value for each reference trajectory is given in the captions of Figs. 2.12a, 2.13a, 2.14a, and 2.15a. These values are at most about 0.285% of the initial length of the robot, indicating the effectiveness of the control strategy.

2.2.4 Discussion

We have proposed a novel discrete approach to kinematic modeling of an octopus-inspired, multi-segment hyper-redundant robot. A bio-inspired solution that exploits the flow of information between interconnected segments is used to determine the instantaneous configuration of the robot, given the tip's pose, without solving the computationally expensive IK problem for the robot.

Chapter 3

COLLISION-FREE POSE STABILIZATION AND TRAJECTORY TRACKING CONTROL OF AUTONOMOUS MOBILE ROBOTS

This chapter contains results from Salimi Lafmejani *et al.* (2020d) (Section 3.1), Salimi Lafmejani *et al.* (2020b) (Section 3.2), Salimi Lafmejani and Berman (2021) (Section 3.3), Salimi Lafmejani *et al.* (2022b) (Section 3.4), Salimi Lafmejani *et al.* (2022a) (Section 3.5), and Salimi Lafmejani *et al.* (2021c) (Section 3.6).

3.1 Pose Stabilization and Trajectory Tracking Control of a Holonomic Robot

In this section, we present an optimal control approach using Linear Matrix Inequalities (LMIs) for trajectory tracking control of a three-wheeled omnidirectional mobile robot in the presence of external disturbances on the robot's actuators and noise in the robot's sensor measurements. First, a state-space representation of the omnidirectional robot dynamics is derived using a point-mass dynamic model. Then, we propose an LMI-based full-state feedback H_∞-optimal controller for the tracking problem. The robot's tracking performance with the H_∞-optimal controller is compared to its performance with a classical full-state feedback tracking controller in simulations with circular and bowtie-shaped reference trajectories. In order to evaluate our proposed controller in practice, we also implement the H_∞-optimal and classical controllers for these reference trajectories on a three-wheeled omnidirectional robot. The H_∞-optimal controller guarantees stabilization of the robot motion and attenuates the effects of frictional disturbances and measurement noise on the robot's tracking performance. Using the H_∞-optimal controller, the robot is able to track the reference trajectories with up to a 47.8% and 45.8% decrease in the maxi-

mum pose and twist errors, respectively, over a full cycle of the trajectory compared to the classical controller. The simulation and experimental results show that our LMI-based H_∞-optimal controller is robust to undesired effects of disturbances and noise on the dynamic behavior of the robot during trajectory tracking and can outperform the classical controller in attenuating their effects.

3.1.1 Formulation and Problem Setup

Figure 3.1 shows a schematic of a three-wheeled omnidirectional robot equipped with omni wheels, which allow the robot move at any direction at each time instant. This holonomic WMR can be modeled as a point-mass with double-integrator dynamics as follows, without including disturbances and noise:

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{u}, \tag{3.1}$$

where $\mathbf{M}_{3\times3} = \text{diag}(m, m, I)$ is a diagonal mass matrix in which m and I are defined as the robot's mass and moment of inertia about the axis perpendicular to the ground, respectively. We assume that m is a known constant and I can be calculated as the moment of inertia of a solid cylinder with radius d as $I = \frac{1}{2}md^2$. As shown in Fig. 3.1, the variables ${}^G x$ and ${}^G y$ are the position coordinates of the robot's center of mass along the $\bar{x}$-axis and $\bar{y}$-axis, and φ is the robot's heading angle. The vector $\mathbf{x} = [{}^G x, \quad {}^G y, \quad \varphi]^\mathrm{T} \in \mathbb{R}^3$, which is defined in the global coordinate frame $\bar{x} - \bar{y}$, indicates the pose of the robot in a planar configuration. Moreover, $\mathbf{u} = [{}^G f_x, \quad {}^G f_y, \quad {}^G \tau_z]^\mathrm{T} \in \mathbb{R}^3$ is the control input vector, which is interpreted as the wrench (forces and torque) applied to the robot's body. We define the state variables as follows:

$$\mathbf{x}_1 = [{}^G x \quad {}^G y \quad \varphi]^\mathrm{T}, \quad \mathbf{x}_2 = [{}^G \dot{x} \quad {}^G \dot{y} \quad \dot{\varphi}]^\mathrm{T}. \tag{3.2}$$

Then, one can readily write the state-space representation of the robot dynamics in Eq. (3.1) as:

$$\dot{\mathbf{x}}_1 = \mathbf{x}_2, \quad \dot{\mathbf{x}}_2 = \mathbf{M}^{-1}\mathbf{u}. \tag{3.3}$$

The vectors $\mathbf{X}_{6\times1} := [\mathbf{x}_1^{\mathrm{T}} \quad \mathbf{x}_2^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^6$ and $\dot{\mathbf{X}}_{6\times1} := [\dot{\mathbf{x}}_1^{\mathrm{T}} \quad \dot{\mathbf{x}}_2^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^6$ are defined to characterize the state-space representation of the robot's dynamics as:

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{A}_p\mathbf{X} + \mathbf{B}_p\mathbf{u} \\ \mathbf{y}_p &= \mathbf{C}_p\mathbf{X} + \mathbf{D}_p\mathbf{u}, \end{aligned} \tag{3.4}$$

where $\mathbf{y}_p$ denotes the output and the four matrices $\mathbf{A}_p$, $\mathbf{B}_p$, $\mathbf{C}_p$, and $\mathbf{D}_p$ for the trajectory tracking control problem are given by:

$$\begin{aligned} \mathbf{A}_p &= \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}, \quad \mathbf{B}_p = \mathbf{M}^{-1} \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{I}_{3\times3} \end{bmatrix} \\ \mathbf{C}_p &= \mathbf{I}_{6\times6}, \qquad \mathbf{D}_p = \mathbf{0}_{6\times3}. \end{aligned} \tag{3.5}$$

Since we are designing a full-state feedback controller, the output $\mathbf{y}_p$ is defined as the state vector $\mathbf{X}$. Note that $\mathbf{I}$ denotes the identity matrix and $\mathbf{0}$ denotes a matrix of zeros.

Figure 3.2 shows the block diagram of the proposed tracking controller when actuation disturbances and output measurement noise are included in the robot model (Scherer, 2001). The plant in Fig. 3.2 consists of the state-space model of the robot's dynamics without undesired exogenous inputs, described in Eq. (3.4). We can expand the state-space model to include the undesired effects of disturbances on the robot, arising from the friction force between the omni wheels and the ground, and noise in the sensor measurements. To do so, we formulate the model as a tracking control problem in the LMI framework, using the notation described in (Caverly and Forbes, 2019). We start by defining the reference trajectory, $\mathbf{r}_{6\times1}$, the disturbance input, $\mathbf{d}_{3\times1}$, and the output measurement noise, $\mathbf{n}_{6\times1}$. The error, $\mathbf{e}_{6\times1}$, between the tracked trajectory and the reference trajectory is defined as $\mathbf{z}_1$. In

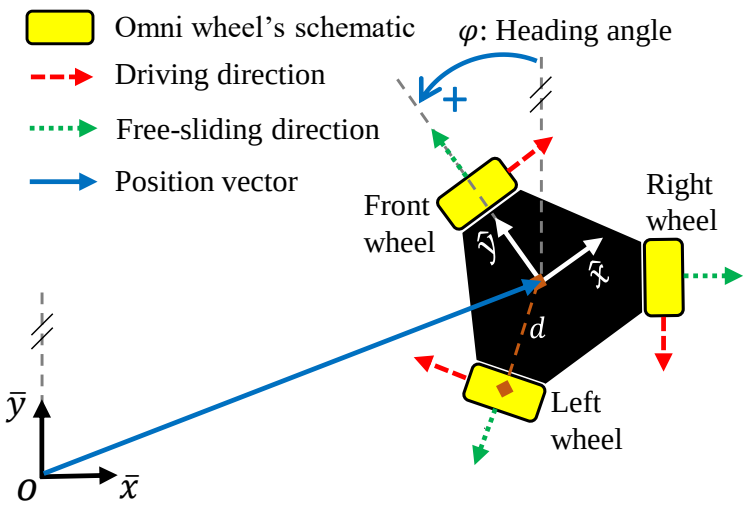


Figure 3.1: Schematic of a Three-Wheeled Omnidirectional WMR, Illustrating the Driving and Free-Sliding Directions of the Robot's Wheels.

addition, $\mathbf{y}_1$ and $\mathbf{y}_2$ are the reference trajectory and the noisy output, respectively. Thus, we have that:

$$\begin{aligned} &\mathbf{w} := [\mathbf{r}^{\mathrm{T}} \quad \mathbf{d}^{\mathrm{T}} \quad \mathbf{n}^{\mathrm{T}}]^{\mathrm{T}}, \quad \mathbf{z}_1 := \mathbf{e} = \mathbf{r} - \mathbf{y}_p, \quad \mathbf{z}_2 := \mathbf{u} \\ &\mathbf{y}_1 := \mathbf{r}, \quad \mathbf{y}_2 := \mathbf{n} + \mathbf{y}_p. \end{aligned} \tag{3.6}$$

As a result, the state-space representation of the robot dynamics for the trajectory tracking problem, including disturbances and sensor noise, can be written as:

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{A}_p\mathbf{X} + [\mathbf{0} \quad \mathbf{B}_p \quad \mathbf{0}]\mathbf{w} + \mathbf{B}_p\mathbf{u} \\ \mathbf{z}_1 &= -\mathbf{C}_p\mathbf{X} + [\mathbf{I} \quad -\mathbf{D}_p \quad \mathbf{0}]\mathbf{w} - \mathbf{D}_p\mathbf{u} \\ \mathbf{z}_2 &= \mathbf{0}\mathbf{X} + [\mathbf{0} \quad \mathbf{0} \quad \mathbf{0}]\mathbf{w} + \mathbf{I}\mathbf{u} \\ \mathbf{y}_1 &= \mathbf{0}\mathbf{X} + [\mathbf{I} \quad \mathbf{0} \quad \mathbf{0}]\mathbf{w} + \mathbf{0}\mathbf{u} \\ \mathbf{y}_2 &= \mathbf{C}_p\mathbf{X} + [\mathbf{0} \quad \mathbf{D}_p \quad -\mathbf{I}]\mathbf{w} + \mathbf{D}_p\mathbf{u}. \end{aligned} \tag{3.7}$$

The realization in Eq. (3.7) results in the following 9-matrix representation for the LMI-

based tracking controller:

$$\begin{aligned}
\mathbf{A} &:= \mathbf{A}_p, \quad \mathbf{B}_1 := \begin{bmatrix} \mathbf{0} & \mathbf{B}_p & \mathbf{0} \end{bmatrix}, \quad \mathbf{B}_2 := \mathbf{B}_p \\
\mathbf{C}_1 &:= \begin{bmatrix} -\mathbf{C}_p & \mathbf{0} \end{bmatrix}^{\mathrm{T}} \quad \mathbf{C}_2 := \begin{bmatrix} \mathbf{0} & \mathbf{C}_p \end{bmatrix}^{\mathrm{T}} \\
\mathbf{D}_{11} &:= \begin{bmatrix} \mathbf{I} & -\mathbf{D}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \mathbf{D}_{12} := \begin{bmatrix} -\mathbf{D}_p & \mathbf{I} \end{bmatrix}^{\mathrm{T}} \\
\mathbf{D}_{21} &:= \begin{bmatrix} \mathbf{I} & -\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_p & -\mathbf{I} \end{bmatrix} \quad \mathbf{D}_{22} := \begin{bmatrix} \mathbf{0} & \mathbf{D}_p \end{bmatrix}^{\mathrm{T}}.
\end{aligned} \tag{3.8}$$

3.1.2 Control Approaches

In the trajectory tracking control problem, the three-wheeled omnidirectional robot should follow predefined reference trajectories that are described by desired poses (position and orientation) and twists (linear and angular velocities) of the robot with respect to time. In this section, we first describe a classical full-state feedback controller for this problem, and then propose a full-state feedback H_∞-optimal controller.

Classical Full-State Feedback Controller

Given the point-mass dynamic model for the omnidirectional robot in Eq. (3.1), define $\mathbf{K} \in \mathbb{R}^{6\times6}$ as a positive definite gain matrix and $\ddot{\mathbf{x}}_r$ as the desired acceleration of the reference trajectory, including linear and angular accelerations. Referring to (Slotine *et al.*, 1991), the control law for the classical controller is described by:

$$\mathbf{u} = -\mathbf{S} + \ddot{\mathbf{x}}_r, \tag{3.9}$$

where $\mathbf{S} \in \mathbb{R}^{6\times1}$ is defined as follows:

$$\mathbf{S} = \mathbf{K}\dot{\mathbf{e}} + \mathbf{\Lambda}\mathbf{e}, \tag{3.10}$$

in which $\mathbf{\Lambda} \in \mathbb{R}^{6\times6}$ is a positive definite gain matrix. Here, the vector $\mathbf{e} \in \mathbb{R}^{6\times1}$ denotes the error between the measured pose of the robot, $\mathbf{x}$, and the reference pose, $\mathbf{x}_r$. In addition, the vector $\dot{\mathbf{e}} \in \mathbb{R}^{6\times1}$ denotes the error between the measured twist of the robot, $\dot{\mathbf{x}}$, and the reference twist, $\dot{\mathbf{x}}_r$. Accordingly, we have that:

$$\mathbf{e} = \mathbf{x}_1 - \mathbf{x}_r, \quad \dot{\mathbf{e}} = \mathbf{x}_2 - \dot{\mathbf{x}}_r. \tag{3.11}$$

Substituting Eq. (3.10) into Eq. (3.9) and using the fact that $\ddot{\mathbf{e}} = \ddot{\mathbf{x}} - \ddot{\mathbf{x}}_r$, we can obtain the equation of the error dynamics for the closed-loop system as a second-order differential equation with positive definite matrix coefficients:

$$\ddot{\mathbf{e}} + \mathbf{K}\dot{\mathbf{e}} + \mathbf{\Lambda}\mathbf{e} = \mathbf{0}. \tag{3.12}$$

In (Sreedhar and Rao, 1968), it was proved that a differential equation of the form Eq. (3.12) is stable if both matrix coefficients are positive definite. This proves that the pose and twist errors will converge to zero.

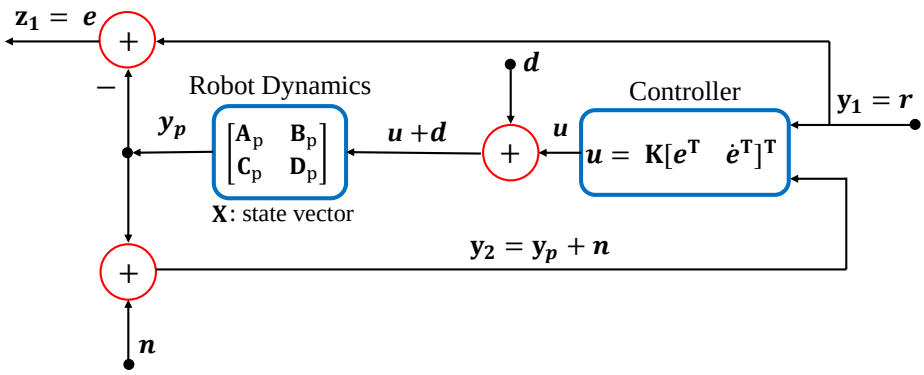


Figure 3.2: Block Diagram of the Tracking Controller, Including Disturbances and Sensor Noise as Input Signals.

Full-State Feedback H_∞-Optimal Controller

The input-output equations in the state-space model of the robot dynamics in Eq. (3.7) can be written in an abbreviated form as follows (Duan and Yu, 2013), where $\mathbf{z} := [\mathbf{z}_1^{\mathrm{T}} \quad \mathbf{z}_2^{\mathrm{T}}]^{\mathrm{T}}$:

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{A}\mathbf{X} + \mathbf{B}_1\mathbf{w} + \mathbf{B}_2\mathbf{u} \\ \mathbf{z} &= \mathbf{C}_1\mathbf{x} + \mathbf{D}_{11}\mathbf{w} + \mathbf{D}_{12}\mathbf{u}. \end{aligned} \tag{3.13}$$

We propose the following control law for the H_∞-optimal controller:

$$\mathbf{u} = -\mathbf{K}_{H_\infty}[\mathbf{e}^{\mathrm{T}} \quad \dot{\mathbf{e}}^{\mathrm{T}}]^{\mathrm{T}} + \ddot{\mathbf{x}}_r \tag{3.14}$$

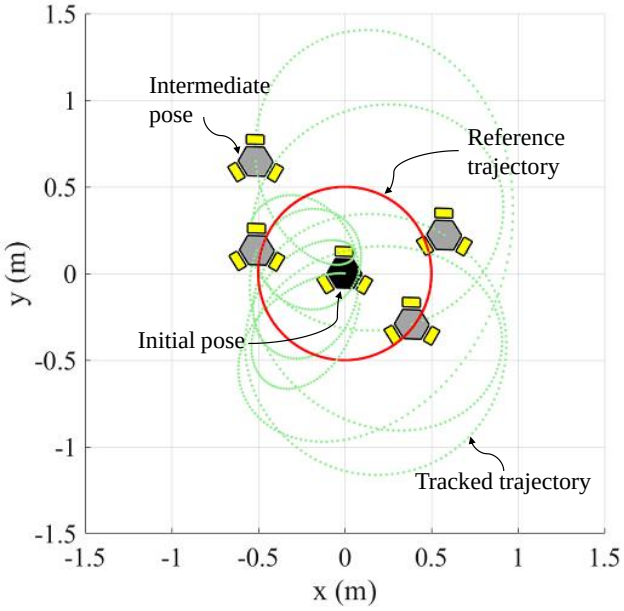
where the controller gain matrix, $\mathbf{K}_{H_\infty}$, is determined such that the H_∞-norm of the closed-loop system is minimized when the exogenous inputs are applied to the robot. The closed-loop system represented in state-space form and its transfer function, $\mathbf{T}(s) = \mathbf{z}(s)/\mathbf{w}(s)$, are described by:

$$\begin{aligned} \dot{\mathbf{X}} &= (\mathbf{A} + \mathbf{B}_2\mathbf{K}_{H_\infty})\mathbf{X} + \mathbf{B}_1\mathbf{w} \\ \mathbf{z} &= (\mathbf{C}_1 + \mathbf{D}_{12}\mathbf{K}_{H_\infty})\mathbf{X} + \mathbf{D}_{11}\mathbf{w} \\ \mathbf{T}(s) &= (\mathbf{C}_1 + \mathbf{D}_{12}\mathbf{K}_{H_\infty})(s\mathbf{I} - (\mathbf{A} + \mathbf{B}_2\mathbf{K}_{H_\infty}))^{-1}\mathbf{B}_1 + \mathbf{D}_{11} \end{aligned} \tag{3.15}$$

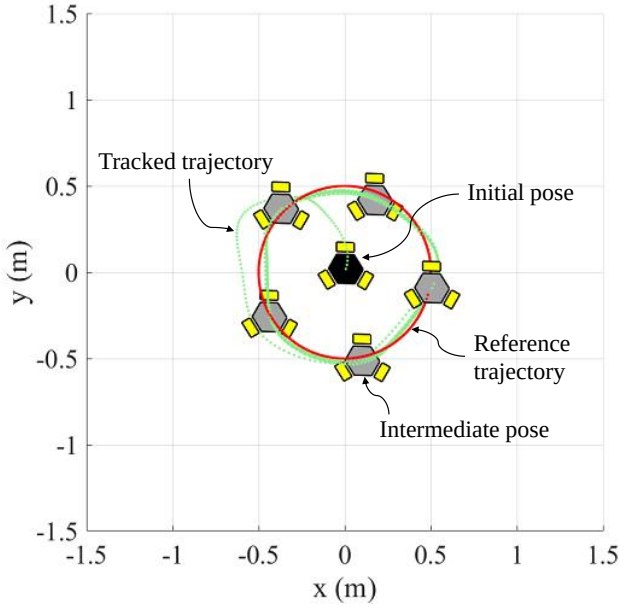
Using the Bounded Real Lemma (Duan and Yu, 2013; Caverly and Forbes, 2019), it can be shown that γ is the minimum value of the H_∞-norm of the closed-loop system if there exist a positive definite matrix $\mathbf{P}$ and a matrix $\mathbf{F}$ which satisfy the LMI in the following optimization problem (here, "$*$" represents a symmetric element of a matrix):

$$\begin{aligned} &\min \quad \gamma \\ &\mathbf{P} > \mathbf{0} \\ &\begin{bmatrix} \mathbf{A}\mathbf{P} + \mathbf{B}_2\mathbf{F} + (\mathbf{A}\mathbf{P} + \mathbf{B}_2\mathbf{F})^{\mathrm{T}} & \mathbf{B}_1 & (\mathbf{C}_1\mathbf{P} + \mathbf{D}_{12}\mathbf{F})^{\mathrm{T}} \\ *^{\mathrm{T}} & -\gamma\mathbf{I} & \mathbf{D}_{11}^{\mathrm{T}} \\ *^{\mathrm{T}} & *^{\mathrm{T}} & -\gamma\mathbf{I} \end{bmatrix} < \mathbf{0} \end{aligned} \tag{3.16}$$

Given such matrices $\mathbf{P}$ and $\mathbf{F}$, we can calculate the H_∞-optimal controller gain matrix as $\mathbf{K}_{H_\infty} = \mathbf{F}\mathbf{P}^{-1}$.

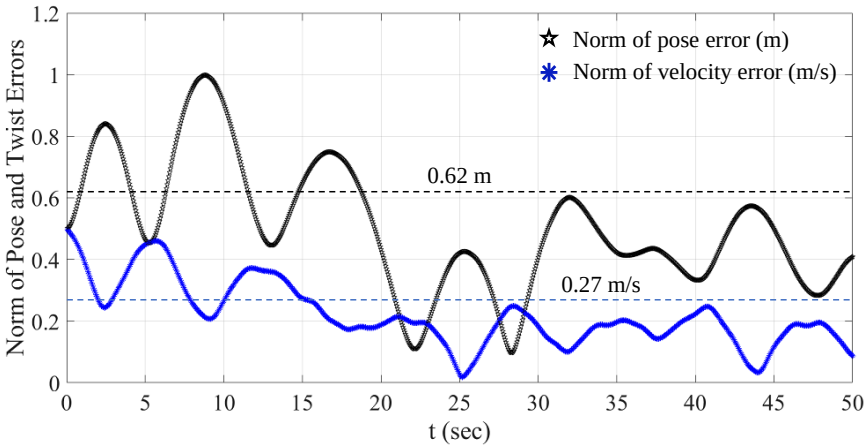


(a)

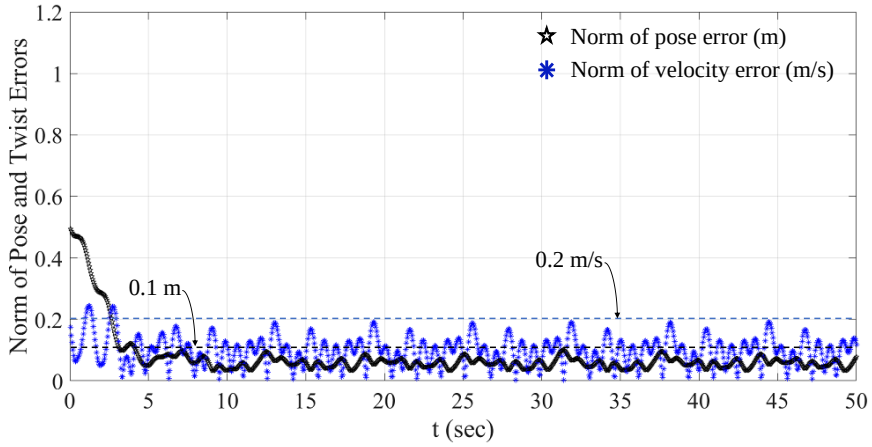


(b)

Figure 3.3: Trajectory of a Simulated Three-Wheeled Omnidirectional Robot that Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller.



(a)



(b)

Figure 3.4: Norm of the Pose and Twist Errors Over 50 s of the Simulations in which the Robot Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller.

3.1.3 Simulation Results

In this section, we implement the controllers on a simulated three-wheeled omnidirectional robot in MATLAB. We compare the performance of the H_∞-optimal controller and the classical controller for two reference trajectories: a circular reference trajectory

described by:

$$\begin{aligned}\mathbf{x}_{r,c} &= [2+\cos(t) \quad 2+\sin(t) \quad 0]^{\mathrm{T}} \\ \dot{\mathbf{x}}_{r,c} &= [-\sin(t) \quad \cos(t) \quad 0]^{\mathrm{T}} \\ \ddot{\mathbf{x}}_{r,c} &= [-\cos(t) \quad -\sin(t) \quad 0]^{\mathrm{T}},\end{aligned} \tag{3.17}$$

and a bowtie-shaped reference trajectory described by:

$$\begin{aligned}\mathbf{x}_{r,b} &= [0.5\cos(t) \quad 0.5\sin(t)\cos(t) \quad 0]^{\mathrm{T}} \\ \dot{\mathbf{x}}_{r,b} &= [-0.5\sin(t) \quad 0.5\cos(2t) \quad 0]^{\mathrm{T}} \\ \ddot{\mathbf{x}}_{r,b} &= [-0.5\cos(t) \quad -\sin(2t) \quad 0]^{\mathrm{T}}.\end{aligned} \tag{3.18}$$

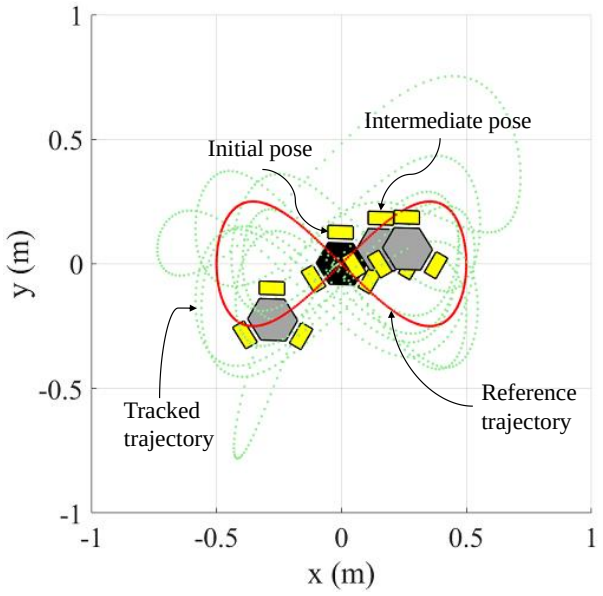
In addition, we simulate the disturbance input $\mathbf{d}$ and output measurement noise $\mathbf{n}$ as the following sinusoidal functions:

$$\begin{aligned}\mathbf{d} =& [0.05\sin(t/2) \quad 0.15\sin(t/4) \quad 0]^{\mathrm{T}} \\ \mathbf{n} =& [0.15\sin(2t) \quad 0.1\sin(3t) \quad 0 \quad 0.25\sin(4t) \quad 0.25\sin(2t) \quad 0]^{\mathrm{T}}.\end{aligned} \tag{3.19}$$

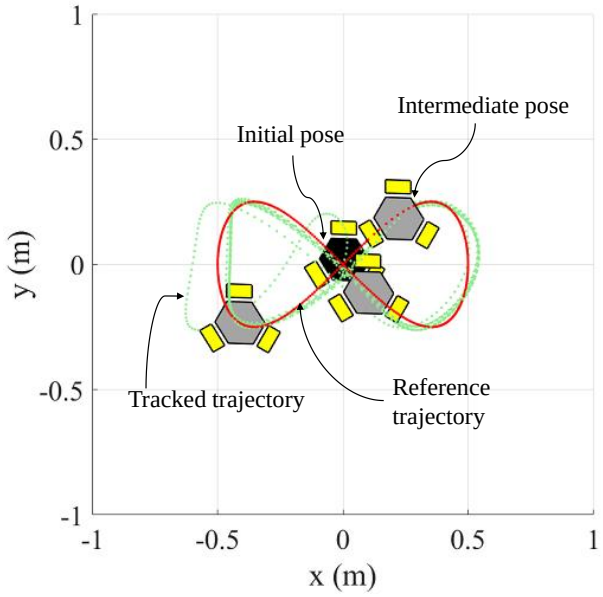
Note that the first three elements of the noise signal are applied to the measurement of the robot's pose, $[{}^{G}x \quad {}^{G}y \quad \varphi]^{\mathrm{T}}$, and the last three elements affect the twist measurements, $[{}^{G}\dot{x} \quad {}^{G}\dot{y} \quad \dot{\varphi}]^{\mathrm{T}}$.

For the classical controller, we tuned the gain matrices to $\mathbf{K} = 0.25\mathbf{I}_{3\times 3}$ and $\mathbf{\Lambda} = 0.5\mathbf{I}_{3\times 3}$, which produce the best tracking performance in terms of the lowest values of $||\mathbf{e}(t)||_2$ and $||\dot{\mathbf{e}}(t)||_2$, the 2-norms of the pose error and twist error. In order to find the controller gain for the H_∞-optimal controller, we solve the LMI in Eq. (3.16) using YALMIP (Lofberg, 2004), which employs SeDuMi (Sturm, 1999) for this purpose. The optimal controller gain matrix of the H_∞ controller is obtained as:

$$\mathbf{K}_{H_\infty} = \begin{bmatrix} 5.01 & 0 & 0 & 12.33 & 0 & 0 \\ 0 & 5.01 & 0 & 0 & 12.33 & 0 \\ 0 & 0 & 35.31 & 0 & 0 & 39.54 \end{bmatrix} \tag{3.20}$$

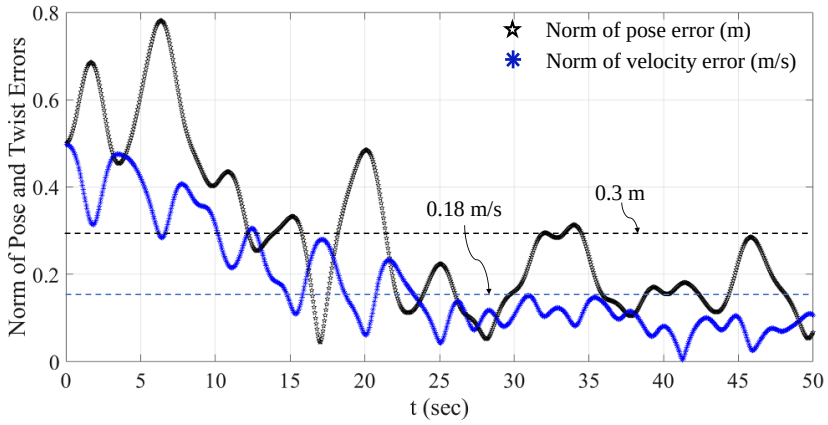


(a)

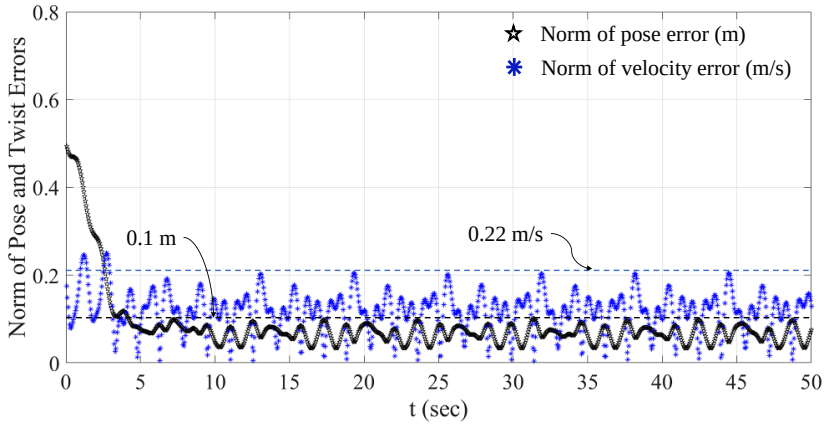


(b)

Figure 3.5: Trajectory of a Simulated Three-Wheeled Omnidirectional Robot that Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller.



(a)



(b)

Figure 3.6: Norm of the Pose and Twist Errors Over 50 s of the Simulations in which the Robot Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller.

Figures 3.3a and 3.3b show the trajectory of the simulated omnidirectional robot over 50 s as it tracks the circular reference trajectory using the classical and H_∞-optimal controllers, respectively, in the presence of the disturbances and noise defined in Eq. (3.19). For these simulations, Figs. 3.4a and 3.4b each plot the time evolution of $||\mathbf{e}(t)||_2$ and $||\dot{\mathbf{e}}(t)||_2$. Similarly, Figs. 3.5a and 3.5b plot the trajectory of the simulated robot over 50 s as it tracks the bowtie reference trajectory using both controllers, and Figs. 3.6a and 3.6b

plot the corresponding time evolution of $||\mathbf{e}(t)||_2$ and $||\dot{\mathbf{e}}(t)||_2$. In the plots of the error norms over time, the black and blue dashed horizontal lines indicate the maximum values of $||\mathbf{e}(t)||_2$ and $||\dot{\mathbf{e}}(t)||_2$, respectively, over one complete cycle of the robot around the reference trajectory after the robot's initial transient behavior as it approached this trajectory. We will refer to these maximum values as $||\mathbf{e}||_2^{\max}$ and $||\dot{\mathbf{e}}||_2^{\max}$. Figs. 3.4a and 3.4b show that for the circular reference trajectory, the H_∞ controller reduces $||\mathbf{e}||_2^{\max}$ from 0.62 m for the classical controller to 0.1 m, and reduces $||\dot{\mathbf{e}}||_2^{\max}$ from 0.27 m/s for the classical controller to 0.2 m/s. Figs. 3.5a and 3.5b show that for the bowtie reference trajectory, the H_∞ controller reduces $||\mathbf{e}||_2^{\max}$ from 0.3 m to 0.1 m, and slightly raises $||\dot{\mathbf{e}}||_2^{\max}$ from 0.18 m/s to 0.22 m/s.

These results demonstrate that the LMI-based full-state feedback H_∞-optimal controller can significantly improve the tracking performance of the robot in simulation compared to the classical full-state feedback controller. This is because the H_∞-optimal controller attenuates the undesired effects of the exogenous inputs, $\mathbf{d}$ and $\mathbf{n}$, whereas the classical controller cannot mitigate the effects of these undesired inputs.

3.1.4 Experimental Implementation and Results

In this section, we implement the classical and H_∞ trajectory tracking controllers on the three-wheeled omnidirectional robot shown in Fig. 3.7. This robot has three omni wheels connected to Dynamixel DC motors that are spaced 120° apart. The rollers around the rims of the omni wheels allow the robot to move freely to any arbitrary configuration. The Dynamixel motors can measure the omni wheels' rotation angles, θ_i ($i = 1, 2, 3$), and their angular velocities, $\dot{\theta}_i$ ($i = 1, 2, 3$), using embedded encoders. The parameters r and d denote the radius of the omni wheels and the distance between the center of each wheel and the origin of the body-fixed coordinate frame, defined as $\hat{x} - \hat{y}$.

Odometry, Twist, and Torque Calculations

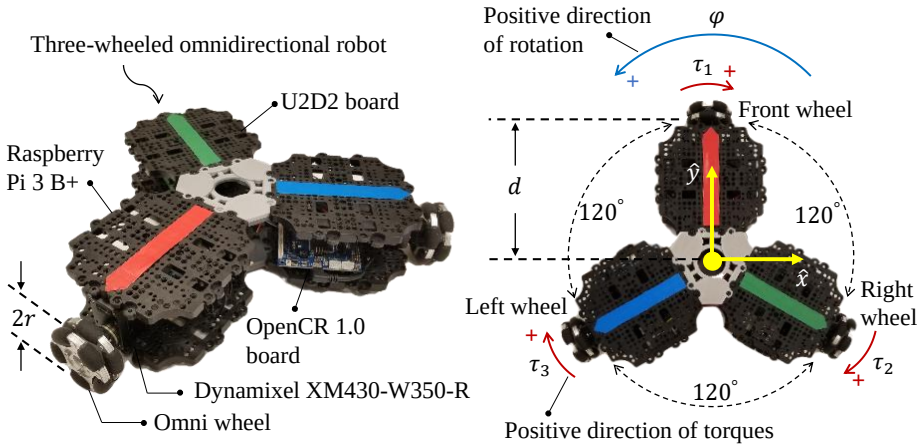


Figure 3.7: (a) Isometric and (b) Overhead Views of the Three-Wheeled Omnidirectional WMR Used in the Experiments. The Global Coordinate Frame is Defined as in Fig. 3.1.

The twist of the omnidirectional robot, $\dot{\mathbf{x}} = [{}^{G}\dot{x} \quad {}^{G}\dot{y} \quad \dot{\varphi}]^{\mathrm{T}}$, represented in the global coordinate frame $\bar{x} - \bar{y}$, can be expressed in terms of the angular velocities of the omni wheels. We first define the rotation matrix, $\mathbf{R}_{-\varphi}(z)$, that describes rotation of the body-fixed coordinate frame, $\hat{x} - \hat{y}$, by an angle $-\varphi$ about the $\bar{z}$-axis. To calculate the robot's twist in the global coordinate frame, we pre-multiply the robot's twist in the body-fixed frame, ${}^{b}\dot{\mathbf{x}} = [{}^{b}\dot{x}, \quad {}^{b}\dot{y}, \quad \dot{\varphi}]^{\mathrm{T}}$, by this rotation matrix as follows (Lynch and Park, 2017):

$$\begin{bmatrix} {}^{G}\dot{x} \\ {}^{G}\dot{y} \\ \dot{\varphi} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}_{-\varphi}(z)} \underbrace{\begin{bmatrix} \frac{2r}{3} & \frac{-r}{3} & \frac{r}{3} \\ 0 & \frac{-\sqrt{3}r}{3} & \frac{\sqrt{3}r}{3} \\ \frac{-r}{3d} & \frac{-r}{3d} & \frac{-r}{3d} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}}_{{}^{b}\dot{\mathbf{x}}} \tag{3.21}$$

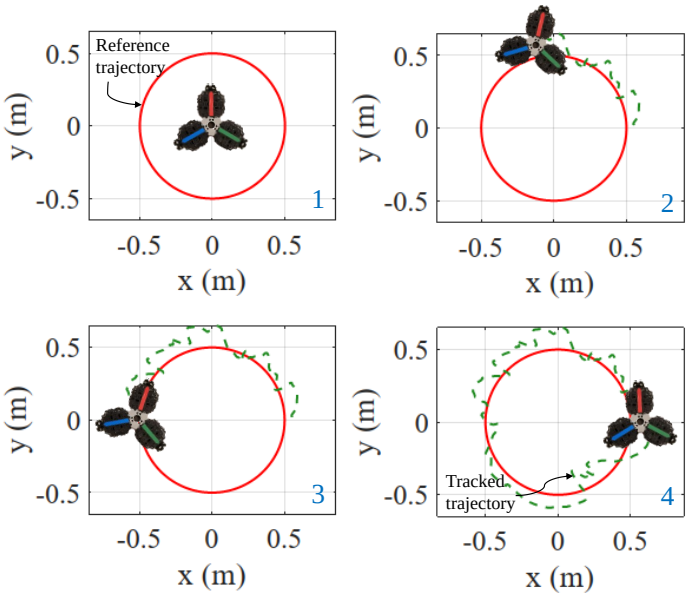
This equation can be used to compute the robot's twist error, $\dot{\mathbf{e}}$, in Eq. (3.11). Odometry to estimate the robot's pose, $[{}^{G}x \quad {}^{G}y \quad \varphi]^{\mathrm{T}}$, is possible using the rotational feedback provided by the Dynamixel motors. This can be done by assuming that the wheels' angular ve-

locities remain constant during the time step Δt between readings, such that $\dot{\theta}_i = \Delta\theta_i / \Delta t$. Regardless of the units used to measure time, we can set $\Delta t = 1$ without loss of generality, which results in $\dot{\theta}_i = \Delta\theta_i$ (Lynch and Park, 2017). Thus, the robot's velocity in the global frame $\bar{x} - \bar{y}$ can be computed in terms of the rotation angles of the omni wheels measured by the encoders. In turn, we can calculate the robot's pose error, $\mathbf{e}$, in Eq. (3.11) given the odometry information of the robot.

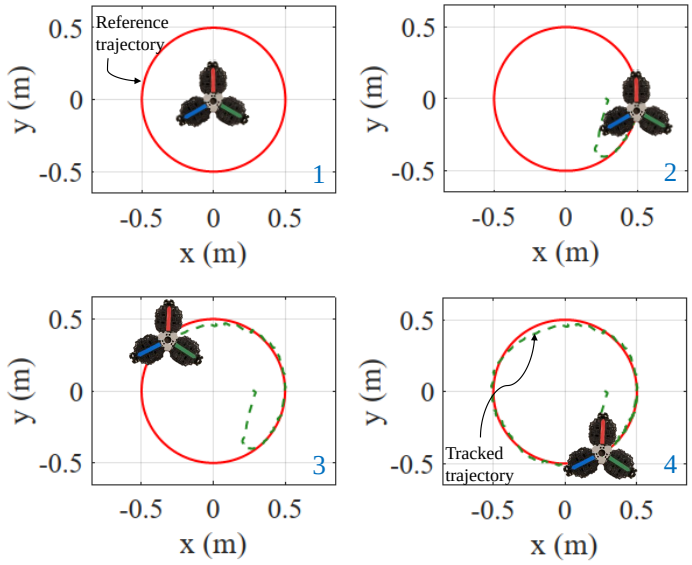
We calculate the control input vector, $\mathbf{u}$, by substituting the robot's pose and twist errors into the control laws of the classical and H_∞ controllers, defined in Eqs. (3.9) and (3.14), respectively. We set the classical controller gains to the values used in the simulations, and the gains of the H_∞ controller were set according to Eq. (3.20). The control inputs are defined as the desired wrench of the robot, $\mathbf{F} = [{}^G f_x \quad {}^G f_x \quad {}^G \tau_z]^{\mathrm{T}}$, represented in the global coordinate frame $\bar{x} - \bar{y}$. This wrench should be produced by the Dynamixel motors. We first transform this desired wrench from the global frame to the body-fixed coordinate frame of the robot by pre-multiplying the wrench in the global frame by the rotation matrix $\mathbf{R}_\varphi(z)$. Then, the wrench in the body-fixed frame can be converted to the desired torques that should be produced by the motors on the omni wheels, $\mathbf{T} = [\tau_1 \quad \tau_2 \quad \tau_3]^{\mathrm{T}}$, as follows:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} \frac{2r}{3} & 0 & \frac{-r}{3d} \\ \frac{-r}{3} & \frac{-\sqrt{3}r}{3} & \frac{-r}{3d} \\ \frac{-r}{3} & \frac{\sqrt{3}r}{3} & \frac{-r}{3d} \end{bmatrix} \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^G f_x \\ {}^G f_y \\ {}^G \tau_z \end{bmatrix} \tag{3.22}$$

The torques applied by the Dynamixel motors are controlled by the Raspberry Pi computer through the U2D2 board, which are both powered by the OpenCR board. We programmed both controllers on the robot in the C++ language based on the Robot Operating System (ROS) platform.

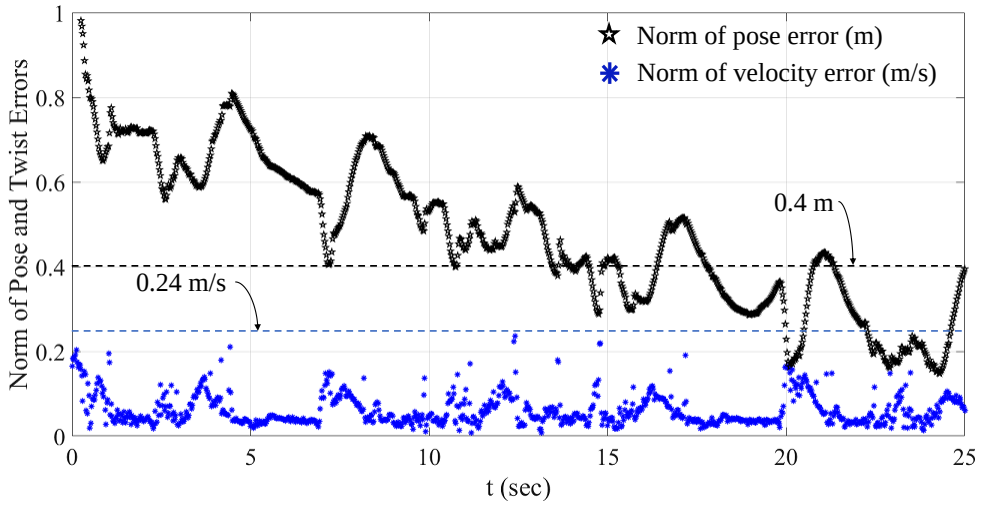


(a)

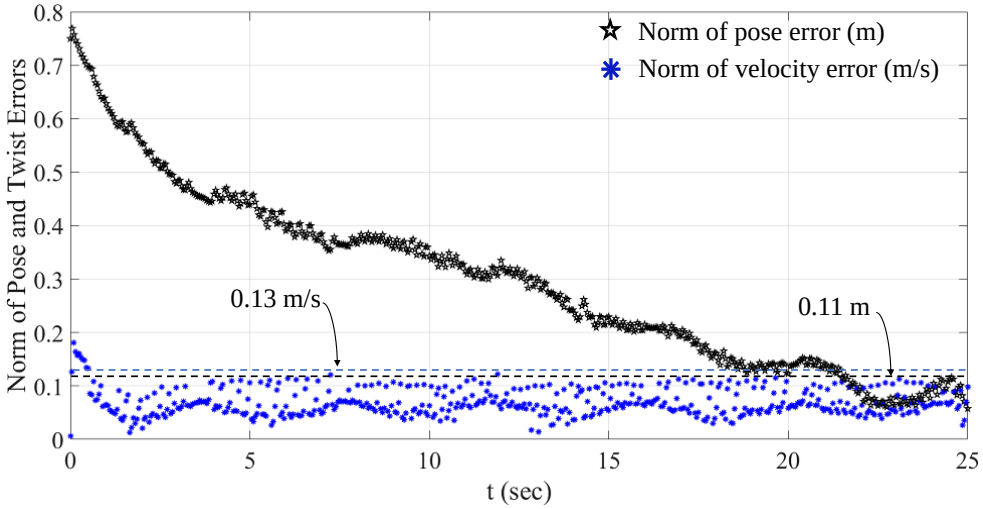


(b)

Figure 3.8: Snapshots of the Trajectory of a Three-wheeled Omnidirectional Robot During an Experiment in which it Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller.



(a)

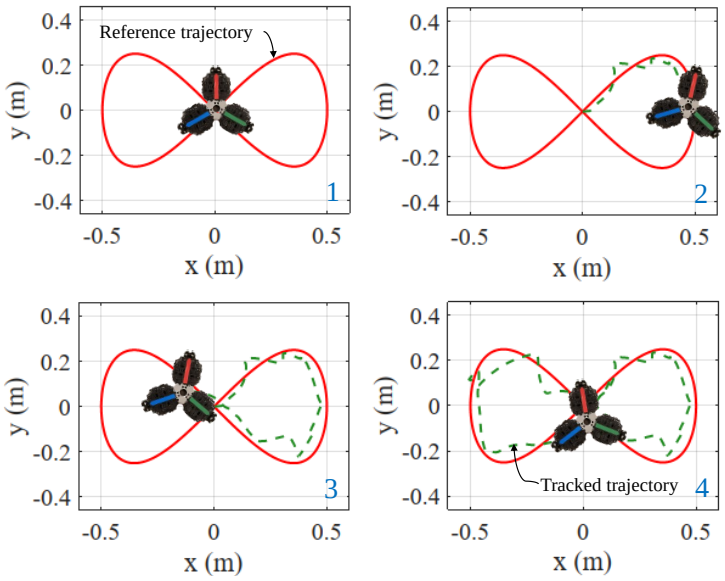


(b)

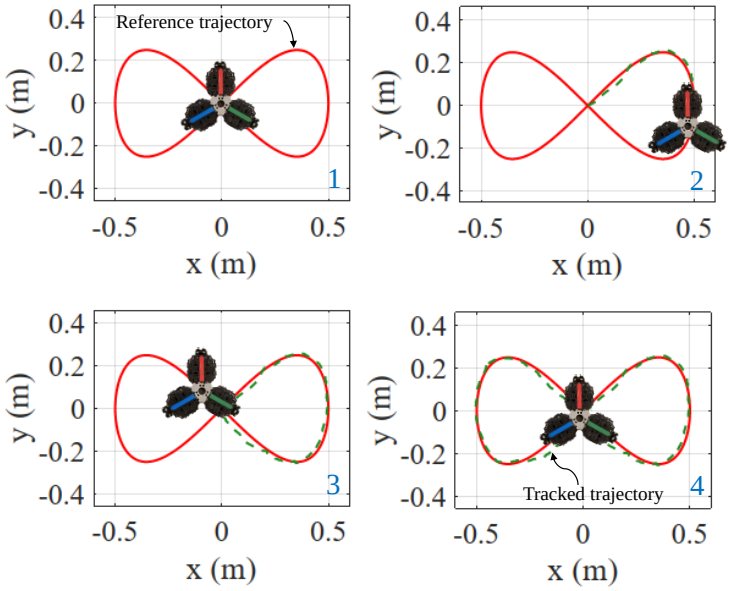
Figure 3.9: Norm of the Pose and Twist Errors Over 25 s of an Experiment in which the Robot Tracks the Circular Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller.

Experimental Results

In our experiments, the three-wheeled omnidirectional robot attempted to track the circular and bowtie reference trajectories defined in Eq. (3.17) and (3.18) in the presence of

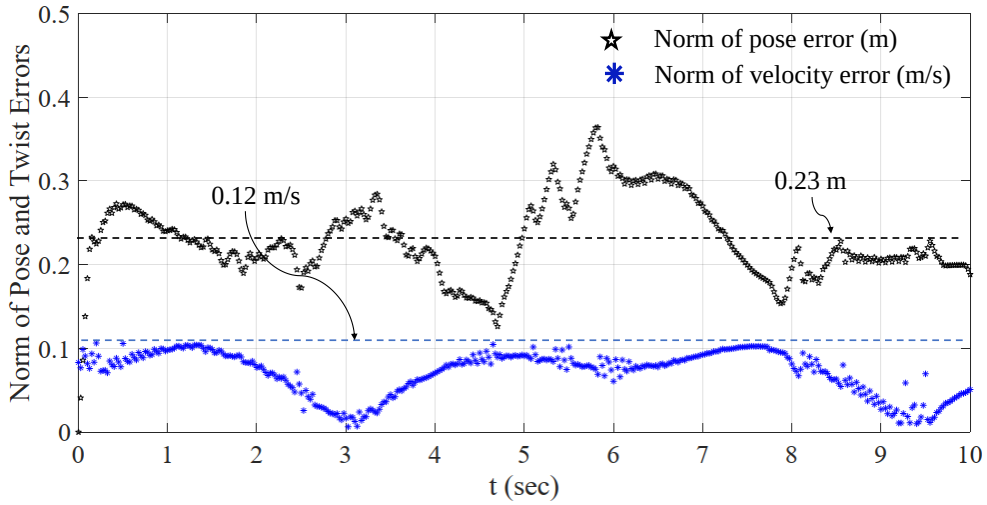


(a)

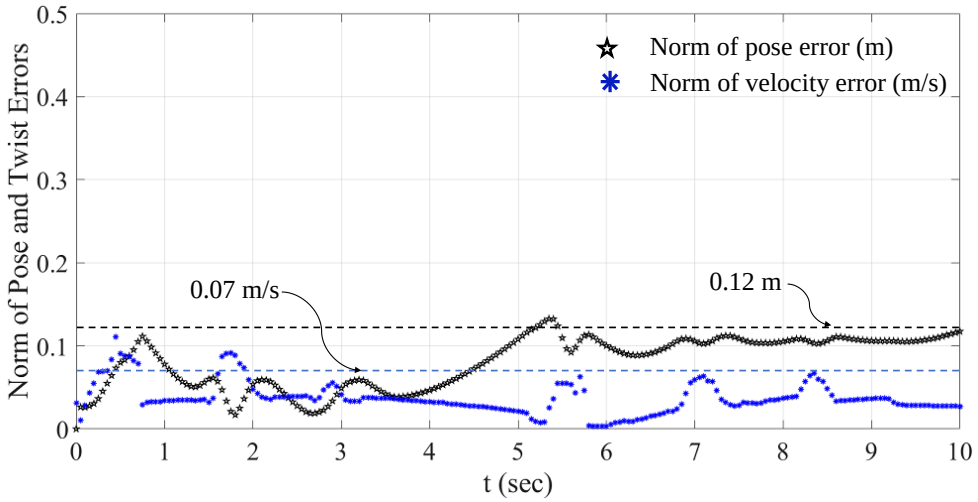


(b)

Figure 3.10: Snapshots of the Robot's Trajectory During an Experiment in which it Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) the Full-State Feedback H_∞-Optimal Controller.



(a)



(b)

Figure 3.11: Norm of the Pose and Twist Errors Over 10 s of an Experiment in which the Robot Tracks the Bowtie Reference Trajectory Using (a) The Classical Full-State Feedback Controller; (b) The Full-State Feedback H_∞-Optimal Controller.

unknown disturbances arising from the friction force between the omni wheels and the ground, as well as noise in the output measurements. We compared the robot's tracking performance with both the classical and H_∞-optimal controllers. Snapshots of the robot's

trajectory over one full cycle of the circular and bowtie reference trajectories for both controllers are plotted using in MATLABin Figs. 3.8a, 3.8b and in Figs. 3.10a, 3.10b. As in the simulations, the H_∞ controller is able to attenuate the undesired effects of exogenous inputs on the tracking performance of the robot, yielding lower tracking errors than the classical controller. Figs. 3.9a and 3.9b show that for the circular reference trajectory, the H_∞ controller reduces $||\mathbf{e}||_2^{\max}$ from 0.4 m for the classical controller to 0.11 m, and reduces $||\dot{\mathbf{e}}||_2^{\max}$ from 0.24 m/s for the classical controller to 0.13 m/s. Figs. 3.10a and 3.10b show that for the bowtie reference trajectory, the H_∞ controller reduces $||\mathbf{e}||_2^{\max}$ from 0.23 m to 0.12 m, and $||\dot{\mathbf{e}}||_2^{\max}$ from 0.12 m/s to 0.07 m/s.

3.1.5 Discussion

In this chapter, we proposed a full-state feedback H_∞-optimal controller based on the LMI framework for trajectory tracking control by a three-wheeled omnidirectional robot. First, a point-mass dynamic model of the robot represented in state-space form was derived, and then the effects of disturbances on its actuators and noise in its output measurements were incorporated into the model. Along with the H_∞-optimal controller, we also described a classical full-state feedback controller for this control problem. We simulated a three-wheeled omnidirectional robot and implemented both controllers in simulation for tracking of circular and bowtie reference trajectories. Furthermore, we experimentally tested the classical and H_∞ controllers on a real three-wheeled omnidirectional robot for the same reference trajectories. We compared the tracking performance of the robot with both controllers in terms of the norms of the robot's pose and twist errors. The simulation and experimental results demonstrate that our proposed full-state feedback H_∞-optimal controller can significantly improve the robot's tracking performance compared to the classical controller by attenuating the effect of undesired exogenous inputs on the robot.

3.2 Position Stabilization of a Nonholonomic Robot with Static Obstacle Avoidance

In this section, we propose a gradient-based nonlinear control approach for stabilizing a nonholonomic Wheeled Mobile Robot (WMR) to a target position in environments with and without obstacles. This approach enables any gradient-based feedback control law (with bounded or unbounded gradients) developed for a holonomic point-mass robot model to be adapted to control a nonholonomic robot. The proposed controller is defined in terms of smooth continuous functions, which produce smooth robot trajectories and can be tuned to stabilize the robot to the goal position at a desired convergence rate. We first prove that the controller will stabilize a nonholonomic robot to a target point in an obstacle-free environment. To stabilize the robot's position in environments with obstacles, we modify our controller to utilize the gradient of an artificial potential function and use Lyapunov stability theory to prove that the robot is guaranteed to converge to the target position under this controller. We demonstrate the effectiveness of our controller for various initial robot positions and environments, and two types of potential fields that are widely used in gradient-based methods for obstacle avoidance, through MATLAB simulations and experiments with a commercial nonholonomic WMR.

3.2.1 Problem Formulations

We consider a nonholonomic WMR with a reference point P at the midpoint of the axis connecting its wheels, as shown in Fig. 3.12. The robot's configuration at time t is defined as $\boldsymbol{\xi}(t) = [\mathbf{x}(t)^T \ \theta(t)]^T = [x(t) \ y(t) \ \theta(t)]^T \in \mathbb{R}^3$, where x and y denote the coordinates of point P in the global coordinate frame and θ denotes the robot's heading angle, defined as the angle of the robot's heading direction with respect to the x_o-axis of the global frame. The unicycle kinematic model of the robot is given by:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \omega, \tag{3.23}$$

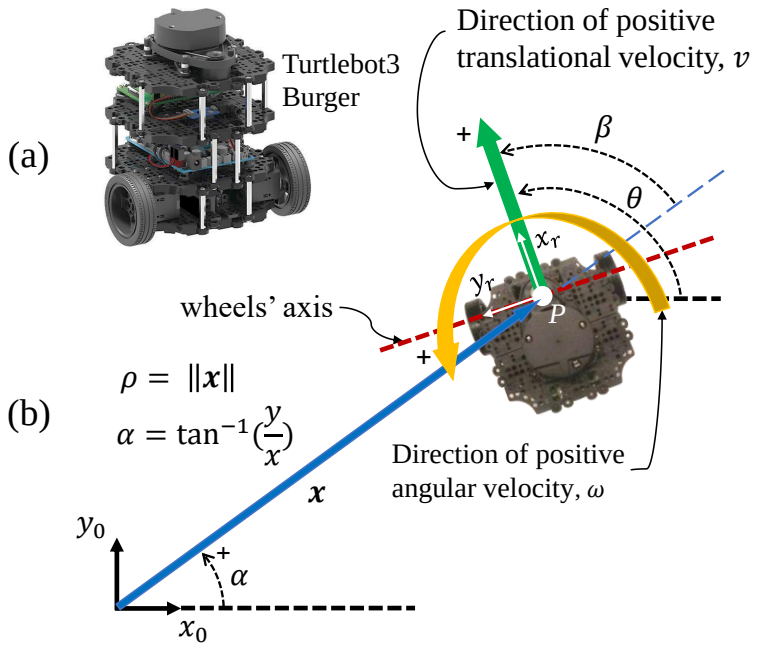


Figure 3.12: (a) 3-D View of the Turtlebot3 Burger Robot (Robotis, 2020), a Nonholonomic WMR that We Used in Our Experiments, and (b) Definitions of the State Variables ρ, α, and θ in an Overhead View of the Robot.

where v is the speed of the reference point and ω is the angular velocity of the robot. The vector of control inputs is defined as $\mathbf{u} = [v\ \omega]^T \in \mathbb{R}^2$. We assume that the robot knows its initial configuration $\boldsymbol{\xi}(0)$ and can use its sensors to determine $\mathbf{x}(t)$ and the quaternion representation (Jia, 2008) of its heading angle $\theta(t)$ at each time $t > 0$. A quaternion $\mathbf{q}$ is defined as the sum of a scalar q_w and a vector $[q_x \quad q_y \quad q_z]^{\mathrm{T}}$ described in an orthonormal basis $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ of $\mathbb{R}^3$:

$$\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}. \tag{3.24}$$

Since the WMR moves in the $x - y$ plane, the elements of $\mathbf{q}$ are given by:

$$q_x = 0, \quad q_y = 0, \quad q_z = \sin\left(\frac{\theta}{2}\right), \quad q_w = \cos\left(\frac{\theta}{2}\right), \tag{3.25}$$

two of which are trigonometric functions of the robot's heading. We will design controllers for the robot that solve the following problems. In both problems, we define the target

position of the robot (specifically, of the point P) as the origin of the global coordinate frame, without loss of generality.

Problem 3.2.1. *Consider a robot with model* (3.23) *that moves in an unbounded domain. Design a feedback control law of the form* $\mathbf{u} = \mathbf{u}(\boldsymbol{\xi})$ *that requires only trigonometric functions of* θ *and drives the robot to the target position from any initial configuration.*

Problem 3.2.2. *Consider a robot with model* (3.23) *that moves in a convex domain that contains one or more convex obstacles. Design a controller of the form* $\mathbf{u} = \mathbf{u}(\boldsymbol{\xi})$*, based on the gradient of a potential field, that requires only trigonometric functions of* θ *and drives the robot to the target position from almost*[1] *any initial configuration while preventing collisions with the obstacles and the domain's boundary.*

Existing solutions to the above problems have some limitations. In particular, they require the controller to be an explicit function of the Euler angle representation of the robot's heading, which can result in large changes in the control input when the robot's heading increases past π rad and is measured as $\theta \in [-\pi, 0]$ rad, or when it decreases past $-\pi$ rad and is measured as $\theta \in [0, \pi]$ rad. These large changes can cause undesired transient behaviors in the robot's navigation, such as sudden reorientations of the robot and chattering in its trajectory that is characteristic of switching controllers (Liberzon, 2003). In order to produce a smooth robot trajectory, our position controller uses trigonometric functions of the robot's heading, i.e. $\sin(\theta)$ and $\cos(\theta)$, in accordance with the quaternion representation Eq. (3.25). Moreover, our approach to collision-free position control can implement any gradient-based controller designed for a holonomic robot on a nonholonomic WMR.

Defining $\rho := \sqrt{x^2 + y^2}$ and $\alpha := \tan^{-1}(\frac{y}{x})$, we can represent the position of the reference point P in a polar coordinate system as $x = \rho\cos(\alpha)$ and $y = \rho\sin(\alpha)$. Accordingly,

[1] See Remark 3.2.6 for a discussion of particular cases in which convergence to the target position is not guaranteed.

we can rewrite the model in Eq. (3.23) in the polar coordinate system as:

$$\begin{aligned}
&\dot{\rho}\cos(\alpha) - \rho\dot{\alpha}\sin(\alpha) = v\cos(\theta),\\
&\dot{\rho}\sin(\alpha) + \rho\dot{\alpha}\cos(\alpha) = v\sin(\theta),\\
&\dot{\theta} = \omega.
\end{aligned} \tag{3.26}$$

We multiply the first and second equations in Eq. (3.26) by $\cos(\alpha)$ and $\sin(\alpha)$, respectively, and add them up. Then we repeat this procedure by multiplying the first and second equations by $-\sin(\alpha)$ and $\cos(\alpha)$, respectively, and adding them up. As a result, we obtain the following representation of the unicycle kinematic model in the polar coordinate system, where ρ, α, and θ are state variables and v and ω are control inputs:

$$\dot{\rho} = v\cos(\theta - \alpha), \quad \dot{\alpha} = \frac{v}{\rho}\sin(\theta - \alpha), \quad \dot{\theta} = \omega, \tag{3.27}$$

We propose the following control laws for v and ω:

$$v = k_v\rho, \quad \omega = k_\omega \sin(\theta - \alpha), \tag{3.28}$$

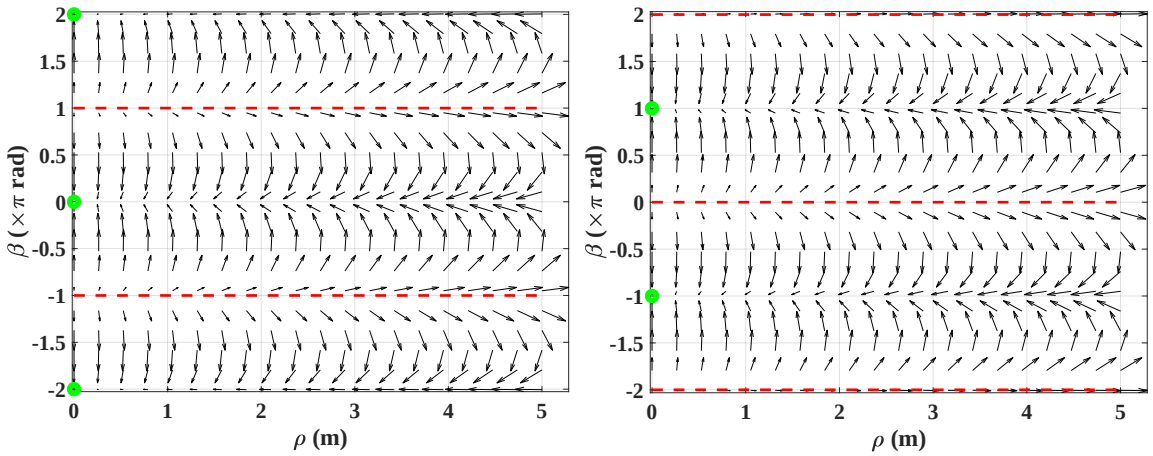
where k_v and k_ω are controller gains. Given Eq. (3.25) and the definition of ρ, Eq. (3.28) can be rewritten as:

$$v = k_v\sqrt{x^2 + y^2} = k_v\|\mathbf{x}\|,$$

$$\begin{aligned}
\omega &= k_\omega \sin(\theta - \alpha) \qquad (3.29)\\
&= k_\omega \sin(\theta)\cos(\alpha) - k_\omega \cos(\theta)\sin(\alpha)\\
&= 2k_\omega q_z q_w \cos(\alpha) - k_\omega(q_w^2 - q_z^2)\sin(\alpha). \qquad (3.30)
\end{aligned}$$

Remark 3.2.3. *As Eq.* (3.29) *shows, we can directly substitute the position and orientation measurements from the robot's sensors, i.e.* x*,* y*,* q_z*, and* q_w*, into the control laws in order to calculate the control inputs without any post-processing of these measurements. Moreover, our proposed control law for* ω *does not have any discontinuities in the robot's heading*

angle θ, since it is based on the quaternion representation of rotations using trigonometric functions of the heading angle measurements.



(a) (b)

Figure 3.13: The Phase Portrait of the Closed-Loop System when the (a) Negative and (b) Positive Controller Gains are Used.

Convergence Analysis

To analyze the stability of the proposed control system, we substitute the control law (3.28) into the robot's kinematic model (3.27) and obtain the equations of the closed-loop system as:

$$\begin{aligned} \dot{\rho} &= k_v \rho \cos(\theta - \alpha), \\ \dot{\alpha} &= k_v \sin(\theta - \alpha), \\ \dot{\theta} &= k_\omega \sin(\theta - \alpha). \end{aligned} \tag{3.31}$$

Furthermore, defining $\beta := \theta - \alpha$ and subtracting the second equation from the third equation in (3.31), we obtain a reduced-dimensional representation of the closed-loop system:

$$\dot{\rho} = k_v \rho \cos(\beta), \tag{3.32}$$

$$\dot{\beta} = (k_\omega - k_v) \sin(\beta). \tag{3.33}$$

The equilibrium points of this system are located where $\rho = 0$ and $\beta = n\pi$ for $n \in \mathbb{Z}$. By linearizing the system about these points, we can confirm that if $k_\omega < k_v < 0$, then the equilibrium points with even values of n are locally exponentially stable, while the equilibria with odd values of n are unstable. A converse statement about even and odd values of n holds true if $0 < k_v < k_\omega$. The next theorem characterizes the region of attraction of the stable equilibrium points.

Theorem 3.2.4. *Consider the closed-loop system* (3.32)-(3.33). *If $k_\omega < k_v < 0$, then an equilibrium point of this system, specified by an even number for n as $n = 2m,\ m \in \mathbb{Z}$, is asymptotically stable, and the set $\mathcal{R} = \{\boldsymbol{\xi} \mid \rho > 0,\ (2m-1)\pi < \beta < (2m+1)\pi\}$ is its basin of attraction.*

Proof. Using the trigonometric identity $\sin(\beta) = \frac{2\tan(\frac{\beta}{2})}{1+\tan^2(\frac{\beta}{2})}$ and the following change of variable,

$$\eta = \tan\left(\frac{\beta}{2}\right) \ \rightarrow\ \dot{\eta} = \frac{1}{2}\dot{\beta}\left(1 + \tan^2\left(\frac{\beta}{2}\right)\right), \tag{3.34}$$

Eq. (3.33) is simplified to

$$\dot{\eta} = k_\beta \eta, \tag{3.35}$$

where $k_\beta := k_\omega - k_v$. The solution of Eq. (3.35) is

$$\eta(t) = \eta_0 e^{k_\beta t}, \tag{3.36}$$

where η_0 denotes the value of η at $t = 0$. This shows that if $k_\beta < 0$, $\eta(t)$ exponentially converges to zero for any value of η_0. This assures monotonic convergence of β to $2m\pi$

from any value in $((2m-1)\pi, (2m+1)\pi)$, since $\tan(\cdot)$ is a strictly monotonic function on $(\frac{(2m-1)\pi}{2}, \frac{(2m+1)\pi}{2})$. In Fig. 3.13a, the projection of the system trajectories for $m = -1, 0, 1$ along the vertical axis is monotonic, which shows this monotonic evolution of β. Furthermore, using the trigonometric identity $\cos(\beta) = \frac{1-\tan^2(\frac{\beta}{2})}{1+\tan^2(\frac{\beta}{2})}$ and Eq. (3.36), we can rewrite Eq. (3.32) as

$$\dot{\rho} = f(t)\rho, \tag{3.37}$$

where

$$f(t) = k_v \left(\frac{1 - \eta_0^2 e^{2k_\beta t}}{1 + \eta_0^2 e^{2k_\beta t}} \right). \tag{3.38}$$

Eq. (3.37) is a first-order linear differential equation with a time-varying coefficient, whose solution is given by

$$\rho(t) = \rho_0 e^{\int f(t)dt}, \tag{3.39}$$

where ρ_0 denotes the value of ρ at $t = 0$. To solve the integral in the exponential term, we use the following change of variable:

$$\zeta(t) := \eta_0^2 e^{2k_\beta t} \;\rightarrow\; d\zeta = 2k_\beta \eta_0^2 e^{2k_\beta t} dt. \tag{3.40}$$

Then we have that $\int f(t)dt$ is equal to:

$$\begin{aligned} \int k_v \frac{1-\zeta(t)}{1+\zeta(t)} dt &= \frac{k_v}{2k_\beta} \int \frac{1-\zeta}{1+\zeta} \frac{d\zeta}{\zeta} \\ &= \frac{k_v}{2k_\beta} \int \left(\frac{1}{\zeta} - \frac{2}{1+\zeta} \right) d\zeta \\ &= \frac{k_v}{2k_\beta} \left(\ln \zeta - 2\ln(1+\zeta) \right) \\ &= \ln \left(\frac{\zeta}{(1+\zeta)^2} \right)^{\frac{k_v}{2k_\beta}} \end{aligned} \tag{3.41}$$

Substituting the expression for $\int f(t)dt$ from Eq. (3.41) into Eq. (3.39), we obtain the solution for $\rho(t)$ as

$$\rho(t) = \rho_0 \left(\frac{\eta_0^2 e^{2k_\beta t}}{(1 + \eta_0^2 e^{2k_\beta t})^2} \right)^{\frac{k_v}{2k_\beta}}. \tag{3.42}$$

If $k_\beta < 0$, we can confirm that

$$\lim_{t\to\infty} \frac{\eta_0^2 e^{2k_\beta t}}{(1+\eta_0^2 e^{2k_\beta t})^2} = 0, \tag{3.43}$$

which means that $\rho(t) \to 0$ as $t \to \infty$ if $\frac{k_v}{k_\beta} > 0$. Therefore, $\rho(t)$ converges to zero for any value of ρ_0 if $k_v, k_\beta < 0$, which is equivalent to $k_\omega < k_v < 0$. This completes the proof. □

Corollary 3.2.5. *The equilibrium points of the closed-loop system are marked by the green circles in Figs. 3.13a and 3.13b. If we set $0 < k_v < k_\omega$, then the equilibrium points that are specified by odd values of n become asymptotically stable, and the others become unstable. This occurs if β is replaced by $\beta + \pi$ in Eqs.* (3.32)-(3.33) *and the procedure in the proof of Theorem 3.2.4 is reapplied. The evolution of the system's trajectories for positive values of k_v and k_ω is illustrated in Fig. 3.13b. Monotonic convergence of β to $(2m+1)\pi$ for $m = -1,\ 0,\ 1$, and convergence of ρ to zero, are seen along the vertical and horizontal axes, respectively.*

Remark 3.2.6. *As shown in the phase portraits in Fig. 3.13a and Fig. 3.13b, the robot's radial coordinate ρ diverges to infinity if and only if the initial angle $\beta(0)$ is exactly $(2m+1)\pi$ rad when using negative controller gains or $2m\pi$ rad when using positive controller gains (for $m \in \mathbb{Z}$). To ensure that the robot converges to the target point from such initial configurations, positive controller gains can be used when $\beta(0) = (2m+1)\pi$ (cases 7 and 10 in Fig. 3.14), and negative gains can be used when $\beta(0) = 2m\pi$ (cases 4 and 6 in Fig. 3.14).*

Theorem 3.2.7. *Consider a nonholonomic WMR with kinematic model* (3.23) *that moves in a convex domain containing one or more convex obstacles. Suppose there exists an artificial potential field $\varphi = \varphi(\mathbf{x})$ that admits a global minimum at the origin $\mathbf{x} = \mathbf{0}$ and for which $\mu := ||\nabla\varphi||$ is maximal on the boundaries of the obstacles. If $k_v < 0$ and*

$k_\omega = 2\,k_v$, *then the following control law drives the robot to the origin from almost any initial position in the domain while preventing it from colliding with the obstacles and the domain's boundary:*

$$v = k_v \mu, \quad \omega = k_\omega \frac{\mu}{\rho} \sin(\theta - \alpha). \tag{3.44}$$

Proof. Inserting the control law in Eq. (3.44) into the robot's kinematic model in Eq. (3.27), and using the fact that $\beta = \theta - \alpha$, the equations of the closed-loop system become:

$$\dot{\rho} = k_v \mu \cos(\beta), \quad \rho\dot{\beta} = k_v \mu \sin(\beta). \tag{3.45}$$

We define a set $\mathcal{C}$ as

$$\mathcal{C} = \left\{ (\rho, \beta) \in \mathbb{R}^2 \mid \sin(\beta) = 0 \ \ \vee \ \ \cos(\beta) = 0 \right\}. \tag{3.46}$$

If the system trajectories start from this set, then the solution of Eq. (3.45) is straightforward to calculate. We will now consider the time evolution of trajectories that start outside this set.

We note that μ is implicitly a function of ρ, and the equilibrium points of system (3.45) are the elements of the set

$$\mathcal{E} := \left\{ (\rho, \beta) \in \mathbb{R}^2 \mid \mu = 0 \right\}. \tag{3.47}$$

This set also represents the critical point of the potential field $\varphi(\mathbf{x})$. We now consider the function

$$V = \rho \tan^2(\beta/2), \tag{3.48}$$

which equals zero if $\rho = 0$ and/or $\beta = 2m\pi, \ m \in \mathbb{Z}$, and is positive otherwise. The time derivative of V along the trajectories of the closed-loop system is

$$\dot{V} = k_v \mu \left(2 + \cos(\beta)\right) \tan^2(\beta/2), \tag{3.49}$$

which is zero when $\mu = 0$ and/or $\beta = 2m\pi,\ m \in \mathbb{Z}$, and is negative otherwise. Invoking *LaSalle's invariance principle* (Khalil, 1996), we can conclude that system trajectories that start outside the set $\mathcal{C}$ converge to the largest invariant set in the set

$$\mathcal{S} := \left\{(\rho, \beta) \in \mathbb{R}^2 \mid \mu = 0 \ \vee \ \beta = 2m\pi,\ m \in \mathbb{Z}\right\}, \tag{3.50}$$

which contains $\mathcal{E}$. Points (ρ, β) for which $\rho > 0$ and $\beta = 2m\pi$, $m \in \mathbb{Z}$, do not comprise an invariant set, since they are not elements of $\mathcal{E}$. Therefore, almost all trajectories of the closed-loop system converge to the largest invariant set in $\mathcal{E}$, which is the location of the global minimum of $\varphi(\mathbf{x})$.

Before we prove that the robot will avoid collisions with obstacles, we state the following two lemmas, which will be used in the proof.

Lemma 3.2.8. *Along system trajectories that start outside the set $\mathcal{C}$, $\sin(\beta)$ never changes sign and converges to zero as $t \to \infty$.*

Proof. We multiply the first and second equations in Eq. (3.45) by $\sin(\beta)$ and $-\cos(\beta)$, respectively, and add them up. Then we obtain:

$$\dot{\rho}\sin(\beta) - \rho\dot{\beta}\cos(\beta) \equiv \frac{d}{dt}\left(\frac{\rho}{\sin(\beta)}\right)\sin^2(\beta) = 0. \tag{3.51}$$

Define $\rho_0 = \rho(0)$ and $\beta_0 = \beta(0)$. Equation (3.51) shows that if $\sin(\beta_0) \neq 0$, then $\frac{\rho}{\sin(\beta)}$ is constant along the robot's entire trajectory, and therefore we have that

$$\rho(t) = \frac{\rho_0}{\sin(\beta_0)}\sin(\beta(t)). \tag{3.52}$$

Since $\rho(t)$ is always positive, the sign of $\sin(\beta(t))$ must be equal to the sign of $\sin(\beta_0)$ for all $t \geq 0$. This implies that $\sin(\beta(t))$ never changes sign. Moreover, from Eqs. (3.48)-(3.49), we have that $\rho(t) \to 0$ as $t \to \infty$. Equation (3.52) then shows that $\sin(\beta(t)) \to 0$ as $t \to \infty$. This completes the proof. □

Lemma 3.2.9. *Along system trajectories that start outside the set $\mathcal{C}$, $\dot{\beta}$ remains bounded and converges to zero.*

Proof. The function V in Eq. (3.48) is positive semi-definite, and $\dot{V}$ in Eq. (3.49) is negative semi-definite. Therefore, V is always bounded, which implies that $\tan(\beta/2)$ remains bounded. Also, using the trigonometric identity $\sin(\beta) = \frac{2\tan(\frac{\beta}{2})}{1+\tan^2(\frac{\beta}{2})}$ and Lemma 3.2.8, we can conclude that $\tan(\beta(t)/2) \to 0$ as $t \to \infty$. This implies that $\lim_{t\to\infty} \tan(\beta(t)/2)$ exists and is finite. Moreover, the time derivative of $\tan(\beta(t)/2)$ is:

$$\frac{d}{dt}(\tan(\beta/2)) = \frac{\dot{\beta}}{2}(1 + \tan^2(\beta/2)). \tag{3.53}$$

By applying *Barbalat's Lemma* (Khalil, 1996) to the function $\tan(\beta/2)$, we can conclude that $\lim_{t\to\infty}(\frac{d}{dt}(\tan(\beta/2))) = 0$. This, along with the boundedness of $\tan(\beta/2)$ in Eq. (3.53), proves that $\dot{\beta}(t)$ is bounded and converges to zero as $t \to \infty$. □

We now substitute the expression for $\rho(t)$ in Eq. (3.52) into the second equation in Eq. (3.45) to obtain

$$\mu = \frac{\sin(\beta_0)}{k_v \rho_0}\dot{\beta}. \tag{3.54}$$

Since $\dot{\beta}$ is bounded by Lemma 3.2.9, this equation shows that μ is bounded. This gives an upper bound on $\mu := ||\nabla\varphi||$ along the robot's trajectory. If $||\nabla\varphi||$ is infinite along the boundaries of the obstacles and the domain (if it is bounded), then the boundedness of μ guarantees the robot's clearance from these boundaries. Alternatively, if $||\nabla\varphi||$ is bounded and has its maximum value along these boundaries, then we must design the parameters of the function φ such that the maximum value of $||\nabla\varphi||$ always exceeds the upper bound in Eq. (3.54). This condition guarantees the robot's clearance from the boundaries of the obstacles and the domain. □

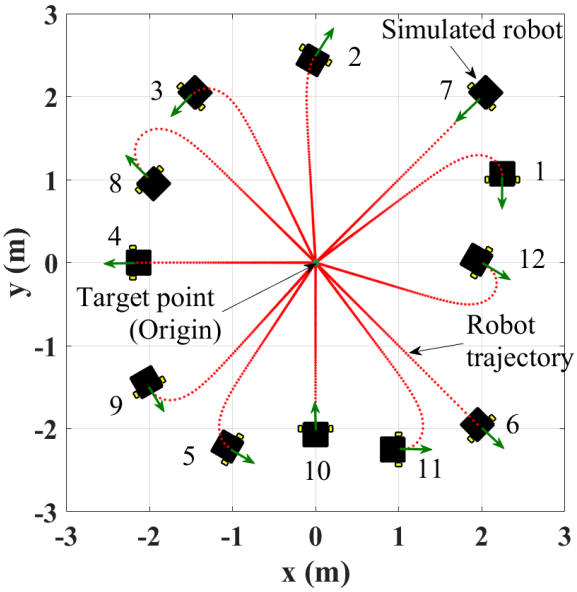


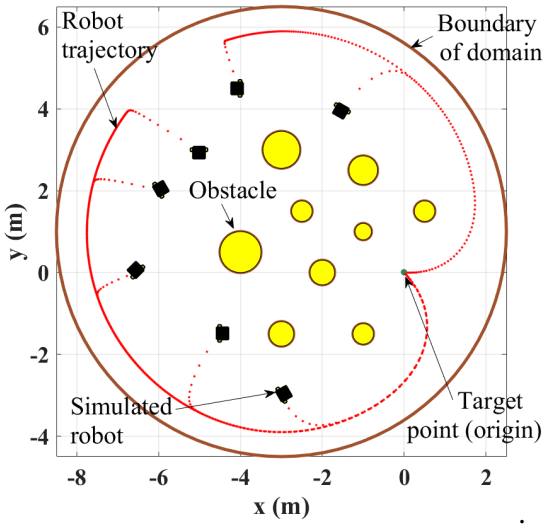
Figure 3.14: Stabilization of a Simulated Nonholonomic WMR to the Origin from Different Initial Configurations, Using Negative Controller Gains $k_v = -0.1$ and $k_\omega = -0.5$ (1 to 6), and Positive Controller Gains $k_v = 0.1$ and $k_\omega = 0.5$ (7 to 12). The Robot's Heading at Each Initial Configuration is Indicated by a Green Arrow.

3.2.2 Simulation and Experimental Results

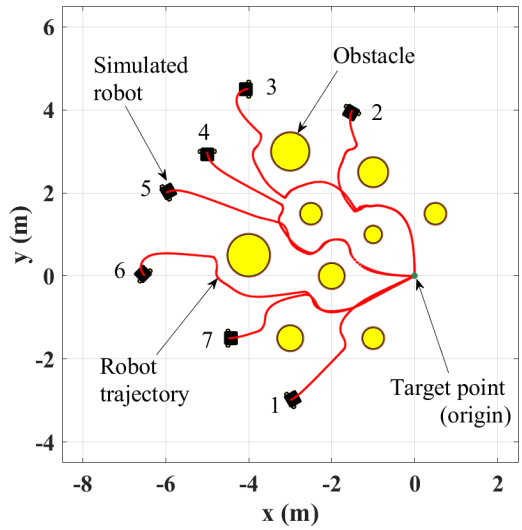
In this section, we validate our controllers for position control and obstacle avoidance with MATLAB® simulations of a nonholonomic robot. We implemented our position controller on the robot with both positive and negative gains and tested it for different initial robot configurations.

Simulation Results

Position Control: First, we simulated the position controller for 12 different initial configurations of the robot, using both possible sets of control gains (positive and negative). The resulting trajectories of the robot are shown in Fig. 3.14. From each initial configuration,

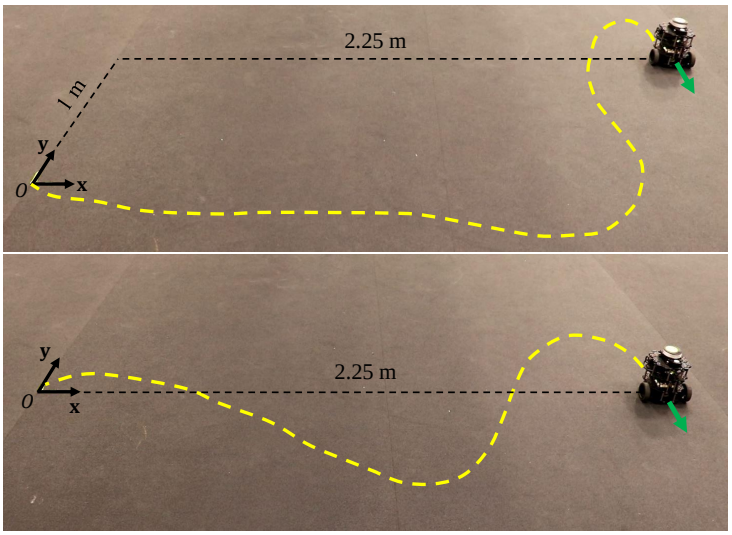


(a) $\varphi = \psi$, a Navigation Function

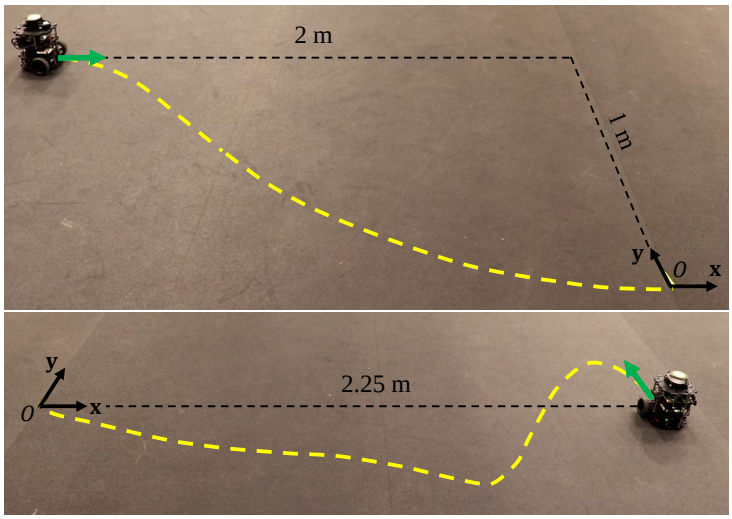


(b) $\varphi = U$, an attractive-repulsive potential field

Figure 3.15: Stabilization of a simulated nonholonomic WMR to the origin from different initial configurations, while avoiding collisions with obstacles, using our proposed controller with the potential field φ defined as **(a)** a navigation function (Koditschek, 1987); **(b)** an attractive-repulsive potential field (Khatib, 1985).



(a) k_v= −0.5, k_ω= −1.5. *Top:* Scenario 1; *Bottom:* Scenario 2.



(b) k_v= 0.5, k_ω= 1.5. *Top:* Scenario 1; *Bottom:* Scenario 2.

Figure 3.16: Experimental Validation of Our Position Controller with **(a)** Negative Controller Gains, Producing Backward Motion, and **(b)** Positive Controller Gains, Producing Forward Motion. The Robot's Heading at its Initial Configuration is Indicated by a Green Arrow, and its Trajectory is Plotted as a Yellow Dashed Line.

shown in black, the robot is successfully stabilized to the origin (dark green point). Note that positive control gains cause the robot to move forward along its trajectory (i.e., in the

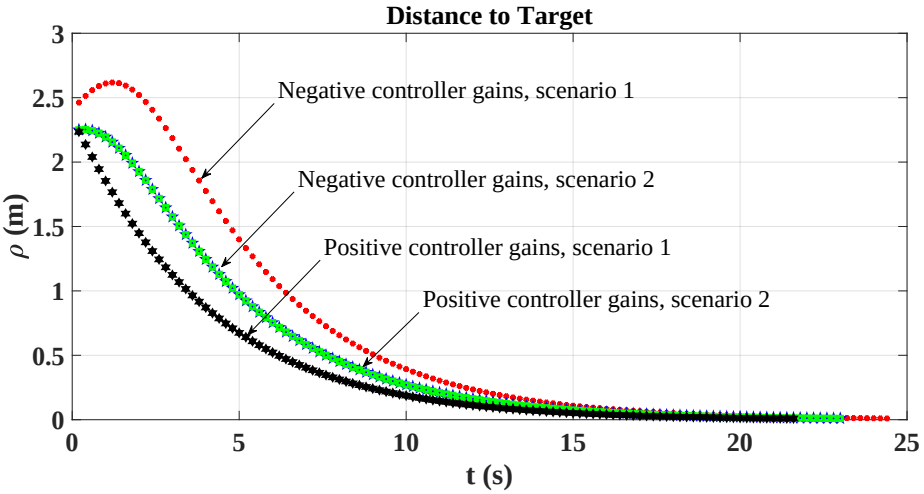


Figure 3.17: Time Evolution of ρ, the Robot's Distance to the Target Position, During the Experimental Tests of Position Stabilization in Fig. 3.16.

direction of the x_r axis in Fig. 3.12), while negative gains cause it to move backward.

Obstacle Avoidance: We evaluated our obstacle avoidance controller by setting the potential field φ to either a *navigation function* ψ (Rimon and Koditschek, 1992) or an *attractive-repulsive potential field* U defined as in (Khatib, 1985). These types of potential fields are widely used in gradient-based control methods for obstacle avoidance, and they satisfy the requirements on φ given in Theorem 3.2.7.

For the case $\varphi = \psi$, we simulated a scenario in which a robot moves to the target position, defined as the origin, through a circular environment with a radius of 4.5 m (navigation functions are defined on bounded domains) while avoiding collisions with 9 circular obstacles. We tested the controller for 7 different initial robot configurations. For the case $\varphi = U$, we simulated the same scenario and initial robot configurations, but in an unbounded environment rather than a bounded one. Figures 3.15a and 3.15b plot the resulting trajectories of the robot for the cases $\varphi = \psi$ and $\varphi = U$, respectively. The trajectories show that in all cases, the robot successfully avoids collisions with the obstacles as it drives to the origin.

Experimental Implementation and Results

We also implemented our controllers for position control and obstacle avoidance on a commercial Robot Operating System (ROS)-compatible nonholonomic WMR, the TurtleBot3 Burger robot produced by Robotis (see Fig. 3.12). This platform is a differential-drive robot with a nonholonomic constraint on its velocity; it cannot produce any controlled motion along the direction of its wheels' axis, shown by the red dashed line in Fig. 3.12b. The robot uses only its onboard sensor measurements to estimate $\boldsymbol{\xi}(t)$, its configuration in the global coordinate frame; no external localization system, such as an overhead camera and vision-based tracking software, is used for this purpose. The robot can use both odometry calculations, based on the measurements of its wheel encoders, and IMU sensor data to estimate its instantaneous configuration with respect to its initial configuration in the global frame, $\boldsymbol{\xi}(0)$. Fusion of the odometry and IMU sensor data improves the accuracy of this estimate. Since we specify that the robot knows $\boldsymbol{\xi}(0)$, it can calculate its configuration $\boldsymbol{\xi}(t)$ in the global frame at any time $t > 0$ by adding the sensor measurements to $\boldsymbol{\xi}(0)$. The robot's heading angle measurement is recorded in quaternion representation, and thus can be directly input to our proposed controllers in Eq. (3.29), making them convenient to implement. A video recording of the experiments described here, along with additional position control experiments, is available online at (Salimi Lafmejani *et al.*, 2020c).

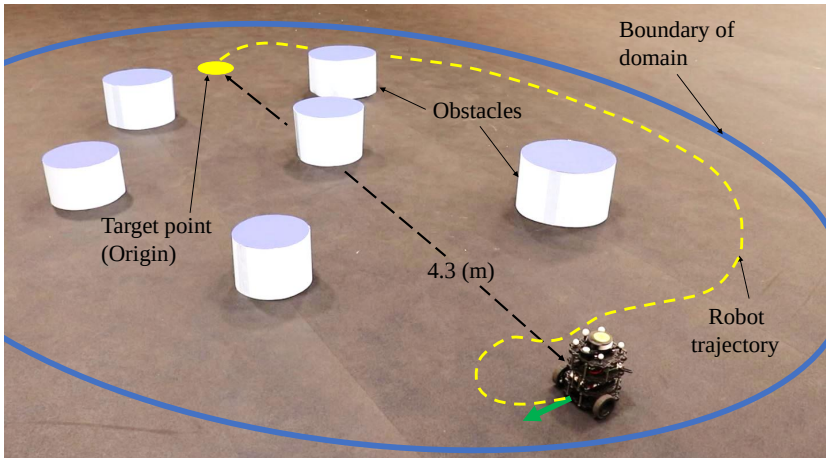
Position Control: Figures 3.16a and 3.16b display the robot's trajectory during four of the experiments, and Fig. 3.17 plots the time evolution of the robot's distance to the target position (the origin) during each of these experiments. The figures show that the controller stabilizes the robot smoothly to the target position from each initial configuration, without producing oscillations or cusps in the robot's trajectory.

Obstacle Avoidance: We also implemented our obstacle avoidance controller with the potential field φ defined as either a navigation function ψ or an attractive-repulsive potential

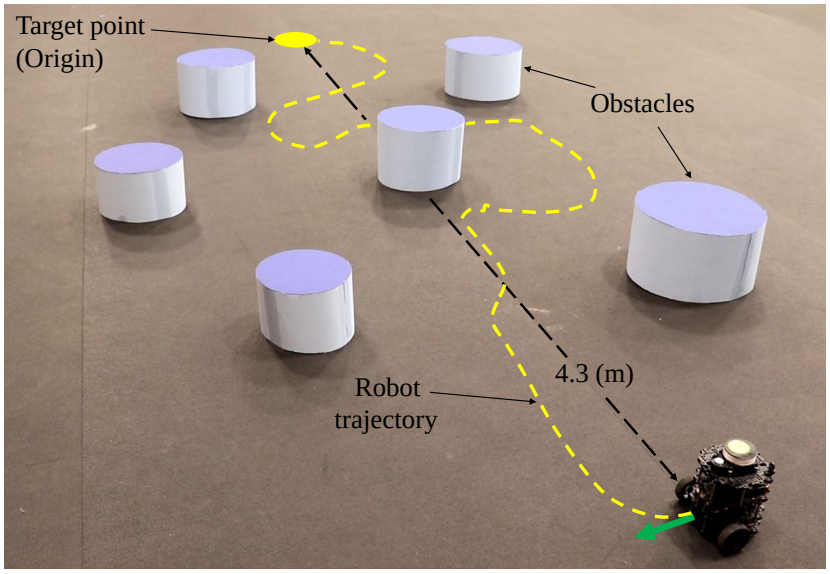
field U. For both cases, we tested the controller in an environment with 6 circular obstacles. In the case where $\varphi = \psi$, we defined a virtual circular boundary with a radius of 2.5 m around the obstacles, and the robot was given the centers and radii of the obstacles and boundary *a priori*. In the case where $\varphi = U$, the robot must be able to measure its distance from the obstacles, which we emulated by giving the robot the same *a priori* information about the obstacles. Figures 3.18a and 3.18b show the robot's trajectory during the experiment for each case. In both experiments, the robot converges to the target location (yellow circle) while avoiding collisions with the obstacles, and for the case where $\varphi = \psi$, it stays within the domain boundary (blue line in Fig. 3.18a). Figure 3.19 plots the time evolution of $\varphi(\mathbf{x})$ and $||\nabla\varphi(\mathbf{x})||$ at the robot's location $\mathbf{x}$ along its trajectory during each experiment. As expected, the values of these functions converge to zero as the robot approaches the target point.

3.2.3 Discussion

In this chapter, we have modified gradient-based controllers designed for holonomic WMRs to enable their implementation on nonholonomic WMRs, in order to achieve collision-free position control in environments with and without obstacles. Our controllers guarantee obstacle avoidance under the same assumptions (e.g., prior information about obstacles) as the corresponding controllers for holonomic robots, and do not impose any additional requirements. We designed a smooth, continuous, nonlinear controller that depends on trigonometric functions of the robot's heading in quaternion form, and thus does not produce undesired transient behaviors during navigation that are exhibited by discontinuous and time-varying controllers which use heading measurements in the Euler angle representation. Any gradient-based obstacle avoidance method that is developed for holonomic robots can be integrated into our controller for nonholonomic robots. We proved that the proposed controllers are stable for two sets of positive and negative control gains and



(a) $\varphi = \psi$, a Navigation Function



(b) $\varphi = U$, an attractive-repulsive potential field

Figure 3.18: Experimental validation of our obstacle avoidance controller, with the potential field φ defined as **(a)** a navigation function (Koditschek, 1987); **(b)** an attractive-repulsive potential field (Khatib, 1985). The robot's initial heading is indicated by a green arrow, and its trajectory is plotted as a yellow dashed line.

demonstrated their effectiveness through simulations and experiments with a nonholonomic WMR.

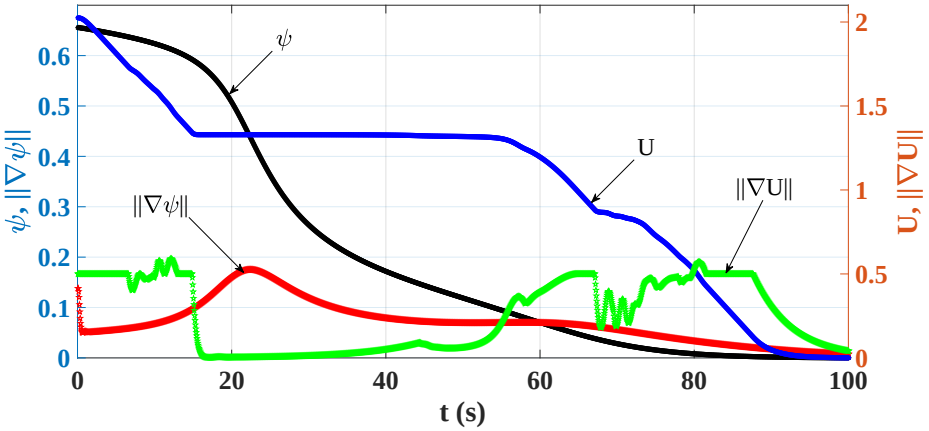


Figure 3.19: Time Evolution of φ and $||\nabla\varphi||$, where $\varphi = \psi$ or $\varphi = U$, During the Experimental Tests of Obstacle Avoidance in Fig. 3.18.

3.3 Pose Stabilization of Multiple Nonholonomic Robots with Collision and Deadlock Avoidance

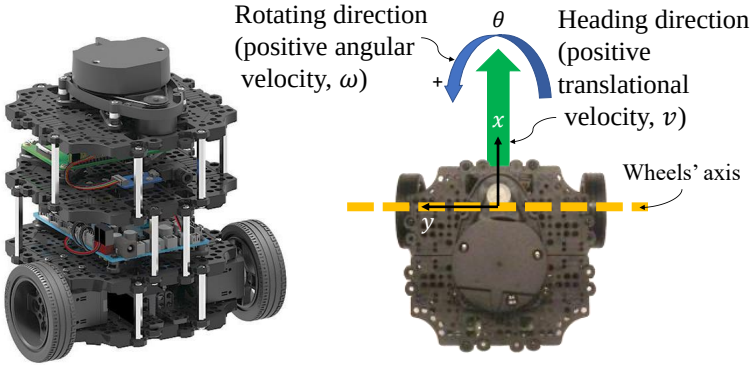
In this section, we present an online nonlinear Model Predictive Control (MPC) method for collision-free, deadlock-free navigation by multiple autonomous nonholonomic Wheeled Mobile Robots (WMRs). Our proposed method solves a nonlinear constrained optimization problem at each time step over a specified horizon to compute a sequence of optimal control inputs that drive the robots to target poses along collision-free trajectories, where the robots' future states are predicted according to a unicycle kinematic model. To reduce the computational complexity of the optimization problem, we formulate it without stabilizing terminal constraints or terminal costs. We describe a computationally efficient approach to programming and solving the optimization problem, using open-source software tools for fast nonlinear optimization and applying the multiple-shooting method. We also provide rigorous proofs of the feasibility of the optimization problem and the stability of the proposed method.

We implement our method using CasADi in a standard software framework, the Robot

Operating System (ROS), and validate its effectiveness at producing collision-free multi-robot navigation in six different scenarios, both in simulations and physical experiments. We also provide recommendations for tuning the parameters of our method, describe how the method can prevent three types of deadlocks, and outline limitations of the method and how they can be overcome. In all scenarios, the robots successfully navigate to their goal poses without colliding with one another or becoming trapped in a deadlock.

3.3.1 Problem Formulation

Figure 3.20 shows an example of a nonholonomic WMR, the Turtlebot3 Burger robot from Robotis®, which we use to validate our control strategy in simulations and physical experiments. The origin of the robot's local coordinate frame, defined in Fig. 3.20b, is located at position $(x(t), y(t))$ in the global coordinate frame at time t. The heading direction of the robot is aligned with the x-axis of the local coordinate frame. The robot's heading angle $\theta(t)$ at time t increases when the robot rotates counter-clockwise about the z-axis of the local frame, and decreases when it rotates clockwise. The control inputs at time t are the robot's linear velocity $v(t)$ and angular velocity $w(t)$. We define the state vector $\mathbf{x} \in \mathbb{R}^3$



(a) (b)

Figure 3.20: (a) 3-D View (Robotis, 2020) of the Turtlebot3 Burger Robot, and (b) Overhead View with Body-Fixed Coordinate Frame.

and the control input vector $\mathbf{u} \in \mathbb{R}^2$ as:

$$\mathbf{x} := [x \quad y \quad \theta]^{\mathrm{T}}, \quad \mathbf{u} := [v \quad w]^{\mathrm{T}}. \tag{3.55}$$

The kinematic model of a nonholonomic WMR can be written as the following unicycle model in state-space form (Lynch and Park, 2017):

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \rightarrow \begin{cases} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= w \end{cases} \tag{3.56}$$

where $f(\cdot)$ denotes a nonlinear function of states and control inputs, representing the twist of the robot. From Eq. (3.56), we obtain the following discrete-time kinematic model of the robot (Keighobadi *et al.*, 2011):

$$\begin{aligned} x(k+1) &= x(k) + v(k)\cos(\theta(k))T_s \\ y(k+1) &= y(k) + v(k)\sin(\theta(k))T_s \\ \theta(k+1) &= \theta(k) + w(k)T_s \end{aligned} \tag{3.57}$$

where T_s is the sampling time and $k \in \mathbb{N} \cup \{0\}$ is an index for the time step, such that $t = kT_s$. We can write this model in state-space form as:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + f(\mathbf{x}(k), \mathbf{u}(k))T_s \tag{3.58}$$

The states and control inputs of the robot may be subject to particular constraints. For instance, if a robot is constrained to navigate in a bounded environment, then its state vector must evolve within the limits $\mathbf{x}_{\text{min}}$ and $\mathbf{x}_{\text{max}}$ defined by the coordinates of the boundaries. Moreover, in real-world implementations, the control inputs of the robot are constrained by a lower limit $\mathbf{u}_{\text{min}}$ and an upper limit $\mathbf{u}_{\text{max}}$, which are determined by the capabilities of the robots' actuators.

MPC Formulation for Single-Robot Navigation

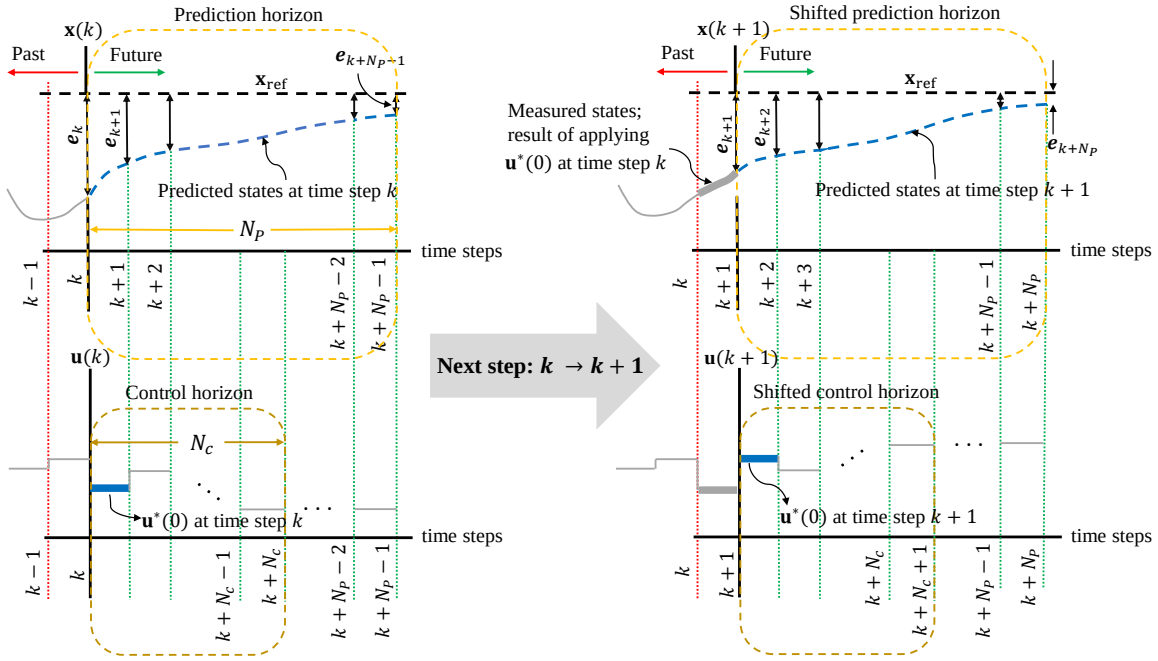


Figure 3.21: Illustration of the Execution of an MPC Method Over two Consecutive Time Steps, k (*left*) and $k+1$ (*right*). The Robot States $\mathbf{x}(k)$ (*Upper Plots*) and Control Inputs $\mathbf{u}(k)$ (*Lower Plots*) are Computed Over the Prediction Horizon N_P and the Control Horizon N_C, Respectively.

In an MPC scheme, the future states of the robot are calculated over a prediction horizon and an objective function is minimized in order to find a sequence of optimal control solutions over this horizon. At each time step, only the first optimal control in the computed sequence is applied to the robot (Borrelli *et al.*, 2017). In the general MPC formulation, a control input $\mathbf{u}(k)$ that is close to a reference control input $\mathbf{u}_{\text{ref}}(k)$ must be computed at each time step k in order to drive the robot's state $\mathbf{x}(k)$ to follow a reference trajectory $\mathbf{x}_{\text{ref}}(k)$. Using the notation $\|\mathbf{x}\|^2_{\mathbf{A}} := \mathbf{x}^T\mathbf{A}\mathbf{x}$ for a square matrix $\mathbf{A}$, we define the following

loss function with weighting matrices $\mathbf{Q} \in \mathbb{R}^{3\times3}$ and $\mathbf{R} \in \mathbb{R}^{2\times2}$:

$$l(\mathbf{x}(k), \mathbf{u}(k)) = \|\mathbf{x}(k) - \mathbf{x}_{\text{ref}}(k)\|^2_{\mathbf{Q}} + \|\mathbf{u}(k) - \mathbf{u}_{\text{ref}}(k)\|^2_{\mathbf{R}} \tag{3.59}$$

where $\mathbf{Q}$ is positive semi-definite and $\mathbf{R}$ is positive definite. In this section, we consider the case where the robot must be stabilized to a fixed goal pose, i.e., the *pose regulation problem*. The reference trajectory $\mathbf{x}_{\text{ref}}(k)$ is defined as this single goal pose, and $\mathbf{u}_{\text{ref}}(k)$ is set to zero so that no control input is driving the robot once it reaches the goal pose.

Given the loss function in Eq. (3.59), the discrete-time kinematic model Eq. (3.58), and the specified limits $\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{max}}$ and $\mathbf{u}_{\text{min}}, \mathbf{u}_{\text{max}}$ on the state vector and control input vector, respectively, we can write the nonlinear optimization problem for the MPC method as follows:

$$\begin{aligned} J^*_{N_P}(\mathbf{x}(0), \mathbf{x}^*, \mathbf{u}^*) = \min_{\mathbf{u}} & \sum_{k=0}^{N_P-1} l(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{x}(k+1) &= \mathbf{x}(k) + f(\mathbf{x}(k), \mathbf{u}(k))T_s \\ \mathbf{x}_{\text{min}} &\leq \mathbf{x}(k) \leq \mathbf{x}_{\text{max}} \\ \mathbf{u}_{\text{min}} &\leq \mathbf{u}(k) \leq \mathbf{u}_{\text{max}} \\ \mathbf{x}(0) &= \mathbf{x}_c(k) \end{aligned} \tag{3.60}$$

where N_P is the prediction horizon, $\mathbf{x}(0)$ is the initial state of the robot, and $\mathbf{x}_c(k)$ is the current measured state of the robot. The optimization problem in Eq. (3.60) is solved at each time step, and the initial state of the robot is updated at each step with the measured state of the robot at time step k. In this MPC design, the optimization problem is solved using the single-shooting method, in which only control inputs are decision variables.

Figure 3.21 illustrates the execution of the MPC method over two consecutive time steps, k and $k+1$. The figure shows representative plots of the robot's state $\mathbf{x}$ and control input $\mathbf{u}$ over time at these two time steps. At time step k, the robot states at times k through $k + N_P - 1$, depicted by the blue dashed line in the upper left plot, are predicted according

to the discrete-time model in Eq. (3.57). We define the error between the robot's state at time k and its current goal pose as

$$\mathbf{e}(k) = \mathbf{x}(k) - \mathbf{x}_{\text{ref}}(k). \tag{3.61}$$

Then, we can compute a matrix $\mathbf{e}(k|k + N_P - 1) \in \mathbb{R}^{3\times N_P}$ that consists of the errors between all future predicted robot states and the corresponding goal poses:

$$\mathbf{e}(k|k + N_P - 1) = [\mathbf{e}(k) \quad \mathbf{e}(k + 1) \quad ... \quad \mathbf{e}(k + N_P - 1)] \tag{3.62}$$

The solution to the optimization problem in Eq. (3.60) is the sequence of optimal controls computed at time step k, $\mathbf{u}^*(k|k+N_P-1) \in \mathbb{R}^{2\times N_P}$. In accordance with the MPC method, the robot receives only the first optimal control input in this sequence, $\mathbf{u}^*(0) \in \mathbb{R}^{2\times 1}$, and applies this input until the next time step, as illustrated in the control input plots in Fig. 3.21.

The robot's measurements of its states will differ from the predicted states due to sensor noise, uncertainties in the kinematic model, and unmodeled dynamics of the environment. This disparity is depicted in the plots of the robot's state between time steps k and $k + 1$ in Fig. 3.21. To initialize the optimization problem at time step $k + 1$, the prediction horizon is shifted forward by one time step and the robot's initial state is updated to its current measured state, $\mathbf{x}(0) = \mathbf{x}_c(k + 1)$. The state errors over the prediction horizon, $\mathbf{e}(k+1|k+N_P)$, are computed for the new time step, and the optimization problem is solved to obtain the optimal control solutions. Again, the robot applies the first optimal control input until the next time step. This procedure is repeated until $||\mathbf{e}(k)|| = ||\mathbf{x}(k) - \mathbf{x}_{\text{ref}}(k)|| < \delta$, for some small positive constant δ.

In order to reduce the computational complexity of the optimization problem, we can compute the control inputs over a control horizon N_C that is shorter than the prediction horizon N_P, while still predicting the future robot states over the prediction horizon. The

solution to the optimization problem is then the sequence of optimal controls $\mathbf{u}^*(k|k + N_C - 1) \in \mathbb{R}^{2\times N_C}$. The optimal controls for the remainder of the prediction horizon are defined as follows:

$$\mathbf{u}^*(\bar{k}) := \mathbf{u}^*(k + N_C - 1), \quad \forall \bar{k} \in [k + N_C, k + N_P - 1] \tag{3.63}$$

At the next time step, this control horizon shifts along with the prediction horizon. In this section, we set the control horizon equal to the prediction horizon, $N_C = N_P$, in all simulations and experiments in order to predict future robot states as accurately as possible, as discussed in Section 3.3.5.

3.3.3 MPC Design for Multiple Nonholonomic Robots

In our proposed nonlinear MPC method for collision-free navigation by multiple nonholonomic WMRs, the optimal control solutions (i.e., the optimal linear and angular velocities) for all robots are obtained by solving a constrained optimization problem at each time step. The objective is to find optimal solutions such that each robot navigates to a preassigned goal pose while avoiding collisions with other robots.

MPC Formulation for Multi-Robot Navigation

Given m robots, we define $\mathbf{x}_i$ and $\mathbf{u}_i$, $i \in \{1, 2, ..., m\}$, as the state and control input vectors, respectively, of the i-th robot. From Eq. (3.58), the discrete-time kinematic model of all robots is given by:

$$\mathbf{x}_i(k + 1) = \mathbf{x}_i(k) + f(\mathbf{x}_i(k), \mathbf{u}_i(k))T_s, \;\; i = 1, ..., m \tag{3.64}$$

We define vectors that contain all the robot states and control inputs at time step $k \in \{0, 1, ..., N_P - 1\}$:

$$\begin{aligned} \mathbf{X}(k) &= [\mathbf{x}_1^{\mathrm{T}}(k) \quad \mathbf{x}_2^{\mathrm{T}}(k) \quad ... \quad \mathbf{x}_m^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{3m} \\ \mathbf{U}(k) &= [\mathbf{u}_1^{\mathrm{T}}(k) \quad \mathbf{u}_2^{\mathrm{T}}(k) \quad ... \quad \mathbf{u}_m^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{2m} \end{aligned} \tag{3.65}$$

Similarly, $\mathbf{X}_c(k)$ will denote the vector containing all the robots' measured states at time step k. We also define $\mathbf{x}_{i,\text{min}}$ and $\mathbf{x}_{i,\text{max}}$ as lower and upper bounds on the i-th robot's state vector, and $\mathbf{u}_{i,\text{min}}$ and $\mathbf{u}_{i,\text{min}}$ as lower and upper bounds on its control input vector.

We consider the pose regulation problem, in which each robot must be stabilized to a fixed goal pose $\mathbf{x}_{g_i}$. We define the vector of reference states as:

$$\mathbf{X}_{\text{ref}} = [\mathbf{x}_{g_1}^{\text{T}} \quad \mathbf{x}_{g_2}^{\text{T}} \quad ... \quad \mathbf{x}_{g_m}^{\text{T}}]^{\text{T}} \in \mathbb{R}^{3m} \tag{3.66}$$

Note that $\mathbf{X}_{\text{ref}}$ is a constant vector, since the goal poses of the robots are fixed. Given that the reference control inputs are set to zero in the pose regulation problem, the loss function in Eq. (3.59) in our multi-robot MPC formulation is defined as:

$$l(\mathbf{X}(k), \mathbf{U}(k)) = \|\mathbf{X}(k) - \mathbf{X}_{\text{ref}}\|_{\mathbf{Q}}^2 + \|\mathbf{U}(k)\|_{\mathbf{R}}^2 . \tag{3.67}$$

In order to achieve collision-free navigation of all m robots, we define a set of constraints in the formulation of the MPC to prevent any collisions that may occur within the prediction horizon. These constraints restrict the optimization problem to compute optimal control commands that drive the robots along collision-free paths toward their goal poses. To this end, at every time step within the prediction horizon, we need to prevent collisions between each pair of robots. We define the minimum allowable distance between two robots, d_{min}, as twice the diameter of the smallest circle that is centered at the origin of a robot's local coordinate frame and completely encloses the robot. Then, the following constraints prevent collisions between each pair of robots during the prediction horizon:

$$||\mathbf{x}_i(k|k+N_P-1) - \mathbf{x}_j(k|k+N_P-1)|| > d_{\text{min}}, \quad \forall i \neq j, \tag{3.68}$$

where $i, j \in \{1, 2, ..., m\}$. We assume that the distance between the initial positions of each pair of robots, and the distance between their goal positions, is at least d_{min}.

Figure 3.22 illustrates examples of collisions between robots that would occur if the robots followed their predicted trajectories. The i-th robot is labeled R$\#i$. In this scenario,

R#1, R#2, and R#3 are moving, and R#4 is stationary. The predicted trajectories of R#1 and R#2 intersect, and R#3 is predicted to collide with R#4. The optimal control solutions which would produce these collisions, and hence violate the collision avoidance constraints in Eq. (3.68), are considered to be infeasible solutions of the optimization problem at the time step k.

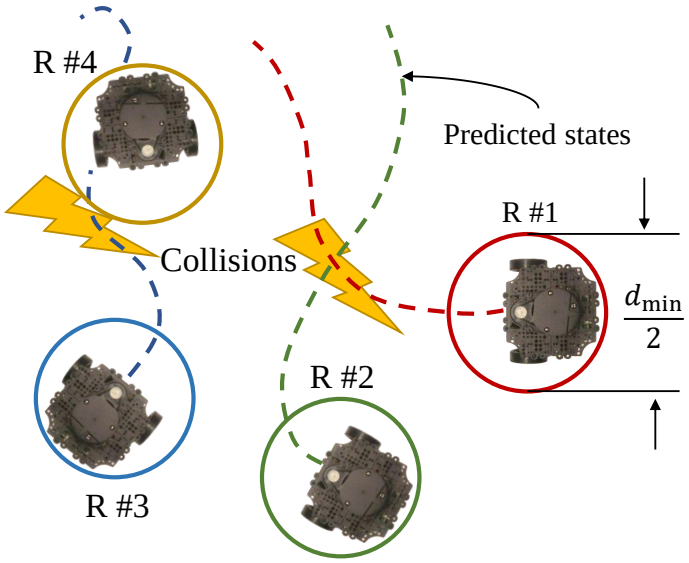


Figure 3.22: Examples of Potential Collisions, Indicated by Lightning Bolts, Between Pairs of Robots.

Given the loss function in Eq. (3.67), the discrete-time kinematic model in Eq. (3.64), the limits on the robots' state and control input vectors, and the collision avoidance constraints in Eq. (3.68), we can write the nonlinear optimization problem for our proposed

MPC method as follows, with $i = 1, ..., m$:

$$
\begin{aligned}
J^*_{N_P}(\mathbf{X}(0), \mathbf{X}^*, \mathbf{U}^*) &= \min_{\mathbf{X},\mathbf{U}} \sum_{i=1}^{m} \sum_{k=0}^{N_P-1} l(\mathbf{X}(k), \mathbf{U}(k)) \\
||\mathbf{x}_i(k+1) - \mathbf{x}_j(k+1)|| &> d_{\text{min}}, \quad \forall i \neq j, \\
\mathbf{x}_i(k+1) &= \mathbf{x}_i(k) + f(\mathbf{x}_i(k), \mathbf{u}_i(k))T_s \\
\mathbf{x}_{i,\text{min}} \leq \mathbf{x}_i(k) &\leq \mathbf{x}_{i,\text{max}} \\
\mathbf{u}_{i,\text{min}} \leq \mathbf{u}_i(k) &\leq \mathbf{u}_{i,\text{max}} \\
\mathbf{X}(0) &= \mathbf{X}_c(k)
\end{aligned}
\tag{3.69}
$$

We solve this optimization problem using the multiple-shooting method.

Feasibility and Stability Analysis

In this section, we investigate the *feasibility* of the optimization problem in Eq. (3.69) and the *stability* of the proposed MPC method. These two properties of MPC schemes have been studied extensively in (Mayne *et al.*, 2000; Gondhalekar *et al.*, 2009). The optimization problem is considered *feasible* if it has at least one solution and all its constraints are satisfied at each time step, and the MPC method is considered *stable* if all robots converge to their preassigned goal poses. A *feasible state* is a state $\mathbf{X}(k)$ that satisfies the bounds $\mathbf{x}_{i,\text{min}} \leq \mathbf{x}_i(k) \leq \mathbf{x}_{i,\text{max}}$ for all $i = 1, ..., m$, and a *feasible control input* is an input $\mathbf{U}(k)$ that satisfies the bounds $\mathbf{u}_{i,\text{min}} \leq \mathbf{u}_i(k) \leq \mathbf{u}_{i,\text{max}}$, $i = 1, ..., m$.

Definition 3.3.1 (Feasible states and admissible control sets)**.** *Define $\mathbb{X}$ as the set of feasible states $\mathbf{X}$ of all robots, and $\mathbb{U}$ as the set of feasible robot control inputs $\mathbf{U}$. Then, the control inputs $\mathbf{U}(k|k+N_P-1)$ are admissible for any initial state $\mathbf{X}(0) \in \mathbb{X}$ if:*

***(1)** the control inputs are a subset of the feasible control set for all time steps in the prediction horizon, i.e. $\mathbf{U}(k|k+N_P-1) \subseteq \mathbb{U}$, and*

***(2)** the predicted states are a subset of the feasible state set for all time steps in the horizon,*

i.e. $\mathbf{X}(k|k+N_P-1) \subseteq \mathbb{X}$.

To simplify the notation, we also define the set of all feasible states and feasible control inputs as $\mathbb{Z} := \mathbb{X} \times \mathbb{U}$.

Theorem 3.3.2 (**Feasibility**). *Suppose that the initial state* $\mathbf{X}(0)$ *is known and is a feasible state, i.e.* $\mathbf{X}(0) \in \mathbb{X}$. *Then, the constrained nonlinear MPC optimization problem in Eq.* (3.69) *has feasible solutions if and only if for all time steps* k *in the prediction horizon, the following statements hold:*

(a) *The optimal control solutions and corresponding predicted states are a subset of the feasible set, i.e.* $\mathbf{X}^*(k|k+N_P-1) \times \mathbf{U}^*(k|k+N_P-1) \subseteq \mathbb{Z}$.

(b) *The collision avoidance constraints are satisfied for all time steps in the horizon, i.e.* $||\mathbf{x}_i(k|k+N_P+1) - \mathbf{x}_j(k|k+N_P+1)|| > d_{min}, \ \ \forall i \neq j$.

Proof. Given the nonlinear system in Eq. (3.56), if the initial state and initial control input are in the feasible set, i.e. $\mathbf{X}(0) \times \mathbf{U}(0) \in \mathbb{Z}$, then the recursive feasibility, defined in (Grüne and Pannek, 2017), of the optimization problem in Eq. (3.69) trivially holds. In other words, the set of optimal control solutions $\mathbf{U}^*(k|k+N_P-1)$ is not the empty set. To prove that the collision avoidance constraints are satisfied for all time steps $k, ..., k+N_P-1$, we consider a contradictory case. Suppose that there exists an optimal control solution for which two robots have a point in common on their predicted paths that they will reach at the same time, leading to a collision (see robots R$\#1$ and R$\#2$ in Fig. 3.22). According to the *Nyquist-Shannon* sampling theorem, a discrete sequence of samples can be used to adequately reconstruct a continuous-time signal given a small enough sampling time, T_s. Thus, for T_s small enough, the MPC method will identify the aforementioned point of collision between the robots and compute an optimal admissible control solution that avoids producing this collision. As a result, satisfying the collision avoidance constraints at each time step is sufficient to guarantee collision-free navigation by all the robots. □

Proving the stability of our proposed NMPC method is challenging, since the method requires solving a finite-horizon optimal control problem at each time step. Most existing NMPC-based approaches for pose stabilization of nonholonomic robots ensure stability by incorporating stabilizing terminal constraints and/or stabilizing terminal costs into the optimization problem (Keighobadi *et al.*, 2011; Faulwasser and Findeisen, 2015; Faulwasser, 2012; Alessandretti *et al.*, 2013; Lam *et al.*, 2013; Yu *et al.*, 2015; Gu and Hu, 2005; Worthmann *et al.*, 2015b). However, constructing suitable stabilizing terminal constraints or costs is a challenging task, and moreover, this approach limits the operating region of the MPC and necessitates a relatively long prediction horizon (Boccia *et al.*, 2014). These issues in turn make the NMPC method computationally prohibitive to implement. The stability of NMPC methods for nonholonomic robots has also been proved without the introduction of stabilizing terminal constraints and terminal costs (Mehrez *et al.*, 2020; Grimm *et al.*, 2005; Rawlings and Mayne, 2009). We establish our proposed MPC method's stability using this approach, which is detailed in (Worthmann *et al.*, 2015a). Our method extends these types of NMPC methods, which are designed for single robots, to multi-robot systems and introduces constraints into the NMPC that enforce inter-robot collision avoidance. The objective is to prove that there exists an optimal value function, defined as:

$$V_{N_P}(\mathbf{X}(0)) := \inf \; J_{N_P}(\mathbf{X}(0), \mathbf{X}, \mathbf{U}) \quad \forall \mathbf{X} \times \mathbf{U} \in \mathbb{Z}, \tag{3.70}$$

that guarantees the asymptotic stability of our NMPC design with prediction horizon N_P. We first state several definitions and assumptions that are needed in the stability proof.

Definition 3.3.3 ($\mathcal{K}_\infty$-function)**.** *Continuous functions $\zeta(r) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that are strictly increasing, and for which $\zeta(0) = 0$, are categorized as the class of $\mathcal{K}$-functions. Those $\mathcal{K}$-functions which satisfy $\lim_{r\to\infty} \zeta(r) = \infty$ are called the class of $\mathcal{K}_\infty$-functions.*

Assumption 3.3.4. *Define J_k as the following cost function, which is calculated over the*

first k steps of the prediction horizon:

$$J_k(\mathbf{X}(0), \mathbf{X}, \mathbf{U}) = \min_{\mathbf{X},\mathbf{U}} \sum_{i=1}^{m} \sum_{\bar{k}=0}^{k} l(\mathbf{x}_i(\bar{k}), \mathbf{u}_i(\bar{k})). \tag{3.71}$$

Then, for any initial state $\mathbf{X}(0) \in \mathbb{X}$, the following holds:

$$V_k(\mathbf{X}(0)) := \textit{inf}\ \ J_k(\mathbf{X}(0), \mathbf{X}, \mathbf{U}) \ \leq\ \eta_k l(\mathbf{X}^*, \mathbf{U}^*), \tag{3.72}$$

where $\eta_k,\ k \in \mathbb{N}$, is a bounded sequence of monotonically increasing natural numbers.

Assumption 3.3.5. *There exist two $\mathcal{K}$-functions ζ_1 and ζ_2 which satisfy the following:*

$$\zeta_1(||\mathbf{X} - \mathbf{X}_{\text{ref}}||) \ \leq\ l(\mathbf{X}^*, \mathbf{U}^*) \ \leq\ \zeta_2(||\mathbf{X} - \mathbf{X}_{\text{ref}}||). \tag{3.73}$$

Definition 3.3.6 (Performance index)**.** *The performance index γ_{N_P}, which determines the minimum value of the prediction horizon N_P that ensures the asymptotic stability of the MPC method (Worthmann* et al.*, 2015a), is defined as:*

$$\gamma_{N_P} :=\ 1 - \frac{(\eta_{N_P} - 1)\prod_{k=1}^{N_P-1}(\eta_k - 1)}{\prod_{k=1}^{N_P-1}\eta_k - \prod_{k=1}^{N_P-1}(\eta_k - 1)}. \tag{3.74}$$

Theorem 3.3.7 (**Stability**)**.** *Suppose that the initial state $\mathbf{X}(0)$ is known and that the initial state and initial control inputs are feasible, i.e. $\mathbf{X}(0) \times \mathbf{U}(0) \in \mathbb{Z}$. Given Assumption 3.3.4, the closed-loop controller computed by the nonlinear constrained MPC in Eq.* (3.69) *is asymptotically stable with the prediction horizon N_P if for a positive performance index $\gamma_{N_P} > 0$, the relaxed Lyapunov inequality holds:*

$$V_{N_P}(\mathbf{X}(k+1)) \ \leq\ V_{N_P}(\mathbf{X}(k)) - \gamma_{N_P} l(\mathbf{X}(k), \mathbf{U}(k)). \tag{3.75}$$

Proof. Suppose that Assumption 3.3.4 and the inequality in Eq. (3.75) hold, and the initial robot states and control inputs are feasible, i.e. $\mathbf{X}(0) \times \mathbf{U}(0) \in \mathbb{Z}$. Given a sequence $c_k \subseteq \mathbb{R}_{\geq 0}$, where $\sum_{k=0}^{\infty} c_k < \infty$, such that

$$l(\mathbf{X}(k), \mathbf{U}(k)) \leq c_k l(\mathbf{X}^*, \mathbf{U}^*), \quad \forall k \in \{1, 2, ..., N_P - 1\}, \tag{3.76}$$

the bounds in Eq. (3.72) are achieved by setting $\eta_k = \sum_{\bar{k}=0}^{k-1} c_{\bar{k}}$ (Mehrez, 2017). Thus, an upper bound on the optimal value function at time step k, i.e. V_k, can be obtained as follows:

$$\begin{aligned} V_k(\mathbf{X}(0)) &\leq \sum_{\bar{k}=0}^{k-1} l(\mathbf{X}(\bar{k}), \mathbf{U}(\bar{k})) \\ &\leq \sum_{\bar{k}=0}^{k-1} c_{\bar{k}} l(\mathbf{X}^*(\bar{k}), \mathbf{U}^*(\bar{k})) = \eta_k l(\mathbf{X}^*(k), \mathbf{U}^*(k)), \end{aligned} \tag{3.77}$$

in which the sequence η_k is monotonically increasing for $c_k \geq 0$. For further details, we refer the reader to (Worthmann *et al.*, 2015b; Boccia *et al.*, 2014; Worthmann *et al.*, 2015a; Worthmann, 2011). □

Corollary 3.3.8. *Theorem 3.3.7 implies that the constrained nonlinear MPC design in Eq.* (3.69) *produces stable optimal control solutions if and only if the predicted sequence of states for each robot, i.e.* $\mathbf{x}_i(k)$, $i = 1, ..., m$, *converges to its desired goal pose* $\mathbf{x}_{g_i}$ *in finite time starting from any feasible initial state* $\mathbf{X}(0) \in \mathbb{X}$.

Algorithms for Implementation of the MPC

Figure 3.23 illustrates the framework for implementing our proposed MPC method in the Robot Operating System (ROS). We first specify the number of robots m; the robots' initial poses $\mathbf{X}(0)$ and goal poses $\mathbf{X}_{\text{ref}}$; the bounds on the robots' states and control inputs, i.e. $\mathbf{x}_{i,\min}$, $\mathbf{x}_{i,\max}$, $\mathbf{u}_{i,\min}$, and $\mathbf{u}_{i,\max}$; the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$; the sampling time T_s; and the distance $d_{\min}$. The bounds on the robots' linear and angular velocities are defined as $v_i \in [-0.22 \quad 0.22]$ (m/s) and $w_i \in [-2.84 \quad 2.84]$ (rad/s), according to the Burger robot's specifications (Robotis, 2020). We use the symbolic programming framework in CasADi (Andersson *et al.*, 2012) to formulate the optimization problem in Eq. (3.60) for the case of a single robot, or the one in Eq. (3.69) for multiple robots. The optimization problem is solved using the Interior Point OPTimizer (IPOPT), an open-source software

library for large-scale nonlinear optimization problems that is available in CasADi. For details about IPOPT options and the tuning parameters in CasADi, see (Andersson *et al.*, 2012; Wächter, 2009).

Algorithms 1, 2, and 3 contain pseudocode that describes the implementation of our proposed MPC method in both simulation and experiment. A central supervisor runs these algorithms to solve the optimization problem and send optimal control commands to the robots. The robots estimate their current poses relative to their initial poses (which are known) using odometry, fusing their wheel encoder measurements and IMU sensor data to improve the accuracy of the pose estimates. Using ROS, the robots *publish* their odometry information on a *topic*, and the central supervisor *subscribes* to each robot's published odometry topic in order to obtain the poses of the robots. The supervisor publishes the optimal control solutions to the robots' velocity commands. These control commands are published to the robots' velocity topics, which are each defined as a ROS topic.

Algorithm 1 executes the main loop of the program. First, it initializes the optimization problem of the MPC, the ROS core node, the subscribers that read the robots' odometry measurements, and the velocity command publishers. It also sets the options for the Nonlinear Programming (NLP) solver, including the acceptable convergence tolerance (acceptable_tol), the stopping criterion based on the change in the objective function value (acceptable_obj_change_tol), and the maximum number of iterations (max_iter) for the IPOPT solver. Next, the following procedure repeats as long as the error between the current robot poses and their goal poses, $||\mathbf{X} - \mathbf{X}_{\text{ref}}||$, exceeds a small positive constant δ. First, Algorithm 2 obtains the robots' measurements of their own poses by subscribing to each robot's odometry callback function. Then, the IPOPT solves the optimization problem in either Eq. (3.60) or Eq. (3.69). As described earlier, only the optimal control solutions computed at the first time step of the prediction horizon, i.e. $\mathbf{U}^*(0)$, are applied to the robots. Finally, Algorithm 3 shifts the prediction and control horizons forward by one time

step and re-initializes the optimization problem using the *warm start* approach, in which the optimal solutions at the previous time step are defined as the initial solutions (also referred to as initial guesses) at the next time step. The robots' states are re-initialized with their current poses.

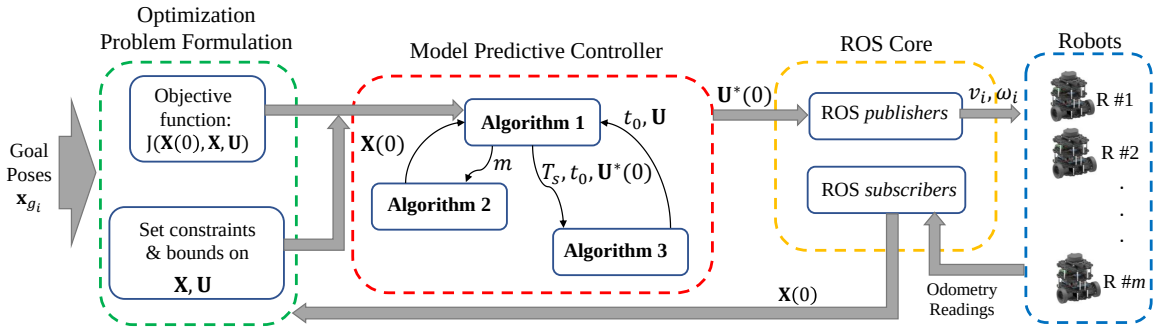


Figure 3.23: Framework for Implementing the Proposed MPC Method for Collision-Free Navigation by Multiple WMRs.

3.3.4 Simulation and Experimental Results

In this section, we validate our NMPC method for collision-free navigation in both simulations and experiments with real-world robots. We performed the simulations in Gazebo, using the Turtlebot3 Burger (see Fig. 3.20) as the robot platform, and conducted the experiments with multiple Burger robots. The Burger robot is a two-wheeled differential-drive WMR that is equipped with Raspberry Pi and OpenCR boards, providing an integrated embedded system of sensors and processors which can be programmed in ROS to implement our MPC method. More details about this robot can be found in (Robotis, 2020). Videos of the simulations and experiments are available online in (Salimi Lafmejani and Berman, 2020b) and (Salimi Lafmejani and Berman, 2020a), respectively.

We tested our MPC method in six different scenarios, each implemented in both simulations and experiments. Three of the scenarios, in which there are $m \in \{1, 2, 3\}$ robots,

Algorithm 1 Main Function of MPC Method

Input: $\mathbf{X}(k|k+N_P-1)$, $\mathbf{X}(0)$, $\mathbf{X}_{\text{ref}}(k|k+N_P-1)$, $\mathbf{U}(k|k+N_P-1)$, $\mathbf{Q}$, $\mathbf{R}$, N_P, m, $\mathbf{x}_{i,\min}$, $\mathbf{x}_{i,\max}$, $\mathbf{u}_{i,\min}$, $\mathbf{u}_{i,\max}$, $f(\mathbf{x}_i(k), \mathbf{u}_i(k))$, T_s, $d_{\min}$

Output: $\mathbf{X}^*(k|k+N_P-1)$, $\mathbf{U}^*(k|k+N_P-1)$

1: Initialize ROS node, odometry subscribers, and velocity command publishers; import CasADi
2: Symbolically formulate the optimization problem in Eq. (3.60) or Eq. (3.69) using CasADi
3: Set NLP solver (IPOPT) options: acceptable_tol $= 10^{-8}$, acceptable_obj_change_tol $= 10^{-6}$, max_iter $= 2000$
4: **while** $||\mathbf{X}(k) - \mathbf{X}_{\text{ref}}(k)|| > \delta$ **do**
5: Algorithm 2: Obtain the robots' pose measurements
6: Solve optimization problem in Eq. (3.60) or Eq. (3.69)
7: Publish $\mathbf{U}^*(0)$ to ROS topics of robots' velocity commands
8: Algorithm 3: Shift the prediction and control horizons, re-initialize the optimization problem with warm start
9: **end while**

are shown in the videos (Salimi Lafmejani and Berman, 2020b,a) but are not discussed here for the sake of conciseness. The other three scenarios, in which there are $m \in \{4, 5, 6\}$ robots, are described below:

Scenario 1 : Collision-free Position Swap of Four Robots. Four robots are initially located at the vertices of a square and are all oriented toward the center of the square (the origin). Each robot must switch positions with the robot at the opposite vertex along the

Algorithm 2 Obtain Robots' Pose Measurements

Input: m, k

Output: $\mathbf{X}_c(k)$

1: $i \leftarrow 1$, $\mathbf{X}_c(k) \leftarrow [\,]$

2: **for** $i \leq m$ **do**

3: Call odometry callback function of i-th robot

4: Read pose measurement $\mathbf{x}_i \leftarrow [x_i \quad y_i \quad \theta_i]$

5: Concatenate $\mathbf{X}_c(k)$ and $\mathbf{x}_i$

6: **end for**

7: Return $\mathbf{X}_c(k)$

Algorithm 3 Shift Horizons & Initialize Next Step

Input: $\mathbf{U}^*(0)$, $\mathbf{X}_c(k)$

Output: $\mathbf{U}(0)$, $\mathbf{X}(0)$

1: Update initial guess: $\mathbf{U}(0) \leftarrow \mathbf{U}^*(0)$

2: Update initial state: $\mathbf{X}(0) \leftarrow \mathbf{X}_c(k)$

3: Increment time step: $k \leftarrow k + 1$

4: Return $\mathbf{U}(0)$, $\mathbf{X}(0)$

diagonal of the square. The robots' goal headings are the same as their initial headings.

Scenario 2 : Collision-free Reconfiguration of Five Robots. Five robots start in a V-formation and must reconfigure into a symmetric formation about the y-axis, with their goal headings the same as their initial headings. The goal positions of the robots are assigned such that they must avoid many potential collisions with one another as they navigate to their goal poses.

Scenario 3 : Collision-free Position Swap of Six Robots. Six robots start at the vertices of a regular hexagon, facing toward the center (the origin). Each robot must switch posi-

Table 3.1: Sampling Time (s) and Prediction Horizon (Time Steps) for Each Scenario in the Simulations and Experiments.

	Simulation			**Experiment**		
Scenario	**1**	**2**	**3**	**1**	**2**	**3**
Sampling time	0.1	0.1	0.35	0.1	0.1	0.3
Prediction horizon	50	35	35	45	40	35

tions with the robot at the diametrically opposite vertex while avoiding collisions with one another. The robots' goal headings are the same as their initial headings. (*Note:* In the experiment for this scenario, the ranges of v_i and w_i were reduced to $v_i \in [-0.15 \;\; 0.15]$ (m/s), $w_i \in [-1.5 \;\; 1.5]$ (rad/s) to accommodate the relatively high density of robots in the testbed.)

Table 3.1 lists the sampling time T_s and the prediction horizon N_P that were used in each of these scenarios in the simulations and experiments. For these scenarios, and for the scenarios with $m = 2$ and $m = 3$ robots, the weighting matrices $\mathbf{Q}_m$ and $\mathbf{R}_m$ were defined as the Kronecker products of the matrices in Eq. (3.79) with the $m \times m$ identity matrix $\mathbf{I}_{m\times m}$:

$$\mathbf{Q}_m = \mathbf{Q} \otimes \mathbf{I}_{m\times m}, \quad \mathbf{R}_m = \mathbf{R} \otimes \mathbf{I}_{m\times m}, \tag{3.78}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are the weighting matrices for the scenario with $m = 1$ robot, defined as:

$$\mathbf{Q} = \text{diag}(1, 5, 0.1), \quad \mathbf{R} = \text{diag}(0.5, 0.05). \tag{3.79}$$

Figures 3.24–3.29 show snapshots of the robots at several time steps during simulations and experimental runs of the *Scenarios 1–3*. The robots' trajectories are plotted as colored

dashed lines, and their goal positions and goal orientations are indicated by colored circles and arrows, respectively, with a different color assigned to each robot. In the figure captions, the goal position coordinates are given in meters, and the goal orientations are in radians. The snapshots show that in each scenario, all robots successfully navigate to their goal poses without colliding with one another or becoming trapped in a deadlock configuration. In addition, Figs. 3.30–3.41 plot the time evolution of the robots' optimal control inputs, i.e. the optimal linear and angular velocities computed by our MPC method, during the simulations and experimental runs of the scenarios.

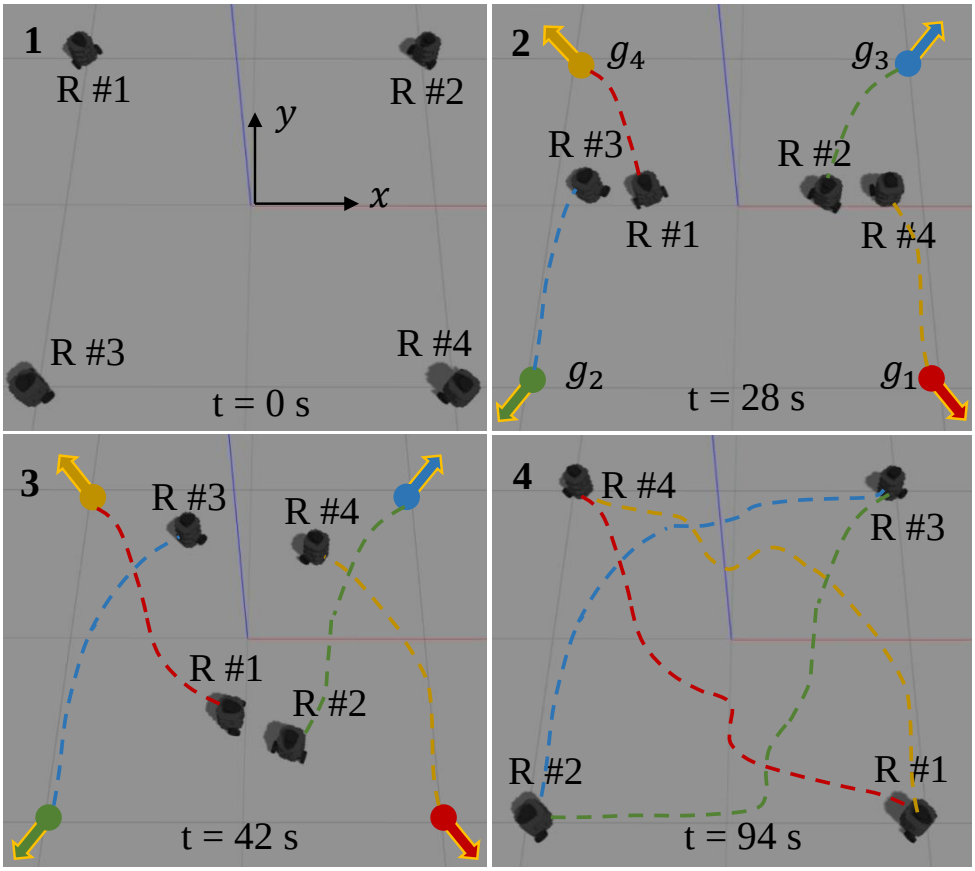


Figure 3.24: Snapshots of the Gazebo Simulation of *Scenario 1*. The robots' Goal Poses Are: $\mathbf{x}_{g_1} = [1 \ -1 \ -0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [-1 \ -1 \ -2.356]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [1 \ 1 \ 0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [-1 \ 1 \ 2.356]^{\mathrm{T}}$.

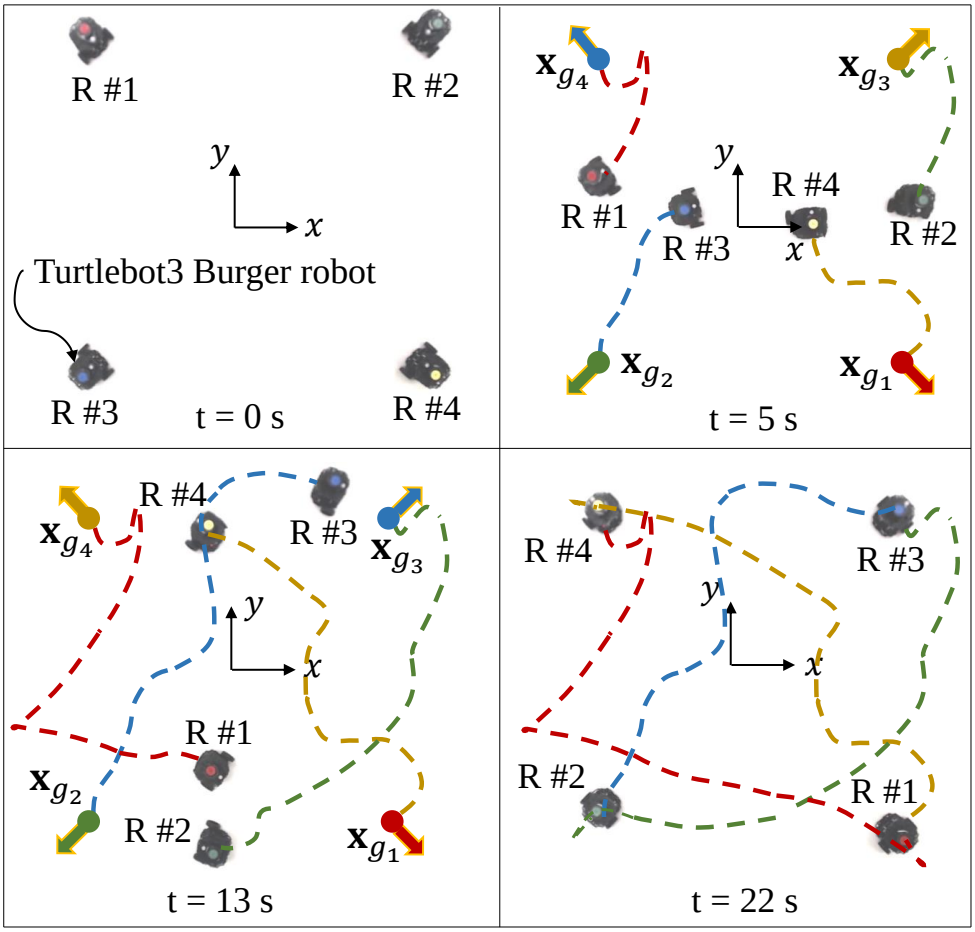


Figure 3.25: Snapshots of the Experimental Run of *Scenario 1*. The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [0.71 \ -0.71 \ -0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0.71 \ 0.71 \ -2.356]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [-0.71 \ -0.71 \ 0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.71 \ -0.71 \ 2.356]^{\mathrm{T}}$.

3.3.5 Analysis and Deadlock

In this section, we first give recommendations on tuning several parameters in our MPC method for the scenarios. Next, we describe types of deadlocks that can occur in each scenario and discuss how our method can resolve these deadlocks. Finally, we identify some limitations of our method.

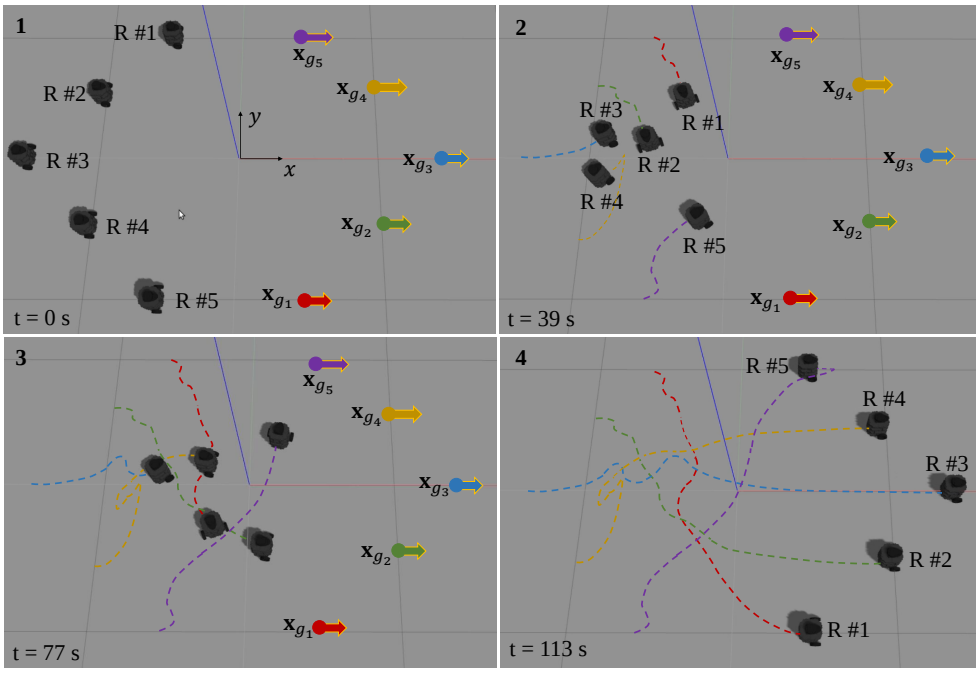


Figure 3.26: Snapshots of the Gazebo Simulation of *Scenario 2*. The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [1 \;\; 0.5 \;\; 0]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0 \;\; 0.75 \;\; -1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [-0.5 \;\; -0.5 \;\; 3.14]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [-0.5 \;\; -0.75 \;\; 0.785]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0.75 \;\; -0.75 \;\; -0.785]^{\mathrm{T}}$.

Tuning MPC Parameters

Proper selection of the MPC parameters T_s, N_P, N_C, $\mathbf{Q}$, and $\mathbf{R}$ is important, since they can affect both the controller performance and the computational complexity of the method. Furthermore, these parameters may need to be retuned for different multi-robot navigation problems.

Sampling time (T_s): The sampling time determines the resolution of the discrete-time kinematic model in Eq. (3.57) and the rate at which the optimization problem in Eq. (3.69) is solved. If T_s is too large, then the kinematic model will not accurately predict the robots' future states, and therefore potential collisions between robots may not be identified and the robots may navigate to points that are far from their goal positions. Moreover, collisions

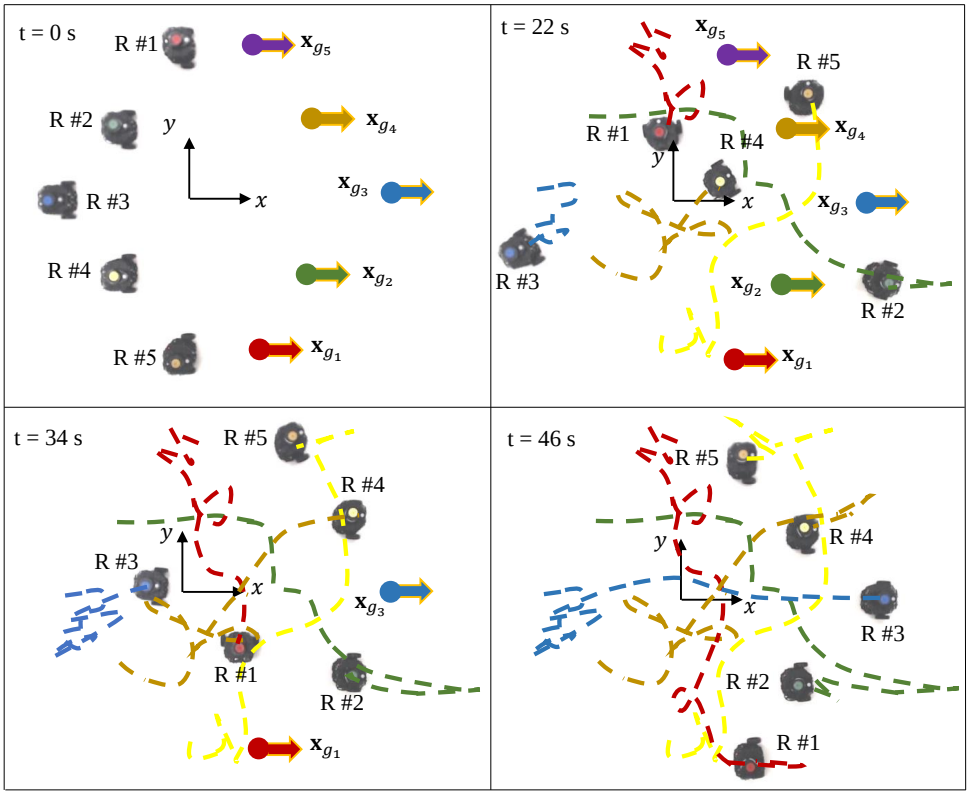


Figure 3.27: Snapshots of the Experimental Run of *Scenario 2*. The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [0.56 \;\; -0.71 \;\; 0]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0.84 \;\; -0.36 \;\; 0]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [1.18 \;\; 0 \;\; 0]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.84 \;\; 0.36 \;\; 0]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0.56 \;\; 0.71 \;\; 0]^{\mathrm{T}}$.

can occur between robots that follow optimal control velocities which are infrequently updated by the MPC method, and therefore are not adjusted for imminent collisions. On the other hand, a very small T_s will result in more accurate predictions of the future states and identification of all possible robot collisions, as discussed in the proof of Theorem 3.3.2. However, reducing T_s increases the computational cost to solve the optimization problem, as we observed in our simulations and experiments.

Prediction horizon (N_P): The prediction horizon determines the number of time steps over which the MPC method predicts the robots' future states. If N_P is too small, then col-

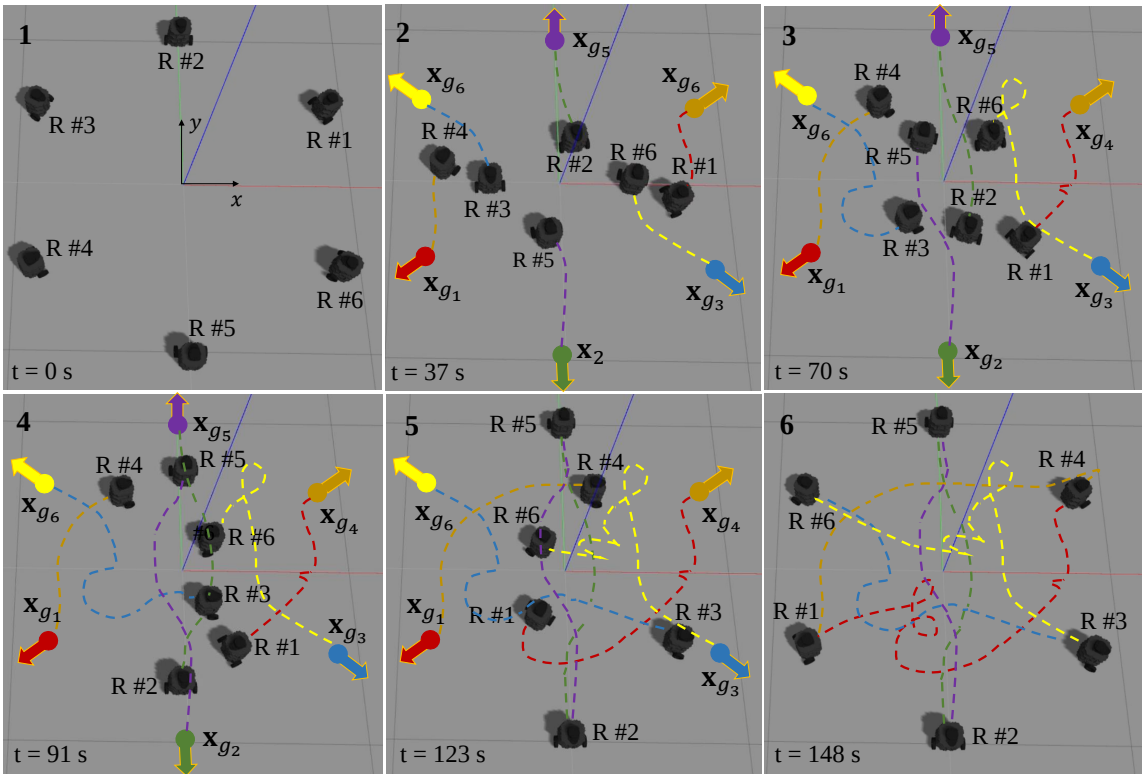


Figure 3.28: Snapshots of the Gazebo Simulation of *Scenario 3*. The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [-0.866 \ -0.5 \ -2.618]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0 \ -1 \ -1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [0.866 \ -0.5 \ -0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.866 \ 0.5 \ 0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0 \ 1 \ 1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_6} = [-0.866 \ 0.5 \ 2.618]^{\mathrm{T}}$.

lisions between approaching robots may not be predicted, or even if robots avoid collisions, they can become stuck in immobile deadlock configurations. If N_P is large, then the MPC method may produce overly conservative optimal control velocities that steer the robots away from each other prematurely, in an effort to prevent future collisions that are in fact too distant in time to accurately predict. Moreover, increasing N_P raises the computational cost to solve the optimization problem, similar to the effect of decreasing T_s. Based on our empirical observations, our recommendation is to set N_P between 25 and 110 time steps.

Control horizon (N_C): The control horizon, N_C, can be set to any value such that $N_C \leq$

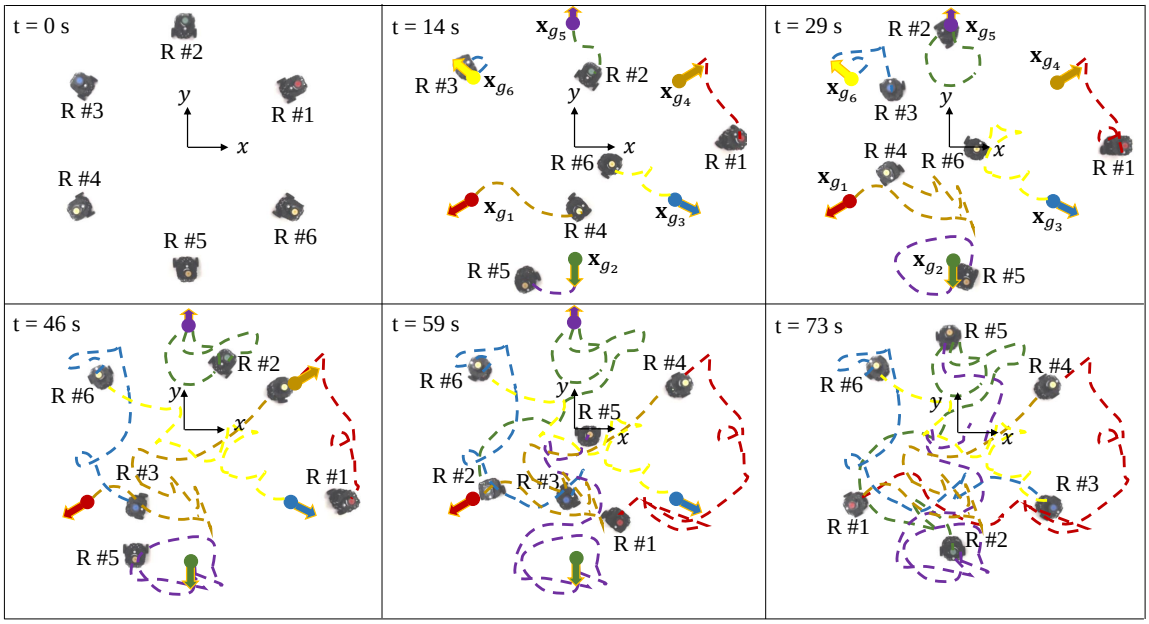


Figure 3.29: Snapshots of the Experimental Eun of *Scenario 3.* The Robots' Goal Poses Are: $\mathbf{x}_{g_1} = [-0.7 \ -0.4 \ -2.618]^{\mathrm{T}}$, $\mathbf{x}_{g_2} = [0 \ -0.8 \ -1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_3} = [0.7 \ -0.4 \ -0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_4} = [0.7 \ 0.4 \ 0.523]^{\mathrm{T}}$, $\mathbf{x}_{g_5} = [0 \ 0.8 \ 1.57]^{\mathrm{T}}$, $\mathbf{x}_{g_6} = [-0.7 \ 0.4 \ 2.618]^{\mathrm{T}}$.

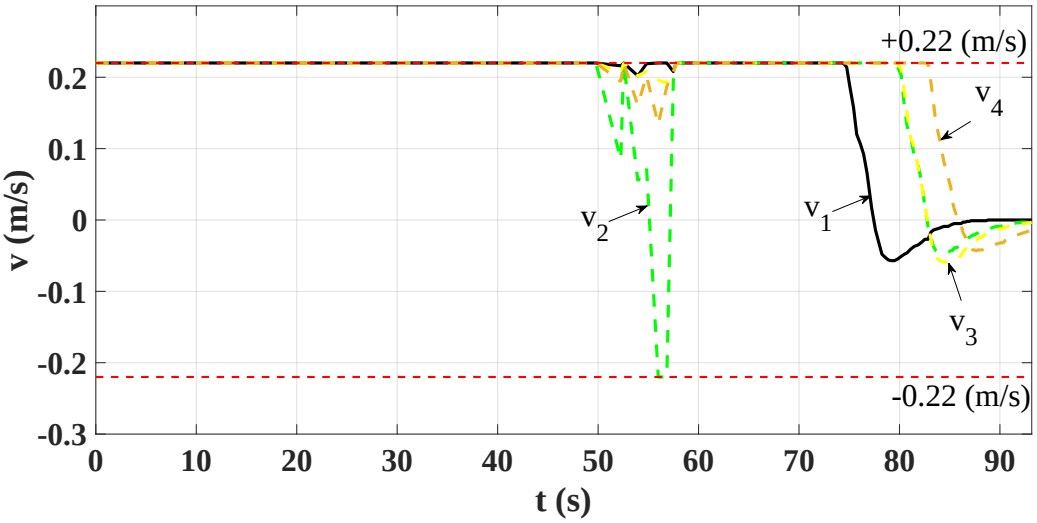


Figure 3.30: Optimal Robot Linear Velocities Over Time in the Simulation of *Scenario 1*

.

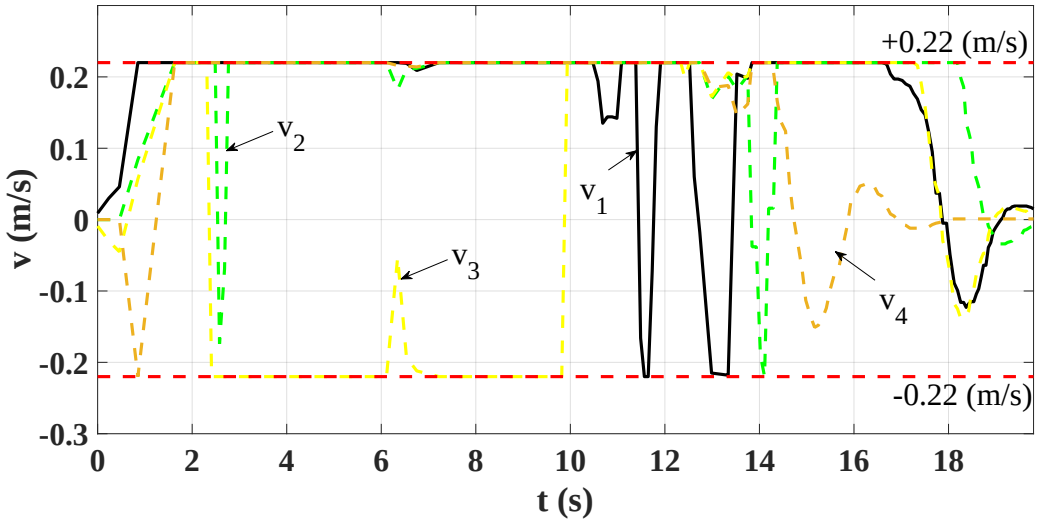


Figure 3.31: Optimal Robot Linear Velocities Over Time During the Experimental Run of *Scenario 1.*

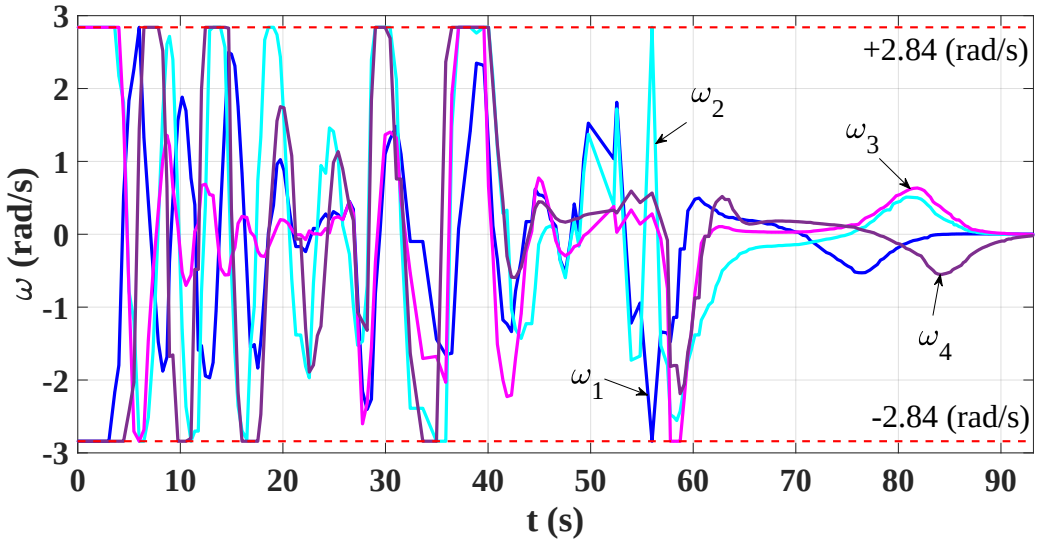


Figure 3.32: Optimal Robot Angular Velocities Over Time in the Simulation of *Scenario 1.*

N_P. In this section, we choose $N_C = N_P$ in all simulations and experiments. In MPC design, the smaller the control horizon, the lower the computational cost to solve the optimization problem. However, increasing the control horizon improves the accuracy of the

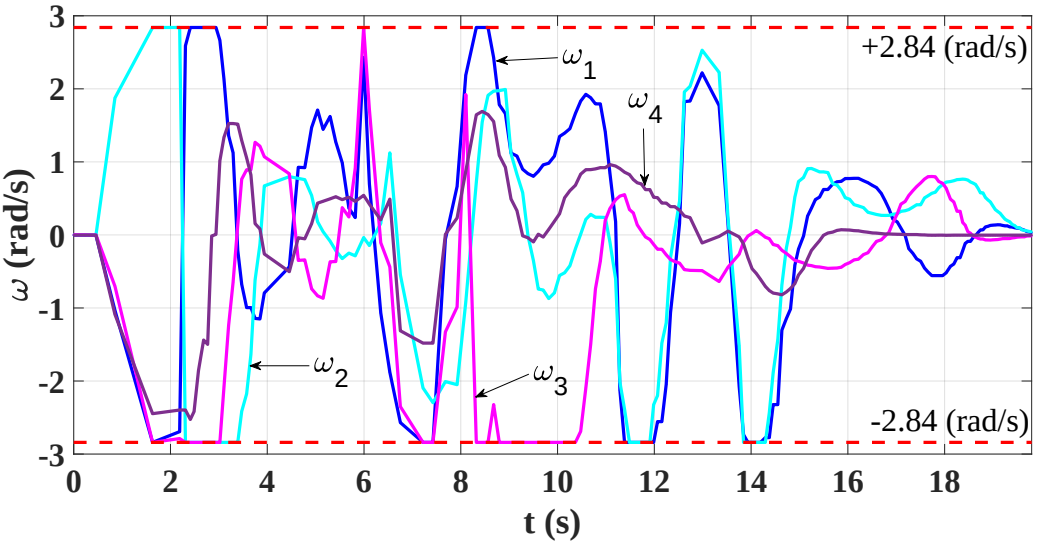


Figure 3.33: Optimal Robot Angular Velocities Over Time During the Experimental Run of *Scenario 1*.

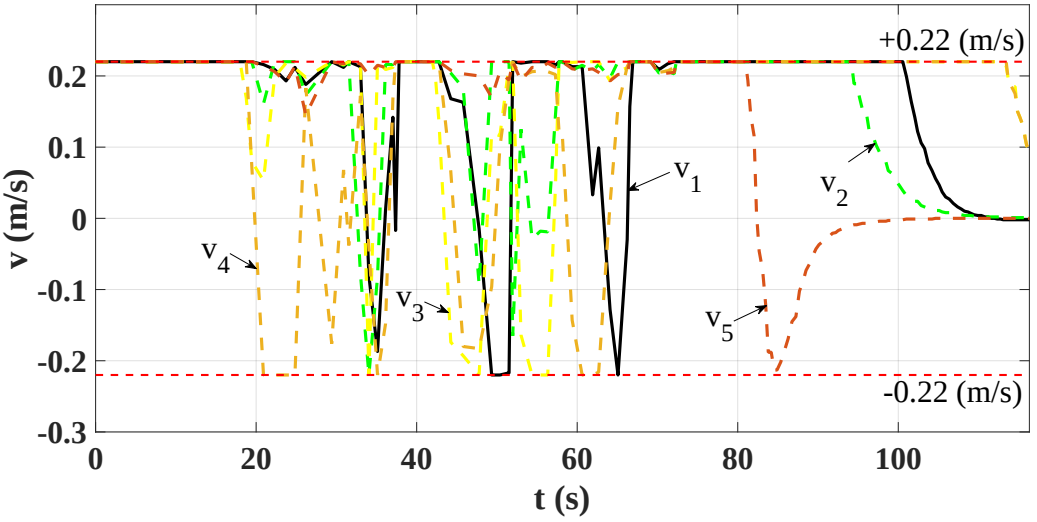


Figure 3.34: Optimal Robot Linear Velocities Over Time in the Simulation of *Scenario 2*.

predicted future robot states, since they are computed using a longer sequence of optimal control inputs. Therefore, if sufficient computational resources are available, we recommend to set $N_C = N_P$ in order to achieve the best possible predictions of the future robot states. Alternatively, we may define N_C as 10%–20% of N_P, as is standard in MPC design.

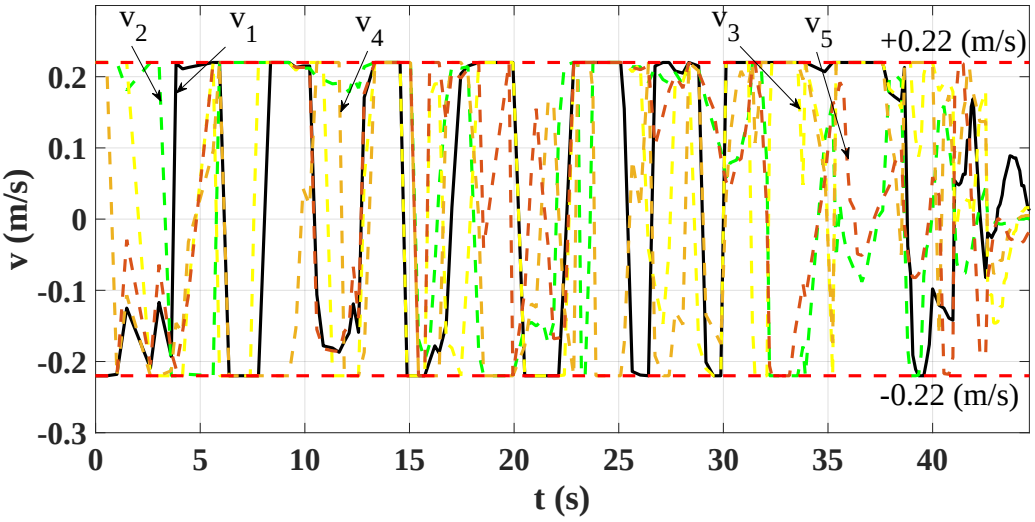


Figure 3.35: Optimal Robot Linear Velocities Over Time During the Experimental Run of *Scenario 2.*

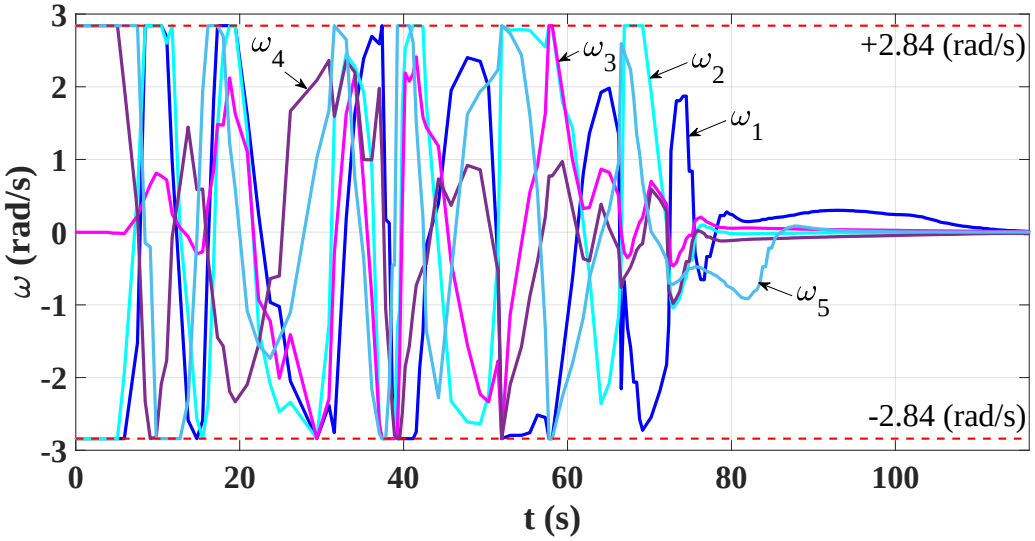


Figure 3.36: Optimal Robot Angular Velocities Over Time in the Simulation of *Scenario 2.*

Weighting matrices (Q and R): The weighting matrices **Q** and **R** are used to define the objective function of the optimization problem in the MPC formulation. The MPC

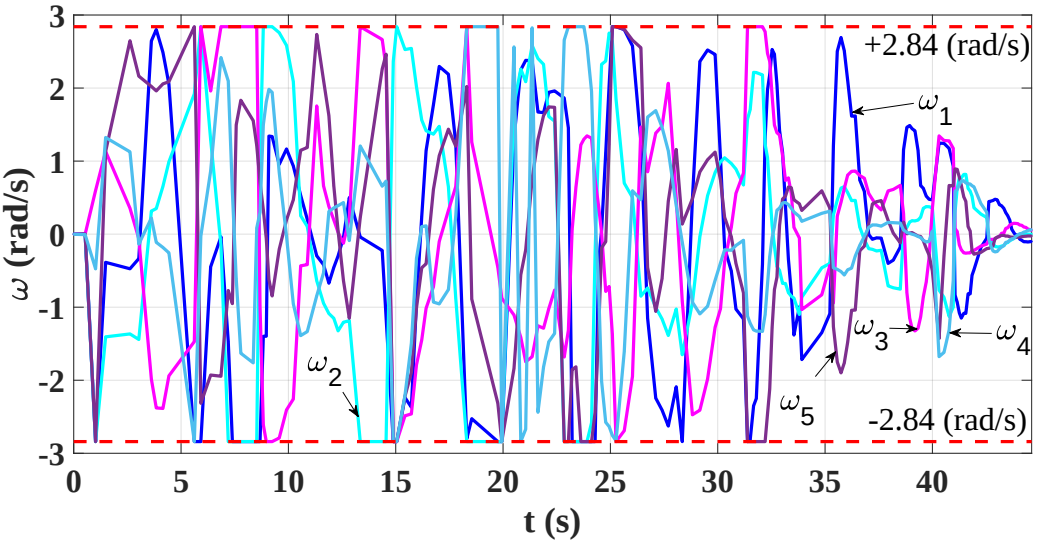


Figure 3.37: Optimal Robot Angular Velocities Over Time During the Experimental Run of *Scenario 2*.

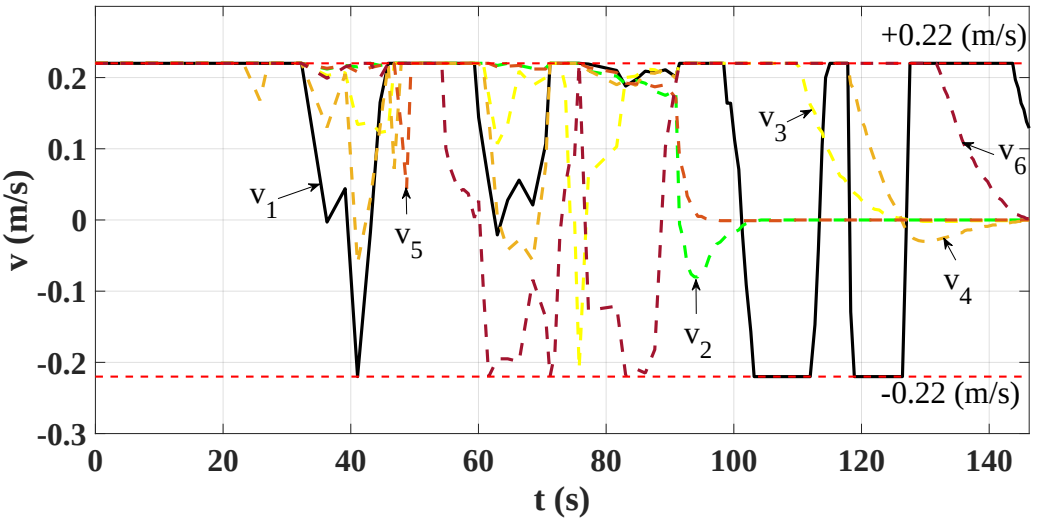


Figure 3.38: Optimal Robot Linear Velocities Over Time in the Simulation of *Scenario 3*.

must simultaneously satisfy two competing objectives: (1) the robots' poses, $\mathbf{X}$, should converge as closely as possible to their goal poses, $\mathbf{X}_{\text{ref}}$; and (2) the optimal control solutions should be sufficiently smooth to avoid aggressive control maneuvers. The weighting matrices establish the relative importance of these two objectives. One way to satisfy the

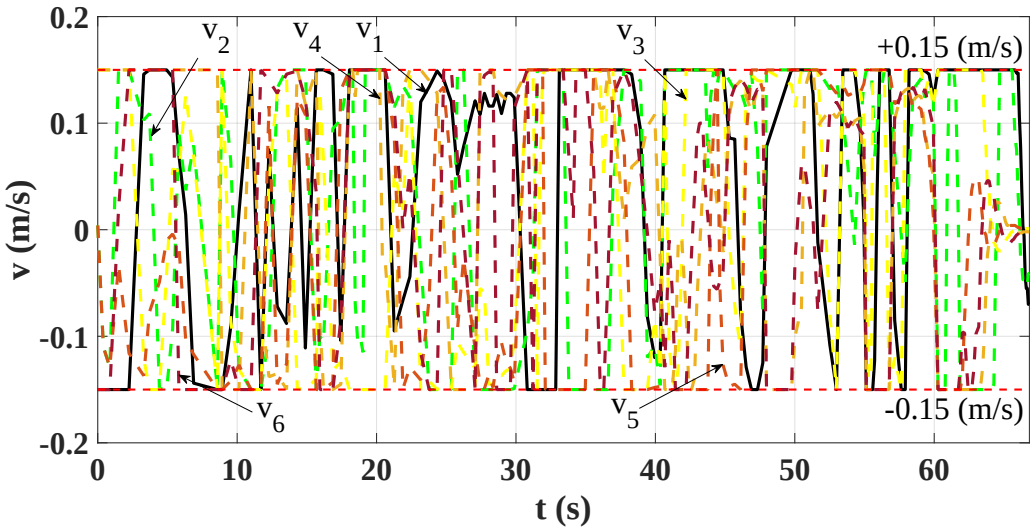


Figure 3.39: Optimal Robot Linear Velocities Over Time During the Experimental Run of *Scenario 3.*

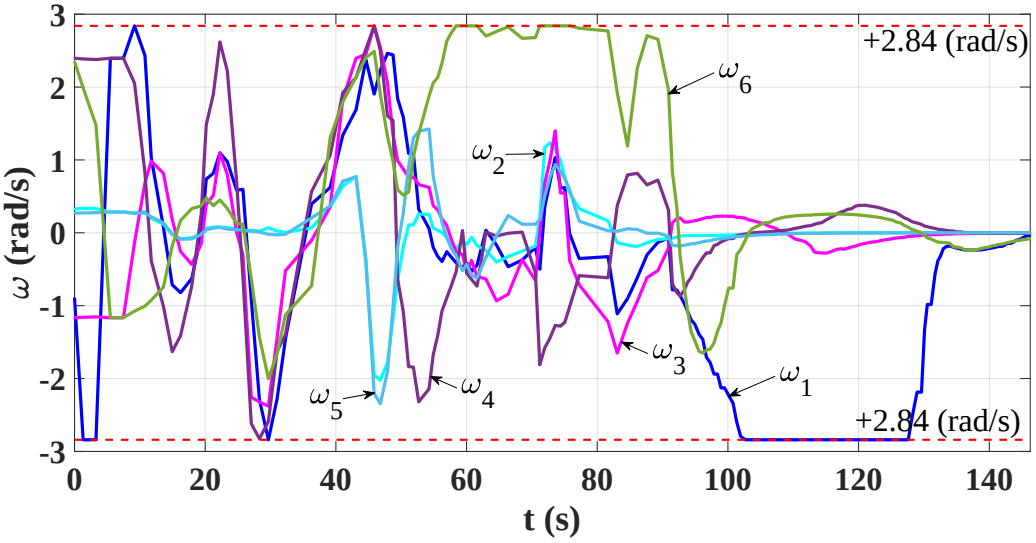


Figure 3.40: Optimal Robot Angular Velocities Over Time in the Simulation of *Scenario 3.*

requirement that $\mathbf{Q}$ is positive semi-definite and $\mathbf{R}$ is positive definite is to define them as diagonal matrices with positive entries q_{ii}, r_{ii} along the diagonal:

$$\mathbf{Q} = \text{diag}(q_{11}, q_{22}, q_{33}), \quad \mathbf{R} = \text{diag}(r_{11}, r_{22}). \tag{3.80}$$

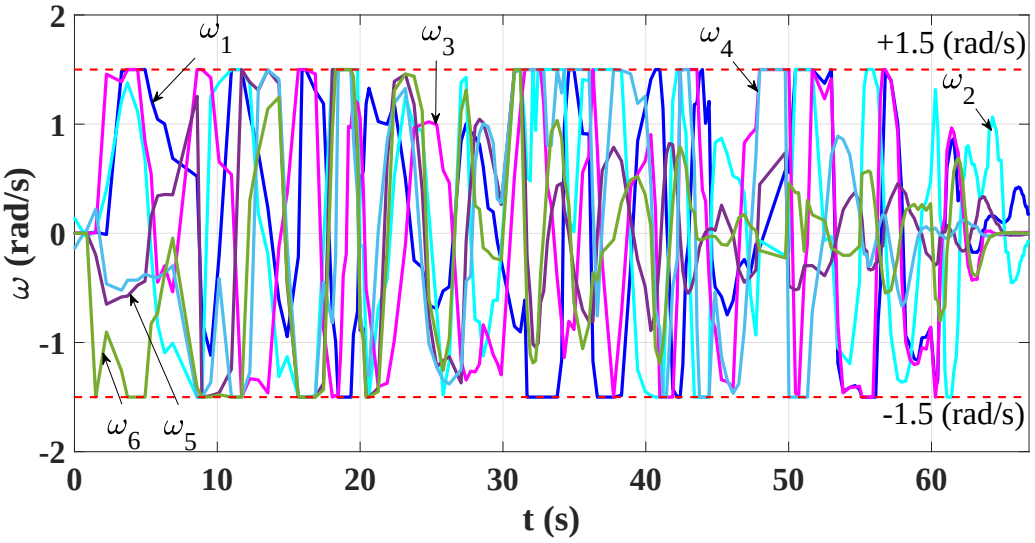


Figure 3.41: Optimal Robot Angular Velocities Over Time During the Experimental Run of *Scenario 3*.

In the navigation problem that we consider, it is more important that the robots reach their goal positions than their goal orientations. Therefore, q_{11} and q_{22} can be defined as much higher weights (e.g., 10 to 50 times larger) than q_{33}. For the same reason, we can define r_{11}, the weight corresponding to a robot's linear velocity, as a much higher value (e.g., 10 times larger) than r_{22}, the weight on its angular velocity, which only affects its orientation. We note that the relative values of the entries of the weighting matrices affect the trajectories of the robots.

Resolving Deadlocks

In multi-robot navigation scenarios, a deadlock occurs when a robot stops moving before it reaches its goal position, as a consequence of becoming trapped in an equilibrium point of the system dynamics (Zhou *et al.*, 2017). Deadlocks inevitably occur in multi-robot systems that use decentralized control approaches, since the robots can only respond to local information within their sensing or communication range (Grover *et al.*, 2019). Three

types of deadlocks that can occur during multi-robot navigation are described and illustrated in (Wang *et al.*, 2017). These types of deadlocks are re-illustrated in Fig. 3.42. Given those three scenarios, deadlock type 1 can occur in *Scenarios 1* and *3*; deadlock type 2 can occur in *Scenario 3*; and deadlock type 3 can occur in *Scenarios 2* and *3*. Our centralized MPC method can predict all three types of deadlocks, since it models the future states of all robots at each time step over the prediction horizon N_P. Given an appropriate selection of N_P and a sufficiently small sampling time T_s, as discussed earlier in this section, our method produces optimal control solutions for deadlock-free navigation.

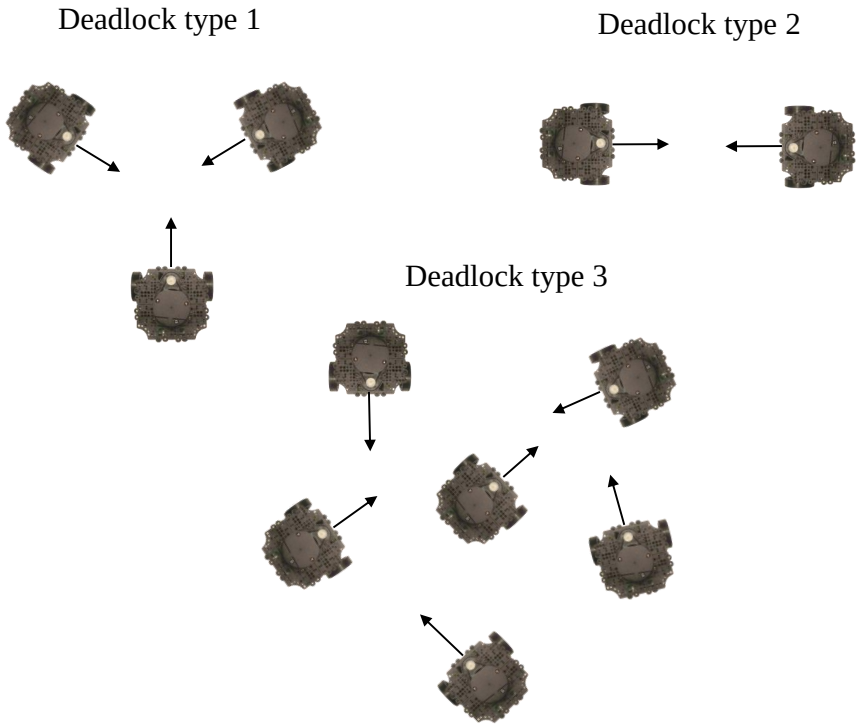


Figure 3.42: Three Types of Deadlocks that Can Occur in Our Multi-Robot Navigation Scenarios. Illustration is Based on Fig. 8 in (Wang *et al.*, 2017).

Limitations of the Proposed MPC Method

Although our MPC method can guarantee deadlock-free, collision-free navigation by multiple nonholonomic WMRs, it has certain limitations that should be considered in the selection of its parameters and its implementation. First, the scalability of the method to large numbers of robots is limited by the available computational resources, since the method uses a centralized approach to computing robot control inputs. Second, as demonstrated in (Worthmann *et al.*, 2015a) with numerical simulations, the quadratic loss function in the optimization problem fails to satisfy the bounds specified in Assumption 3.3.4 when the prediction horizon N_P is very large. This problem can be addressed by careful tuning of N_P and the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ for each navigation scenario. Third, errors in the robots' measured poses can arise from sensor noise in the robots' odometry readings, wheel skidding and sliding, and external mechanical disturbances such as unknown friction forces. In our experiments, the robots fuse their wheel encoder readings and IMU sensor data to improve the accuracy of the odometry, although this sensor fusion did not completely eliminate errors in the odometry measurements. Techniques such as direct decentralized integration (Nourmohammadi and Keighobadi, 2017) can improve real-time estimation of the robots' positions and orientations. In addition, a robust controller such as an H_∞ controller and an observer could be incorporated into our method to attenuate the effects of measurement noise and disturbances on the robots' trajectories.

3.3.6 Discussion

In this chapter, we proposed a nonlinear MPC method for collision-free and deadlock-free navigation by multiple nonholonomic WMRs. This method incorporates an optimization problem for computing velocity control inputs that drive the robots to goal poses while avoiding collisions. We analyzed the feasibility of the optimization problem and proved the

stability of the resulting controller. To reduce the computational complexity of the MPC, we did not include any stabilizing terminal constraints or costs in the optimization problem. We implemented the MPC using open-source software tools that provide a symbolic programming framework, which significantly accelerates the solution of the optimization problem. We demonstrated the effectiveness of our proposed method in both realistic 3D simulations and physical experiments for six different multi-robot navigation scenarios.

3.4 Trajectory Tracking Control of Multiple Nonholonomic Robots with Collision and Deadlock Avoidance

In this section, we present a centralized predictive control approach for safe navigation of multiple nonholonomic Wheeled Mobile Robots (WMRs) in environments with known obstacles. Our method is based on a Nonlinear Model Predictive Control (NMPC) framework and uses Barrier Functions (BFs) corresponding to the obstacles to construct Control Barrier Conditions (CBCs), which are included as constraints in the NMPC optimization problem. We prove that the method stabilizes the robots to target trajectories while preventing inter-robot collisions, robot-obstacle collisions, and deadlocks among the robots during navigation. We validate the effectiveness of our approach at enforcing safe multi-robot navigation in 3D simulations of three different trajectory tracking scenarios with four nonholonomic WMRs and three static obstacles.

3.4.1 Problem Formulation

In this section, we describe the trajectory tracking control problem and formulate an NMPC method for navigation of a single WMR. Then, we extend this formulation to collision-free and deadlock-free navigation of multiple WMRs in environments with static obstacles that have known locations and geometries. We assume that there is a central

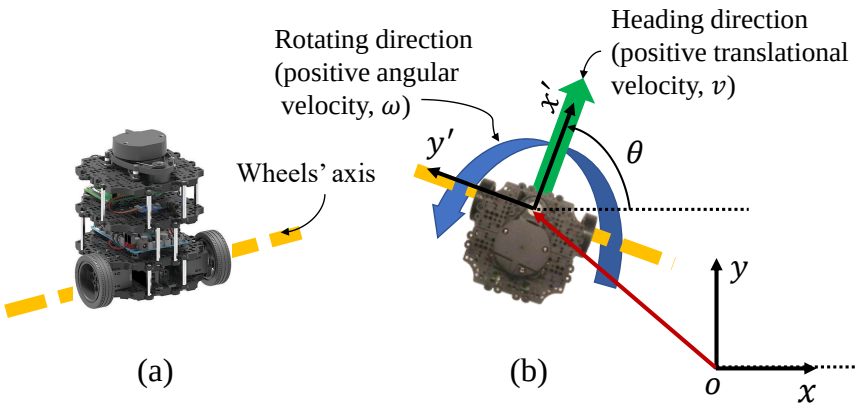


Figure 3.43: (a) Isometric View (Robotis, 2020) of the Turtlebot3 Burger Robot, and (b) Overhead View with Body-Fixed and Global Coordinate Frames.

supervisor that solves the optimization problem in our NMPC method and transmits the resulting optimal control inputs to the robots. The robots estimate their current poses relative to their initial poses (which are known) using odometry measurements and transmit these pose estimates to the central supervisor.

Figure 3.43(a) shows an isometric view of a commercial nonholonomic WMR, the Turtlebot3 (TB3) Burger robot (Robotis, 2020). The nonholonomic constraint does not allow the robot to move along its wheels' axis. As shown in Fig. 3.43(b), the origin of the robot's local coordinate frame $x' - y'$ is located at the position defined as $\bar{\mathbf{x}} = [x \quad y]^{\mathrm{T}}$ in the global coordinate frame $x - y$, and the robot's heading angle θ describes the rotation of the local frame about the z-axis with respect to the global frame. The positive direction of rotation is counter-clockwise. The control inputs of the robot are its linear velocity, v, and angular velocity, ω. We define the state vector representing the robot's pose as $\mathbf{x} := [\bar{\mathbf{x}} \quad \theta]^{\mathrm{T}}$ and the control input vector as $\mathbf{u} := [v \quad \omega]^{\mathrm{T}}$. We use the following modified version of a

discrete-time unicycle model to describe the kinematics of the nonholonomic WMR:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + \mathbf{g}(\mathbf{x}(k))\mathbf{u}(k)T_s \ , \\ \mathbf{g}(\mathbf{x}(k)) &= \begin{bmatrix} \cos(\theta(k)) - a\omega\sin(\theta(k)) & 0 \\ \sin(\theta(k)) - a\omega\cos(\theta(k)) & 0 \\ 0 & 1 \end{bmatrix} \ , \end{aligned} \tag{3.81}$$

where a is a small positive constant, T_s is the sampling time, and $k \in \mathbb{N} \cup \{0\}$ is an index for the time step. Moreover, $\mathbf{g}(\mathbf{x}(k))$ denotes the vector field representing the twist of the robot.

In the trajectory tracking control problem, the control inputs should stabilize the robot's position to a desired path and its heading to the direction of the tangent to the path. The states and control inputs of the robot may be subject to particular constraints. For instance, we may need to constrain the position of the robot if it must navigate within a bounded environment with dimensions $[x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}]$, and the robot's heading may need to be limited to the range $\theta \in [\theta_{\text{min}}, \theta_{\text{max}}]$. We also need to restrict the control inputs to the ranges $v \in [v_{\text{min}}, v_{\text{max}}]$ and $\omega \in [\omega_{\text{min}}, \omega_{\text{max}}]$ defined by the minimum and maximum values of the robot's linear and angular speeds. The constraints on the robot's angular velocity and heading angle should enable smooth navigation of the robot along the trajectory. Given a prediction horizon of N_P time steps, we define $\mathbf{e}(k) = \mathbf{x}(k) - \mathbf{x}_d(k)$, $k \in \{0, 1, 2, ..., N_P - 1\}$, as the error between the robot's current pose, $\mathbf{x}(k)$, and its target position at time k along the desired trajectory, $\mathbf{x}_d(k)$. Moreover, the desired control inputs, $\mathbf{u}_d(k)$, consist of the desired linear and angular velocities of the robot for tracking the trajectory. We define the error between the current and desired control inputs at time k as $\mathbf{e}_{\mathbf{u}}(k) = \mathbf{u}(k) - \mathbf{u}_d(k)$.

In a standard NMPC method for tracking control problems, the optimal control solutions are computed by minimizing the loss function $l(\mathbf{x}(k), \mathbf{u}(k)) = \|\mathbf{e}_{\mathbf{x}}(k)\|^2_{\mathbf{Q}} + \|\mathbf{e}_{\mathbf{u}}(k)\|^2_{\mathbf{R}}$. In this expression, the first and second terms denote $\mathbf{e}^{\mathrm{T}}_{\mathbf{x}}(k)\mathbf{Q}\mathbf{e}_{\mathbf{x}}(k)$ and $\mathbf{e}^{\mathrm{T}}_{\mathbf{u}}(k)\mathbf{R}\mathbf{e}_{\mathbf{u}}(k)$, where $\mathbf{Q} \in \mathbb{R}^{3\times3}$ is positive semi-definite and $\mathbf{R} \in \mathbb{R}^{2\times2}$ is positive definite. The minimization

problem is subject to the constraints imposed by the kinematic model of the robot and the bounds on its states and control inputs. We formulate the NMPC as:

$$\begin{aligned}
&\mathbf{u}^* = \text{argmin}_{\mathbf{u}} \sum_{k=0}^{N_P-1} l(\mathbf{x}(k), \mathbf{u}(k)) \\
&\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{g}(\mathbf{x}(k))\mathbf{u}(k)T_s \\
&\mathbf{x}_{\text{min}} \leq \mathbf{x}(k) \leq \mathbf{x}_{\text{max}} \\
&\mathbf{u}_{\text{min}} \leq \mathbf{u}(k) \leq \mathbf{u}_{\text{max}} \\
&\mathbf{x}(0) = \mathbf{x}_c,
\end{aligned} \tag{3.82}$$

where $\mathbf{x}(0)$ is the initial pose of the robot, is set to $\mathbf{x}_c$, the robot's current measurement of its pose via odometry readings. The solution of the optimization problem is a sequence of control inputs $\mathbf{u}^* \in \mathbb{R}^{2\times N_p}$. In accordance with MPC methods, only the first column of this matrix, $\mathbf{u}^*(0) \in \mathbb{R}^2$, is applied to the robot. The solution of Eq. (3.82) gives the optimal controller for trajectory tracking by a single WMR in an obstacle-free environment. In a trajectory tracking problem for multiple robots that navigate an environment with obstacles, the robots must avoid collisions with one another and with the obstacles while tracking their desired trajectories. We use *barrier certificates* to ensure the safety of the robots during navigation, i.e., to prevent the robots from colliding with other robots or obstacles.

Prior to describing our proposed method in the next section, we provide some definitions. Barrier Functions (BFs) enable the synthesis of safe controllers by guaranteeing the forward invariance of a specified *safe set* based on a Lyapunov-like condition. We define a safe set $\mathcal{C}$ that represents the free space of the domain, where the robot can move without colliding with an obstacle. This set is described by the super-level set of a continuous differentiable function $h(\bar{\mathbf{x}})$, which is known as a barrier function (Ames *et al.*, 2019). The

safe set $\mathcal{C}$, its boundary $\partial\mathcal{C}$, and the unsafe set $\mathcal{U}$ are defined as:

$$\begin{aligned}\mathcal{C} &= \{\bar{\mathbf{x}}(k)) \in \mathbb{R}^2 |\ h(\bar{\mathbf{x}}(k)) \geq 0\} \\ \partial\mathcal{C} &= \{\bar{\mathbf{x}}(k) \in \mathbb{R}^2 |\ h(\bar{\mathbf{x}}(k)) = 0\} \\ \mathcal{U} &= \{\bar{\mathbf{x}}(k) \in \mathbb{R}^2 |\ h(\bar{\mathbf{x}}(k)) \leq 0\}\end{aligned} \tag{3.83}$$

In order to guarantee the robot's safety, the control inputs must maintain the robot's position $\bar{\mathbf{x}}$ within the safe set $\mathcal{C}$. In other words, the set $\mathcal{C}$ must be *forward invariant* with respect to the vector field $\mathbf{g}(\mathbf{x}(k))$. To ensure forward invariance, the following inequality, called the Control Barrier Condition (CBC), should be satisfied for all $k \in \{1, ..., N_P\}$:

$$\mathcal{L}_{\mathbf{g}(\bar{\mathbf{x}}(k))} h(\bar{\mathbf{x}}(k))\mathbf{u}(k) + \gamma\Big(h(\bar{\mathbf{x}}(k))\Big) \geq 0, \tag{3.84}$$

where $\mathcal{L}_{\mathbf{g}(\bar{\mathbf{x}}(k))} h(\bar{\mathbf{x}}(k)) := \nabla h(\bar{\mathbf{x}}(k))^{\mathrm{T}} \mathbf{g}(\bar{\mathbf{x}}(k))$ are the Lie derivatives of the barrier function $h(\bar{\mathbf{x}})$ along the directions of the vectors $\mathbf{f}(\bar{\mathbf{x}}(k))$ and $\mathbf{g}(\bar{\mathbf{x}}(k))$, respectively. Moreover, $\gamma(\cdot)$ is a class $\mathcal{K}$ function, which is a strictly increasing function with $\gamma(0) = 0$. If one uses the standard unicycle kinematic model of a nonholonomic WMR as in (Salimi Lafmejani *et al.*, 2020b), then the angular velocity of the robot does not show up in the CBC (Xiao and Belta, 2019). Thus, we use the modified kinematic model in Eq. (3.81) so that the system has relative degree 1.

3.4.2 Safe NMPC Control Design and Analysis

The objective of safe trajectory tracking by multiple robots in environments with obstacles is to drive the robots to track their desired trajectories while avoiding collisions with one another and with static or dynamic obstacles in the environment. Figure 3.44 illustrates an example of trajectory tracking by multiple robots in which collisions would occur between two robots or between a robot and an obstacle if the robots followed their desired trajectories (dashed lines). In our model predictive control framework, the central supervisor calculates optimal control inputs for the robots that achieve trajectory tracking

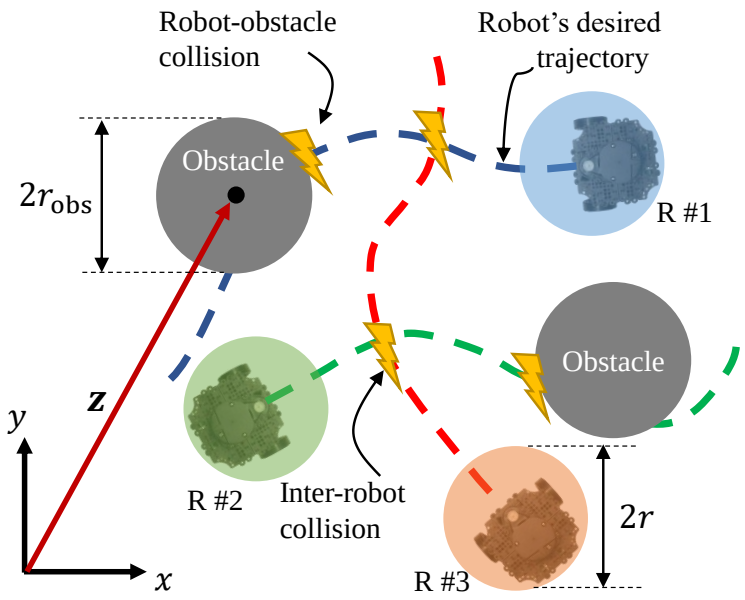


Figure 3.44: Illustration of Three Robots Following their Desired Trajectories, with Potential Collisions Between Two Robots (Collisions of R $\#1$ and R $\#3$, R $\#2$ and R $\#3$) and Between a Robot and an Obstacle (Collisions of R $\#1$ and R $\#2$ with Obstacles).

while preventing inter-robot and robot-obstacle collisions over the prediction horizon N_P. If a potential collision is detected along a robot's desired trajectory within the next N_P time steps, then the computed control inputs temporarily reroute the robot along a collision-free path and re-stabilize the robot to its desired trajectory once the collision is avoided.

We assume a scenario with m robots and s obstacles in the environment. We denote the state and control input vectors of the i-th robot, $i \in \{1, 2, ..., m\}$, by $\mathbf{x}_i$ and $\mathbf{u}_i$, respectively. We also define the state vector of the j-th obstacle, $j \in \{1, 2, ..., s\}$, by its position in the global frame, $\mathbf{z}_j = [x_{\text{obs},j} \quad y_{\text{obs},j}]^{\text{T}}$. If the obstacle is moving, then it has velocity vector $\dot{\mathbf{z}}_j = [\dot{x}_{\text{obs},j} \quad \dot{y}_{\text{obs},j}]^{\text{T}}$. Given the kinematic model of a single robot in Eq. (3.81), we define the following state space model of the entire system, describing the kinematics of all robots

and obstacles:

$$\begin{aligned}
&\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + (\mathbf{f}(\mathbf{x}_i(k)) + \mathbf{g}(\mathbf{x}_i(k))\mathbf{u}_i(k))T_s \\
&\mathbf{z}_j(k+1) = \mathbf{z}_j(k) + \dot{\mathbf{z}}_j(k)T_s \\
&i = 1, 2, ..., m, \quad j = 1, 2, ..., s.
\end{aligned} \tag{3.85}$$

In order to enforce inter-robot collision avoidance, we define the corresponding CBCs as follows:

$$\begin{aligned}
\nabla h^{\mathrm{T}}(\bar{\mathbf{x}}_i(k), \bar{\mathbf{x}}_{i'}(k))[\mathbf{g}^{\mathrm{T}}(\bar{\mathbf{x}}_i(k)) \;\; \mathbf{g}^{\mathrm{T}}(\bar{\mathbf{x}}_{i'}(k))]^{\mathrm{T}} \\
+ \gamma\Big(h(\bar{\mathbf{x}}_i(k), \bar{\mathbf{x}}_{i'}(k))\Big) \geq 0, \quad \forall i \neq i',
\end{aligned} \tag{3.86}$$

in which the BF for inter-robot collision avoidance and its gradient are defined as:

$$\begin{aligned}
h(\bar{\mathbf{x}}_i(k), \bar{\mathbf{x}}_{i'}(k)) &= ||\bar{\mathbf{x}}_i(k) - \bar{\mathbf{x}}_{i'}(k)||^2 - (r_i + r_{i'})^2 \\
\nabla h(\bar{\mathbf{x}}_i(k), \bar{\mathbf{z}}_j(k)) &= [2(x_i(k) - x_{i'}(k)) \;\; 2(y_i(k) - y_{i'}(k)) \\
&\quad - 2(x_i(k) - x_{i'}(k)) \;\; - 2(y_i(k) - y_{i'}(k))]^{\mathrm{T}},
\end{aligned} \tag{3.87}$$

where r_i and $r_{i'}$ are the radii of the smallest circles that contain the i-th and i'-th robots and are centered at the origins of their local coordinate frames. In order to enforce robot-obstacle collision avoidance, we define the corresponding CBCs as:

$$\begin{aligned}
&\nabla h^{\mathrm{T}}(\bar{\mathbf{x}}_i(k), \mathbf{z}_j(k))[\mathbf{0}^{\mathrm{T}} \;\; \dot{\mathbf{z}}_j^{\mathrm{T}}(k)]^{\mathrm{T}} \\
&\qquad + \nabla h^{\mathrm{T}}(\bar{\mathbf{x}}_i(k), \mathbf{z}_j(k))[\mathbf{g}^{\mathrm{T}}(\bar{\mathbf{x}}_i(k)) \;\; \mathbf{0}^{\mathrm{T}}]^{\mathrm{T}} \\
&\qquad + \gamma\Big(h(\bar{\mathbf{x}}_i(k), \mathbf{z}_j(k))\Big) \geq 0, \quad \forall i, j
\end{aligned} \tag{3.88}$$

in which the BF for robot-obstacle collision avoidance and its gradient are defined as:

$$\begin{aligned}
h(\bar{\mathbf{x}}_i(k), \bar{\mathbf{z}}_j(k)) &= ||\bar{\mathbf{x}}_i(k) - \bar{\mathbf{z}}_j(k)||^2 - (r_i + r_{\text{obs},j})^2 \\
\nabla h(\bar{\mathbf{x}}_i(k), \bar{\mathbf{z}}_j(k)) &= \\
&\quad [2(x_i(k) - x_{\text{obs},j}(k)) \quad 2(y_i(k) - y_{\text{obs},j}(k)) \\
&\quad -2(x_i(k) - x_{\text{obs},j}(k)) \quad - 2(y_i(k) - y_{\text{obs},j}(k))]^{\mathrm{T}},
\end{aligned} \tag{3.89}$$

where $r_{\text{obs},j}$ is the radius of the smallest circle that contains the j-th obstacle with the center of its local coordinate frame. We note that this formulation does not require the obstacles to be circular; depend on the shape of circular obstacles. In fact, one can consider $\mathbf{z}_j$ to be the position of the center of the smallest circle that contains an obstacle with arbitrary shape. In the CBCs (3.86) and (3.88), we set the function $\gamma(\cdot)$ to $\gamma(h(\cdot,\cdot)) := \beta h(\cdot,\cdot)$, where $\beta > 0$ is a small positive constant. Defining small values for β ensures that the robot moves slowly and avoids the boundary of the safe set, $\partial\mathcal{C}$, by a wide margin.

Given the CBCs (3.86) and (3.88), we now extend the NMPC method for trajectory tracking control of a single WMR to our proposed NMPC method for collision-free trajectory tracking control of multiple WMRs in an environment with obstacles. We define vectors that contain all m robots' positions and control inputs at time step k, $k \in \{0, 1, ..., N_P - 1\}$, by:

$$\begin{aligned} \mathbf{X}(k) &= [\mathbf{x}_1^{\mathrm{T}}(k) \quad \mathbf{x}_2^{\mathrm{T}}(k) \quad ... \quad \mathbf{x}_m^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{3m} \\ \mathbf{U}(k) &= [\mathbf{u}_1^{\mathrm{T}}(k) \quad \mathbf{u}_2^{\mathrm{T}}(k) \quad ... \quad \mathbf{u}_m^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{2m} \end{aligned} \tag{3.90}$$

In addition, we define the vectors of the robots' desired states and desired control inputs at each time step k as:

$$\begin{aligned} \mathbf{X}_d(k) &= [\mathbf{x}_{d_1}^{\mathrm{T}}(k) \quad \mathbf{x}_{d_2}^{\mathrm{T}}(k) \quad ... \quad \mathbf{x}_{d_m}^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{3m} \\ \mathbf{U}_d(k) &= [\mathbf{u}_{d_1}^{\mathrm{T}}(k) \quad \mathbf{u}_{d_2}^{\mathrm{T}}(k) \quad ... \quad \mathbf{u}_{d_m}^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{2m} \end{aligned} \tag{3.91}$$

Then, the loss function in our proposed NMPC method is defined as:

$$\begin{aligned} l(\mathbf{X}(k), \mathbf{U}(k)) &= \|\mathbf{E}_{\mathbf{X}}(k)\|_{\mathbf{Q}}^2 + \|\mathbf{E}_{\mathbf{U}}(k)\|_{\mathbf{R}}^2, \\ \mathbf{E}_{\mathbf{X}}(k) = \mathbf{X}(k) - \mathbf{X}_d(k), &\quad \mathbf{E}_{\mathbf{U}}(k) = \mathbf{U}(k) - \mathbf{U}_d(k). \end{aligned} \tag{3.92}$$

We define $\mathbf{X}_c$ as the vector containing all robots' measured states via odometry at time step k, $\mathbf{x}_{i,\text{min}}$ and $\mathbf{x}_{i,\text{max}}$ as lower and upper bounds on the i-th robot's state vector, and $\mathbf{u}_{i,\text{min}}$ and $\mathbf{u}_{i,\text{max}}$ as lower and upper bounds on its control input vector. Using the loss function l in Eq. (3.92), the discrete-time kinematic model in Eq. (3.85), the bounds on the

robots' state and control input vectors, and the collision avoidance constraints in Eq. (3.86) and Eq. (3.88), we formulate our NMPC method for safe multi-robot trajectory tracking control as follows:

$$
\begin{aligned}
&\mathbf{X}^*, \mathbf{U}^* = \text{argmin}_{\mathbf{X},\mathbf{U}} \sum_{i=1}^{m} \sum_{k=0}^{N_P-1} l(\mathbf{X}(k), \mathbf{U}(k)) \\
&\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{g}(\mathbf{x}_i(k))\mathbf{u}_i(k)T_s \\
&\text{Eq. (3.86)} \qquad \text{Inter-robot CBC} \\
&\text{Eq. (3.88)} \qquad \text{Robot-obstacle CBC} \\
&\mathbf{x}_{i,\text{min}} \leq \mathbf{x}_i(k) \leq \mathbf{x}_{i,\text{max}} \\
&\mathbf{u}_{i,\text{min}} \leq \mathbf{u}_i(k) \leq \mathbf{u}_{i,\text{max}} \\
&\mathbf{X}(0) = \mathbf{X}_c
\end{aligned}
\tag{3.93}
$$

To reduce the computational complexity of the optimization problem in Eq. (3.93), we solve it using the multiple-shooting method (Mehrez *et al.*, 2017a) and defining its decision variables as both the robot states and control inputs. The pseudo-codes in Algorithms 1 and 2 describe the implementation of our proposed NMPC method. It is crucial to carefully tune the MPC parameters, T_s, N_p, $\mathbf{Q}$, and $\mathbf{R}$, and the tuning parameter β in the CBCs, since they directly affect the controller's performance and the computational complexity of the optimization problem.

The stability of the proposed NMPC method can be established using proofs similar to those in our previous work (Salimi Lafmejani and Berman, 2021) if we can show that the associated optimization problem in Eq. (3.93), with CBCs as constraints, has recursive feasibility (Grüne and Pannek, 2017). Let us denote $\mathbb{X}_i$ and $\mathbb{U}_i$ as the feasible sets of states and control inputs for the i-th robot. Then we define $\mathbb{X} = \Pi_{i=1}^{m}\mathbb{X}_i$ and $\mathbb{U} = \Pi_{i=1}^{m}\mathbb{U}_i$ as the feasible sets of states and control inputs for the multi-robot system, and we use the simplified notation $\mathbb{Z} = \mathbb{X} \times \mathbb{U}$. Suppose that the initial state $\mathbf{X}(0)$ is known and

Algorithm 4 Main Function of NMPC Method

Input: $\mathbf{X}$, $\mathbf{X}(0)$, $\mathbf{X}_{\text{d}}$, $\mathbf{U}$, $\mathbf{U}_{\text{ref}}$, $\mathbf{Q}$, $\mathbf{R}$, N_P, m, $\mathbf{x}_{i,\text{min}}$, $\mathbf{x}_{i,\text{max}}$, $\mathbf{u}_{i,\text{min}}$, $\mathbf{u}_{i,\text{max}}$, $\mathbf{g}(\bar{\mathbf{x}}_i(k), \mathbf{u}_i(k))$, T_s, T_{max}

Output: $\mathbf{X}^*$, $\mathbf{U}^*$

1: Initialize ROS node, robot's odometry subscribers, and velocity command publishers
2: Symbolically formulate the optimization problem in Eq. (3.93) using CasADi (Andersson *et al.*, 2019)
3: **while** $kT_s < T_{\text{max}}$ **do**
4: Algorithm 2: Obtain the robots' pose and velocity measurements
5: Solve optimization problem in Eq. (3.93)
6: Publish $\mathbf{U}^*(0)$ to ROS topics of robots' velocity commands
7: Update initial guess: $\mathbf{U}(0) \leftarrow \mathbf{U}^*(0)$
8: Update initial state: $\mathbf{X}(0) \leftarrow \mathbf{X}_c$
9: Increment time step: $k \leftarrow k + 1$
10: **end while**

is a feasible state, i.e., $\mathbf{x}_i(0) \in \mathbb{X}_i$ for all $i = 1, ..., m$. Then, the constrained NMPC optimization problem in Eq. (3.93) has feasible solutions if and only if for all time steps $k, ..., k + N_P - 1$: **(a)** the optimal control solutions and corresponding predicted states are a subset of the feasible set, i.e., $\mathbf{X}^* \times \mathbf{U}^* \subseteq \mathbb{Z}$, and **(b)** the CBCs in Eqs. (3.86) and (3.88) are satisfied.

To prove statement **(a)**, we consider the kinematic model of each robot in Eq. (3.81). If the initial states and initial control inputs are in the feasible set, i.e., $\mathbf{X}(0) \times \mathbf{U}(0) \in \mathbb{Z}$, then the recursive feasibility of the optimization problem in Eq. (3.93) trivially holds. In other words, the set of optimal control solutions $\mathbf{U}^*(k)$ is not the empty set. To prove statement **(b)**, we consider a contradictory case. Suppose that there exists an optimal con-

Algorithm 5 Obtain Robots' Pose & Velocity Measurements

Input: m, k

Output: $\mathbf{X}_c$, $\mathbf{U}(k)$

1: $i \leftarrow 1$, $\mathbf{X}_c \leftarrow [\,]$

2: **for** $i \leq m$ **do**

3: Call odometry callback function of i-th robot

4: Read pose measurement $\mathbf{x}_i \leftarrow [x_i \quad y_i \quad \theta_i]$

5: Read velocity measurement $\mathbf{u}_i \leftarrow [v_i \quad \omega_i]$

6: Concatenate $\mathbf{X}_c$ and $\mathbf{x}_i$, $\mathbf{U}(k)$ and $\mathbf{u}_i$

7: **end for**

8: Return $\mathbf{X}_c$, $\mathbf{U}(k)$

trol solution for which two robots have a point in common on their trajectories that they will reach at the same time, or for which a robot's trajectory will intersect an obstacle. By the *Nyquist-Shannon* sampling theorem, a discrete sequence of samples can be used to adequately reconstruct a continuous-time signal given a small enough sampling time T_s. Thus, for a sufficiently small T_s, the NPMC method will identify the aforementioned point of inter-robot or robot-obstacle collision and compute an optimal admissible control solution that avoids producing this collision. As a result, satisfying the collision avoidance constraints at each time step is sufficient to guarantee collision-free navigation by all the robots. Moreover, since our method is centralized, deadlock-free navigation by the robots is guaranteed if we set a large enough prediction horizon N_P. In fact, the centralized framework is able to plan collision-free paths for the robots that allow them to escape a deadlock.

3.4.3 Simulation Results

We implemented our NMPC method on four simulated TB3 Burger robots (see Fig. 3.43) using ROS and the Gazebo robot simulator, programmed in Python. We simulated three different scenarios that may occur in automated warehouses and autonomous vehicle traffic: *Intersection Cross*, *Lane Merging*, and *Lane Changing*. In each scenario, the robots must track specified trajectories in environments with or without stationary obstacles. The videos of all simulated scenarios are available online at (Salimi Lafmejani *et al.*, 2021b).

For all three scenarios, we set the sampling time to $T_s = 0.05$ s, the prediction horizon to $N_P = 30$ time steps, and the parameter β in the CBCs to $\beta \in [0.1, 0.2]$. We selected the weighting matrices $\mathbf{Q}_m = \mathbf{Q} \otimes \mathbf{I}_{m\times m}$ and $\mathbf{R}_m = \mathbf{R} \otimes \mathbf{I}_{m\times m}$, where $\mathbf{Q} = \text{diag}(2, 2, 0.5)$, $\mathbf{R} = \text{diag}(0.5, 0.05)$, and $\otimes$ is the Kronecker product. The translational and rotational velocities of Burger robots are restricted to the ranges $v \in [-0.22 \quad 0.22]$ m/s and $\omega \in [-2.84 \quad 2.84]$ rad/s. Accordingly, we set the minimum and maximum values of the control inputs as $v_{\text{min}} = 0$ m/s, $v_{\text{max}} = 0.2$ m/s and $\omega_{\text{min}} = -1.5$ rad/s, $\omega_{\text{max}} = 1.5$ rad/s. (Note that the constraint $v_{\text{min}} = 0$ prevents the robots from moving backward.)

In the *Intersection Cross* scenario, the desired trajectory of one robot intersects those of the other three robots, which run in parallel. In the case with obstacles in the environment, shown in Fig. 3.45, each of three robots' desired trajectories intersects a circular obstacle. The videos of this scenario in (Salimi Lafmejani *et al.*, 2021b) show that when a collision is predicted at a point along a robot's desired trajectory, the robot is steered away from this trajectory to avoid the collision and thereafter is stabilized again to the trajectory. In the *Lane Merging* scenario, the robots' desired trajectories all lie along the same line, which in one case is tangent to three circular obstacles as shown in Fig. 3.46. The videos of this scenario demonstrate that robots are temporarily driven away from this line in order to prevent collisions with other robots and obstacles. Finally, in the *Lane Changing* scenario,

the robots exchange their desired trajectories, which run in parallel, at two times during the simulation, and circular obstacles may lie tangent to some trajectories as shown in Fig. 3.47. In the videos of this scenario, the robots are again steered away from their desired trajectories when confronted with an obstacle or another robot. In all scenarios, the NMPC method computes control inputs that successfully navigate the robots through their environment without collisions.

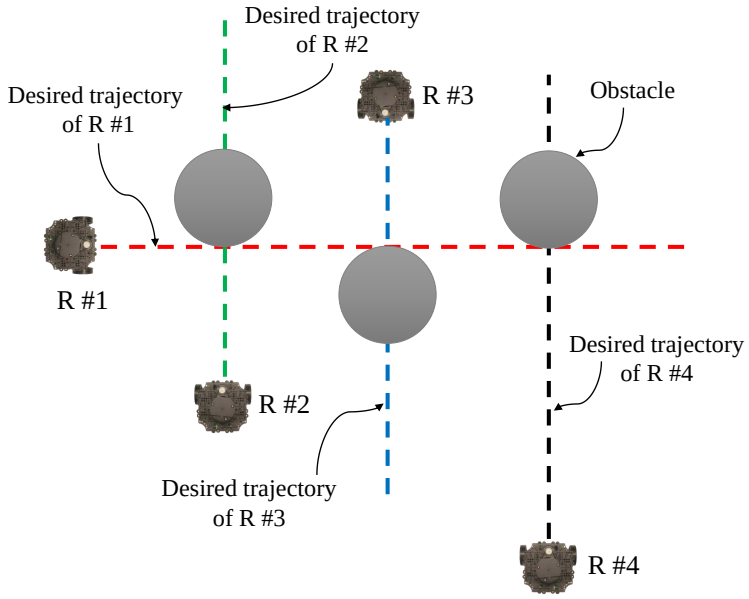


Figure 3.45: Overhead View of the Initial Configurations and Desired Trajectories of the Robots in an *Intersection Cross* Scenario with Obstacles (gray circles).

3.4.4 Discussion

In this section, we propose a control approach based on an NMPC method and barrier functions for safe navigation of multiple WMRs through environments with static obstacles. We incorporated Control Barrier Conditions (CBCs) into the constraints of the NMPC optimization problem in order to prevent inter-robot and robot-obstacle collisions. We tested the proposed method in simulation for three different trajectory tracking scenarios.

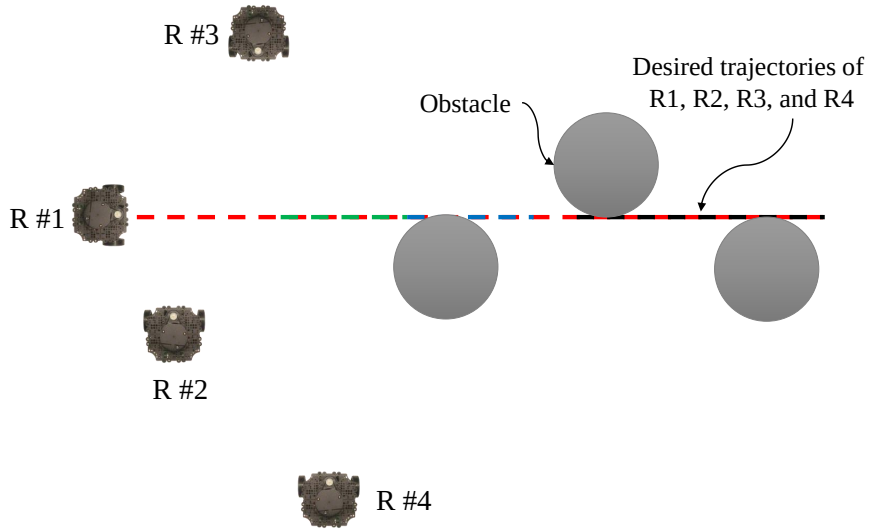


Figure 3.46: Overhead View of the Initial Configurations and Desired Trajectories of the Robots in a *Lane Merging* Scenario with Obstacles (Gray Circles).

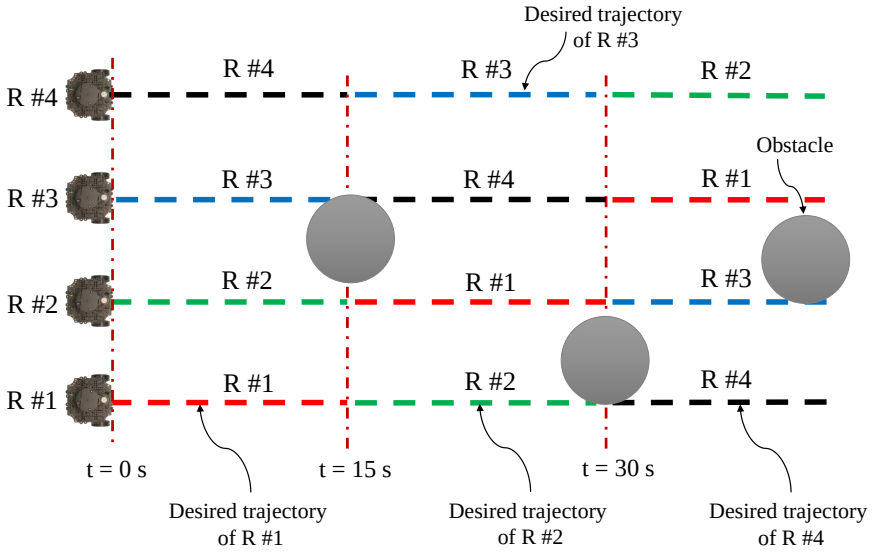


Figure 3.47: Overhead View of the Initial Configurations and Desired Trajectories of the Robots in a *Lane Changing* Scenario with Obstacles (Gray Circles).

3.5 Pose Stabilization of Multiple Nonholonomic Robots with Static and Dynamic Obstacle Avoidance

In this section, we present a decentralized control approach based on a Nonlinear Model Predictive Control (NMPC) method that employs barrier certificates for safe navigation of

multiple nonholonomic wheeled mobile robots in unknown environments with static and/or dynamic obstacles. This method incorporates a Learned Barrier Function (LBF) into the NMPC design in order to guarantee safe robot navigation, i.e., prevent robot collisions with other robots and the obstacles. We refer to our proposed control approach as NMPC-LBF. Since each robot does not have a priori knowledge about the obstacles and other robots, we use a Deep Neural Network (DeepNN) running in real-time on each robot to learn the Barrier Function (BF) only from the robot's LiDAR and odometry measurements. The DeepNN is trained to learn the BF that separates safe and unsafe regions. We implemented our proposed method on simulated and actual Turtlebot3 Burger robot(s) in different scenarios. The implementation results show the effectiveness of the NMPC-LBF method at ensuring safe navigation of the robots.

3.5.1 Control Problem Formulations

Nonlinear Model Predictive Control (NMPC)

Our proposed control approach utilizes a nonlinear MPC (NMPC) method. We use the following modified version of a discrete-time unicycle model that describes the kinematics of a nonholonomic wheeled mobile robot (WMR):

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))T_s \\ \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) &= \begin{bmatrix} \cos(\theta(k)) & -a\sin(\theta(k)) \\ \sin(\theta(k)) & a\cos(\theta(k)) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \end{aligned} \tag{3.94}$$

where k denotes the time step; T_s is the sampling time; a is a small positive constant; the state vector $\mathbf{x}(k) = [x(k) \quad y(k) \quad \theta(k)]^{\mathrm{T}}$ is the robot's pose, i.e., its position $\bar{\mathbf{x}} = [x \quad y]^{\mathrm{T}}$ and heading angle θ in the global coordinate frame at time step k; and the control input vector $\mathbf{u}(k) = [v(k) \quad \omega(k)]^{\mathrm{T}}$ contains the robot's control inputs, which are its linear

velocity v and angular velocity ω at time step k. If one uses the standard unicycle kinematic model of a nonholonomic WMR as in (Salimi Lafmejani *et al.*, 2020b), then the angular velocity of the robot does not show up in the barrier constraint (Xiao and Belta, 2019). Thus, we use the modified kinematic model in Eq. (3.94) so that the system has relative degree 1.

In an NMPC method, we first solve a nonlinear constrained optimization problem that minimizes a loss function $l(\mathbf{x}, \mathbf{u})$ over a prediction horizon of N_p time steps:

$$\begin{aligned} \mathbf{U}^* = \text{argmin}_{\mathbf{u}} & \sum_{k=0}^{N_p-1} l(\mathbf{x}(k), \mathbf{u}(k)) \\ & \mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))T_s \\ & \mathbf{x}_{\text{min}} \leq \mathbf{x}(k) \leq \mathbf{x}_{\text{max}} \\ & \mathbf{u}_{\text{min}} \leq \mathbf{u}(k) \leq \mathbf{u}_{\text{max}} \\ & \mathbf{x}(0) = \mathbf{x}_c \end{aligned} \tag{3.95}$$

where $\mathbf{x}_c$ is the robot's current odometry measurement of its pose and $\mathbf{U}^* \in \mathbb{R}^{2\times N_p}$ is a sequence of optimal control inputs for the future N_p time steps. The bounds $\mathbf{x}_{\text{min}}$ and $\mathbf{x}_{\text{max}}$ on the state vector can be imposed to restrict the robot to move within a specific region, and the bounds $\mathbf{u}_{\text{min}}$ and $\mathbf{u}_{\text{max}}$ on the control inputs are determined by the capabilities of the robot's actuators. If the control objective is to drive the robot to a target pose $\mathbf{x}_{\text{ref}}$, then the loss function can be defined as the sum of two quadratic terms that quantify the normed distance of the robot from the target pose and the control effort:

$$l(\mathbf{x}(k), \mathbf{u}(k)) = ||\mathbf{x}(k) - \mathbf{x}_{\text{ref}}||^2_{\mathbf{Q}} + ||\mathbf{u}(k)||^2_{\mathbf{R}}, \tag{3.96}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are square weighting matrices and $||\mathbf{x}||^2_{\mathbf{A}} \equiv \mathbf{x}^T\mathbf{A}\mathbf{x}$. Given the optimal control inputs $\mathbf{U}^*$, only the first control input $\mathbf{U}^*(0)$ is applied to the robot's actuators. Then, the time step is incremented from k to $k+1$, and optimization problem (3.95) is solved again.

Control Barrier Functions (CBFs)

We define *safety* as a criterion that prevents the control inputs from driving the robot into a collision with other robots or obstacles. Control barrier functions enable safety for control synthesis by providing forward invariance property of a specified *safe set* based on a Lyapunov-like condition. We define a safe set $\mathcal{C}$ that represents the free space of the domain, where the robot can move without colliding with an obstacle. This set is described by the superlevel set of a continuous differentiable function $h(\bar{\mathbf{x}}(k))$, which is known as a barrier function (Ames *et al.*, 2019, 2014):

$$\mathcal{C} = \{\forall k \in \mathbb{Z}_0,\ \bar{\mathbf{x}}(k) \in \mathbb{R}^n |\ h(\bar{\mathbf{x}}(k)) \geq 0\}. \tag{3.97}$$

In order to prevent collisions, the control inputs must maintain the robot's position $\bar{\mathbf{x}}$ within the safe set $\mathcal{C}$. In other words, the set $\mathcal{C}$ must be *forward invariant* with respect to the vector field $\mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k))$ defined in Eq. (3.94). The closed set $\mathcal{C}$ is called forward invariant if for every $\bar{\mathbf{x}}(0) \in \mathcal{C}$, $\bar{\mathbf{x}}(k) \in \mathcal{C}$ for all $k \in \mathbb{Z}_0$. Then, the safety certificate at time step k during robot navigation can be encoded as a Control Barrier Condition (CBC) as follows:

$$\mathcal{L}_{f(\bar{\mathbf{x}}(k)),\mathbf{u}(k))} h(\bar{\mathbf{x}}(k)) + \gamma\Big(h(\bar{\mathbf{x}}(k))\Big) \geq 0, \tag{3.98}$$

where $\mathcal{L}_{f(\bar{\mathbf{x}}(k),\mathbf{u}(k))} h(\bar{\mathbf{x}}(k))$, the Lie derivative of barrier function $h(\bar{\mathbf{x}})$ with respect to the vector field $\mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k))$, is defined as:

$$\mathcal{L}_{f(\bar{\mathbf{x}}(k)),\mathbf{u}(k))} h(\bar{\mathbf{x}}(k)) := \nabla h(\bar{\mathbf{x}}(k))^{\mathrm{T}} \mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k)) \tag{3.99}$$

and $\gamma(\cdot)$ is a class $\mathcal{K}$ function, that is, a strictly increasing function with $\gamma(0) = 0$. In (Zeng *et al.*, 2020), it was proved that adding the CBC (3.98) to the constraints of the MPC optimization problem (3.95) ensures the safety of the computed optimal control inputs.

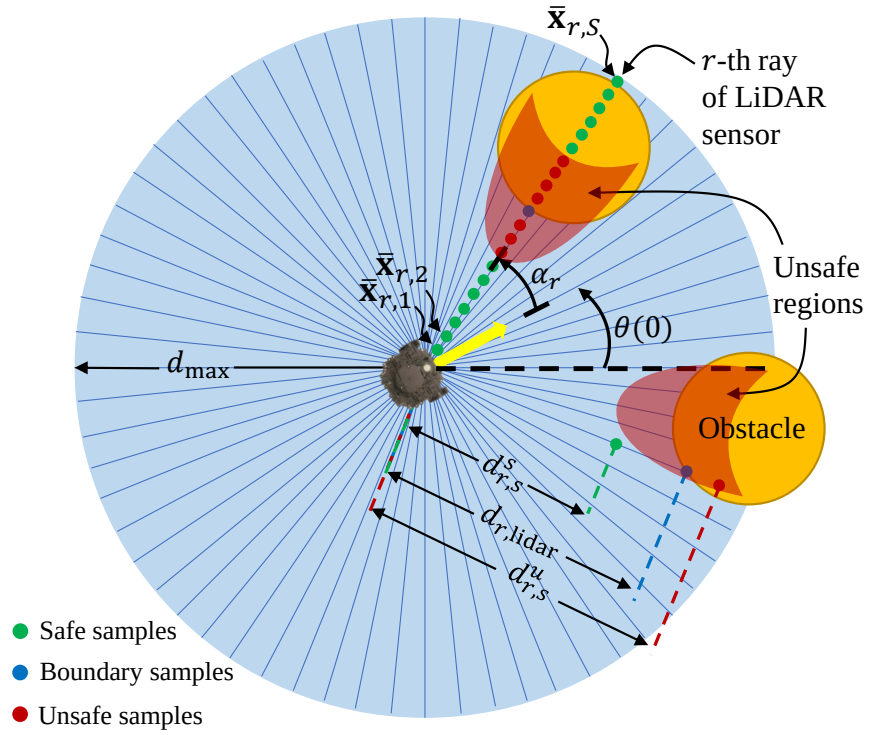


Figure 3.48: LiDAR Sensor Rays and Point Sampling in Safe and Unsafe Regions within the Sensing Range of the LiDAR. The Unsafe Samples are Taken from the Red Shaded Regions and the Safe Samples are in Areas Excluding the Unsafe Regions. $d^s_{r,s}$, $d_{r,\text{lidar}}$, $d^u_{r,s}$ Show the Distances of Robot to the Safe, Boundary, and Unsafe Samples on a Ray of LiDAR.

3.5.2 Decentralized NMPC-LBF Method

In our NMPC-LBF method, the BF is learned by training a DeepNN in real-time on each robot. The trained DeepNN provides an approximation of the true BF defined by $\hat{h}(\bar{\mathbf{x}}(k))$ given future predicted states of the robot. In our method, we conservatively choose the control horizon as $N_s = N_p$ to enforce the safety of the robot over all N_p future time steps.

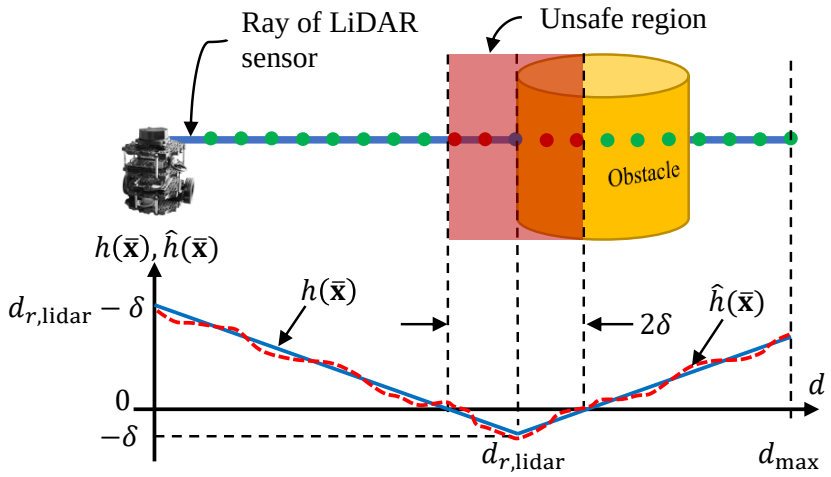


Figure 3.49: An Illustration of Ground-Truth BF, $h(\bar{\mathbf{x}})$, and Approximation of BF by the DeepNN, $\hat{h}(\bar{\mathbf{x}})$, on a Single Ray of LiDAR Along with a 2D Visualization of Safe and Unsafe Samples and a Sample on the Boundary of Obstacle. We Note that the Sample Points are Defined in $x - y$ Plane with the Height of Zero.

Data Sampling and Training DeepNN

The DeepNN should be trained to learn an approximation of the BF. To train the DeepNN, we need to collect samples from safe and unsafe regions that are observed by the robot during navigation. Figure 3.48 demonstrates our method to collect samples at each time instant. These samples are obtained in real-time only based on the LiDAR sensor readings and the current pose of the robot via odometry readings. We define R as the number of rays on the LiDAR sensor, $r = \{1, 2, ..., R\}$ as the index for the rays, α_r as the angle between the direction of the robot's heading and the r-th ray of the LiDAR, and $d_{r,\text{lidar}}(k)$ as the distance measured by the LiDAR in the direction of the r-th ray at the k-th time step. Then, we sample points within the sensing range of the robot along each ray of the LiDAR as illustrated in Fig. 3.48. Let us define $d_{\max}$ as the maximum distance that the robot can sense. Then, we take S number of samples on each ray of the LiDAR. We define $\bar{\mathbf{x}}_{r,s}(k) \in \mathbb{R}^2$ as the position in the global coordinate frame of the s-th sample, $s \in \{1, 2, ..., S\}$, on the r-th

ray at the k-th time step. Therefore, we have in total $N_{\text{samples}} = R \times S$ samples available in the dataset at each time step to train the DeepNN. By calculating $d_{r,s} = d_{\max} s/S$ as the distance of sample position $\bar{\mathbf{x}}_{r,s}(k)$ to the origin of the robot's local frame, we can readily compute the position of the sample as:

$$\begin{aligned} \bar{\mathbf{x}}_{r,s}(k) &= \bar{\mathbf{x}}_c + \mathbf{R}_z(\theta(k))\mathbf{d}_{r,s}(k) \\ \mathbf{R}_z(\theta(k)) &= \begin{bmatrix} \cos(\theta(k)) & -\sin(\theta(k)) \\ \sin(\theta(k)) & \cos(\theta(k)) \end{bmatrix} \\ \mathbf{d}_{r,s}(k) &= d_{r,s} \begin{bmatrix} \cos(\alpha_r(k)) \\ \sin(\alpha_r(k)) \end{bmatrix} \end{aligned} \tag{3.100}$$

where $\theta(k)$ is the heading angle of the robot at the k-th time step and $\mathbf{R}_z(\theta(k))$ is the standard rotation matrix that performs a rotation through angle $\theta(k)$ around about the positive z axis. This rotation matrix transforms vectors described in the local frame of the robot into vectors described in the global frame $x - y$. We note that $\bar{\mathbf{x}}_{r,s}$ is located in the safe region, the boundary of an obstacle, or an unsafe region within the sensing range of the LiDAR. Thus, the input for training the DeepNN is the set of all sampled points, defined as the vector $\mathbf{X}_s \in \mathbb{R}^{N_{\text{samples}} \times 2}$. We describe the ground-truth outputs as the value of the BF at the input samples:

$$h(\bar{\mathbf{x}}_{r,s}(k)) = |d_{r,s} - d_{r,\text{lidar}}(k)| - \delta, \tag{3.101}$$

where $\delta > 0$ denotes half of the width of unsafe region around the distance measured by the LiDAR and is set to a positive value greater than the radius of the smallest circle surrounding the robot. Thus, the input-output set of data for training the DeepNN can be described by $\mathcal{D} = \{\bar{\mathbf{X}}_s, \mathbf{H}_s\}$ in which each row of $\bar{\mathbf{X}}_s = \mathbb{R}^{N_{\text{samples}} \times 2}$ is the position vector of the sampled point $\bar{\mathbf{x}}_{r,s}$, $s = \{1, 2, ..., S\}$, and each row of $\mathbf{H}_s \in \mathbb{R}^{N_{\text{samples}}}$ is the corresponding ground-truth output value of BF, $h(\bar{\mathbf{x}}_{r,s})$. We employ the *incremental*

learning method to train the DeepNN that provides an approximation of the BF. We use a fully-connected DeepNN that is implemented in TensorFlow (Abadi *et al.*, 2016) and Keras (Chollet *et al.*, 2015) with 2 inputs (prediction of the robot's future states, i.e. $\bar{\mathbf{x}}(k)$) and $n_l = \{32, 32, 16, 16, 8\}$ as the number of nodes on 5 consecutive hidden layers, and 1 node at the output layer that gives a scalar value of the approximated BF.

Approximation of Barrier Function

We compute the approximation of the BF and its derivative given the trained DeepNN. Recalling that $\bar{\mathbf{x}}(k)$, where $k \in \{0, 1, ..., N_p - 1\}$, denotes the predicted future states of the robot, we can obtain a symbolic expression of the approximated BF for each predicted future state, given the activation functions on each node of the DeepNN and the optimal weights after each training. We give the symbolic expression of $\bar{\mathbf{x}}(k)$ as the input to the DeepNN and obtain a symbolic expression for the approximation of BF, i.e. $\hat{h}(\bar{\mathbf{x}}(k))$, at the output of the DeepNN. Figure 3.49 illustrates samples on a single ray of the robot's LiDAR and the change of the ground-truth BF and its approximation with respect to the distance to an obstacle. The approximated BF can be promptly computed by using the Forward Propagation (FP) technique. In contrast, computation of the derivative of approximated BF, i.e. $\nabla\hat{h}(\bar{\mathbf{x}}(k))$, is not trivial. To do so, we use the Back Propagation (BP) technique to calculate a symbolic expression for the gradient of $\hat{h}(\bar{\mathbf{x}}(k))$ with respect to the DeepNN's inputs, i.e., $\bar{\mathbf{x}}(k)$. In our method, we use $\tanh$ as the activation function of all nodes of the DeepNN, since $\tanh$ is a continuously differentiable function, which makes possible an analytical computation of gradient. We note that using ReLU activation function hinders computation of the gradient in an analytical closed-form. In (Zhao *et al.*, 2020), Bent-ReLU is used to solve this issue. The required computations in FP to obtain symbolic expression

of the approximated BF are:

$$\begin{aligned} &\mathbf{A}_l = \tanh(\mathbf{Z}_l), \quad \mathbf{Z}_l =^{l-1} \mathbf{W}_l^{\mathrm{T}} \mathbf{A}_{l-1} + \mathbf{b}_l \\ &\mathbf{A}_0 = \bar{\mathbf{x}}(k),\ \mathbf{A}_L = \hat{h}(\bar{\mathbf{x}}(k)), \quad l = 1, 2, ..., L \end{aligned} \tag{3.102}$$

where 0 is the index for the input layer, l is the index for hidden layers, L denotes the number of hidden layers, and $\tanh$ is the activation function on each node. The weight matrix including the weights on the connections between two consecutive layers $l-1$ and l that is defined by ${}^{l-1}\mathbf{W}_l$. Moreover, $\mathbf{Z}_l$ is an affine transformation of the previous layer's output $\mathbf{A}_{l-1}$ and $\mathbf{A}_l$ describes the output of l layer after applying the activation function on $\mathbf{Z}_l$. In the BP computations, we obtain a symbolic expression for the approximated gradient of the BF with respect to the inputs by:

$$\begin{aligned} \frac{\partial \hat{h}(\bar{\mathbf{x}}(k))}{\partial \bar{\mathbf{x}}(k)} &= \frac{\partial \mathbf{A}_L}{\partial \mathbf{Z}_L} . \frac{\partial \mathbf{Z}_L}{\partial \mathbf{A}_{L-1}} \cdots \frac{\partial \mathbf{A}_1}{\partial \mathbf{Z}_1} . \frac{\mathbf{Z}_1}{\partial \bar{\mathbf{x}}(k)} \\ \frac{\partial \mathbf{A}_l}{\partial \mathbf{Z}_l} &= 1 - \tanh^2(\mathbf{Z}_l), \quad \frac{\partial \mathbf{Z}_l}{\partial \mathbf{A}_{l-1}} =^{l-1} \mathbf{W}_l^{\mathrm{T}}. \end{aligned} \tag{3.103}$$

Incorporating BF into NMPC

Given the BF approximated by the DeepNN, we define sets that correspond to the safe region, its boundary, and the unsafe region of the environment, respectively:

$$\begin{aligned} &\mathcal{C} = \{\bar{\mathbf{x}} \mid \hat{h}(\bar{\mathbf{x}}) > \delta\},\ \ \partial\mathcal{C} = \{\bar{\mathbf{x}} \mid \hat{h}(\bar{\mathbf{x}}) = \delta\}, \\ &\mathcal{U} = \{\bar{\mathbf{x}} \mid \hat{h}(\bar{\mathbf{x}}) < \delta\} \end{aligned} \tag{3.104}$$

For all future N_p time steps, we calculate $\hat{h}(\bar{\mathbf{x}}(k))$ and $\partial \hat{h}(\bar{\mathbf{x}}(k))/\partial \bar{\mathbf{x}}(k) = \nabla \hat{h}(\bar{\mathbf{x}}(k))$ via FP and BF computations over the DeepNN, which also determines whether each future state will be in the set $\mathcal{C}$, $\partial\mathcal{C}$, or $\mathcal{U}$. The constrained optimization problem of our NMPC-

LBF method is formulated as:

$$
\begin{aligned}
\mathbf{U}^*, \mathbf{X}^* = \text{argmin}_{\mathbf{x},\mathbf{u}} & \sum_{k=0}^{N_P-1} l(\mathbf{x}(k), \mathbf{u}(k)) \\
& \mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))\mathbf{T}_s \\
& \mathbf{x}_{\text{min}} \leq \mathbf{x}(k) \leq \mathbf{x}_{\text{max}} \\
& \mathbf{u}_{\text{min}} \leq \mathbf{u}(k) \leq \mathbf{u}_{\text{max}} \\
& \mathbf{x}(0) = \mathbf{x}_c \\
& \nabla \hat{h}(\bar{\mathbf{x}}(k))^{\mathrm{T}} \mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k)) + \gamma\Big(\hat{h}(\bar{\mathbf{x}}(k))\Big) \geq 0 \text{ (CBC)}
\end{aligned}
\tag{3.105}
$$

where $\mathbf{X}^*$ is a sequence of optimal states. In order to reduce the computational complexity in this optimization problem, we use multiple shooting method and lift up the problem by considering both $\mathbf{x}$ and $\mathbf{u}$ as decision variables in the minimization. Moreover, we choose the function $\gamma(.)$ as $\gamma(\hat{h}(\bar{\mathbf{x}}(k))) = \beta\hat{h}(\bar{\mathbf{x}}(k))$ where $\beta > 0$ is a small positive value. We implement and solve this constrained nonlinear optimization problem via CasADi that is a powerful framework specialized for solving NMPC problems by providing symbolic expression of the problem. In the decentralized framework of our method, each robot solves the optimization problem described in Eq. (3.105) independently in which the BF is being learned by the DeepNN online in the loop at each time step. Figure 3.50 shows a block diagram of the NMPC-LBF method.

Implementation and Parameter Tuning

We describe implementation of our method using the pseudo codes in Algorithms 1 and 2. Algorithm 1 explains the NMPC-LBF method step by step. After initialization of the problem and formulating the optimization problem (line 1), the main loop (line 2 to 9) is executed online to collect the samples and train the DeepNN as described in Algorithm 2 given the current robot's pose from odometry and LiDAR measurements. We obtain optimal control inputs after solving Eq. (3.105) and apply it to the robot. Then, we shift the

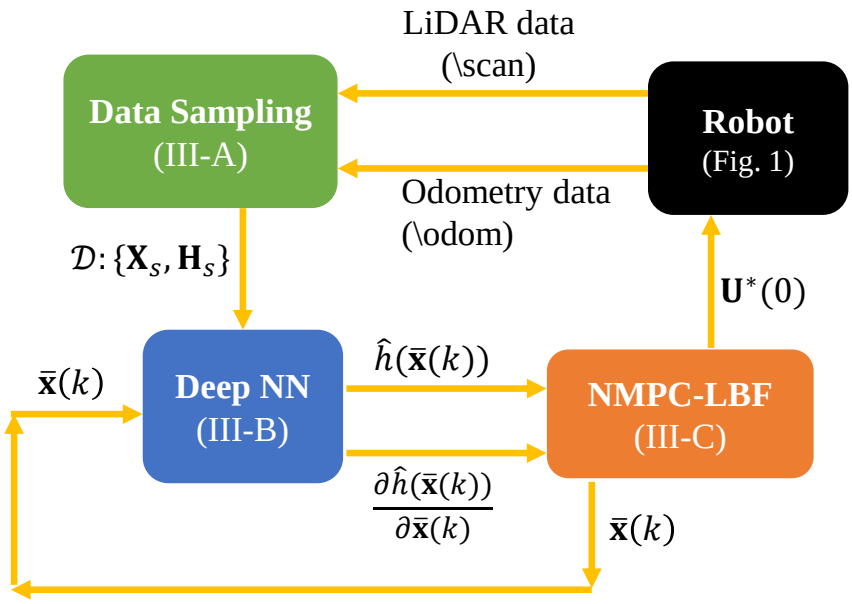


Figure 3.50: Block Diagram of the NMPC-LBF Method Described in Eq. (3.105). \Scan and \Odom are ROS Topics Through Which the States of the Robot and the LiDAR Measurements Would be Available.

prediction horizon and initialize the states and control inputs in the optimization problem with a warm start. This loop is executed up to the point that the distance of the robot to its goal becomes less than $e_{\text{ref}} = 0.1$. There are several parameters in the NMPC-LBF method that should be tuned carefully to achieve the expected performance of the robot to safely navigate in the environment. To implement the NMPC-LBF method, we recommend the following values for parameters to achieve the desired performance; prediction horizon as $N_p = 10 \sim 20$, sampling time as $T_s = 0.01 \sim 0.05$, weight matrices as $\mathbf{Q} = \text{diag}(5, 5, 0.05)$ and $\mathbf{R} = \text{diag}(2, 0.5)$, learning rate as $l_r = 0.01$, number of samples on each ray of LiDAR $S = 50$, half width of unsafe region $\delta = 0.2$ m, number of epochs $n_{\text{epochs}} = 20$ in the training of DeepNN, and $\beta = 0.1 \sim 0.2$ in the CBC.

Algorithm 6 NMPC-LBF Method

Input: $\mathbf{x}(0)$, $\mathbf{x}_{\text{ref}}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{x}_{\text{min}}$, $\mathbf{x}_{\text{max}}$, $\mathbf{u}_{\text{min}}$, $\mathbf{u}_{\text{max}}$, T_s, N_p

Output: $\mathbf{U}^*(0)$

1: Initialize ROS node, odometry and LiDAR scan subscribers, and velocity command publishers, symbolically formulate Eq. (3.105) using CasADi
2: **while** $||\mathbf{x}(k) - \mathbf{x}_{\text{ref}}(k)|| > e_{\text{ref}}$ **do**
3: Obtaining robots' pose and LiDAR measurements
4: Algorithm 2: Data Sampling and DeepNN training
5: Solve optimization problem in Eq. (3.105) to obtain $\mathbf{U}^*$
6: Publish $\mathbf{U}^*(0)$ to ROS topics of robots' velocity commands
7: Update initial guess: $\mathbf{U}(0) \leftarrow \mathbf{U}^*(0)$, $\mathbf{X}(0) \leftarrow \mathbf{x}_c$
8: Increment time step: $k \leftarrow k + 1$
9: **end while**

3.5.3 Simulation and Experimental Results

To evaluate the effectiveness of our method, we implement it on simulated TurtleBot3 (TB3) Burger robots (Robotis, 2020) in Gazebo and on an actual TB3 Burger robot. We simulated different scenarios for navigation of single and multiple robots in unknown environments. Due to space limitations, we only present results for two of these scenarios in this section. Videos of the experiments with the real robot and all simulations, including the simulation not discussed here, are available at (Salimi Lafmejani *et al.*, 2021a). In scenario 1, a single robot should stabilize to a goal pose in an environment with six unknown static obstacles. Figure 3.51 shows a snapshot of the initial configuration of the robot in simulation of scenario 1. The same scenario has been implemented in our previous work (Salimi Lafmejani *et al.*, 2020b) where the robot has to have a priori knowledge about

Algorithm 7 Data Sampling and DeepNN Training

Input: $\mathbf{x}_c$, $\mathbf{x}(k)$, $d_{r,\text{lidar}}$, $d_{\max}$, R, S, l_r, n_{epochs}

Output: $\hat{h}(\mathbf{x}(k)), \partial\hat{h}(\mathbf{x}(k))/\partial\mathbf{x}(k)$

$\bar{\mathbf{X}}_s = [\,]$, $r = 0$, $s = 0$
for $r < R$ **do**
  Computing $d_{r,s}$, α_r given r and s
  for $s < S$ **do**
    Computing $\mathbf{d}_{r,s}$ in Eq. (3.100)
    Obtaining sample positions $\mathbf{x}_{r,s}$ in Eq. (3.100)
    Collect data samples $\bar{\mathbf{X}}_s \leftarrow \bar{\mathbf{x}}_{r,s}$
  end for
end for
Calculating $\hat{h}(\bar{\mathbf{x}}(k))$ via FP
Calculating $\partial\hat{h}(\bar{\mathbf{x}}(k))/\partial\bar{\mathbf{x}}(k)$ via BP
Return $\hat{h}(\bar{\mathbf{x}}(k))$, $\partial\hat{h}(\bar{\mathbf{x}}(k))/\partial\bar{\mathbf{x}}(k)$

the position and geometry of the obstacles, unlike the NMPC-LBF method. In scenario 2, four robots should stabilize to their goal poses while avoiding collisions with one another and two unknown static obstacles in the environment. In scenario 2, four robots should stabilize to their goal poses in an environment with two unknown static obstacles. Figure 3.52 shows a snapshot of the initial configuration of the robots (R $\#i, \quad i = \{1, 2, 3, 4\}$) in simulation of scenario 2. Figure 3.53 shows distance of the robot to the goal and optimal control inputs over time in scenario 1. The plots for distance of the robots to their goal poses in scenario 2 are shown in Fig. 3.54. In experimental tests, we implemented the NMPC-LBF method on a single robot in three different scenarios in which the robot should stabilize to its goal pose in environment with one or two unknown static obstacle(s). We also tested

our control method in the case where the environment changes over time (Salimi Lafmejani *et al.*, 2021a).

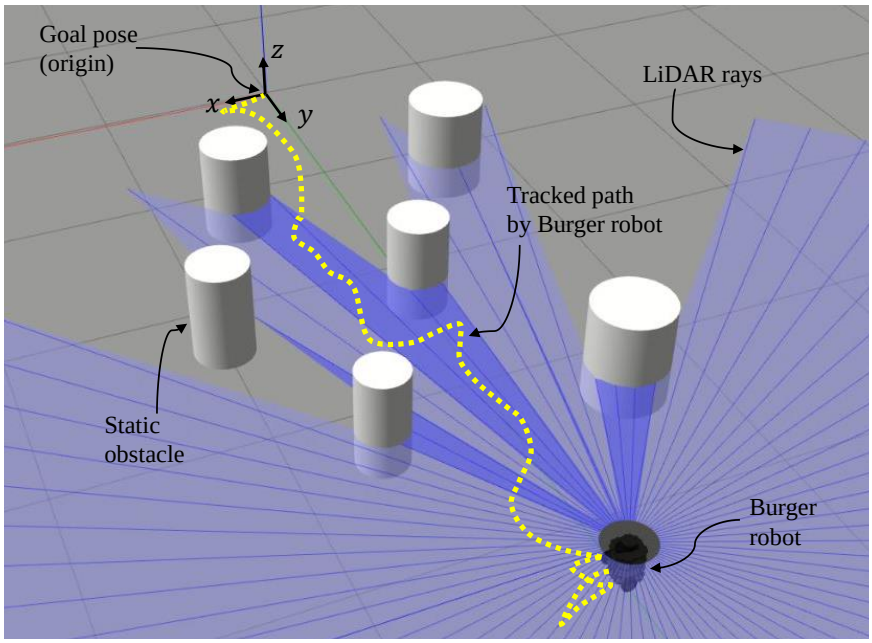


Figure 3.51: A Snapshot of the Simulation of Scenario 1 in Gazebo. The Robot Should Stabilize to the Goal Pose at the Origin of the Global Frame.

3.5.4 Analysis of Proposed Control Approach

Stability and Feasibility

The stability of the proposed control approach can be established using proofs similar to those in our previous work (Salimi Lafmejani and Berman, 2021) if we can show that the associated optimization problem in Eq. (3.105), with CBCs as constraints, has recursive feasibility (Grüne and Pannek, 2017). Let us denote $\mathbb{X}$ and $\mathbb{U}$ as the feasible sets of states and control inputs for the robot. We use the simplified notation $\mathbb{Z} = \mathbb{X} \times \mathbb{U}$. Suppose that $\bar{\mathbf{x}}(0)$ is known and is in a feasible state, i.e., $\bar{\mathbf{x}}(0) \in \mathbb{X}$. Then, the optimization problem in Eq. (3.105) has feasible solutions if and only if for all time steps $0, ..., N_p - 1$: **(a)** the

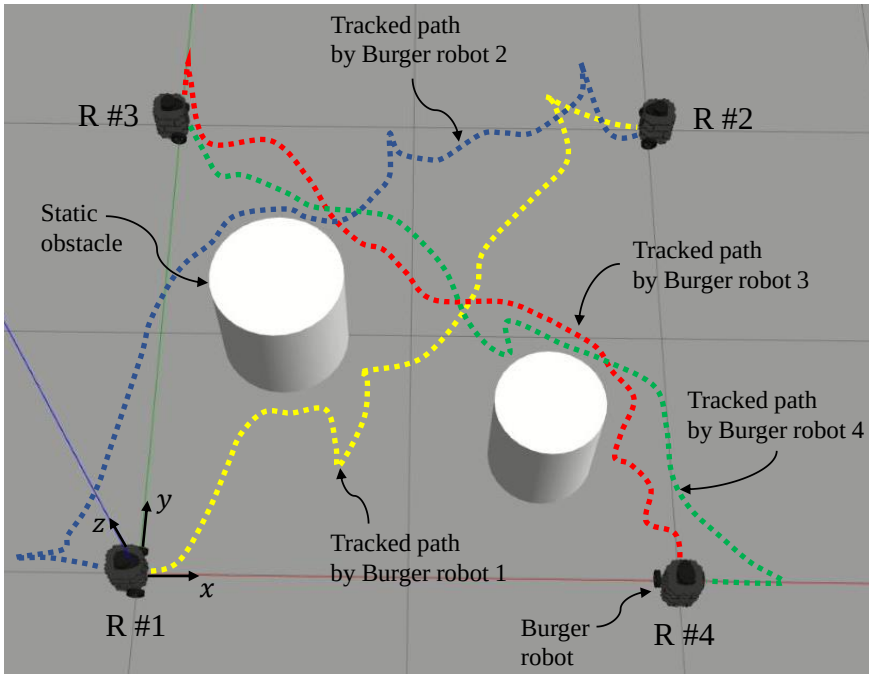


Figure 3.52: A Snapshot for Simulated Scenario 2 in Gazebo. All Robots Should Stabilize to Their Goal Poses while Avoiding Collisions with Other Robots and Static Obstacles.

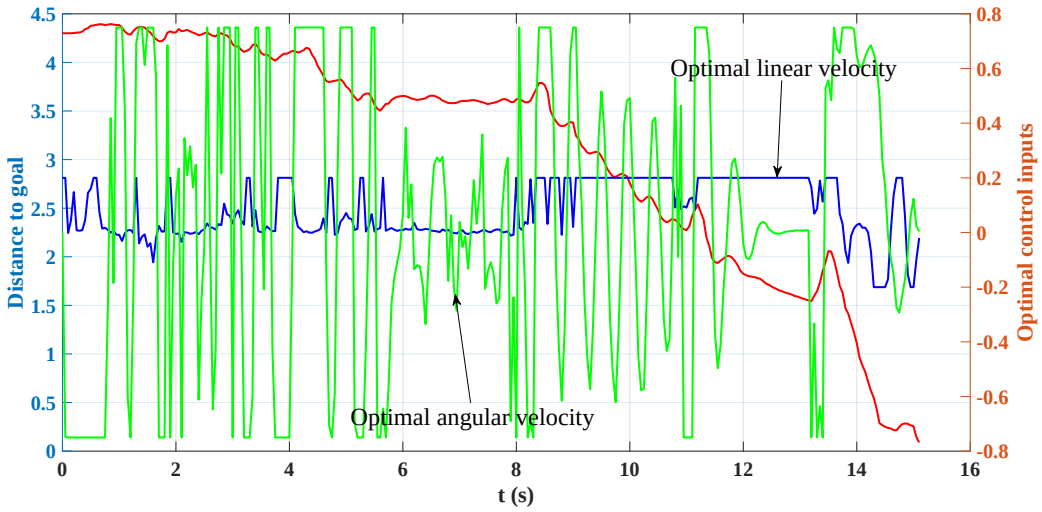


Figure 3.53: Plots of Distance to Goal and Optimal Control Inputs of the Robot in Scenario 1.

optimal control solutions and corresponding predicted states are a subset of the feasible set, i.e., $\mathbf{X}^* \times \mathbf{U}^* \subseteq \mathbb{Z}$, and **(b)** the CBCs in Eq. (3.98) are satisfied.

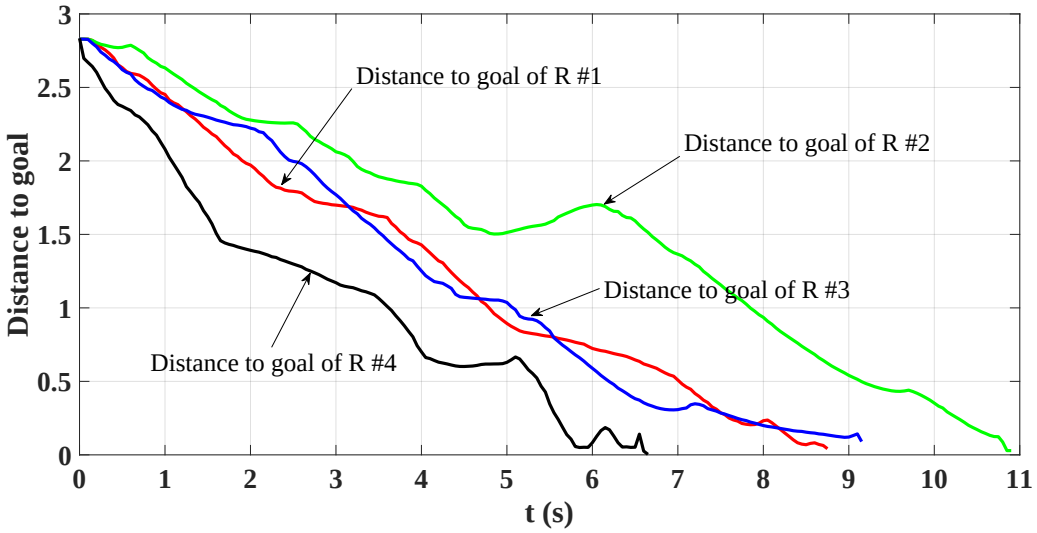


Figure 3.54: The Distance of Robots to Their Goal Poses in Scenario 2 Over Time.

To prove statement **(a)**, we consider the kinematic model of each robot in Eq. (3.94). If the initial state and initial control input are in the feasible set, i.e., $\bar{\mathbf{x}}(0) \times \mathbf{u}(0) \in \mathbb{Z}$, then the recursive feasibility of the optimization problem in Eq. (3.105) trivially holds. In other words, the sequence of optimal control inputs $\mathbf{U}^*$ is not the empty set. To prove statement **(b)**, we consider a contradictory case. Suppose that there exists an optimal control solution for which the robot and an obstacle has a point in common on their trajectories that they will reach at the same time, or for which a robot's trajectory will intersect an obstacle. By the *Nyquist-Shannon* sampling theorem, a discrete sequence of samples can be used to adequately reconstruct a continuous-time signal given a small enough sampling time T_s. Since the approximated BF can be computed for any given predicted states, for a sufficiently small T_s, the NMPC-LBF method will identify the aforementioned point of collision and compute an optimal admissible control solution that avoids producing this collision. As a result, satisfying the collision avoidance constraints at each time step is sufficient to guarantee collision-free navigation by the robot. Moreover, deadlock-free navigation by the robot is guaranteed theoretically if we set a large enough prediction horizon N_P and if there exist a trajectory by which the robot could avoid deadlock.

Conditions to Guarantee Safe Navigation

Let us define $e_{h(\bar{\mathbf{x}}(k))}$ and $e_{\nabla h(\bar{\mathbf{x}}(k))}$ as the errors between the approximated and ground-truth values of the BF and the partial derivative of the BF as:

$$\begin{aligned} e_{h(\bar{\mathbf{x}}(k))} &= h(\bar{\mathbf{x}}(k)) - \hat{h}(\bar{\mathbf{x}}(k)) \\ e_{\nabla h(\bar{\mathbf{x}}(k))} &= \partial h(\bar{\mathbf{x}}(k))/\partial \bar{\mathbf{x}}(k) - \partial \hat{h}(\bar{\mathbf{x}}(k))/\partial \bar{\mathbf{x}}(k), \end{aligned} \tag{3.106}$$

Proposition 1: We assume that the learned BF via DeepNN is σ_h-close to the true BF in Eq. (3.101). Then the CBC in Eq. (3.105) that uses learned BF and its derivative provided by FP and BP computations in DeepNN would guarantee safety of the robot navigation on the condition that the parameter β is tuned such that it satisfies the following inequality:

$$\beta \leq \frac{||e_{\nabla h(\bar{\mathbf{x}}(k))}||\ ||\mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k))||}{\sigma_h} \tag{3.107}$$

Proof. According to the assumption, we know that σ_h is the upper bound to the absolute value of the approximation error as $|e_h| < \sigma_h$. We also know that the ground-truth BF satisfies the CBC defined in Eq. (3.98). Given definition of the errors in Eq. (3.106), we can rewrite the CBC as:

$$(\nabla \hat{h}(\bar{\mathbf{x}}(k)) + e_{\nabla h(\bar{\mathbf{x}}(k))})^{\mathrm{T}} \mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k)) \geq -\beta(\hat{h}(\bar{\mathbf{x}}(k)) + e_{h(\bar{\mathbf{x}}(k))}) \tag{3.108}$$

which can be rearranged to separate the terms with the errors as follows:

$$\begin{aligned} \nabla \hat{h}(\bar{\mathbf{x}}(k))^{\mathrm{T}} \mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k)) \geq &- \beta \hat{h}(\bar{\mathbf{x}}(k)) - \beta e_{h(\bar{\mathbf{x}}(k))} \\ &- e^{\mathrm{T}}_{\nabla h(\bar{\mathbf{x}}(k))} \mathbf{f}(\bar{\mathbf{x}}(k), \mathbf{u}(k)). \end{aligned} \tag{3.109}$$

Now, we use the condition in Eq. (3.107) and the bounds on the BF approximation error to obtain:

$$\beta \leq \frac{||e_{\nabla h(\mathbf{x}(k))}||\ ||\mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))||}{\sigma_h} \leq \frac{|e^{\mathrm{T}}_{\nabla h(\mathbf{x}(k))} \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))|}{|e_h|} \tag{3.110}$$

which gives us the following inequalities:

$$-e^{\mathrm{T}}_{\nabla h(\bar{\mathbf{x}}(k))} h(\bar{\mathbf{x}}(k)) \leq \beta |e_h| \leq e^{\mathrm{T}}_{\nabla h(\bar{\mathbf{x}}(k))} h(\bar{\mathbf{x}}(k)). \tag{3.111}$$

If we consider both positive and negative values of e_h in Eq. (3.111), we can readily obtain:

$$e^{\mathrm{T}}_{\nabla h(\bar{\mathbf{x}}(k))}\mathbf{f}(\bar{\mathbf{x}}(k),\mathbf{u}(k)) + \beta e_h \geq 0. \tag{3.112}$$

Considering Eq. (3.109) and Eq. (3.112), we can achieve:

$$\begin{aligned}\nabla\hat{h}(\bar{\mathbf{x}}(k))^{\mathrm{T}}\mathbf{f}(\bar{\mathbf{x}}(k),\mathbf{u}(k)) \geq &-\beta\hat{h}(\bar{\mathbf{x}}(k)) - \beta e_{h(\bar{\mathbf{x}}(k))} \\ &- e^{\mathrm{T}}_{\nabla h(\bar{\mathbf{x}}(k))}\mathbf{f}(\bar{\mathbf{x}}(k),\mathbf{u}(k)) \\ \geq &-\beta\hat{h}(\bar{\mathbf{x}}(k)).\end{aligned} \tag{3.113}$$

The inequality in Eq. (3.113) is the CBC for the approximated learned BF and its approximated derivative, which concludes the safety of the robot navigation while using the DeepNN to construct the CBC. □

Limitations of Proposed NMPC-LBF Method

As a limitation of our decentralized method, a powerful computational resource on the robot is required to train the DeepNN and solve the optimization problem of NMPC-LBF in real-time. In turn, this allows us to increase the resolution of sampling points on each ray of the LiDAR to collect more data, which could preempt the over-fitting problem in training of the DeepNN. Another limitation of our method is the challenging task of tuning the parameters of the NMPC-LBF method, which significantly affects the safety and stability of the robot during navigation. Although inter-robot collision avoidance can be ensured in our method, there are some special cases in which collision avoidance between the robot and dynamic obstacles is not guaranteed. If the velocity vector of a dynamic obstacle aligns with the velocity vector of the robot and has a greater magnitude, then they might collide since the robot's control inputs are constrained and there are no known constraints on the obstacle's dynamics. We provide a solution to this limitation as future work in section 4. Due to the generalization error in learning of the BF by the DeepNN, there always exists a difference between the learned BF and its derivative and their ground-truth values. Given

the ground-truth values of the BF, computing the approximation errors in Eq. (3.106) for training sample positions would be possible, whereas we are not able to compute them for test sample positions that are not in our dataset. Thus, we cannot exploit the results in Proposition 1 for predicted future samples that are outside of our dataset. This issue could be solved to some extent if we take enough samples to cover almost all spots in the sensing range of the robot. Last but not least, the NMPC-LBF method does not necessarily generate smooth trajectories while approaching the obstacles. This issue could be solved by using the stochastic NMPC method as described in (Park, 2016) to generate graceful motions of the robot during navigation.

3.5.5 Discussion

In this section, we presented a decentralized control approach based on an NMPC method leveraged by a DeepNN to learn BF to ensure safety for navigation of mobile robots in unknown environments in the presence of other robots and obstacles. The proposed method does not require inter-robot communication and it can be scaled up to be implemented on any number of robots.

3.6 Velocity Tracking of a Ground Target by Multiple Unmanned Aerial Vehicles with Collision Avoidance

In this section, we present a controller for collision-free velocity tracking of a moving ground target by multiple unmanned aerial vehicles (UAVs). The controller combines a feedforward proportional-derivative (PD) control term and a term that is based on the gradient of an artificial potential function. We use Lasalle's invariance principle to analytically prove the convergence of the UAVs to a fixed formation above the target that tracks the target's velocity and provide mathematical guarantees on the UAVs' collision avoidance. As a result, the Euclidean distance between each pair of UAVs approaches a constant value at

equilibrium. In the event of UAV failure, the remaining UAVs reconfigure to a new fixed formation and maintain collision-free tracking of the target's velocity, demonstrating the robustness of our control approach to failure. We validate this control approach on different simulated scenarios in MATLAB and the Gazebo (Koenig and Howard, 2004) robot simulator. We also experimentally test the performance of the control approach on physical robots, using Crazyflie quadrotors as the UAVs and a Turtlebot3 Burger robot as the moving ground target. The simulation and experimental results demonstrate the effectiveness of our control approach at collision-free tracking and its robustness to UAV failure.

3.6.1 Problem Formulation

We first describe the assumed capabilities of the UAVs and define the model that is used to represent their dynamics. Figure 3.55 shows an illustration of our multi-robot target tracking problem, in which a group of N UAVs must track a wheeled mobile robot (WMR), the target, which is moving on the ground. Let $\mathbf{x}_t = [x_t \;\; y_t \;\; 0]^T$ denote the position of the target in the global coordinate frame, whose origin is defined on the ground. We assume that the UAVs have access to the instantaneous position $\mathbf{x}_t$, velocity $\dot{\mathbf{x}}_t$, and acceleration $\ddot{\mathbf{x}}_t$ of the target. In real-world scenarios, measurements of $\mathbf{x}_t$, $\dot{\mathbf{x}}_t$, and $\ddot{\mathbf{x}}_t$ can be obtained by cameras on the UAVs or by a GPS sensor on the moving target. We also assume that each UAV, indexed by $i \in \{1, ..., N\}$, can use measurements from its onboard sensors to accurately estimate its current position $\mathbf{x}_i \in \mathbb{R}^3$ and velocity $\dot{\mathbf{x}}_i \in \mathbb{R}^3$ in the global coordinate frame, its distance to every other UAV, and its angle with respect to every other UAV in the global coordinate frame. The UAVs can receive information from a central computational unit, but they do not communicate with one another. We also assume that the UAVs always operate close to a hovering condition and do not need to perform maneuvers with large changes in orientation, which allows us to represent the dynamics of

each UAV as a point-mass double-integrator model,

$$m\ddot{\mathbf{x}}_i = \mathbf{u}_i, \tag{3.114}$$

where $m \in \mathbb{R}$ is the UAV's mass, assumed to be the same for all UAVs, and $\mathbf{u}_i \in \mathbb{R}^3$ is its control input vector. Using this double-integrator model facilitates the implementation of the control approach on any type of multi-rotor UAV. However, if the UAVs must perform aggressive maneuvers to track the moving target, then each UAV should instead be described by a nonlinear dynamic model (Mellinger, 2012).

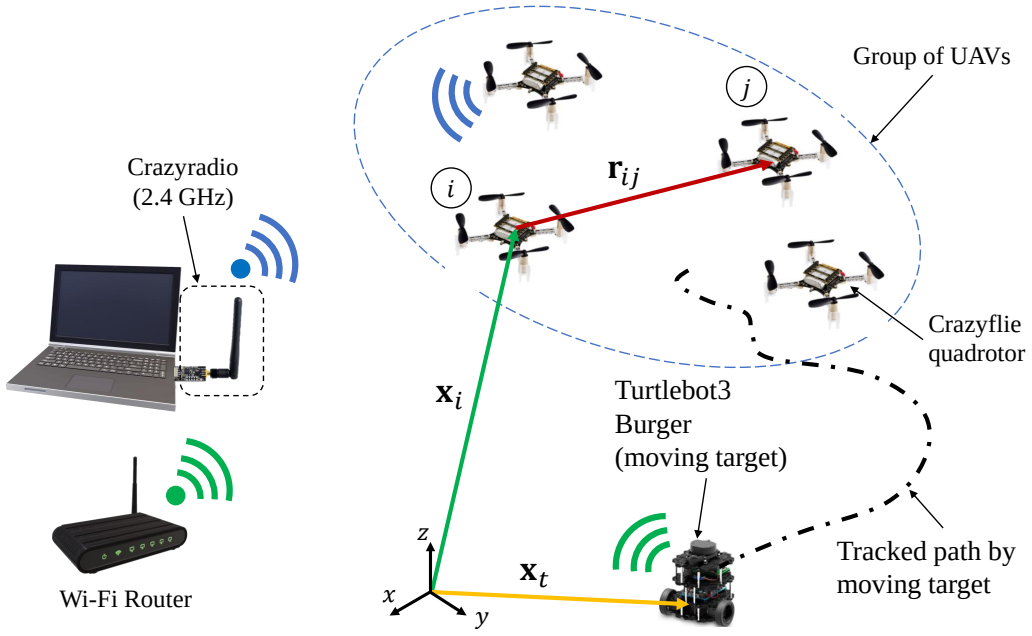


Figure 3.55: Schematic of Our Experimental Setup for the Target Tracking Problem, in which Four Crazyflie Quadrotors Track a Turtlebot3 Burger Robot Moving on the Ground.

3.6.2 Control Approaches

We design a control law to achieve collision-free tracking of a moving ground target by the UAVs and prove that the control law achieves this objective. Our proposed control law $\mathbf{u}_i$ for the i^{th} UAV consists of two components, each designed to achieve one of the follow-

ing objectives: (**1**) tracking the velocity of the moving target, and (**2**) avoiding collisions with the other UAVs. The component of the control law that achieves the tracking objective is defined as a feedforward Proportional-Derivative (PD) controller. The component that ensures collision avoidance is defined as a potential-based controller with the following potential function V_{ij}:

$$V_{ij} = r_{ij} + \frac{a}{r_{ij}}, \tag{3.115}$$

where $a \in \mathbb{R}_{>0}$ is a positive constant and $r_{ij} \in \mathbb{R}_{>0}$ is the Euclidean distance between the i^{th} and j^{th} UAVs, i.e., $r_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||_2$. The full control law for the i^{th} UAV is given by:

$$\mathbf{u}_i = -\mathbf{K}_1\mathbf{s}_i + m\ddot{\mathbf{x}}_t - \mathbf{K}_2\mathbf{h}_i, \tag{3.116}$$

where $\mathbf{K}_1 = k_1\mathbf{I}_{3\times3}$ and $\mathbf{K}_2 = k_2\mathbf{I}_{3\times3}$ are positive definite gain matrices, with $k_1, k_2 \in \mathbb{R}_{>0}$ and $\mathbf{I}_{3\times3}$ denoting the 3×3 identity matrix; $\mathbf{s}_i \in \mathbb{R}^3$ is defined as

$$\mathbf{s}_i = \mathbf{e}_i + \mathbf{\Lambda}\dot{\mathbf{e}}_i, \quad \mathbf{\Lambda} = \lambda\mathbf{I}_{3\times3}, \quad \lambda \in \mathbb{R}_{>0}, \tag{3.117}$$

where $\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}_t$ is the error between the position of the i^{th} UAV and the target; and $\mathbf{h}_i$ is the sum

$$\mathbf{h}_i = \sum_{i\neq j} \nabla V_{ij} = \sum_{i\neq j} \left(1 - \frac{a}{r_{ij}^2}\right)\mathbf{p}_{ij}, \tag{3.118}$$

where $\mathbf{p}_{ij}$ is the unit vector along the distance vector $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, i.e., $\mathbf{p}_{ij} = \mathbf{r}_{ij}/r_{ij}$. To execute this controller, each UAV must use its measurements of the position error $\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}_t$ and its time derivative $\dot{\mathbf{e}}_i = \dot{\mathbf{x}}_i - \dot{\mathbf{x}}_t$, the target's acceleration $\ddot{\mathbf{x}}_t$, and the magnitudes and directions of the vectors $\mathbf{r}_{ij}$.

Analysis of Closed-Loop Dynamics

We obtain the closed-loop dynamics of our target tracking control system by substituting our proposed control law in Eq. (3.116) into the UAV model in Eq. (3.114):

$$m\ddot{\mathbf{x}}_i = -\mathbf{K}_1\mathbf{s}_i + m\ddot{\mathbf{x}}_t - \mathbf{K}_2\mathbf{h}_i. \tag{3.119}$$

Using the expressions for $\mathbf{s}_i$ from Eq. (3.117) and $\mathbf{h}_i$ from Eq. (3.118), we obtain:

$$\begin{aligned} m\ddot{\mathbf{x}}_i &= -\mathbf{K}_1(\mathbf{e}_i + \mathbf{\Lambda}\dot{\mathbf{e}}_i) + m\ddot{\mathbf{x}}_t - \mathbf{K}_2 \sum_{j\neq i} \nabla V_{ij} \\ &= -\mathbf{K}_1\mathbf{e}_i - \mathbf{K}_1\mathbf{\Lambda}\dot{\mathbf{e}}_i + m\ddot{\mathbf{x}}_t - \mathbf{K}_2 \sum_{j\neq i} \left(1 - \frac{a}{r_{ij}^2}\right) \mathbf{p}_{ij}. \end{aligned} \tag{3.120}$$

By rearranging this equation, the closed-loop error dynamics for the i^{th} UAV can be computed as follows:

$$m\ddot{\mathbf{e}}_i + \mathbf{K}_1\mathbf{\Lambda}\dot{\mathbf{e}}_i + \mathbf{K}_1\mathbf{e}_i + \mathbf{K}_2 \sum_{j\neq i} \left(1 - \frac{a}{r_{ij}^2}\right) \mathbf{p}_{ij} = \mathbf{0}. \tag{3.121}$$

Without loss of generality, we assume that $m = 1$. We define the following positive definite Lyapunov function:

$$H = \frac{1}{2}k_2 \sum_{i=1}^{N} \sum_{j\neq i} V_{ij} + \frac{1}{2} \sum_{i=1}^{N} \left(\dot{\mathbf{e}}_i^T \dot{\mathbf{e}}_i + \mathbf{e}_i^T \mathbf{K}_1 \mathbf{e}_i\right). \tag{3.122}$$

The time derivative of H can be calculated as (see Appendix at the end of this chapter):

$$\dot{H} = -\sum_{i=1}^{N} \dot{\mathbf{e}}_i^{\mathrm{T}} \mathbf{\Lambda}\mathbf{K}_1 \dot{\mathbf{e}}_i, \tag{3.123}$$

The positive definiteness of the matrix $\mathbf{\Lambda}\mathbf{K}_1$ and the absence of $\mathbf{e}_i$ in Eq. (3.123) show that $\dot{H}$ is negative semidefinite. This fact, along with the positive definiteness of H, imply the boundedness of H, and consequently, the boundedness of $\mathbf{e}_i$, $\dot{\mathbf{e}}_i$, and V_{ij} for $i = 1, ..., N$, $j \neq i$. In addition, given the continuity of H, we can conclude that

$$\dot{\mathbf{e}}_i \to \mathbf{0} \quad \text{as} \quad t \to \infty, \quad \forall\, i = 1, ..., N. \tag{3.124}$$

This demonstrates that the velocities of all UAVs converge to the velocity of the moving target. Moreover, Eq. (3.124) implies that $\mathbf{e}_i$ converges to a constant value for each UAV, and by using the fact that $\mathbf{x}_i - \mathbf{x}_j = \mathbf{e}_i - \mathbf{e}_j$, we conclude that

$$r_{ij} \to \text{constant} \quad \text{as} \quad t \to \infty, \quad \forall\, i \neq j. \tag{3.125}$$

Eq. (3.125) shows that the UAVs converge to a fixed formation above the moving target at steady-state. Furthermore, given the definition of V_{ij} in Eq. (3.115), the boundedness of V_{ij} for each UAV implies that r_{ij} never equals zero for any pair of UAVs i, j, which guarantees collision avoidance for all UAVs.

Finally, using *LaSalle's invariance principle* (Khalil, 1996), we can confirm that the trajectories of the closed-loop system converge to the largest invariant set in

$$\mathcal{E} = \left\{ \dot{\mathbf{e}}_i \in \mathbb{R}^3 \mid \dot{H} \equiv 0 \right\}, \quad i = 1, \ldots, N. \tag{3.126}$$

Taking into account the closed-loop dynamics in Eq. (3.119) and the fact that $\dot{H} \equiv 0 \Rightarrow \dot{\mathbf{e}}_i, \ddot{\mathbf{e}}_i \equiv \mathbf{0},\ \forall i \in \{1, \ldots, N\}$, $\mathcal{E}$ can be rewritten as

$$\mathcal{E} = \left\{ \mathbf{e}_i \in \mathbb{R}^3 \mid \mathbf{K}_1 \mathbf{e}_i + \mathbf{K}_2 \mathbf{h}_i = \mathbf{0} \right\}, \quad i = 1, \ldots, N. \tag{3.127}$$

Summing the equations $\mathbf{K}_1 \mathbf{e}_i + \mathbf{K}_2 \mathbf{h}_i = \mathbf{0}$ over all UAVs $i = 1, ..., N$, we obtain

$$\mathbf{K}_1 \sum_{i=1}^{N} \mathbf{e}_i + \mathbf{K}_2 \sum_{i=1}^{N} \mathbf{h}_i = \mathbf{0}. \tag{3.128}$$

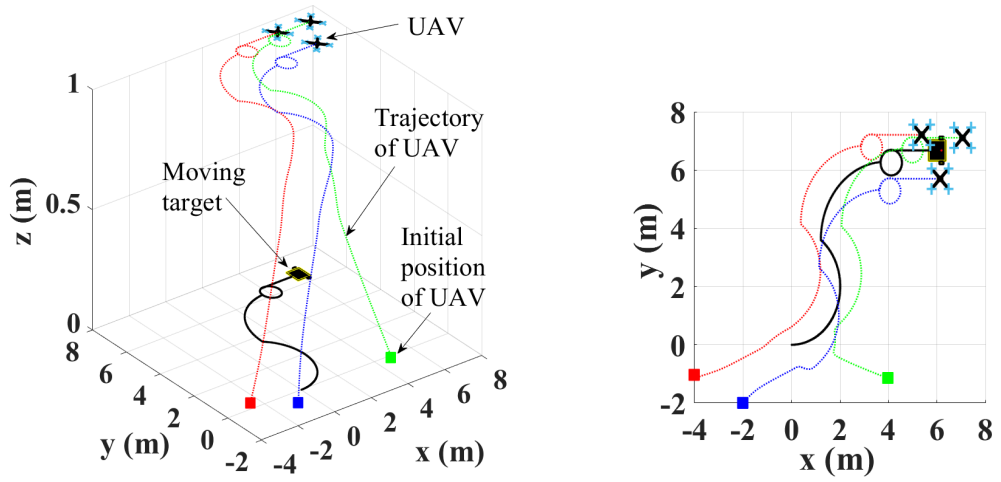
We can confirm that $\mathbf{K}_2 \sum_{i=1}^{N} \mathbf{h}_i = \mathbf{0}$, since it is the sum of the mutual repulsion forces of every pair of UAVs on each other, which cancel out. Hence, Eq. (3.128) yields

$$\sum_{i=1}^{N} \mathbf{e}_i = \mathbf{0}, \tag{3.129}$$

which shows that the geometric center of the polygon defined by the UAVs (the polygon vertices are the position coordinates of the UAVs) tracks the moving target.

3.6.3 Simulation and Experimental Results

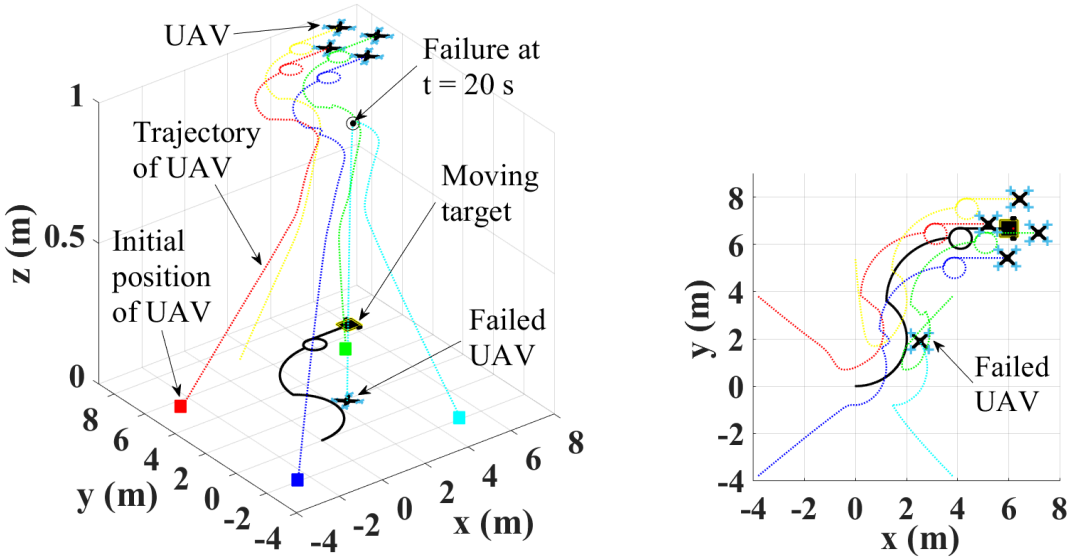
We evaluated our proposed controller for multi-robot target tracking in different simulated scenarios, both in MATLAB and in the Gazebo robot simulator. A video recording of all simulations described here, as well as additional simulations, is available online at (Salimi Lafmejani *et al.*, 2020g).



(a) 3D view

(b) View in $x - y$ Plane

Figure 3.56: Robot Trajectories During a MATLAB Simulation of *Scenario 1*.



(a) 3D view

(b) View in $x - y$ Plane

Figure 3.57: Robot Trajectories During a MATLAB Simulation of *Scenario 2*. The Position at which One UAV Fails is Indicated by a Small Black Circle.

MATLAB Simulations

We simulated two scenarios in which several quadrotor UAVs track a moving nonholonomic WMR while avoiding collisions with one another. The UAVs start from arbitrary

initial configurations on the ground and then take off simultaneously to fly at a constant altitude above the WMR before starting the tracking task.

Scenario 1: Three UAVs track the target WMR as it drives along a trajectory that includes a small circle. Figure 3.56 plots the trajectories of the UAVs and the WMR over the duration of this simulation (60 s). As expected, the UAVs successfully converge to a fixed formation that tracks the trajectory of the WMR, enclosing its position in the $x - y$ plane, while avoiding collisions with one another.

Scenario 2: Five UAVs initially take off to track the target WMR as it drives along the same trajectory as in *Scenario 1*, as shown by the plots of the UAV and WMR trajectories in Figure 3.57. The UAVs converge to a fixed formation above the WMR and track its trajectory. At time $t = 20$ s, one UAV fails and drops to the ground. The remaining four UAVs reconfigure themselves into a new formation and continue to track the trajectory of the WMR for the duration of the simulation (60 s). The UAVs avoid collisions with one another throughout the simulation. This scenario demonstrates the robustness of our multi-robot target tracking approach to UAV failures.

Gazebo Simulations

In order to evaluate our control approach in a more realistic simulation environment, we simulated two scenarios in Gazebo that were similar to *Scenarios 1* and *2* in MATLAB. There were four UAVs instead of five in *Scenario 2*, still with one UAV failure. We plot the trajectories of the moving target and the UAVs in the $x - y$ plane for both scenarios in Figs. 3.59a and 3.60a. To better visualize the tracking performance of the UAVs, we also computed the geometric center of the convex hull of the UAVs during the simulation and plotted its trajectory in Figs. 3.59b and 3.60b, along with the trajectory of the target.

Figure 3.59b shows that the geometric center of the UAV formation successfully tracks the target's trajectory. Figure 3.60b shows that initially, the center of the four UAVs closely tracks the target, and right after a UAV fails at time $t = 30$ s, the center of the remaining three UAVs soon converges back to the target's trajectory. The UAVs avoid collisions throughout the duration of both simulations (100 s).

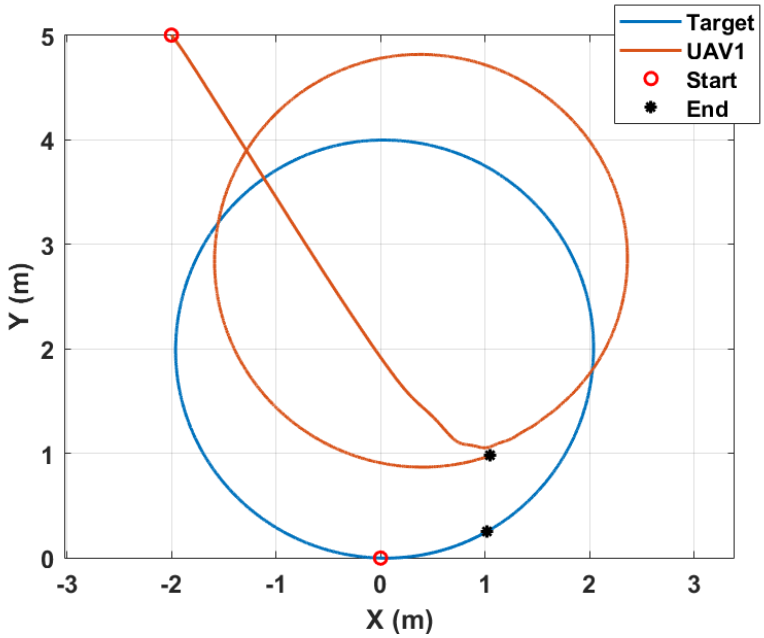
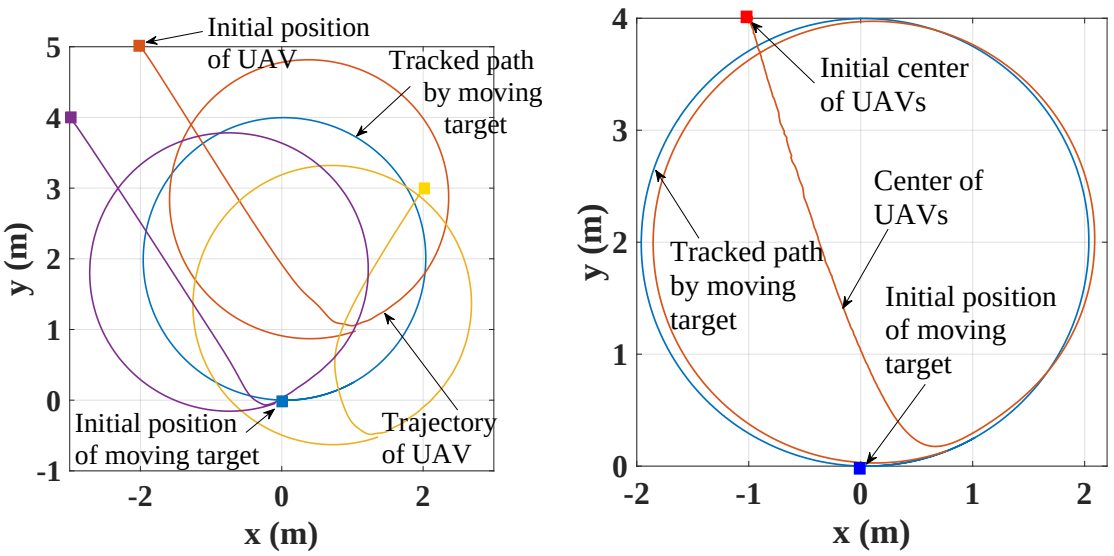


Figure 3.58: Scenario 1: One UAV Tracking the Ground Mobile Target. Total Duration: 20 seconds.

For experimental validation our control approach, we first describe the experimental setup for all tested scenarios and provide corresponding plots of the results for each experiment. A video recording of the experimental tests described here, as well as additional tests, is also available online at (Salimi Lafmejani *et al.*, 2020f).



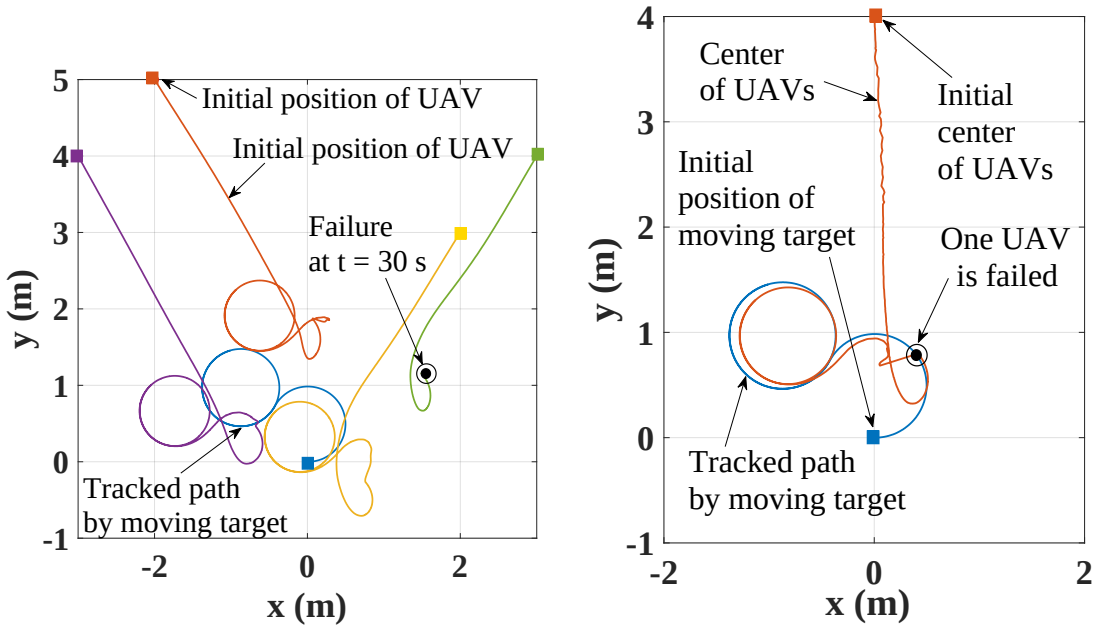
(a) View in $x-y$ Plane of the Target and UAV Trajectories

(b) View in $x-y$ Plane of Trajectories of the Target and Center of UAV Formation

Figure 3.59: Robot Trajectories During a Gazebo Simulation of *Scenario 1*.

Experimental Setup

In order to validate the effectiveness of our proposed control method in practice, we implement our proposed control law in Eq. (3.116) on physical robots. As shown in Fig. 3.55, Crazyflie 2.1 (Bitcraze, 2011) quadrotors are used as the UAVs. Crazyflie is a small, lightweight, open-source platform commonly used for swarm and multi-robot applications. We employ multiple Crazyflies as the tracker UAVs and a Turtlebot3 Burger robot (Robotis, 2020) as the moving target on the ground. We first plan curved trajectories for the Burger robot by setting arbitrary linear and angular velocities. This planned trajectory is not known by the UAVs. Then, we read the Burger robot's odometry information in real-time using the Robot Operating System (ROS). We then calculate the instantaneous position, velocity, and acceleration of the Burger robot and compute the control commands for the Crazyflie quadrotors accordingly. The control commands are sent to the Crazyflies using Crazyradio,



(a) View in $x-y$ Plane of the Target and UAV Trajectories

(b) View in $x-y$ Plane of Trajectories of the Target and Center of UAV Formation

Figure 3.60: Robot Trajectories During a Gazebo Simulation of *Scenario 2*. The Position at which One UAV Fails is Indicated by a Small Black Circle.

a 2.4 GHz radio USB dongle. There is no explicit communication between the Crazyflies. Each Crazyflie is equipped with a Flow deck v2, which uses a VL53L1x ToF sensor to measure the distance to the ground (z-axis) and a PMW3901 optical flow sensor to measure displacement in the plane of the ground ($x-y$ plane). State estimation and PID controllers have been implemented on the Crazyflie firmware, which enable the Crazyflie to follow given position setpoints. At each time instant, the desired position setpoints are sent to the Crazyflies as control commands from a central computer through Wi-Fi communication provided by the Crazyradio between the computer and the Crazyflies.

Experimental Results

For the experimental tests, two scenarios were implemented. In *Scenario 1*, three UAVs track the moving target while avoiding collisions with one another. In *Scenario 2*, four UAVs track the target, and after 30 s, one of the UAV lands (fails), and the tracking continues with three UAVs. The second scenario is designed to demonstrate the robustness of our controller at tracking the moving target in the event of UAV failure. In all experiments, the UAVs start from arbitrary initial configurations on the ground and then take off simultaneously to fly at a constant altitude before starting to track the ground robot. The real-time positions of the UAVs and the target were collected at 20 Hz frequency, and a Butterworth low-pass filter was applied to these data with a 5 Hz cut-off frequency in MATLAB.

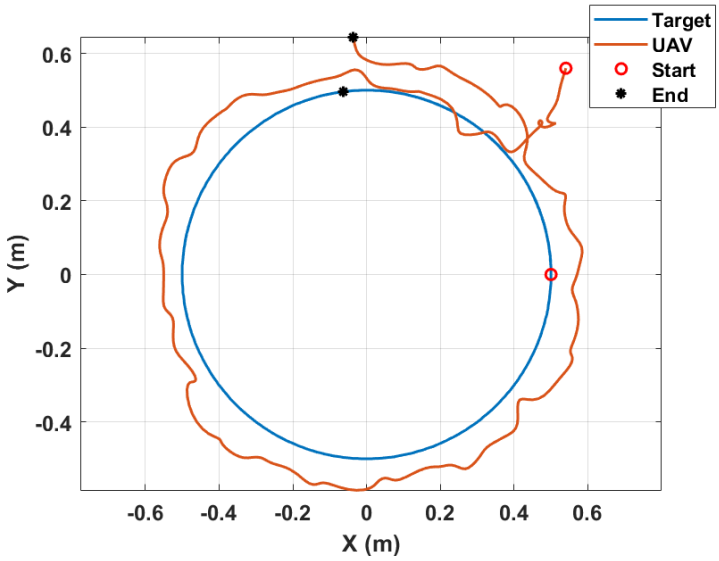
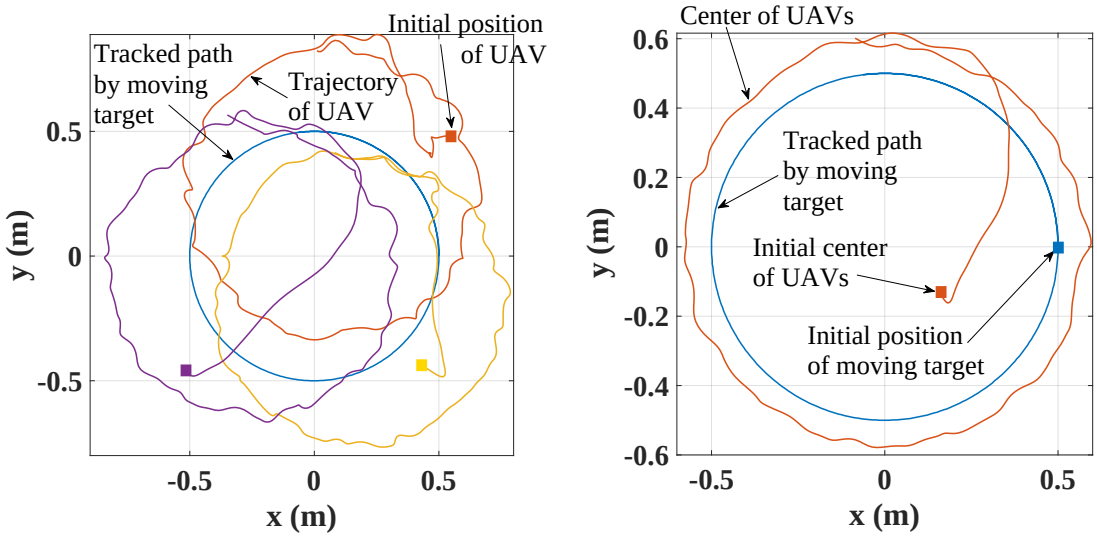


Figure 3.61: Scenario 1: One UAV Tracking the Ground Mobile Target. Total Duration: 20 Seconds.

Figure 3.61 shows the trajectories for the Scenario 1. The start and end points of the UAV and the moving target are pointed out in the figure. As it can be observed, the UAV tracks the circular trajectory of the target robot while keeping a specific distance.

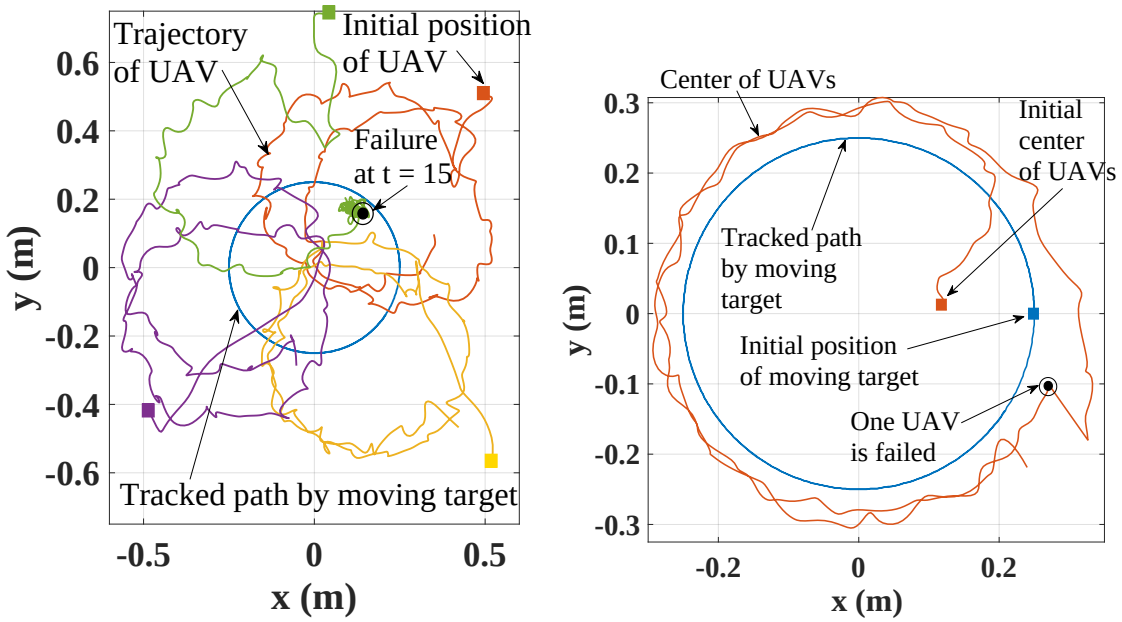


(a) View in $x - y$ Plane of the Target and UAV Trajectories

(b) View in $x - y$ Plane of Trajectories of the Target and Center of UAV Formation

Figure 3.62: Robot Trajectories During an Experimental Trial (Duration: 30 s) of *Scenario 1*.

The experimental results show that the UAVs successfully tracked the ground robot in each scenario while avoiding collisions. The fluctuations in the UAV trajectories can be attributed to the UAVs' low-level position control loop and near-ground altitude. For *Scenario 1*, along with the plots for trajectory of each UAV in Fig. 3.62a, we also plotted the trajectory of the geometric center of the group of UAVs in Fig. 3.62b. The group of UAVs effectively tracked the moving target without colliding with one another. *Scenario 2* was performed with four UAVs. Figure 3.63 shows the trajectories of the UAVs and their geometric center. At $t = 15$ s, one UAV stops tracking and lands on the ground, emulating the failure of one member of the UAV team. This event is indicated in Fig. 3.63 with black arrows. After this event, the remaining three UAVs continue tracking the target and ignore the failed UAV in their collision avoidance controller. As shown in Fig. 3.63b, the failure event causes a sudden shift in the distance between the center of UAVs and the



(a) View in $x-y$ plane of the target and UAV trajectories

(b) View in $x-y$ Plane of Trajectories of the Target and Center of UAV Formation

Figure 3.63: Robot trajectories during an experimental trial (duration: 40 s) of *Scenario 2*. The position at which one UAV fails is indicated by a small black circle.

target. Nevertheless, this distance decreases to approximately the same distance exhibited before the failure event. Comparing the experimental results with the simulation results, we observe similar tracking behavior by the UAVs. The main difference is the presence of fluctuations in the UAV trajectories during the experiments, which is expected in real-world implementations.

3.6.4 Discussion

We addressed the problem of tracking a moving ground target with multiple UAVs that do not have prior information about the target's motion. We analytically proved the convergence of the UAVs to a fixed formation above the target that tracks its trajectory, and we provided mathematical guarantees that the UAVs avoid collisions with one another. We

validated our control approach for different scenarios in MATLAB and Gazebo simulations and in physical experiments with ground and aerial robots. In our simulations and experiments, we also demonstrated the robustness of the UAVs' target tracking performance and collision avoidance to the failure of one or more UAVs during the task.

Appendix

We define the following Lyapunov function:

$$H = \frac{1}{2}k_2 \sum_{i=1}^{N} \sum_{j \neq i} V_{ij} + \frac{1}{2} \sum_{i=1}^{N} \left(\dot{\mathbf{e}}_i^{\mathrm{T}} \dot{\mathbf{e}}_i + \mathbf{e}_i^{\mathrm{T}} \mathbf{K}_1 \mathbf{e}_i \right). \tag{3.130}$$

Given the expression for V_{ij} in Eq. (3.115), which is a continuously differentiable function, the time derivative of H can be calculated as:

$$\dot{H} = \frac{1}{2}k_2 \sum_{i=1}^{N} \sum_{j \neq i} \dot{V}_{ij} + \sum_{i=1}^{N} \left(\dot{\mathbf{e}}_i^{\mathrm{T}} \ddot{\mathbf{e}}_i + \mathbf{e}_i^{\mathrm{T}} \mathbf{K}_1 \dot{\mathbf{e}}_i \right). \tag{3.131}$$

Considering the following representation for the time derivative of V_{ij},

$$\dot{V}_{ij} = \left(\nabla_{\mathbf{r}_{ij}} V_{ij} \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ij}, \tag{3.132}$$

and given that $\nabla_{\mathbf{r}_{ij}} V_{ij} = \nabla_{\mathbf{x}_i} V_{ij} = -\nabla_{\mathbf{x}_j} V_{ij}$, we have that:

$$\dot{V}_{ij} = \left(\nabla_{\mathbf{x}_i} V_{ij} \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ij} = \left(-\nabla_{\mathbf{x}_j} V_{ij} \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ij}. \tag{3.133}$$

Furthermore, the term $\sum_{i=1}^{N} \sum_{j \neq i} \dot{V}_{ij}$ includes both $\dot{V}_{ij}$ and $\dot{V}_{ji}$ for each pair of robots i, j, where $i \neq j$. Taking into account the fact that $V_{ij} = V_{ji}$, the sum of these two terms is calculated as:

$$\begin{aligned} \dot{V}_{ij} + \dot{V}_{ji} &= \left(\nabla_{\mathbf{x}_i} V_{ij} \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ij} + \left(\nabla_{\mathbf{x}_j} V_{ji} \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ji} = \left(\nabla_{\mathbf{x}_i} V_{ij} \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ij} + \left(-\nabla_{\mathbf{x}_i} V_{ji} \right)^{\mathrm{T}} \left(-\dot{\mathbf{r}}_{ij} \right) \\ &= \left(\nabla_{\mathbf{x}_i} (V_{ij} + V_{ji}) \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ij} = 2 \left(\nabla_{\mathbf{x}_i} V_{ij} \right)^{\mathrm{T}} \dot{\mathbf{r}}_{ij}. \end{aligned} \tag{3.134}$$

Moreover, if we define $\mathbf{e}_{ij} := \mathbf{e}_i - \mathbf{e}_j$, we can confirm that $\dot{\mathbf{e}}_{ij} = \dot{\mathbf{r}}_{ij}$. Incorporating this into Eq. (3.134), we can write:

$$\frac{1}{2}\left(\dot{V}_{ij} + \dot{V}_{ji}\right) = \left(\nabla_{\mathbf{x}_i} V_{ij}\right)^{\mathrm{T}} \dot{\mathbf{e}}_{ij} = \left(\nabla_{\mathbf{x}_i} V_{ij}\right)^{\mathrm{T}} \dot{\mathbf{e}}_i + \left(\nabla_{\mathbf{x}_j} V_{ij}\right)^{\mathrm{T}} \dot{\mathbf{e}}_j.$$

Therefore, the first summation in Eq. (3.131) can be written as:

$$\frac{1}{2} k_2 \sum_{i=1}^{N} \left(\sum_{j \neq i} \dot{V}_{ij} \right) = k_2 \sum_{i=1}^{N} \sum_{j \neq i} \left(\nabla_{\mathbf{x}_i} V_{ij}\right)^{\mathrm{T}} \dot{\mathbf{e}}_i. \tag{3.135}$$

Then, solving Eq. (3.121) for $\ddot{\mathbf{e}}_i$ and substituting this expression into Eq. (3.131), we obtain Eq. (3.123) for $\dot{H}$:

$$\begin{aligned}
\dot{H} &= k_2 \sum_{i=1}^{N} \sum_{i \neq j} \left(\nabla_{\mathbf{x}_i} V_{ij}\right)^{\mathrm{T}} \dot{\mathbf{e}}_i + \sum_{i=1}^{N} \left(-\dot{\mathbf{e}}_i^{\mathrm{T}} \mathbf{\Lambda} \mathbf{K}_1 \dot{\mathbf{e}}_i - \dot{\mathbf{e}}_i^{\mathrm{T}} \left(k_2 \sum_{j \neq i} \nabla_{\mathbf{x}_i} V_{ij} \right) \right) \\
&= \underbrace{k_2 \sum_{i=1}^{N} \sum_{i \neq j} \left(\nabla_{\mathbf{x}_i} V_{ij}\right)^{\mathrm{T}} \dot{\mathbf{e}}_i - \dot{\mathbf{e}}_i^{\mathrm{T}} k_2 \sum_{i=1}^{N} \sum_{j \neq i} \nabla_{\mathbf{x}_i} V_{ij}}_{=\,0} + \sum_{i=1}^{N} \left(-\dot{\mathbf{e}}_i^{\mathrm{T}} \mathbf{\Lambda} \mathbf{K}_1 \dot{\mathbf{e}}_i \right) \\
&= -\sum_{i=1}^{N} \left(\dot{\mathbf{e}}_i^{\mathrm{T}} \mathbf{\Lambda} \mathbf{K}_1 \dot{\mathbf{e}}_i \right).
\end{aligned}$$

Chapter 4

CONCLUSION AND FUTURE WORK

In this dissertation, we have taken some significant steps to tackle the challenges of controlling multi-agent robotic systems using decentralized methods, which we hope will be useful for future researchers in multi-agent robotics. We show how decentralized control approaches, which have significant advantages in terms of scalability and computational affordability compared to centralized methods, can be designed to achieve trajectory tracking by multi-segment manipulators and collision-free navigation by multiple mobile robots. In recent years, a variety of approaches for control of multi-agent robotic systems have been developed; however, the existing centralized and distributed methods are largely intractable. In this research, we design different control approaches, ranging from classical control to optimization-based control, to study the performance of our proposed methods in control and motion planning of multi-segment manipulators and mobile robot collectives. We validate our approaches in simulations of multi-segment manipulators, ground robots, and aerial robots, and in experiments with ground and aerial robotic platforms.

We summarize possible directions for future work on the research presented in this dissertation as follows:

- The decentralized position stabilization and trajectory tracking control approach for a 2D multi-segment manipulator proposed in Section 2.1 has the potential to be implemented on a variety of distributed robotic systems that are composed of multiple connected components or multiple freely-moving agents. Possible future work includes: (1) deriving similar consensus-based decentralized controllers for dynamic models of continuum robots; (2) extending the controllers to continuum robots that

move in 3D space; (3) identifying bounds on the controller gain, K, to satisfy mechanical constraints such as limits on displacements of the robot's joints; and (4) incorporating secondary objectives into the controller design such as obstacle avoidance, joint torque reduction, joint limits, and increased manipulability.

- Possible future work on the trajectory tracking controller for an octopus-inspired 3D multi-segment manipulator in Section 2.2 includes developing a dynamic model for the robot and designing a control approach based on task priorities (Soylu *et al.*, 2007) for the robot. Furthermore, a decentralized control approach could be developed in order to enable the robot to operate with autonomous functionality. In this decentralized framework, we can employ Linear Matrix Inequality (LMI)-based optimal control approaches, such as H_2 and H_∞ methods, in order to attenuate the effects of undesired exogenous inputs, such as actuation disturbances and measurement noise, on the performance of the robot.

- Possible future work on the pose stabilization and trajectory tracking control approach for a holonomic mobile robot in Section 3.1 includes the extension of this approach to multiple mobile robots that must perform tasks such as flocking and target tracking while avoiding collisions with each other and with objects in the environment. Moreover, the state-space model of the closed-loop system can be modified to include weights on the reference, output, disturbance, and noise signals, which can be tuned to improve the robot's trajectory tracking performance.

- Possible future work on the position stabilization controller for a nonholonomic mobile robot in Section 3.2, which prevents collisions with static obstacles, includes designing controllers for stabilizing the pose of a nonholonomic robot while preventing collisions with obstacles using approaches other than gradient-based methods.

- Possible future work on the nonlinear Model Predictive Control (NMPC) approach to pose stabilization of multiple nonholonomic robots in Section 3.3, which achieves collision and deadlock avoidance, includes the modification of this method to create a decentralized MPC scheme for multi-robot navigation, in which robots use their own distance measurements (e.g., from a LiDAR sensor) to detect nearby robots and unknown obstacles in the environment and autonomously adjust their velocities to avoid potential collisions and deadlocks. We have developed one such approach in Section 3.5. As another direction of future work, an LMI-based NMPC method could be designed with linear models of the robots' motion, which would reduce the computational complexity of our approach and thus enable controller synthesis for larger numbers of robots without a corresponding significant increase in computational effort.

- Possible future work on the NMPC trajectory tracking controller for multiple nonholonomic robots in Section 3.4, which uses barrier functions to achieve safe multi-robot navigation in environments with known obstacles, is to extend this method to prevent robot collisions with *unknown* static and dynamic obstacles in the environment, e.g., by using Deep Neural Networks (DeepNNs) to learn the corresponding barrier functions in the Control Barrier Conditions (CBCs). We have developed one such approach in Section 3.5.

- Possible future work on the NMPC pose stabilization controller for multiple nonholonomic robots in Section 3.5, which uses learned barrier functions (LBFs) to achieve safe multi-robot navigation in environments with unknown static and/or dynamic obstacles, includes modifying the method for learning the BF over the prediction horizon, and therefore learning the unsafe regions at future time steps, by using a history of the robots' LiDAR readings as inputs to the DeepNN. This would improve

the robots' ability to avoid collisions with moving objects in dynamic environments, since the method could more accurately predict the trajectories of the objects based on their past positions. Another direction for future work is to redesign the optimization problem in order to encourage smoothness in the robots' navigation while guaranteeing safety.

- Possible future work on the velocity tracking controller for multiple UAVs in Section 3.6 includes the design of a fully decentralized tracking controller and extension of the collision avoidance strategy to enable the UAVs to perform collision-free tracking in environments with unknown and dynamic obstacles.

REFERENCES

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning", in "12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)", pp. 265–283 (2016).

Agha-Mohammadi, A.-A. and K. Ebadi, "Rover localization in Mars helicopter-generated aerial maps: Experimental results in a Mars-analogue environment", in "International Symposium on Experimental Robotics", (Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space, 2018).

Alessandretti, A., A. P. Aguiar and C. N. Jones, "Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control", in "2013 European Control Conference (ECC)", pp. 1371–1376 (IEEE, 2013).

Alonso-Mora, J., P. Beardsley and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots", IEEE Transactions on Robotics **34**, 2, 404–420 (2018a).

Alonso-Mora, J., J. A. DeCastro, V. Raman, D. Rus and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles", Autonomous Robots **42**, 4, 801–824 (2018b).

Ames, A. D., S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath and P. Tabuada, "Control barrier functions: Theory and applications", in "2019 18th European Control Conference (ECC)", pp. 3420–3431 (IEEE, 2019).

Ames, A. D., J. W. Grizzle and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control", in "IEEE Conference on Decision and Control (CDC)", pp. 6271–6278 (2014).

Ananthanarayanan, H. and R. Ordóñez, "Real-time inverse kinematics of (2n+ 1) dof hyper-redundant manipulator arm via a combined numerical and analytical approach", Mechanism and Machine Theory **91**, 209–226 (2015).

Andersson, J., J. Åkesson and M. Diehl, "CasADi: A symbolic package for automatic differentiation and optimal control", in "Recent Advances in Algorithmic Differentiation", pp. 297–307 (Springer, 2012).

Andersson, J. A. E., J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control", Mathematical Programming Computation **11**, 1, 1–36 (2019).

Andersson, S. B., "Discrete approximations to continuous curves", in "Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.", pp. 2546–2551 (IEEE, 2006).

Antonelli, G., F. Arrichiello, F. Caccavale and A. Marino, "Decentralized time-varying formation control for multi-robot systems", The International Journal of Robotics Research **33**, 7, 1029–1043 (2014).

Araújo, H. X., A. G. Conceição, G. H. Oliveira and J. Pitanga, "Model predictive control based on LMIs applied to an omni-directional mobile robot", IFAC Proceedings Volumes **44**, 1, 8171–8176 (2011).

Arslan, O. and D. E. Koditschek, "Sensor-based reactive navigation in unknown convex sphere worlds", The International Journal of Robotics Research **38**, 2-3, 196–223 (2019).

Astolfi, A., "Exponential stabilization of a wheeled mobile robot via discontinuous control", Journal of Dynamic Systems, Measurement, and Control **121**, 1, 121–126 (1999).

Ataka, A., H.-K. Lam and K. Althoefer, "Reactive magnetic-field-inspired navigation for non-holonomic mobile robots in unknown environments", in "IEEE International Conference on Robotics and Automation", pp. 6983–6988 (IEEE, 2018).

Azizi, M. R. and J. Keighobadi, "Point stabilization of nonholonomic spherical mobile robot using nonlinear model predictive control", Robotics and Autonomous Systems **98**, 347–359 (2017).

Bitcraze, "Crazyflie", `https://www.bitcraze.io/` (2011).

Boccia, A., L. Grüne and K. Worthmann, "Stability and feasibility of state constrained MPC without stabilizing terminal constraints", Systems & Control Letters **72**, 14–21 (2014).

Bock, H. G. and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems", IFAC Proceedings Volumes **17**, 2, 1603–1608 (1984).

Borrelli, F., A. Bemporad and M. Morari, *Predictive control for linear and hybrid systems* (Cambridge University Press, 2017).

Boyd, S., L. El Ghaoui, E. Feron and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, vol. 15 (SIAM, 1994).

Bräunl, T., "Omni-directional robots", Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems pp. 147–156 (2008).

Brockett, R. W. *et al.*, "Asymptotic stability and feedback stabilization", Differential Geometric Control Theory **27**, 1, 181–191 (1983).

Bullo, F. and R. M. Murray, "Proportional derivative (PD) control on the Euclidean group", in "European Control Conference", vol. 2, pp. 1091–1097 (1995).

Burgner-Kahrs, J., D. C. Rucker and H. Choset, "Continuum robots for medical applications: A survey", IEEE Transactions on Robotics **31**, 6, 1261–1280 (2015).

Buss, S. R., "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods", IEEE Journal of Robotics and Automation **17**, 1-19, 16 (2004).

Camacho, E. F. and C. B. Alba, *Model predictive control* (Springer Science & Business Media, 2013).

Caverly, R. J. and J. R. Forbes, "LMI properties and applications in systems, stability, and control theory", arXiv preprint arXiv:1903.08599 (2019).

Chang, D. E. and J. E. Marsden, "Gyroscopic forces and collision avoidance with convex obstacles", in "New trends in nonlinear dynamics and control and their applications", pp. 145–159 (Springer, 2003).

Chollet, F. *et al.*, "Keras", `https://github.com/fchollet/keras` (2015).

Cristofalo, E., K. Leahy, C.-I. Vasile, E. Montijano, M. Schwager and C. Belta, "Localization of a ground robot by aerial robots for GPS-deprived control with temporal logic constraints", in "International Symposium on Experimental Robotics", pp. 525–537 (Springer, 2016).

Cui, M., J. Zhao and H. Liu, "Design and implementation of trajectory tracking controller for nonholonomic mobile robots based on the Lyapunov method", in "Chinese Control Conference", pp. 4332–4337 (IEEE, 2019).

De Luca, A. and G. Oriolo, "Local incremental planning for nonholonomic mobile robots", in "IEEE International Conference on Robotics and Automation", pp. 104–110 (IEEE, 1994).

De Luca, A., G. Oriolo and M. Vendittelli, "Stabilization of the unicycle via dynamic feedback linearization", IFAC Proceedings Volumes **33**, 27, 687–692 (2000).

Della Santina, C., R. K. Katzschmann, A. Bicchi and D. Rus, "Dynamic control of soft robots interacting with the environment", IEEE International Conference on Soft Robotics (RoboSoft) (2018).

Di Carlo, J., P. M. Wensing, B. Katz, G. Bledt and S. Kim, "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control", in "2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 1–9 (IEEE, 2018).

Dietrich, A. and C. Ott, "Hierarchical impedance-based tracking control of kinematically redundant robots", IEEE Transactions on Robotics **36**, 1, 204–221 (2020).

Dille, M., "Search and pursuit with unmanned aerial vehicles in road networks", Tech. rep., Carnegie Mellon University, The Robotics Institute, Pittsburgh, PA (2013).

Dixon, W. E., D. M. Dawson, F. Zhang and E. Zergeroglu, "Global exponential tracking control of a mobile robot system via a PE condition", IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **30**, 1, 129–142 (2000).

Doroudchi, A., S. Shivakumar, R. E. Fisher, H. Marvi, D. Aukes, X. He, S. Berman and M. M. Peet, "Decentralized control of distributed actuation in a segmented soft robot arm", in "IEEE Conference on Decision and Control", pp. 7002–7009 (IEEE, 2018).

Duan, G.-R. and H.-H. Yu, *LMIs in control systems: analysis, design and applications* (CRC Press, 2013).

Engel, Y., P. Szabo and D. Volkinshtein, "Learning to control an octopus arm with Gaussian process temporal difference methods", in "Advances in Neural Information Processing Systems", pp. 347–354 (2006).

Faulwasser, T., *Optimization-based solutions to constrained trajectory-tracking and path-following problems*, Ph.D. thesis, Otto von Guericke University Magdeburg (2012).

Faulwasser, T. and R. Findeisen, "Nonlinear model predictive control for constrained output path following", IEEE Transactions on Automatic Control **61**, 4, 1026–1039 (2015).

Fiorini, P. and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles", The International Journal of Robotics Research **17**, 7, 760–772 (1998).

Godage, I. S., D. T. Branson, E. Guglielmino and D. G. Caldwell, "Path planning for multisection continuum arms", in "IEEE International Conference on Mechatronics and Automation", pp. 1208–1213 (2012).

Godage, I. S., E. Guglielmino, D. T. Branson, G. A. Medrano-Cerda and D. G. Caldwell, "Novel modal approach for kinematics of multisection continuum arms", in "IEEE/RSJ International Conference on Intelligent Robots and Systems", pp. 1093–1098 (2011).

Gondhalekar, R., J.-i. Imura and K. Kashima, "Controlled invariant feasibility—a general approach to enforcing strong feasibility in MPC applied to move-blocking", Automatica **45**, 12, 2869–2875 (2009).

Grasso, F. W., "The octopus with two brains: how are distributed and central representations integrated in the octopus central nervous system", Cephalopod Cognition pp. 94–122 (2014).

Grimm, G., M. J. Messina, S. E. Tuna and A. R. Teel, "Model predictive control: for want of a local control Lyapunov function, all is not lost", IEEE Transactions on Automatic Control **50**, 5, 546–558 (2005).

Grissom, M. D., V. Chitrakaran, D. Dienno, M. Csencits, M. Pritts, B. Jones, W. McMahan, D. Dawson, C. Rahn and I. Walker, "Design and experimental testing of the OctArm soft robot manipulator", in "Unmanned Systems Technology VIII", vol. 6230, p. 62301F (International Society for Optics and Photonics, 2006).

Grover, J., C. Liu and K. Sycara, "Deadlock analysis and resolution in multi-robot systems (extended version)", arXiv preprint arXiv:1911.09146 (2019).

Grüne, L. and J. Pannek, "Nonlinear model predictive control", in "Nonlinear Model Predictive Control", pp. 45–69 (Springer, 2017).

Gu, D. and H. Hu, "A stabilizing receding horizon regulator for nonholonomic mobile robots", IEEE Transactions on Robotics **21**, 5, 1022–1028 (2005).

Hannan, M. W. and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots", Journal of Robotic Systems **20**, 2, 45–63 (2003).

Hatton, R. L. and H. Choset, "Generating gaits for snake robots: annealed chain fitting and keyframe wave extraction", Autonomous Robots **28**, 3, 271–281 (2010).

Hedengren, J., J. Mojica, W. Cole and T. Edgar, "APOPT: MINLP solver for differential and algebraic systems with benchmark testing", in "Proceedings of the INFORMS National Meeting, Phoenix, AZ, USA", vol. 1417, p. 47 (2012).

Huang, C., X. Chen, Y. Zhang, S. Qin, Y. Zeng and X. Li, "Hierarchical model predictive control for multi-robot navigation", in "International Joint Conferences on Artificial Intelligence (IJCAI), AAAI Press.", p. 3140–3146 (Conference Proceedings, 2016).

Hussein, A. S., C. M. Elias and E. I. Morgan, "A realistic model predictive control using single and multiple shooting in the formulation of non-linear programming model", in "2019 IEEE International Conference of Vehicular Electronics and Safety (ICVES)", pp. 1–6 (IEEE, 2019).

Ishimura, K., M. Natori and M. Wada, "Decentralized control of redundant manipulator based on the analogy of heat and wave equations", in "Distributed Autonomous Robotic Systems 5", pp. 227–236 (Springer, 2002).

Jia, Y.-B., "Quaternions and rotations", Com S **477**, 577, 15 (2008).

Johnson, S. G., "The NLopt nonlinear-optimization package", URL `http://github.com/stevengj/nlopt` (2020).

Kang, R., D. T. Branson, E. Guglielmino and D. G. Caldwell, "Dynamic modeling and control of an octopus inspired multiple continuum arm robot", Computers & Mathematics with Applications **64**, 5, 1004–1016 (2012).

Kano, T., K. Sakai, K. Yasui, D. Owaki and A. Ishiguro, "Decentralized control mechanism underlying interlimb coordination of millipedes", Bioinspiration & biomimetics **12**, 3, 036007 (2017).

Kapitanyuk, Y. A., A. V. Proskurnikov and M. Cao, "A guiding vector-field algorithm for path-following control of nonholonomic mobile robots", IEEE Transactions on Control Systems Technology **26**, 4, 1372–1385 (2017).

Keerthi, S. a. and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations", Journal of optimization theory and applications **57**, 2, 265–293 (1988).

Keighobadi, J., M. S. Sadeghi and K. A. Fazeli, "Dynamic sliding mode controller for trajectory tracking of nonholonomic mobile robots", IFAC Proceedings Volumes **44**, 1, 962–967 (2011).

Khalil, H. K., *Nonlinear systems* (Prentice Hall, Upper Saddle River, N.J., 1996).

Khan, A., B. Rinner and A. Cavallaro, "Cooperative robots to observe moving targets", IEEE Transactions on Cybernetics **48**, 1, 187–198 (2016).

Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots", in "IEEE International Conference on Robotics and Automation", vol. 2, pp. 500–505 (IEEE, 1985).

Khatib, O., X. Yeh, G. Brantner, B. Soe, B. Kim, S. Ganguly, H. Stuart, S. Wang, M. Cutkosky, A. Edsinger *et al.*, "Ocean one: A robotic avatar for oceanic discovery", IEEE Robotics & Automation Magazine **23**, 4, 20–29 (2016).

Kier, W. M., "The musculature of coleoid cephalopod arms and tentacles", Frontiers in Cell and Developmental Biology **4**, 10 (2016).

Kimura, S., S. Tuchiya and Y. Suzuki, "Decentralized autonomous mechanism for fault-tolerant control of a kinematically redundant manipulator", in "1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century", vol. 3, pp. 2540–2545 (IEEE, 1995).

Koditschek, D., "Exact robot navigation by means of potential functions: Some topological considerations", in "IEEE International Conference on Robotics and Automation", vol. 4, pp. 1–6 (IEEE, 1987).

Koenig, N. and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator", in "IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", vol. 3, pp. 2149–2154 (IEEE, 2004).

Kowalczyk, W., "Rapid navigation function control for two-wheeled mobile robots", Journal of Intelligent & Robotic Systems **93**, 3-4, 687–697 (2019).

Kuhne, F., W. F. Lages and J. G. da Silva Jr, "Model predictive control of a mobile robot using linearization", in "Proceedings of Mechatronics and Robotics", pp. 525–530 (2004).

Kuwata, Y., A. Richards, T. Schouwenaars and J. P. How, "Distributed robust receding horizon control for multivehicle guidance", IEEE Transactions on Control Systems Technology **15**, 4, 627–641 (2007).

Lages, W. F. and J. A. V. Alves, "Real-time control of a mobile robot using linearized model predictive control", IFAC Proceedings Volumes **39**, 16, 968–973 (2006).

Lam, D., C. Manzie and M. C. Good, "Multi-axis model predictive contouring control", International Journal of Control **86**, 8, 1410–1424 (2013).

Lamiraux, F., D. Bonnafous and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots", IEEE Transactions on Robotics **20**, 6, 967–977 (2004).

Laumond, J.-P., S. Sekhavat and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots", in "Robot motion planning and control", pp. 1–53 (Springer, 1998).

Levy, G. and B. Hochner, "Embodied organization of Octopus vulgaris morphology, vision, and locomotion", Frontiers in Physiology **8**, 164 (2017).

Li, C., Z. Zhang, A. Nesrin, Q. Liu, F. Liu and M. Buss, "Instantaneous local control barrier function: An online learning approach for collision avoidance", arXiv preprint arXiv:2106.05341 (2021).

Liberzon, D., *Switching in systems and control* (Springer Science & Business Media, 2003).

Liu, Y.-H. and S. Arimoto, "Decentralized adaptive and nonadaptive position/force controllers for redundant manipulators in cooperations", The International Journal of Robotics Research **17**, 3, 232–247 (1998).

Lofberg, J., "YALMIP: a toolbox for modeling and optimization in MATLAB", in "2004 IEEE International Conference on Robotics and Automation (ICRA)", pp. 284–289 (IEEE, 2004).

Long, K., C. Qian, J. Cortés and N. Atanasov, "Learning barrier functions with memory for robust safe navigation", IEEE Robotics and Automation Letters **6**, 3, 4931–4938 (2021).

Lynch, K. M. and F. C. Park, *Modern Robotics* (Cambridge University Press, 2017).

Mademlis, I., N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner and A. Messina, "Autonomous unmanned aerial vehicles filming in dynamic unstructured outdoor environments [applications corner]", IEEE Signal Processing Magazine **36**, 1, 147–153 (2018).

Mahony, R., V. Kumar and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor", IEEE Robotics & Automation Magazine **19**, 3, 20–32 (2012).

Maithripala, D. S., J. M. Berg and W. P. Dayawansa, "Almost-global tracking of simple mechanical systems on a general class of Lie groups", IEEE Transactions on Automatic Control **51**, 2, 216–225 (2006).

Majd, K., S. Yaghoubi, T. Yamaguchi, B. Hoxha, D. Prokhorov and G. Fainekos, "Safe navigation in human occupied environments using sampling and control barrier functions", arXiv preprint arXiv:2105.01204 (2021).

Mao, R., H. Gao and L. Guo, "A novel collision-free navigation approach for multiple nonholonomic robots based on ORCA and linear MPC", Mathematical Problems in Engineering **2020** (2020).

Marcotte, R., *Adaptive Communication for Mobile Multi-Robot Systems*, Ph.D. thesis, University of Michigan (2019).

Mayne, D. Q., J. B. Rawlings, C. V. Rao and P. O. Scokaert, "Constrained model predictive control: Stability and optimality", Automatica **36**, 6, 789–814 (2000).

McEvoy, M. A. and N. Correll, "Distributed inverse kinematics for shape-changing robotic materials", Procedia Technology **26**, 4–11 (2016).

McEvoy, M. A. and N. Correll, "Shape-changing materials using variable stiffness and distributed control", Soft Robotics **5**, 6, 737–747 (2018).

Mehrez, M., G. Mann, R. Gosine, K. Worthmann and T. Faulwasser, "Predictive path following of mobile robots without stabilizing terminal constraints", in "20th IFAC World Congress", No. CONF in 1 (2017a).

Mehrez, M. W., *Optimization Based Solutions for Control and State Estimation in Nonholonomic Mobile Robots: Stability Distributed Control and Localization*, Ph.D. thesis, Memorial University of Newfoundland (2017).

Mehrez, M. W., T. Sprodowski, K. Worthmann, G. K. Mann, R. G. Gosine, J. K. Sagawa and J. Pannek, "Occupancy grid based distributed MPC for mobile robots", in "IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 4842–4847 (2017b).

Mehrez, M. W., K. Worthmann, J. P. Cenerini, M. Osman, W. W. Melek and S. Jeon, "Model predictive control without terminal constraints or costs for holonomic mobile robots", Robotics and Autonomous Systems **127**, 103468 (2020).

Mellinger, D. W., *Trajectory generation and control for quadrotors*, Ph.D. thesis, University of Pennsylvania (2012).

Mesbahi, M. and M. Egerstedt, *Graph theoretic methods in multiagent networks* (Princeton University Press, 2010).

Mochiyama, H., E. Shimemura and H. Kobayashi, "Control of serial rigid link manipulators with hyper degrees of freedom: shape control by a homogeneously decentralized scheme and its experiment", in "Proceedings of IEEE International Conference on Robotics and Automation", vol. 3, pp. 2877–2882 (IEEE, 1996).

Murray, R. M., *A mathematical introduction to robotic manipulation* (CRC Press, 2017).

Neppalli, S. and B. A. Jones, "Design, construction, and analysis of a continuum robot", in "Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on", pp. 1503–1507 (IEEE, 2007).

Nho Cho, C., H. Jung, J. Son and K. Gi Kim, "A modular control scheme for hyper-redundant robots", International Journal of Advanced Robotic Systems **12**, 7, 91 (2015).

Nourmohammadi, H. and J. Keighobadi, "Decentralized INS/GNSS system with MEMS-grade inertial sensors using QR-factorized CKF", IEEE Sensors Journal **17**, 11, 3278–3287 (2017).

Olfati-Saber, R., J. A. Fax and R. M. Murray, "Consensus and cooperation in networked multi-agent systems", Proceedings of the IEEE **95**, 1, 215–233 (2007).

Omidshafiei, S., A.-A. Agha-Mohammadi, C. Amato, S.-Y. Liu, J. P. How and J. Vian, "Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions", The International Journal of Robotics Research **36**, 2, 231–258 (2017).

Oriolo, G., A. De Luca and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation", IEEE Transactions on Control Systems Technology **10**, 6, 835–852 (2002).

Pandey, A., S. Pandey and D. Parhi, "Mobile robot navigation and obstacle avoidance techniques: A review", International Robotics and Automation Journal **2**, 3, 96–105 (2017).

Papadopoulos, E., I. Poulakakis and I. Papadimitriou, "On path planning and obstacle avoidance for nonholonomic platforms with manipulators: A polynomial approach", The International Journal of Robotics Research **21**, 4, 367–383 (2002).

Park, J. J., *Graceful Navigation for Mobile Robots in Dynamic and Uncertain Environments.*, Ph.D. thesis, University of Michigan (2016).

Park, J. J., C. Johnson and B. Kuipers, "Robot navigation with model predictive equilibrium point control", in "2012 IEEE/RSJ International Conference on Intelligent Robots and Systems", pp. 4945–4952 (IEEE, 2012).

Qu, Z., J. Wang and C. E. Plaisted, "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles", IEEE Transactions on Robotics **20**, 6, 978–993 (2004).

Radford, N. A., P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater *et al.*, "Valkyrie: NASA's first bipedal humanoid robot", Journal of Field Robotics **32**, 3, 397–419 (2015).

Rawlings, J. B. and D. Q. Mayne, *Model predictive control: Theory and design* (Nob Hill Pub., 2009).

Rawlings, J. B., E. S. Meadows and K. R. Muske, "Nonlinear model predictive control: A tutorial and survey", IFAC Proceedings Volumes **27**, 2, 185–197 (1994).

Reis, J. C., P. U. Lima and J. Garcia, "Efficient distributed communications for multi-robot systems", in "Robot Soccer World Cup", pp. 280–291 (Springer, 2013).

Ren, C., Y. Ding, X. Li, X. Zhu and S. Ma, "Extended state observer based robust friction compensation for tracking control of an omnidirectional mobile robot", Journal of Dynamic Systems, Measurement, and Control **141**, 10 (2019).

Ren, C. and S. Ma, "Trajectory tracking control of an omnidirectional mobile robot with friction compensation", in "IEEE International Workshop on Intelligent Robots and Systems (IROS)", pp. 5361–5366 (IEEE, 2016).

Richards, A. and J. P. How, "Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility", in "Proceedings of the 2003 American Control Conference, 2003.", vol. 5, pp. 4034–4040 (IEEE, 2003).

Rigatos, G. and P. Siano, "An H-infinity feedback control approach to autonomous robot navigation", in "IECON 2014-40th Annual Conference of the IEEE Industrial Electronics Society", pp. 2689–2694 (IEEE, 2014).

Rigatos, G. G., "Differential flatness theory and flatness-based control", in "Nonlinear Control and Filtering Using Differential Flatness Approaches", pp. 47–101 (Springer, 2015).

Rimon, E. and D. E. Koditschek, "Exact robot navigation using artificial potential functions", IEEE Transactions on Robotics and Automation **8**, 5, 501–518 (1992).

Robotis, "Turtlebot3", `http://www.robotis.us/turtlebot-3/` (2020).

Rösmann, C., A. Makarow and T. Bertram, "Online motion planning based on nonlinear model predictive control with non-Euclidean rotation groups", arXiv preprint arXiv:2006.03534 (2020).

Rosolia, U. and A. D. Ames, "Multi-rate control design leveraging control barrier functions and model predictive control policies", arXiv preprint arXiv:2004.01761 (2020).

Saber, R. O. and R. M. Murray, "Consensus protocols for networks of dynamic agents", in "Proceedings of the 2003 American Control Conference, 2003.", vol. 2, pp. 951–956 (2003).

Sadati, S. H., L. Sullivan, I. D. Walker, K. Althoefer and T. Nanayakkara, "Three-dimensional-printable thermoactive helical interface with decentralized morphological stiffness control for continuum manipulators", IEEE Robotics and Automation Letters **3**, 3, 2283–2290 (2018).

Salimi Lafmejani, A. and S. Berman, "Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots (experiments)", `https://www.youtube.com/watch?v=eHKpTs5h3wY` (2020a).

Salimi Lafmejani, A. and S. Berman, "Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots (simulations)", `https://www.youtube.com/watch?v=8fVpX0D_OH4` (2020b).

Salimi Lafmejani, A. and S. Berman, "Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots", Robotics and Autonomous Systems **141**, 103774 (2021).

Salimi Lafmejani, A., S. Berman and G. Fainekos, "NMPC-LBF: Nonlinear MPC with learned barrier function for decentralized safe navigation of multiple robots in unknown environments", `https://www.youtube.com/watch?v=I3mAqJKf5dk` (2021a).

Salimi Lafmejani, A., S. Berman and G. Fainekos, "Nonlinear MPC with barrier certificates for safe multi-robot navigation in environments with obstacles", `https://www.youtube.com/watch?v=c1k0Yl-0hL4` (2021b).

Salimi Lafmejani, A., S. Berman and G. Fainekos, "NMPC-LBF: Nonlinear MPC with learned barrier function for decentralized safe navigation of multiple robots in unknown environments", Under review for the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2022a).

Salimi Lafmejani, A., S. Berman and G. Fainekos, "Nonlinear MPC with barrier certificates for safe multi-robot navigation in environments with obstacles", In preparation (2022b).

Salimi Lafmejani, A., B. Danaei, A. Kalhor and M. T. Masouleh, "An experimental study on control of a pneumatic 6-DoF Gough-Stewart robot using backstepping-sliding mode and geometry-based quasi-forward kinematic method", in "5th RSI International Conference on Robotics and Mechatronics (ICRoM)", pp. 239–245 (IEEE, 2017).

Salimi Lafmejani, A., A. Doroudchi, H. Farivarnejad, X. He, D. Aukes, M. M. Peet, H. Marvi, R. E. Fisher and S. Berman, "Kinematic modeling and trajectory tracking control of an octopus-inspired hyper-redundant robot", IEEE Robotics and Automation Letters **5**, 2, 3460–3467 (2020a).

Salimi Lafmejani, A., H. Farivarnejad and S. Berman, "Adaptation of gradient-based navigation control for holonomic robots to nonholonomic robots", IEEE Robotics and Automation Letters **6**, 1, 191–198 (2020b).

Salimi Lafmejani, A., H. Farivarnejad and S. Berman, "Adaptation of gradient-based navigation control for holonomic robots to nonholonomic robots", `https://www.youtube.com/watch?v=IzGPDI_aoXw` (2020c).

Salimi Lafmejani, A., H. Farivarnejad and S. Berman, "H_∞-optimal tracking controller for three-wheeled omnidirectional mobile robots with uncertain dynamics", in "2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 7587–7594 (2020d).

Salimi Lafmejani, A., H. Farivarnejad, A. Doroudchi and S. Berman, "A consensus strategy for decentralized kinematic control of multi-segment soft continuum robots", in "2020 American Control Conference (ACC)", pp. 909–916 (IEEE, 2020e).

Salimi Lafmejani, A., H. Farivarnejad, M. Rezayat, F. Z. Sorkhabadi, A. Doroudchi and S. Berman, "Tracking of a ground moving target by multiple unmanned aerial vehicles (experiments)", `https://www.youtube.com/watch?v=G1DW2Pq_p7M` (2020f).

Salimi Lafmejani, A., H. Farivarnejad, M. Rezayat, F. Z. Sorkhabadi, A. Doroudchi and S. Berman, "Tracking of a ground moving target by multiple unmanned aerial vehicles (simulations)", `https://www.youtube.com/watch?v=GqF56jXwRIs` (2020g).

Salimi Lafmejani, A., H. Farivarnejad, M. Rezayat, F. Z. Sorkhabadi, A. Doroudchi and S. Berman, "Collision-free velocity tracking of a moving ground target by multiple unmanned aerial vehicles", 4th International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM) (2021c).

Salimi Lafmejani, A., M. T. Masouleh and A. Kalhor, "Trajectory tracking control of a pneumatically actuated 6-DOF Gough-Stewart parallel robot using backstepping-sliding mode controller and geometry-based quasi forward kinematic method", Robotics and Computer-Integrated Manufacturing **54**, 96–114 (2018).

Samson, C., "Time-varying feedback stabilization of car-like wheeled mobile robots", The International Journal of Robotics Research **12**, 1, 55–64 (1993).

Sartoretti, G., Y. Wu, W. Paivine, T. S. Kumar, S. Koenig and H. Choset, "Distributed reinforcement learning for multi-robot decentralized collective construction", in "Distributed Autonomous Robotic Systems", pp. 35–49 (Springer, 2019).

Saska, M., V. Spurnỳ and V. Vonásek, "Predictive control and stabilization of nonholonomic formations with integrated spline-path planning", Robotics and Autonomous Systems **75**, 379–397 (2016).

Saveriano, M. and D. Lee, "Learning barrier functions for constrained motion planning with dynamical systems", in "IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 112–119 (2019).

Savkin, A. V., A. S. Matveev, M. Hoy and C. Wang, *Safe robot navigation among moving and steady obstacles* (Butterworth-Heinemann, 2015).

Scherer, C., "Theory of robust control", Delft University of Technology pp. 1–160 (2001).

Sharifzadeh, M., R. Khodambashi and D. M. Aukes, "An integrated design and simulation environment for rapid prototyping of laminate robotic mechanisms", in "ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference", (2018).

Shim, H.-S. and Y.-G. Sung, "Asymptotic control for wheeled mobile robots with driftless constraints", Robotics and Autonomous Systems **43**, 1, 29–37 (2003).

Siciliano, B., "Kinematic control of redundant robot manipulators: A tutorial", Journal of Intelligent and Robotic Systems **3**, 3, 201–212 (1990).

Slotine, J.-J. E., W. Li *et al.*, *Applied nonlinear control*, vol. 199 (Prentice Hall Englewood Cliffs, NJ, 1991).

Soylu, S., B. J. Buckham and R. P. Podhorodeski, "Dexterous task-priority based redundancy resolution for underwater manipulator systems", Transactions of the Canadian Society for Mechanical Engineering **31**, 4, 519–533 (2007).

Sreedhar, N. and S. Rao, "Stability of a system of linear differential equations", IEEE Transactions on Automatic Control **13**, 3, 307–308 (1968).

Srinivasan, M., A. Dabholkar, S. Coogan and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach", in "IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 7139–7145 (2020).

Stella, L., A. Themelis, P. Sopasakis and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control", in "2017 IEEE 56th Annual Conference on Decision and Control (CDC)", pp. 1939–1944 (IEEE, 2017).

Stellato, B., G. Banjac, P. Goulart, A. Bemporad and S. Boyd, "OSQP: An operator splitting solver for quadratic programs", Mathematical Programming Computation URL `https://doi.org/10.1007/s12532-020-00179-2` (2020).

Sturm, J. F., "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones", Optimization Methods and Software **11**, 1-4, 625–653 (1999).

Tallamraju, R., S. Rajappa, M. J. Black, K. Karlapalem and A. Ahmad, "Decentralized MPC based obstacle avoidance for multi-robot target tracking scenarios", in "2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)", pp. 1–8 (IEEE, 2018).

Tamba, T. A., B. Hong and K.-S. Hong, "A path following control of an unmanned autonomous forklift", International Journal of Control, Automation and Systems **7**, 1, 113–122 (2009).

Tamimi, J. and P. Li, "Nonlinear model predictive control using multiple shooting combined with collocation on finite elements", IFAC Proceedings Volumes **42**, 11, 703–708 (2009).

Tanner, H. G. and K. J. Kyriakopoulos, "Discontinuous backstepping for stabilization of nonholonomic mobile robots", in "IEEE International Conference on Robotics and Automation", vol. 4, pp. 3948–3953 (IEEE, 2002).

Tayefi, M. and Z. Geng, "Logarithmic control, trajectory tracking, and formation for nonholonomic vehicles on Lie group SE(2)", International Journal of Control **92**, 2, 204–224 (2019).

Tzafestas, S. G., *Introduction to mobile robot control* (Elsevier, 2013).

Urakubo, T., "Feedback stabilization of a nonholonomic system with potential fields: application to a two-wheeled mobile robot among obstacles", Nonlinear Dynamics **81**, 3, 1475–1487 (2015).

Vázquez, J. and M. Velasco-Villa, "Path-tracking dynamic model based control of an omnidirectional mobile robot", IFAC Proceedings Volumes **41**, 2, 5365–5370 (2008).

Vittor, T. and R. Willgoss, "Motion analysis for decentralized control of n-module hyperredundant manipulators", in "Australian Conference on Robotics and Automation", (2005).

Wächter, A., "Short tutorial: getting started with IPOPT in 90 minutes", in "Dagstuhl Seminar Proceedings", (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009).

Walker, I. D., "Continuous backbone "continuum" robot manipulators", ISRN Robotics **2013** (2013).

Walker, I. D., "The importance of torsion in robot backbones", Recent Advances in Circuits, Systems, Signal Processing and Communications. World Scientific and Engineering Academy and Society **4**, 25–31 (2014).

Walker, I. D., H. Choset and G. S. Chirikjian, "Snake-like and continuum robots", in "Springer Handbook of Robotics", pp. 481–498 (Springer, 2016).

Wang, C., X. Liu, X. Yang, F. Hu, A. Jiang and C. Yang, "Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy", Applied Sciences **8**, 2, 231 (2018).

Wang, L., A. D. Ames and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems", IEEE Transactions on Robotics **33**, 3, 661–674 (2017).

Wang, Y. and S. Boyd, "Fast model predictive control using online optimization", IEEE Transactions on Control Systems Technology **18**, 2, 267–278 (2009).

Webster III, R. J. and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review", The International Journal of Robotics Research **29**, 13, 1661–1683 (2010).

Widyotriatmo, A. and K.-S. Hong, "Navigation function-based control of multiple wheeled vehicles", IEEE Transactions on Industrial Electronics **58**, 5, 1896–1906 (2011).

Williams, R. L., B. E. Carter, P. Gallina and G. Rosati, "Dynamic model with slip for wheeled omnidirectional robots", IEEE Transactions on Robotics and Automation **18**, 3, 285–293 (2002).

Williams, R. L. and D. A. Lawrence, *Linear State-Space Control Systems* (John Wiley and Sons, Inc., Hoboken, N.J., 2007).

Worthmann, K., *Stability analysis of unconstrained receding horizon control schemes*, Ph.D. thesis, University of Bayreuth (2011).

Worthmann, K., M. W. Mehrez, M. Zanon, G. K. Mann, R. G. Gosine and M. Diehl, "Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs", IEEE Transactions on Control Systems Technology **24**, 4, 1394–1406 (2015a).

Worthmann, K., M. W. Mehrez, M. Zanon, G. K. Mann, R. G. Gosine and M. Diehl, "Regulation of differential drive robots using continuous time MPC without stabilizing constraints or costs", IFAC-PapersOnLine **48**, 23, 129–135 (2015b).

Xiao, W. and C. Belta, "Control barrier functions for systems with high relative degree", in "2019 IEEE 58th Conference on Decision and Control (CDC)", pp. 474–479 (IEEE, 2019).

Xu, K., N. Simaan *et al.*, "An investigation of the intrinsic force sensing capabilities of continuum robots", IEEE Transactions on Robotics **24**, 3, 576–587 (2008).

Yaghoubi, S., G. Fainekos and S. Sankaranarayanan, "Training neural network controllers using control barrier functions in the presence of disturbances", in "IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)", pp. 1–6 (2020).

Yu, S., X. Li, H. Chen and F. Allgöwer, "Nonlinear model predictive control for path following problems", International Journal of Robust and Nonlinear Control **25**, 8, 1168–1182 (2015).

Yu, X., X. Wang, D. Meng, H. Liu and B. Liang, "Collision free path planning for multi-section continuum manipulators based on a modal method", in "IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems", pp. 236–242 (2018).

Zarei, M., N. Kashi, A. Kalhor and M. T. Masouleh, "Experimental study on shared-control of a mobile robot via a haptic device with an optimal velocity obstacle based receding horizon control approach", Journal of Intelligent & Robotic Systems **97**, 2, 357–372 (2020).

Zeng, J., B. Zhang and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function", arXiv preprint arXiv:2007.11718 (2020).

Zhao, H., X. Zeng, T. Chen and Z. Liu, "Synthesizing barrier certificates using neural networks", in "Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control", pp. 1–11 (2020).

Zheng, T., D. T. Branson, E. Guglielmino and D. G. Caldwell, "A 3d dynamic model for continuum robots inspired by an octopus arm", in "IEEE International Conference on Robotics and Automation", pp. 3652–3657 (IEEE, 2011).

Zheng, T., D. T. Branson, R. Kang, M. Cianchetti, E. Guglielmino, M. Follador, G. A. Medrano-Cerda, I. S. Godage and D. G. Caldwell, "Dynamic continuum arm model for use with underwater robotic manipulators inspired by Octopus vulgaris", in "IEEE International Conference on Robotics and Automation", pp. 5289–5294 (IEEE, 2012).

Zhou, Y., H. Hu, Y. Liu and Z. Ding, "Collision and deadlock avoidance in multirobot systems: A distributed approach", IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**, 7, 1712–1726 (2017).

Zohdi, T. I., "Multiple UAVs for mapping: A review of basic modeling, simulation, and applications", Annual Review of Environment and Resources **43**, 523–543 (2018).