Design, Analysis, Planning, and Control of a Novel Modular Self-Reconfigurable Robotic System

Shumin Feng

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

Pinhas Ben-Tzvi, Chair
Alan T Asbeck
Corina Sandu
Daniel J Stilwell

December 2, 2021
Blacksburg, Virginia

Keywords: Modular robotics, Self-reconfiguration, Trajectory optimization, Trajectory tracking

Design, Analysis, Planning, and Control of a Novel Modular Self-Reconfigurable Robotic System

Shumin Feng

(ABSTRACT)

This dissertation describes the design, analysis, planning, and control of a self-reconfigurable modular robotic system. The proposed robotic system mainly contains three major types of robotic modules: load carrier, manipulation module, and locomotion module. Each module is capable of navigation and interaction with the environment individually. In addition, the robotic system is proposed to reassemble autonomously into various configurations to perform complex tasks such as humanoid configuration to enable enhanced functionality to reconfigure into a configuration that would enable the system to cross over a ditch. A non-back drivable active docking mechanism with two Degrees of Freedom (DOFs) was designed to fit into the tracked units of the robot modules for achieveing the reconfiguration. The quantity and location of the docking mechanisms are customizable and selectable to satisfy various mission requirements and adapt to different environments. During the reconfiguration process, the target coupling mechanism of each module reconfigurable with each other autonomously. A Lyapunov function-based precision controller was developed to align the target docking mechanisms in a close range and high precision for assembling the robot modules autonomously into other configurations.

Additionally, an trajectory optimization algorithm was developed to help the robot determine when to switch the locomotion modes and find the fastest path to the destination with the desired pose.

Design, Analysis, Planning, and Control of a Novel Modular Self-Reconfigurable Robotic System

Shumin Feng

(GENERAL AUDIENCE ABSTRACT)

Though the capabilities of individual robot platforms have advanced greatly from their original rigid construction to more modern reconfigurable platforms, it is still difficult to build a singular platform capable of adapting to all situations and environments that users may want or need it to function in. To help improve the versatility of robot systems, modular robots have become an active area of research. These modular robotic systems are made up of multiple robotic platforms capable of docking together, breaking apart, or otherwise reconfiguring to form a multitude of shapes to overcome and adapt to many diverse challenges and environments. This dissertation describes the design of a new modular robotic system with autonomous docking and reconfiguration. Existing technologies and motivations for the creation of a new modular robotic system are covered. Then the physical design, with a primary focus on the docking mechanism, is reviewed. A validation of the proposed robotic system in a virtual environment with real physical properties is demonstrated. Following this, the development of a Lyapunov function-based controller to autonomously align the docking mechanisms is presented. The overall docking process was also preliminarily verified using a prototype of a locomotion module and an active docking mechanism. In addition, the trajectory optimization and tracking methods are presented in the end.

DEDICATION

This dissertation is dedicated to my parents, Quanbin Feng and Cuimei Li for their support, encouragement and endless love.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Dr. Pinhas Ben-Tzvi for helping and supporting my research. I would like to thank my doctoral committee members Dr. Pinhas Ben-Tzvi, Dr. Alan Asbeck, Dr. Corina Sandu and Dr. Daniel Stilwell for providing me with constructive comments and helping me complete my Ph.D. degree. I would like to thank all of the faculty and staff of Virginia Tech who took care of my life and study.

I would like to thank my parents for their supports and encourages whenever I face challenges and difficulties. I am grateful for my colleagues in the Robotics and Mechatronics Lab during my stay. I would like to thank Isaac Pressgrove for greatly helping with building the prototype and analyzing the active docking mechanism. I would like to thank Ruichang Chen's help with conducting tests of the robot models in simulation. I would also like to thank Yujiong Liu for his support and help with completing the dissertation.

This research is based on work partially supported by the Defense Advanced Research Projects Agency under grant number FA8100-19-P-0022.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background

This dissertation presents the design, analysis, planning, and control of a self-reconfigurable modular robotic system. The proposed robotic system was conceived to leverage the advantages of modular, reconfigurable robotic systems to solve real-world problems. Three major types of robotic modules were designed to satisfy various mission requirements and adapt to different environments. In addition, the modules are capable of assembling autonomously into other configurations to perform complex tasks cooperatively.

This research is motivated by the question of how to combine the advantages of a group of small robots and rigid-structured large robots. Multiple mobile robot systems have drawn increasing research attention. Researchers generally agree that a group of mobile robots have several advantages over single-robot systems in some cases.

In the past decades, various multi-robot architectures have been developed and analyzed for completing diverse missions. For example, an early multi-robot system, named the Nerd herd [1, 2], as shown in Figure 1.1a, demonstrated social behaviors of a group of homogeneous robots with simple individual capabilities; another early work successfully completed a mock clean-up task using a robot team in the ALLIANCE [3, 4] architecture (Figure 1.1b); a group of warehouse robots [5, 6] can navigate through compact spaces and transport shelves

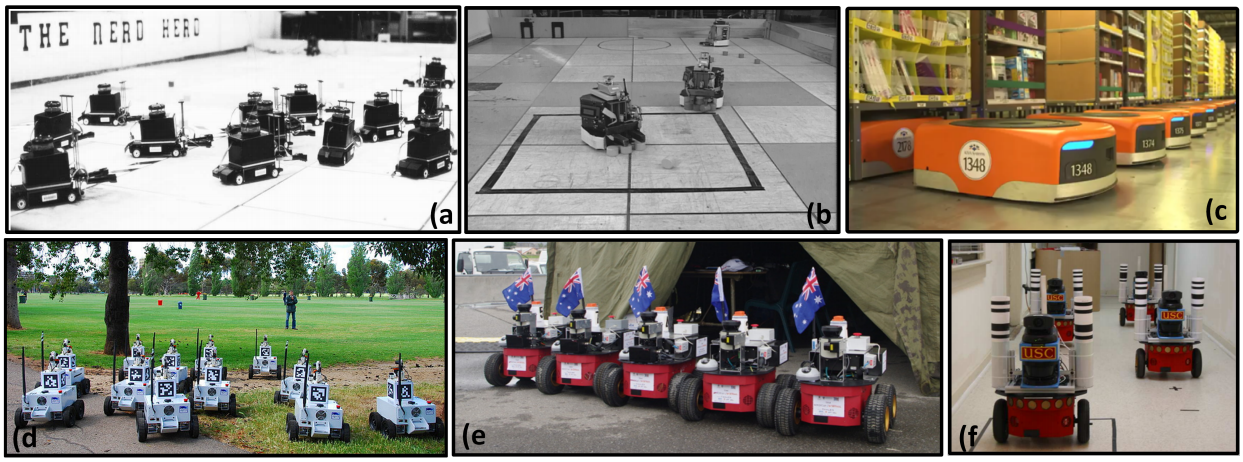

Figure 1.1: Multi-robot systems, (a) The Nerd Herd [1, 2], (b) ALLIANCE architecture [3, 4], (c) Ware house robots [5, 6], (d) Robot Wars [7], (e) WAMbot [8], and f) a robot team built for the DARPA Software for Distributed Robotics (SDR) program [9]

of products simultaneously to achieve high efficiency, as shown in Figure 1.1c; Robot Wars [7], the winner of the magic 2010 robotic competition, was developed to collaboratively map an urban area, find and catch bad guys, and identify and "neutralize" bombs (Figure 1.1d); WAMbots [5, 6], designed to navigate, explore and map large-scale urban environments while performing visual object recognition (Figure 1.1e); A team of robots (Figure 1.1f) was built for mapping, and intruder detection in an indoor environment under the US Defense Advanced Research Projects Agency (DARPA) funded software for distributed robotics (SDR) program [9].

In conclusion, a multi-robot system can complete tasks that are not possible to be completed by a single robot or can accomplish tasks more efficiently than a single robot. Besides, these small robots have the ability to navigate through compact spaces.

However, compared to large robots, such as Matilda [12], Packbot [10], TAROS [14], TALON[11], SherpaTT [15], and Ralter [13], as shown in Figure 1.2, they do lack characteristics such as adaptation to uneven ground and manipulation capabilities.

In order to bridge the gap between multiple small robot teams and large rigid-structure

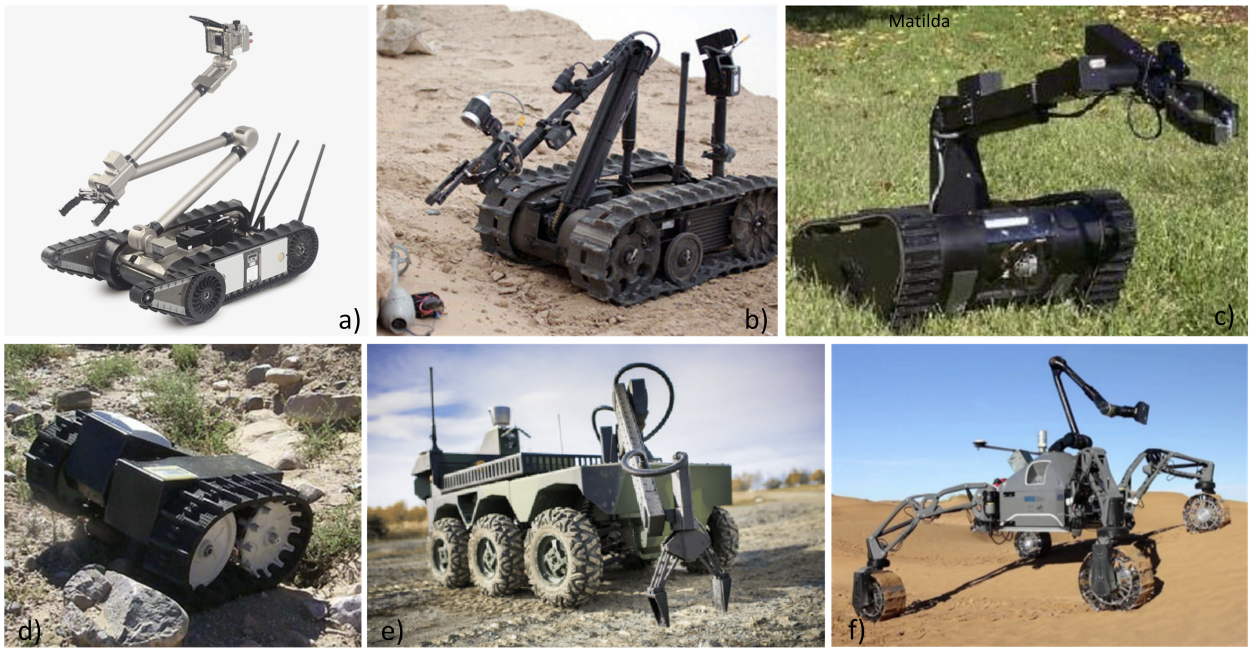

Figure 1.2: Rigid-structure robots, (a) Packbot [10], (b) TALON [11], (c) Matilda [12], (d) Ralter [13], (e) TAROS [14], and f) SherpaTT [15]

robot functionalities to better respond to the challenges of complex tasks and environments, modular robots with reconfigurable capabilities are proposed.

1.2 Summary of Contributions

This dissertation describes the design, analysis, and preliminary validation of a novel modular robotic system. In addition, a pose stabilization controller, a direct collocation-based trajectory optimization approach, and a Model Predictive Control (MPC)-based trajectory tracking method were developed to improve the autonomy of the proposed system, especially to achieve the autonomous self-reconfiguration of the proposed robot modules. The major contributions of this dissertation can be broadly summarized as follows:

1. Modular and reconfigurable robots: a review of existing modular robotic systems along

with various coupling interfaces is presented. Two applications of the proposed modular robotic system are presented: Self-configurable and Transformable Omni-Directional Robotic Modules (STORM) and Self-driving Modular AI-based Robot for Rough Terrain (SMARRT). Mechanical design of the robot modules and docking mechanism of the SMARRT system is described in detail.

2. Validation in a 3D simulator: The proposed SMARRT robot modules were simulated with real physical parameters in the CoppeliaSim simulator. A 3D simulation environment with several different terrain types was built to test the proposed system in different configurations. User commands were sent to the simulator via the Robot Operating System (ROS) to deploy the robot module for reconfiguration and to navigate the various terrains.

3. Pose Stabilization Controller: A review of controllers for mobile robots to achieve trajectory tracking and pose stabilization is presented. A Lyapunov function-based pose stabilization controller was developed to steer the robot to the desired pose and align the docking mechanism in preparation for achieving reconfiguration. Simulations in Matlab and real-world experiments were conducted for validation of the proposed controller.

4. Direct Collocation-based Trajectory Optimization: A review of trajectory optimization methods is described. A direct collocation-based optimization method was developed for determining the best position for switching the system dynamics of the hybrid locomotion module and planning the shortest path to the desired pose.

5. MPC-based Trajectory Tracking: To ensure that the robot module can follow the optimal trajectory from the proposed direct collocation-based optimization and arrive at the desired pose in the end, a MPC-based trajectory tracking controller was developed.

1.3 Dissertation Structure

The outline of this dissertation is described as follows:

Chapter 1 introduces the motivation behind the research and main contributions of the dissertation.

Chapter 2 presents a literature review of existing state-of-the-art modular robotic systems, relevant control theories and optimization methods.

Chapter 3 demonstrates applications of the proposed modular robotic system and the detailed design of the SMARRT architecture of the proposed modular, reconfigurable robot modules, along with detailed analysis of the docking mechanism.

Chapter 4 presents the validations and analysis of the SMARRT system in simulator with physics engine to prove the feasibility of the proposed SMARRT modules in different configurations.

Chapter 5 presents the development of techniques for achieving autonomous alignment and further completing self-reconfiguration tasks. The overall autonomous docking process were demonstrated in real-world using a prototype of an active docking mechanism and a prototype of a STORM locomotion module. The proposed pose stabilization controller was implemented using python and ROS running on an on-board computer of the locomotion module for the autonomous alignment task.

Chapter 6 presents the proposed trajectory optimization method for discovering an optimal path towards autonomous alignment along with choosing the best position to change the dynamics of the locomotion module, followed by the introduction of the proposed MPC-based trajectory tracking.

Chapter 7 concludes the dissertation by providing a summary of the presented work and

includes a discussion about potential directions for future research.

1.4 Related Publications

Disclosure: This dissertation uses content directly adapted from the following peer-reviewed journal publications and conference proceedings:

Peer-reviewed Journals:

- Feng, S., Sebastian, B., Ben-Tzvi, P., "A Collision Avoidance Method Based on Deep Reinforcement Learning", Robotics Journal, Vol. 10, Issue 2, pp. 1-19, May 2021.
- Feng, S., Pressgrove, I., Ben-Tzvi, P., "A Modular Robotic System with Self-Reconfiguration Strategies Demonstating Autonomous Module-to-Module Alignment and Docking ", Journal of Mechanisms and Robotics. (Submitted).
- Feng, S., Liu, Y., Ben-Tzvi, P., "A Hybrid Navigation Strategy for a Multi-directional Mobile Robot with Switching Kinematics Towards Autonomous Alignment and Docking", Journal of Robotics and Autonomous Systems. (In Preparation)
- Liu, Y., Feng, S., Ben-Tzvi, P., "Time-optimal Trajectories for a Multi-directional Vehicle with Bounded Curvatures", Automatica. (In Preparation)

Peer-reviewed Conference Papers:

- Feng, S., Ren, H., Wang, X., Ben-Tzvi, P., "Mobile Robot Obstacle Avoidance Based on Deep Reinforcement Learning", Proceedings of the 2019 ASME IDETC/CIE, 43rd Mechanisms and Robotics Conference, Anaheim, CA, Aug. 18-21, 2019.

Chapter 2

LITERATURE REVIEW AND PROBLEM STATEMENT

This chapter presents a literature review of the state-of-the-art modular, reconfigurable robotic systems focusing on classifications, docking mechanisms, and locomotion types. In addition, the related control and optimization techniques are reviewed in preparation for developing controllers to maneuver the modules, especially to aid the autonomous alignment process, followed by a problem statement and proposed solution.

2.1 Modular and Reconfigurable Robots

A group of autonomous kinematic machines makes up a modular, reconfigurable robotic system with variable morphology. As opposed to traditional fixed structure robots, modular and reconfigurable robots are able to alter their shapes by changing the connections between their own components or forming into a new structure by combining with other robot modules via a variety of docking mechanisms. This allows modular and reconfigurable robots to provide diverse functionalities, achieve complex tasks, and adapt to new environments.

2.1.1 Classification of Modular, Reconfigurable Robots

According to the characteristics and variation of modular, reconfigurable robots, they can be generally classified into two categories: Whole Body Locomotion (WBL) and Mobile Configuration Change (MCC). The WBL is further differentiated by various architectures, such as lattice architecture, chain architecture, and the combination of both, whereas the MCC typically results in a chain-like formation.

Besides, reconfigurable modules have various mobilities and docking interfaces that can be considered features for classifying reconfigurable robots. For example, the groups of robot modules classified as WBL usually change their morphology in one reconfigured formation to provide various locomotion types, such as undulation, quadruped, rolling, and Biped. The mechanism and structure of a single module is generally simple and less expensive to build. However, the simple mechanism reduces the DOFs that can be achieved via a single robot. Otherwise, modules classified as MCC, always have individual locomotion capabilities, and their mobilities, such as track and wheel, can be enhanced after reconfiguration [30].

2.1.2 Docking Interfaces

Various docking interfaces [16] have been developed to satisfy different requirements in terms of reconfiguration. Some can be classified as mechanical couplers, such as pin and hole, hooks, lock and key, as well as shape matching. Some other couplers relay on magnetic connections. The key factors towards designing and analyzing the coupling mechanism can be mainly summzrized as: gender, acutation, and misalignment tolerance.

The gender of the docking mechanism is related to the mechanical structure and reconfigurability of the system. Gendered coupling mechanisms generally consist of male and female connectors with active or passive interfaces. A more flexible version of the docking

mechanisms are equipped with bi-genered coupling interfaces with anti-symmetric interfaces with active elements from both sides. The connection can be established between any pair of docking mechanisms. Another type of coupling is similar with bi-gendered mechanisms, known as genderless coupling that has anti-symmetric interfaces and failsafe characteristic. Namely, a genderless coupling mechanism enables the modules to decouple by actuating one side.

In terms of actuation, the typical methods to activate the docking mechanisms are through motors, magnets, as well as Shape Memory Alloy (SMA) wire. Geared motors are selected if the connection between the modules requires high torque and force. The motors, together with non-back drivable mechanisms establish strong and reliable connection and maintain the connectors in place without consuming power. Due to the complexcity and high-torque features, these types of actuation usually apply to relatively large modules and docking mechanisms. Other actuation types, such as electro-magnets and SMAs, need to consume power to main the connection and mainly apply to micro-scale coupling interfances.

If the modules are required to align the docking mechanisms autonomously, the allowable maximum misalignment along different directions, as shown in Figure 2.1 becomes one of the key features to analyze for improving the success rate when aligning the robot modules and the docking interfaces without human interaction.

2.1.3 Examples and Comparison of the Previous Work

Several modular robot designs were studied and classified, as shown in Figure 2.2 and Table 2.1. The primary points of interest of this study are in robotic mobility, docking interfaces [16, 31, 32], and reconfiguration strategies [33, 34]. In previous research, ATRON [35, 36] modules were designed with a near-spherical geometry to provide smoother and more continuous

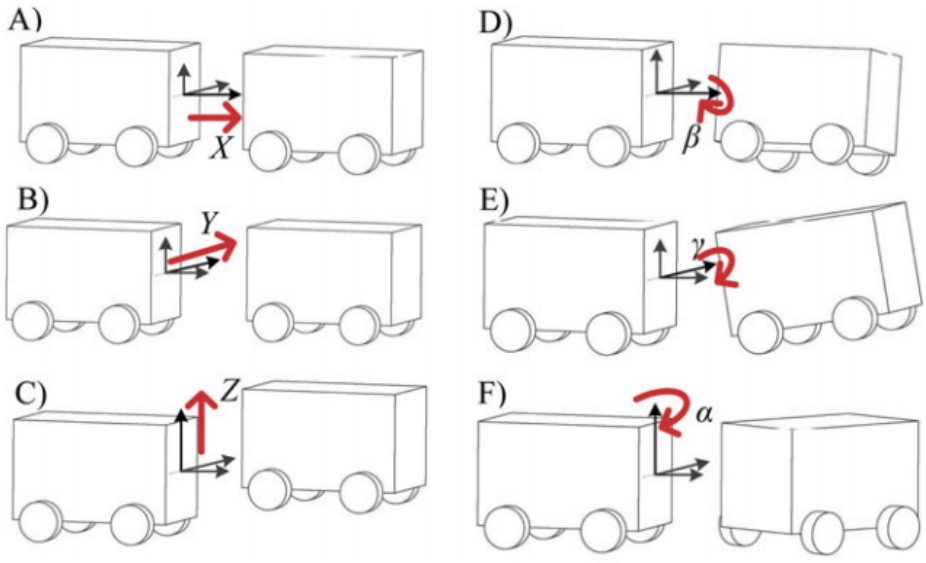

Figure 2.1: Misalignment along different directions A) X, B) Y, C) Z, D Roll β, E) Pitch γ, F) Yaw α [16]

mobility patterns. The reconfiguration capability of this system was achieved through the use of electromechanical active and passive connectors.

M-TRAN [37, 38] consists of semi-cylindrical modules attached in either a lattice or a chain

Table 2.1: Study of Modular Reconfigurable Robotic Systems

Robot Name	Category	Mobility	Docking Interface
AMOEBA [20]	MCC	Track, Train Formation	Link-type interface
JL-1 [24]	MCC	Track, Train Formation, Undulations	Coupler mechanism
Swarm-bots [22]	MCC	Track, Wheel, Train Formation	Robotic gripper
ATRON [17]	WBL	Planar Linear or Rotary, Quadruped, Train Formation	Electromechanical male/female connectors
CKBot [23]	WBL	Rolling, Quadruped, Undulations	Magnetic docking interface
M-TRAN [18]	WBL	Rolling, Quadruped, Spider, Undulations	Link-type interface
Roombots [19]	WBL	Biped, Quadruped, Spider	Mechanical latching fingers
SMORES-EP [21]	MCC/WBL	Wheel, Rolling, Spider, Undulations	Electro-permanent magnets

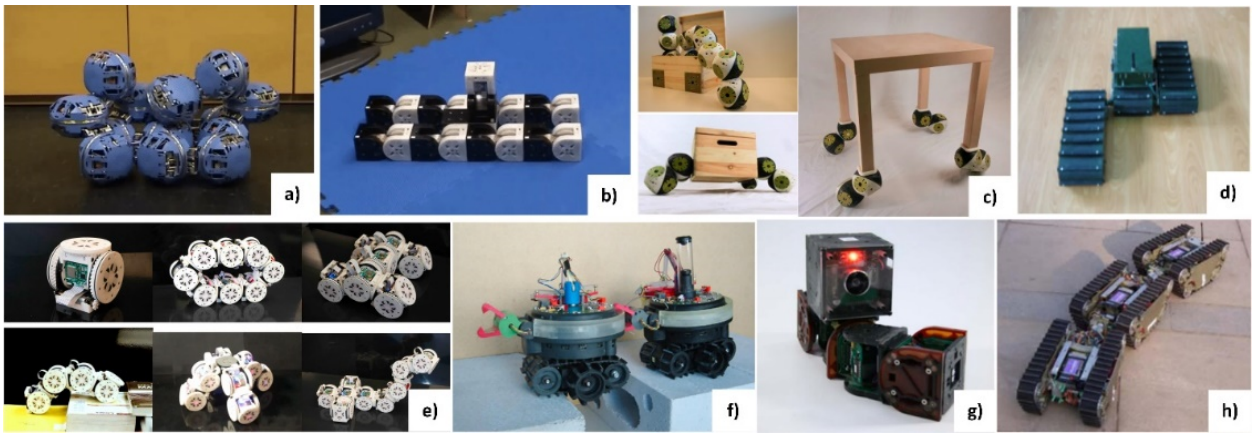

Figure 2.2: Modular and reconfigurable robotic systems. a) ATRON [17], b) M-TRAN [18], c) Roombots [19], d) AMOEBA [20], e) SMORES-EP [21], f) Swarm-bot [22], g) CKBOT [23], h) JL [24]

architecture. Each M-TRAN module was composed of a link and two boxes with three connection surfaces utilizing permanent magnets. Although an M-TRAN module cannot provide individual mobility, complex locomotion patterns such as rolling, quadrupedal walking, and undulation were demonstrated after docking.

Roombots [39, 40, 41] can reconfigure into furniture-like structures that can move, self-assemble, and self-repair.

AMOEBA module [20] is a tracked robot with one track covering the whole body. It is able to move only in one direction without the capability to change direction on its own. Once docking with other modules, it can implement differential drive to make turns.

CKBot modules have magnetic docking interfaces to achieve reconfiguration. Besides, the modules can be connected physically and manually using screws. After connection, the CKBot modules can be self-reconfigured using permanent magnets embedded in the faces to exhibit walking, undulating and rolling capabilities.

Swarm-bots [42, 43, 44] were able to use their grippers to hold on to one another and form train-like formations to achieve reconfigurability. JL [45, 46, 47, 48] was designed to adapt

to different tasks and environments, especially for tasks on rough terrain. A cone-shaped connector and a matching coupler incorporated in the module's center between the tracked units enabled the reconfigurability to provide diverse locomotion patterns.

2.2 Related Control Theories and Optimization Methods

The self-reconfiguration challenges pertaining to the proposed modular robotic system can be defined as trajectory planning and motion planning in general. Namely, a robot module is supposed to be maneuvered to the desired position to align with the other in order to achieve reconfiguration via a docking mechanism. An efficient self-reconfiguration controller is supposed to command the robotic modules to the desired poses. Basically, a well-defined feedback controller should be capable of minimizing the differences between the current pose and the desired pose and choose proper control inputs to the robot. The robot motions rely on the tracking error at the current time step that reduces the effects of external disturbances, such as unexpected passive robot motions when performing autonomous docking on uneven terrain. An optimal trajectory from the initial position to the desired position can be planned for the robot to track, which is known as trajectory optimization. After the optimal trajectory is obtained, an effective trajectory tracking controller is required to plan the robot motions. In summary, three main concepts are considered to solve the self-reconfiguration problem: feedback control, trajectory optimization, and trajectory tracking.

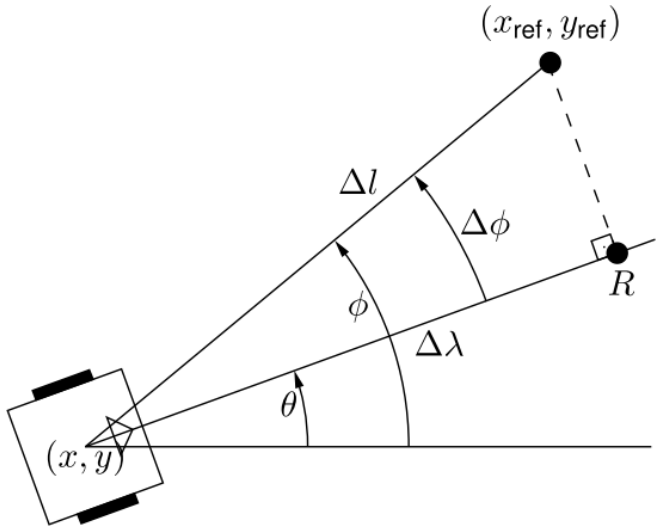

Figure 2.3: Robot position problem [25]

2.2.1 Feedback Control for Mobile Robot Stabilization

The objective of the feedback control is to minimize the error between an initial pose (x_0, y_0, θ_0), including the orientation, to the desired pose (x_d, y_d, θ_d). This control process is also known as mobile robot stabilization. One of the developed control strategies is based on generating a moving reference [25]. The control problem is divided into robot positioning control and robot orientation control. Figure 2.3 illustrates the positioning problem, where the robot needs to reach the desired position (x_{ref}, y_{ref}). This problem can be solved by applying any control strategy to minimize the tracking error $e_c = \Delta\lambda$ and $e_\theta = \Delta\phi$, defined as:

$$\begin{aligned} e_s &= \Delta\lambda = \Delta l \cos\Delta\phi = \sqrt{(\Delta x_{ref})^2 + (\Delta y_{ref})^2}\cos\left[\tan^{-1}\left(\frac{\Delta y_{ref}}{\Delta x_{ref}}\right) - \theta\right] \\ e_\theta &= \tan^{-1}\left(\frac{\Delta y_{ref}}{\Delta x_{ref}}\right) - \theta \end{aligned} \tag{2.1}$$

A moving reference is defined, as shown in Figure 2.4, to overcome the robot orientation problem. The main idea is to drive the robot to another location until the robot can go

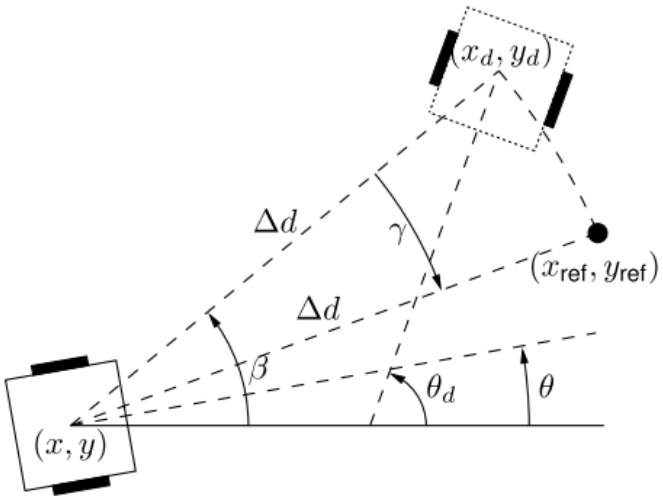

Figure 2.4: Robot orientating problem [25]

straight to the final position with the desired orientation. The moving reference is calculated as,

$$\begin{aligned} \Delta d &= \sqrt{(x_d - x)^2 + (y_d - y)^2} = \sqrt{(\Delta x_d)^2 + (\Delta y_d)^2} \\ \beta &= \tan^{-1}\left(\frac{y_d - y}{x_d - x}\right) = \tan^{-1}\left(\frac{\Delta y_d}{\Delta x_d}\right) \end{aligned} \tag{2.2}$$

Some other controllers are designed based on Lyapunov functions [26, 49]. In these methods, the kinematics of the robot should be converted into polar coordinates as shown in Equation 2.3 and Figure 2.5.

$$\begin{cases} \dot{\rho} = -v\cos\alpha \\ \dot{\phi} = v\dfrac{\sin\alpha}{\rho} \\ \dot{\alpha} = -\omega + v\dfrac{\sin\alpha}{\rho} \end{cases} \tag{2.3}$$

where ρ is the distance between O_1 and O_2, $\phi = \text{atan}\,2(Y_2 - Y, X_2 - X) - \theta_2$, $\alpha = \text{atan}\,2(Y_2 - Y, X_2 - X) - \theta$.

The control low is then derived by constructing appropriate Lyapunov functions. Consider a

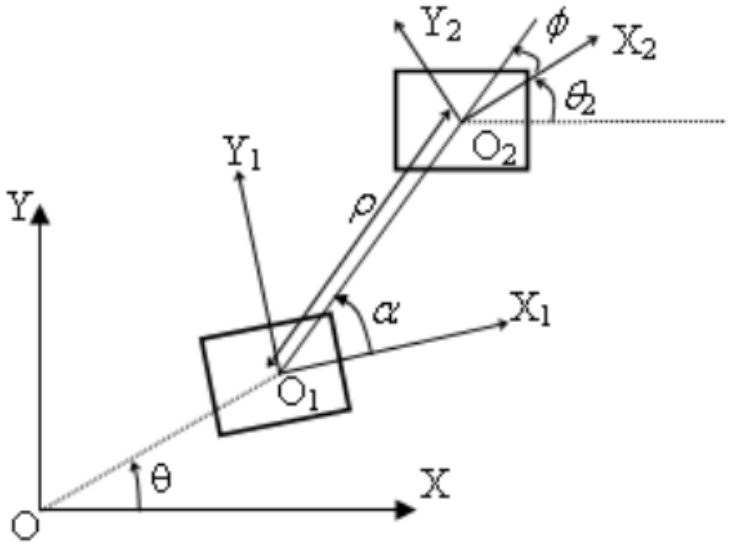

Figure 2.5: The robot poses and parameters [26]

time-invariant system $\dot{x}(t) = f(x(t))$, where $x(t) \in \mathbf{U} \subseteq \mathbb{R}^n$. $\mathbf{U}$ is an open subset containing the origin and belongs to the Euclidean n-space $\mathbb{R}^n$. Suppose an equilibrium point of this system exits, where $f(x_e) = 0$. The equilibrium is Lyapunov stable [50] if the solutions are within a distance from the equilibrium and remain close enough to that point. The definition of Lyapunov stability can be rewritten using mathematical expressions as: $\forall \varepsilon > 0 \, \exists \delta > 0 \ni if \ \|x(0) - x_e\| < \delta, \, then \, \forall t \geqslant 0 \, we \, have \ \|x(t) - x_e\| < \varepsilon$.

If the equilibrium point is Lyapunov stable and satisfies the following requirements, the equilibrium point can be considered as asymptotically stable: $\exists \delta > 0 \ni if \ \|x(0) - x_e\| < \delta, \, then \lim_{t \to \infty} \|x(t) - x_e\| = 0$. An asymptotically stabled system is exponentially stable if: $\exists \alpha > 0, \, \beta > 0, \, \delta > 0 \ni if \ \|x(0) - x_e\| < \delta, \, then \, \forall t \geqslant 0 : \|x(t) - x_e\| \leqslant \alpha \|x(0) - x_e\| e^{-\beta t}$.

To analyze the stability of the equilibrium point, the Lyapunov function [51, 52, 53, 54] is defined as a continuous scalar function $\mathrm{V} : \mathbf{U} \subseteq \mathbb{R}^n \to \mathbb{R}$. The following definitions should be made to find out a proper Lyapunov function:

Definition 2.1. A continuous function $\mathrm{V} : \mathbf{U} \subseteq \mathbb{R}^n \to \mathbb{R}$ is positive definite in $\mathbf{U}$ if :

$\mathrm{V}(0) = 0 \quad and \quad \forall x \in \mathbf{U} \cap x \neq 0 : \mathrm{V}(x) > 0.$

Definition 2.2. Locally positive definite function: A continuous function $\mathrm{V} : \mathbf{U} \subseteq \mathbb{R}^n \to \mathbb{R}$ is locally positive definite in $\mathbf{U}$ if: $\mathrm{V}(0) = 0 \quad and \quad \forall x \in \mathbf{U} \cap x \neq 0 : \mathrm{V}(x) > 0,$ in a neighborhood of the orig

With the definitions above, the basic Lyapunov theorems can be concluded as:

Theorem 2.1. *Globally asymptotically stable: if a continuous function* $\mathrm{V} : \mathbf{U} \subseteq \mathbb{R}^n \to \mathbb{R}$ *is positive definite and* $\mathrm{V}(x) \to \infty$ *as* $\|x\| \to \infty$*, if its derivative* $-\dot{V}$ *is positive definite, then the origin of the system is globally asymptotically stable.*

Theorem 2.2. *Locally asymptotically stable: if a continuous function* $\mathrm{V} : \mathbf{U} \subseteq \mathbb{R}^n \to \mathbb{R}$ *and* $-\dot{V}$ *are locally positive definite, then the origin of the system is locally asymptotically stable.*

2.2.2 Trajectory Optimization

The self-reconfiguration problem can be defined as a trajectory optimization problem as well. The purpose is to find the optimal trajectory for the robot module to reach the target position that satisfies the requirements of reconfiguration. The solutions to the trajectory optimization problems can be classified into indirect or direct method. A direct method will convert the trajectory optimization problem into a constrained parameter optimization problem directly. By contrast, an indirect method needs to construct the necessary and sufficient conditions for optimality analytically before discretizing the trajectory optimization problem. Shooting methods and collocation methods, known as transcription methods, are developed to convert a trajectory optimization problem into a parameter optimization problem. This literature review focuses on numerical trajectory optimization based on direct collocation methods. Generally, the trajectory optimization problem can be converted into a nonlinear program by discretizing the trajectory optimization problem. Two typical direct

collocation methods, known as trapezoidal collocation and Hermite-Simpson collocation, were studied and summarized in this section. Basically, the trajectory of a continuous-time problem is supposed to be discretized into a finite set of decision variables by representing the continuous states by their values at specific points in time, known as collocation points. The continuous system dynamics should be converted into a set of constraints and applied to the state and control at the collocation points using trapezoidal quadrature as:

$$\begin{aligned} \int_{t_k}^{t_{k+1}} \dot{x} dt &= \int_{t_k}^{t_{k+1}} f dt \\ x_{k+1} - x_k &= \frac{1}{2} h_k (f_{k+1} + f_k), \quad k \in 0, 1, \dots N-1, \end{aligned} \tag{2.4}$$

where $h_k = t_{k+1} - t_k$, x_k is the number of collocation points, $f_k = f(t_k, x_k, u_k)$ is a decision variable in the nonlinear program, $f_k = f(t_k, x_k, u_k)$ is the result of evaluating the system dynamics at each point. In addition to the collocation constraints that enforce the system dynamics, the boundary conditions, path constraints, and the limitations of the state and control should be enforced at specific collocation points. The Hermit-Simpson method also can be used to enforce the system dynamics at each collocation point as,

$$x_{k+1} - x_k = \frac{1}{6} h_k (f_k + 4 f_{k+\frac{1}{2}} + f_{k+1}), \quad k \in 1, 2, \dots N-1. \tag{2.5}$$

A second collocation equation is also required to enforce the dynamics at the midpoint of the $f_{k+\frac{1}{2}}$ as a function of the state $x_{k+\frac{1}{2}}$:

$$x_{k+\frac{1}{2}} = \frac{1}{2}(x_k + x_{k+1}) + \frac{h_k}{8}(f_k - f_{k+1}), \quad k \in 1, 2, \dots N-1. \tag{2.6}$$

Interpolation is required in order to get continuous trajectories after determining the values of the state and control trajectories at each collocation points by solving the nonlinear program.

In trapezoidal collocation, the control and state trajectories are approximated as piecewise linear functions, represented as:

$$\begin{aligned} u(t) &= u_k + \frac{\tau}{h_k}(u_{k+1} - u_k) \\ x(t) &= \int \dot{x}(t)d\tau \approx c + f_k\tau + \frac{\tau^2}{2h_k}(f_{k+1} - f_k), \end{aligned} \tag{2.7}$$

where $k \in 1, 2, \ldots N-1$, $\tau = t - t_k$, and $c = x_k$ if $t_0 = 0$. The Hermit-Simpson collocation method can be adapted to interpolate the solution between the collocation points to achieve higher-order accuracy:

$$\begin{aligned} u(t) &= \frac{2}{{h_k}^2}(\tau - \frac{h_k}{2})(\tau - h_k)u_k - \frac{4}{{h_k}^2}(\tau)(\tau - h_k)u_{k+\frac{1}{2}} + \frac{2}{{h_k}^2}(\tau)(\tau - \frac{h_k}{2})u_{k+1} \\ x(t) &= x_k + f_k(\frac{\tau}{h_k}) + \frac{1}{2}(-3f_k + 4f_{k+\frac{1}{2}} - f_{k+1})(\frac{\tau}{h_k})^2 + \frac{1}{3}(2f_k - 4f_{k+\frac{1}{2}} + 2f_{k+1})(\frac{\tau}{h_k})^3, \end{aligned} \tag{2.8}$$

where $k \in 1, 2, \ldots N-1$, $\tau = t - t_k$ and $c = x_k$ if $t_0 = 0$.

2.2.3 Trajectory Tracking

In the trajectory tracking problem, the robot should follow a trajectory from a starting point to the destination, as shown in Figure 2.6.

The simplest trajectory tracking control design is based on tangent linearization along the reference trajectory [55]. The tracking error can be defined as:

$$e_p = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}. \tag{2.9}$$

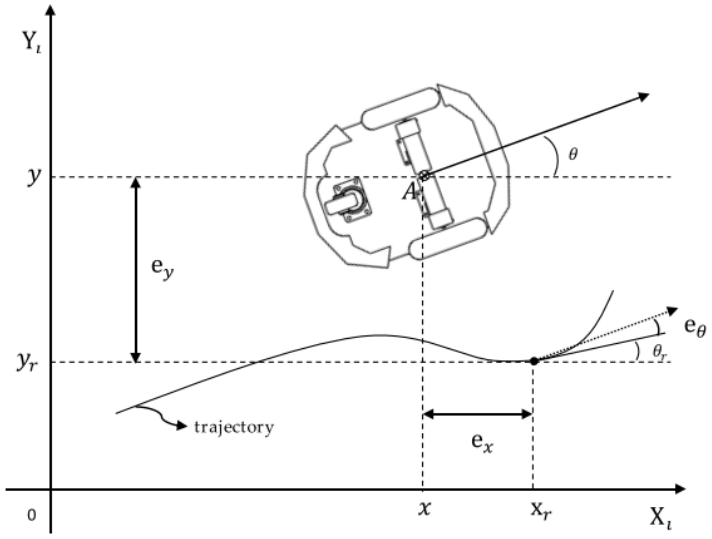

Figure 2.6: Trajectory tracking control [27]

A kinematic based control law [27] is developed and defined as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_r \cos e_\theta + K_x e_x \\ w_r + K_y v_r e_y + K_\theta v_r \sin e_\theta \end{bmatrix}, \tag{2.10}$$

where K_x,k_y and k_θ are positive gains of the backstepping controller. This problem also can be solved using some nonlinear methods, such as the control law in [55]:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_r \cos e_\theta + k_x(v_r, \omega_r) e_x \\ w_r + k_y v_r \frac{\sin e_\theta}{e_\theta} e_y + k_\theta(v_r, \omega_r) e_\theta \end{bmatrix}, \tag{2.11}$$

where the gian $k_x = k_\theta = 2\zeta\sqrt{{\omega_r}^2 + b{v_r}^2}$, $k_y = b$ with $b > 0$ and $\zeta \in (0, 1)$. These trajectory tracking controllers focus on position control regardless of the robot orientation. By contrast, Model Predictive Control (MPC) is an effective optimal control strategy that is capable of dealing with coupled Multi-input Multi-output (MIMO) systems with constraints. It has a multi-step look-ahead capability for the mobile robot to track the trajectory. Model predictive control, as the name implies, consists of the following three main parts:

1. Model: It can be a mathematic model to describe the target system or a data-based model obtained from training, such as a neural network.

2. Prediction: The model can be utilized to predict the future behavior of the target system, as well as generate controls according to the tracking errors.

3. Decision making (Control): An MPC-based controller will tell the system to take the optimal motions based on the predictions. Take an example from the real world, when a human being is crossing the road, he will predict whether the road is safe to pass according to his own walking speed and the traffic. After the prediction, one will pass the road at a proper speed.

Generally, optimal control needs to be optimized in the entire time domain in order to ensure optimality. Several solutions commonly used in optimal control include (1) variational method, (2) maximum principle, and (3) dynamic programming. Generally speaking, the variational method and the maximum principle can only deal with linear models without considering complicated constraints. Dynamic programming is an exhaustive method that is good at solving optimal control problems. However, the computational complexity is often high. Due to the excessive emphasis on optimality, optimal control exposes two problems: (1) it is difficult to solve nonlinear and complicated constraints; (2) The model of the system needs to be accurately known.

2.3 Problem Statement and Proposed Solution

Motivated by combining the advantages of a group of small robots and rigid-structure large robot, as well as responding to the Small Business Innovation Research (SBIR) and Small Business Technology Transfer (STTR) Opportunity Announcement HR001119S0035-23, un-

der the Defense Advanced Research Projects Agency (DARPA), related to investigating the feasibility of self-reconfigurable modular ground robots, this work aims to solve the following challenges:

1. Design of a group of small ground robots with reconfigurable capability.
2. Analyze the feasibility of the proposed robot modules and possible configurations.
3. Motion and dynamic analysis problem of the modular self-reconfigurable robotic system.
4. Motion planning and motion control of the robot module to achieve autonomous alignment in preparation for establishing the physical connection.
5. Trajectory optimization and tracking for improving the autonomous alignment process via taking the robot mechanisms and dynamics into consideration and looking for optimal solutions.

The proposed design, analysis and simulation are demonstrated in Chapter 3 and Chapter 4. The approaches to solve the control and optimization problems are presented in Chapter 5 and Chapter 6.

Chapter 3

MODULAR SELF-RECONFIGURABLE ROBOTS

This chapter describes a novel modular self-reconfigurable robotic system that offers greater versatility and functional advantages over traditional fixed structure robots and typical multiple mobile robot systems. This chapter also presents a detailed introduction to the robotic system, including the mechanical and electrical design overview.

3.1 Overview of the Proposed Modular Robotic System

The proposed robotic system is made up of three types of robotic modules: load carrier, manipulation module, and locomotion module. Each module is capable of navigation and interaction with the environment individually. In addition, the modules are capable of assembling autonomously into other configurations to perform complex tasks cooperatively. A non-backdriveable active docking mechanism with two Degrees of Freedom (DOF) was designed to fit inside the pulleys of the track units of the robot modules. The quantity

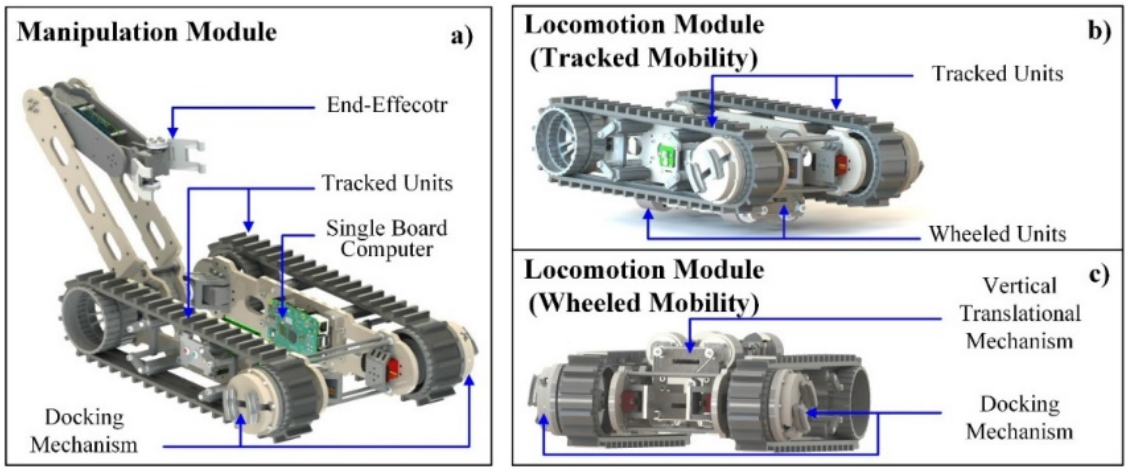

Figure 3.1: 3D models of the individual STORM modules: (a) The manipulation module, (b) The locomotion module in wheeled mobility mode, (c) The locomotion module in tracked mobility mode

and location of the docking mechanisms are customizable and selectable in order to satisfy various mission requirements and adapt to different environments.

3.1.1 Application I: Self-configurable and Transformable Omni-Directional Robotic Modules (STORM)

One application of the proposed modular self-reconfigurable robots, known as Self-configurable and Transformable Omni-Directional Robotic Modules (STORM), is shown in Figure 3.1.

STORM is composed of locomotion modules and manipulation modules. The manipulation module is a tracked mobile robot with an integrated manipulator arm and an end-effector capable of lifting heavy payloads. The arm can also be used as leverage to enhance mobility in rugged terrain and while traversing obstacles. The locomotion module was designed as a multi-directional hybrid mobility robot with tracked and wheeled locomotion modes that can be toggled via a Vertical Translational Mechanism (VTM). Figure 3.2 shows possible multi-robot configurations of STORM for performing different tasks. After docking, the

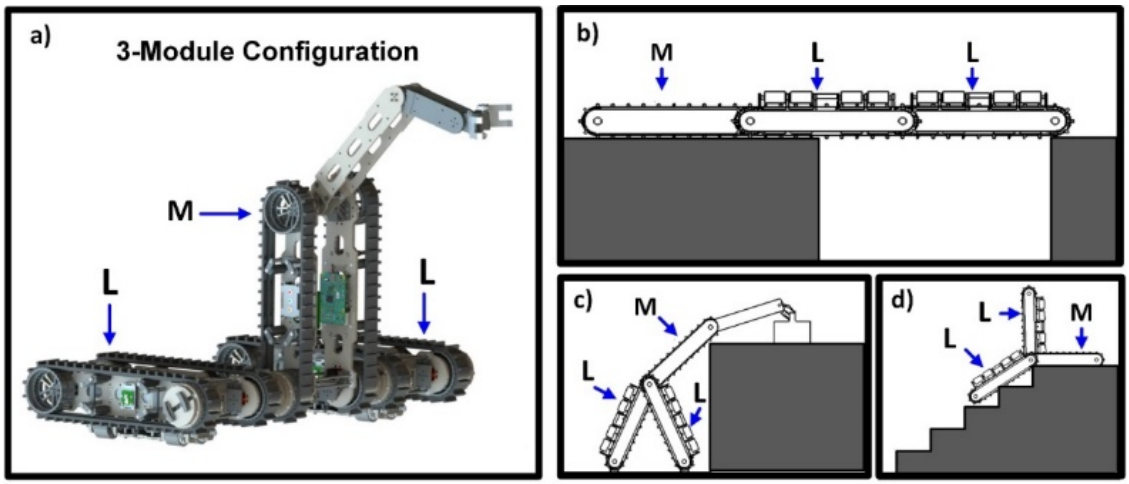

Figure 3.2: Possible multi-robot configurations of STORM for performing different tasks. M denotes the Manipulation module and L denotes the locomotion module. (a) Three-module configuration for carrying objects: (b) crossing wide ditches, (c) retrieving objects from high ledges, (d) stair climbing

modules can form a larger configuration capable of climbing stairs, crossing wide ditches, and retrieving objects from high ledges. The STORM architecture was proposed and presented in the previous work [56]. Sophisticated mechanisms [28, 57, 58] and autonomous control algorithms [59, 60, 61] were developed to enable the proposed reconfiguration capability, validate the feasibility, and improve the autonomy of the entire system.

To specify the functionalities and advantages of the proposed modular self-reconfigurable robotic system, the following presents one of the complex tasks assigned to the STORM system. As shown in Figure 3.3, the STORM modules are exploring a building to search for target objects. Each module should have the ability to detect the objects and determine if the target object is reachable or not. If the object is found and determined as unreachable, other modules should be deployed to the desired location and reconfigured into a large robot to augment the manipulation and grasping capabilities. Figure 3.4 shows the control flow diagram for each STORM module. A map generated simultaneously during the exploration process is fed into a multi-robot exploration algorithm [62] to determine the current goal

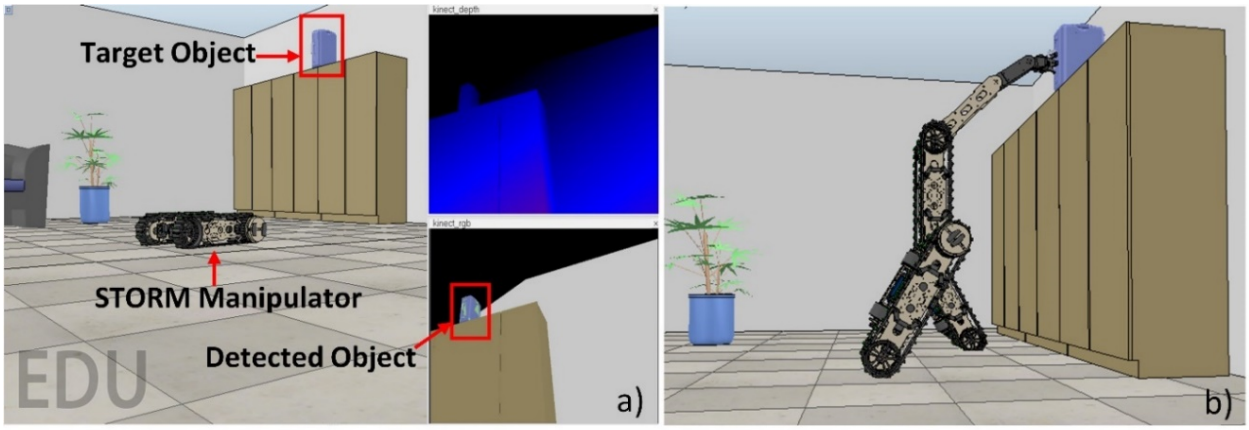

Figure 3.3: STORM modules in a 3D simulation environment. a) STORM Manipulator Model detected the suitcase (Target Object) and determined it was not reachable. b) Target Object is gripped by the 3-module configuration

for each module. The module should go to the current goal position with a path planner and an obstacle avoidance algorithm. If a target object is detected on the way, the module determines to grasp and transports the target object to the destination, sends out commands for reconfiguration and steers to the detected object, or keeps heading to the current goal position in light of the object reachability and the module's capability. After transporting the object to the destination, the module needs to return to the exploration task to search for other objects.

Sample Task Assigned to STORM: Goal-Orientated Exploration

An RRT-based goal-oriented exploration method was implemented in Gazebo [63] to explore an unknown environment and generate a map while going to the destination. This methodology was developed based on the ROS RRT_exploration package [64]. A cost function was defined to balance the exploration while heading to the goal. This algorithm was implemented and validated in a 3D world with a virtual locomotion module built in a robot simulator named CoppeliaSim [65]. The proposed navigation task was achieved through the

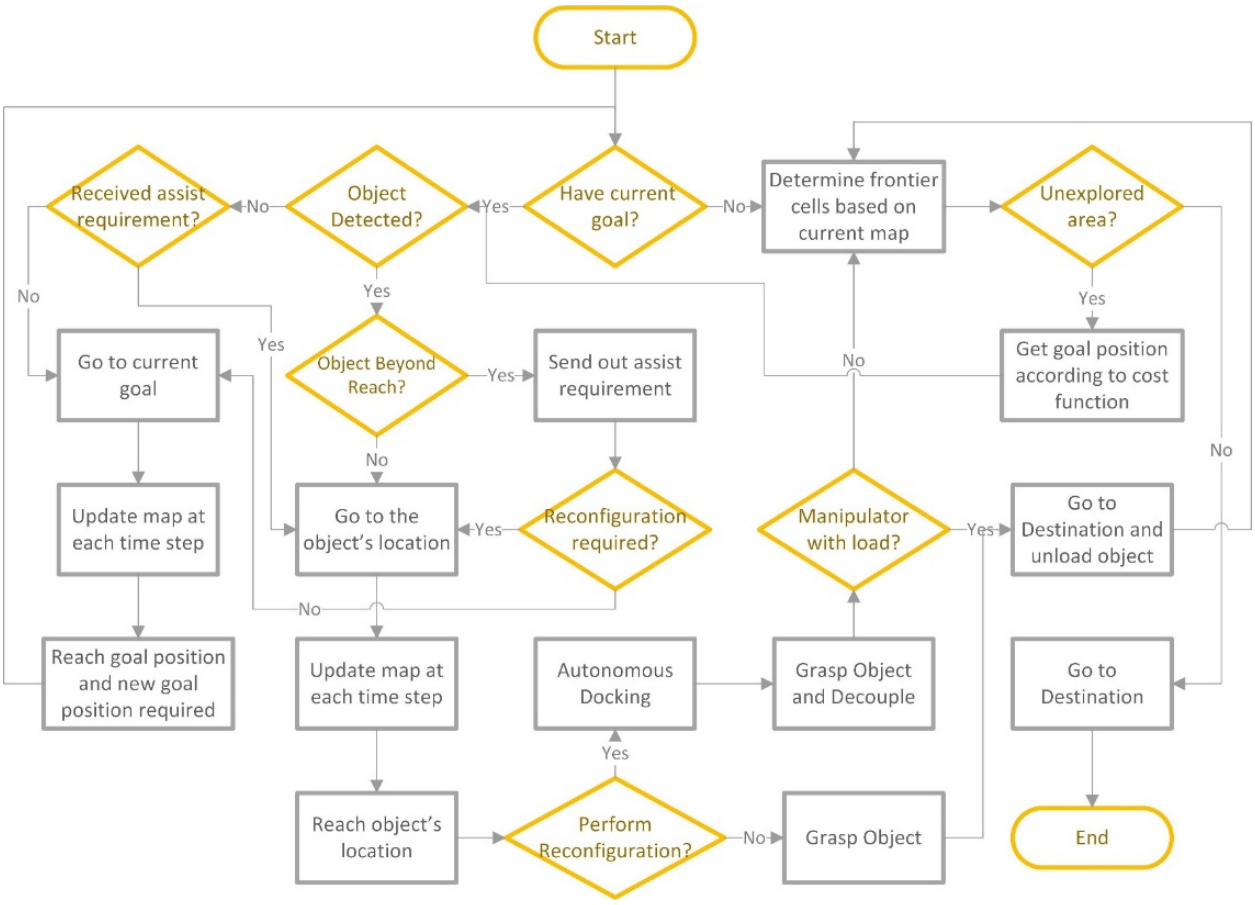

Figure 3.4: Control Flow Diagram for building exploration task assigned to the STORM system

following steps:

1. Collect and update the environmental information as an occupancy grid map at each time step.

2. Detect frontier points located on the boundary between the unexplored and known spaces of the current obtained map.

3. Set the temperate goal position to one of the frontier points or the destination by evaluating each point and the final goal position according to a properly defined cost function.

4. Maneuver the mobile robot to the current goal until the list of frontier points is updated.

5. Terminate the program when the robot arrives to the destination.

Several techniques were utilized to implement the proposed work. For example, the ROS gmapping package provided laser-based Simultaneous Localization and Mapping (SLAM) to generate the occupancy grid map. The frontier detection task was accomplished based on the RRT algorithm. The ROS move_base package played a critical role when maneuvering the mobile robot to the assigned goal. Detailed information about the related methodology is presented as follows:

The SLAM is generally defined as mapping an unknown environment and localizing the robot with the obtained sensory data simultaneously. Various SLAM methods were developed with different sensors and theories [66, 67, 68, 69, 70, 71, 72]. For example, the laser-based Gmapping algorithm estimates the robot pose $x_{1:t}$ and builds a grid map m that relies on particle filters by predicting a posterior probability $p(x_{1:t}, m|z_{1:t}, u_{1:t})$ using the obtained sensory data $z_{1:t}$ and the given controls $u_{1:t}$. Cartographer [69] is another laser-based SLAM system that builds the map of the environment by constructing the submaps representing different small chunks of the world. Each laser scan is inserted into a proper submap with a scan matching method and an estimated pose of that scan. In addition, the estimated poses of the scans and submaps are optimized via loop closure optimization. Except for laser-based SLAM, some SLAM methods use different types of cameras, such as monocular SLAM method: Direct Sparse Odometry (DSO) [72], and stereo SLAM method: Stereo Parallel Tracking and Mapping (S-PTAM) [71].

ROS Gmapping package was utilized to accomplish the mapping and localization subtasks for the navigation strategy implemented in a virtual environment with real physical properties.

The output occupancy grid map and the pose of the robot were feed to a frontier detector and a goal assigner to determine a temperate goal for the ROS move_base package to plan a path.

The occupancy grid map from the ROS Gmapping package can be expressed as: $p(M) = \prod p(m_k)$. The 2D-grid map m is stored as an array in row-major order. Each element in the array ranges from 0 to 100 to represent the probability of a map cell m_k being occupied. A probability value $p(m_k) = -1$ indicates that the corresponding location in the world has not been explored yet. In this case, the frontier points represent the boundary of the unexplored area and the known space on the map. It is critical to detect the frontier points fast and efficiently as a prerequisite to assign the temporary goal for the mobile robot.

An RRT-based algorithm was utilized to find all the frontier points. An RRT tree can be represented using a Graph $G = (V, E)$, where V is a set of vertices indicating the points in the map. E is the edge set that stores the tree branches connecting the vertices. During the exploration in the simulation. A global RRT tree was started at the point $X_{00} \subset V$, where $X_{ij} = \{x_i, y_j\}$ represent the coordinates of the Map cells m_k in the Map frame. At each time step, the global RRT tree was growing according to the following procedures:

1. Select a random point x_{rand} from the free space in the map M as a reference.
2. Find the nearest point $x_{nearest} \subset V$ to x_{rand} to decide the proper vertex and direction of growing the branch of the tree.
3. Choose a point x_{new} in the direction from the nearest point to the random point with the length equal to pre-defined step size.
4. Check the branch from $x_{nearest}$ to x_{new}: if the branch, including the point x_{new}, lies in the unknown region, x_{new} is a frontier point; If the new branch is located in the known region and no obstacles are in the path, add x_{new} to V.

5. Repeat the steps above to find a new vertex and the frontier point.

Simultaneously, a local RRT tree was generated from the mobile robot's current position, following the same procedures as the global RRT tree to improve the efficiency of the frontier detection process. The local RRT tree resets every time after a frontier point was detected.

To determine the temperate goal, a cost function was defined to balance the exploration of the environment and heading to the goal. The frontier with the highest value will be set as the current goal if that value is higher than a predefined constant threshold value. The cost of heading to the goal directly is related to the distance from the current position to the destination and the distances from the frontiers to the destination. If the cost of the goal satisfied the requirements such as higher than the Revenue, the current goal will be set as the temperate goal.

3.1.2 Application II: Self-driving Modular AI-based Robot for Rough Terrain (SMARRT)

In this work, another application of the modular, reconfigurable system is proposed: Self-driving Modular AI-based Robot for Rough Terrain (SMARRT). The proposed SMARRT system was developed to carry a payload and overcome many kinds of rough terrain, such as forests, sand, narrow paths and hills. The core features of the proposed robotic system are terrain classification and the ability to self-reconfigure. A perception algorithm was proposed to classify the terrain and determine the robot system configuration needed for the terrain ahead. This dissertation focuses on the mechanical design of the robotic system and the reconfiguration strategy.

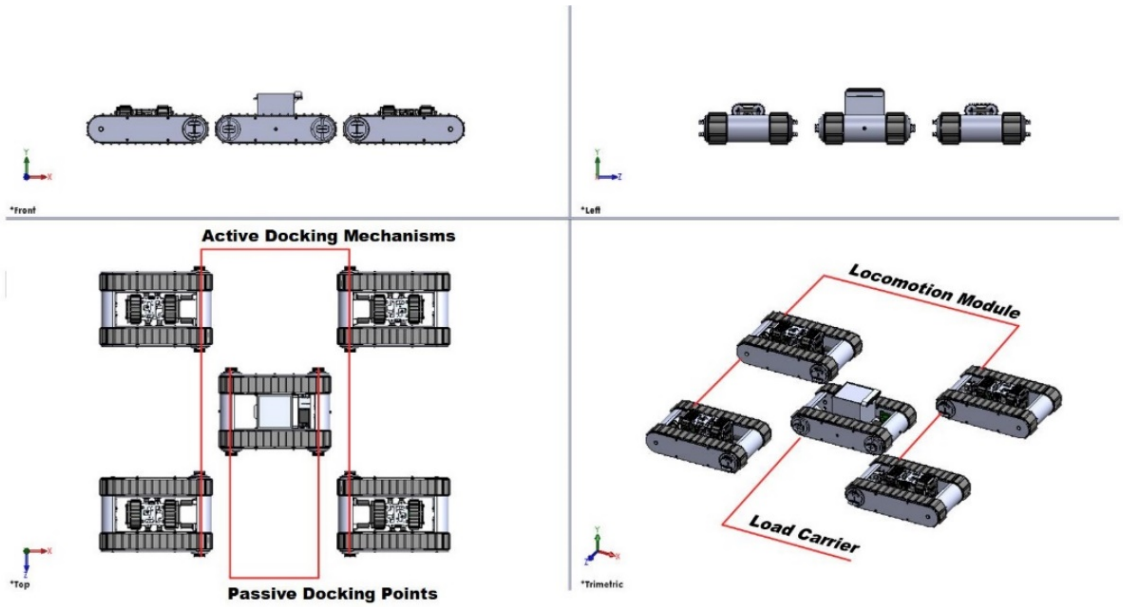

Figure 3.5: SMARRT robotic system-3D models of the robot modules

3.2 Mechanical Design of the SMARRT System

The objective of designing the SMARRT system is to deliver an important package to a remote location that might have several types of obstructions, such as ditches, rocks, mud, and narrow tunnels. A typical portable robot will not be able to navigate such terrains. It will either fall into the ditch or get stuck in between the rocks. The proposed SMARRT system solves some of these challanges by deploying several portable robotic modules. These robots can dock together to create a configuration that is suitable to navigate the environment ahead of the robot. The SMARRT robotic system consists of two types of robot modules: load carrier and locomotion module with longitudinal and lateral tracked units. Active and passive docking mechanisms were customized and fixed on the locomotion modules and the load carrier, respectively. 3D models of the robot modules are shown in Figure 3.5.

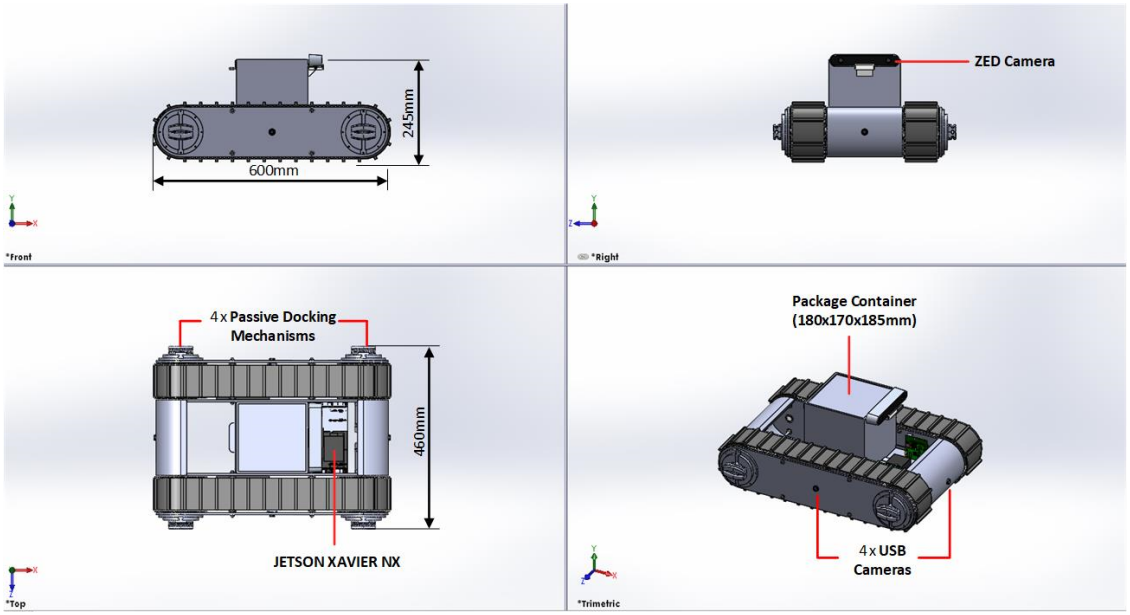

Figure 3.6: SMARRT-Mechanical design of load carrier

3.2.1 Load Carrier

The load carrier, shown in Figure 3.6, is able to carry a package with a volume of up to 300 cubic inches. A ZED camera [73] will be mounted on the package container around 24 cm above the ground to detect and classify the terrain. A JETSON XAVIER NX is attached to the load carrier as the supervisor of the robotic system. Four passive docking points and four USB cameras are located onboard to enable autonomous docking. Two track units were designed to propel the load carrier. Each track unit, as shown in Figure 3.7, consists of two parallel frames to support an asynchronous pulley set driven by a gear motor. The track is further supported by passive rollers to protect and isolate internal components from damage. Besides driving the entire robot, the track units work as the housing for motor drivers and battery packs.

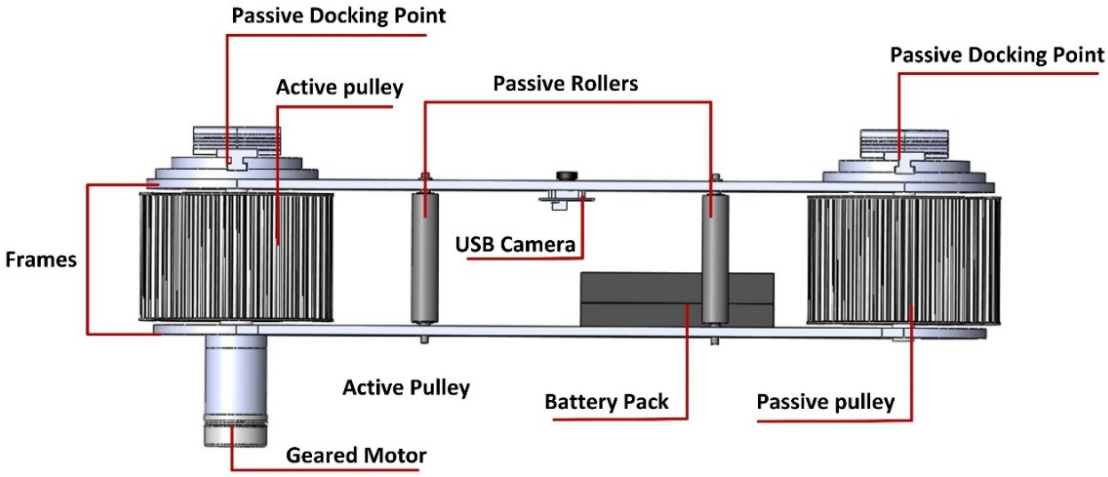

Figure 3.7: The tracked unit of the load carrier

3.2.2 Locomotion Module

Multi-directional mobility was taken into consideration when designing the mechanical system of the locomotion module to improve the spatial alignment capabilities, thereby improving autonomous docking capabilities. As such, the locomotion module consists of two longitudinal tracked units, two lateral tracked units, and a VTM that toggles between the two modes of locomotion. An alternate version of the units to provide the lateral motion is to replace the tracks with wheels. Figure 3.8 shows the 3D model of the locomotion module. As shown in Figure 3.9, the locomotion module consists of a VTM, a set of longitudinal track units, and a set of lateral track units. The VTM allows for both lifting of the longitudinal tracks and engagement/disengagement of the lateral track units, which are rigidly attached to it. The two longitudinal track units have active docking mechanisms built in to their passive pulley. The VTM, as shown in Figure 3.10 is able to switch the locomotion modes and adjust the height of the docking mechanism through the use of a pair of belt driven lead screws. The lead screws enable motion along the Z-axis and ensure equal driving forces are applied to the VTM from both sides to avoid unexpected roll of the VTM or jamming. The lead screw nuts are mounted to the inside frames of the two longitudinal tracks along

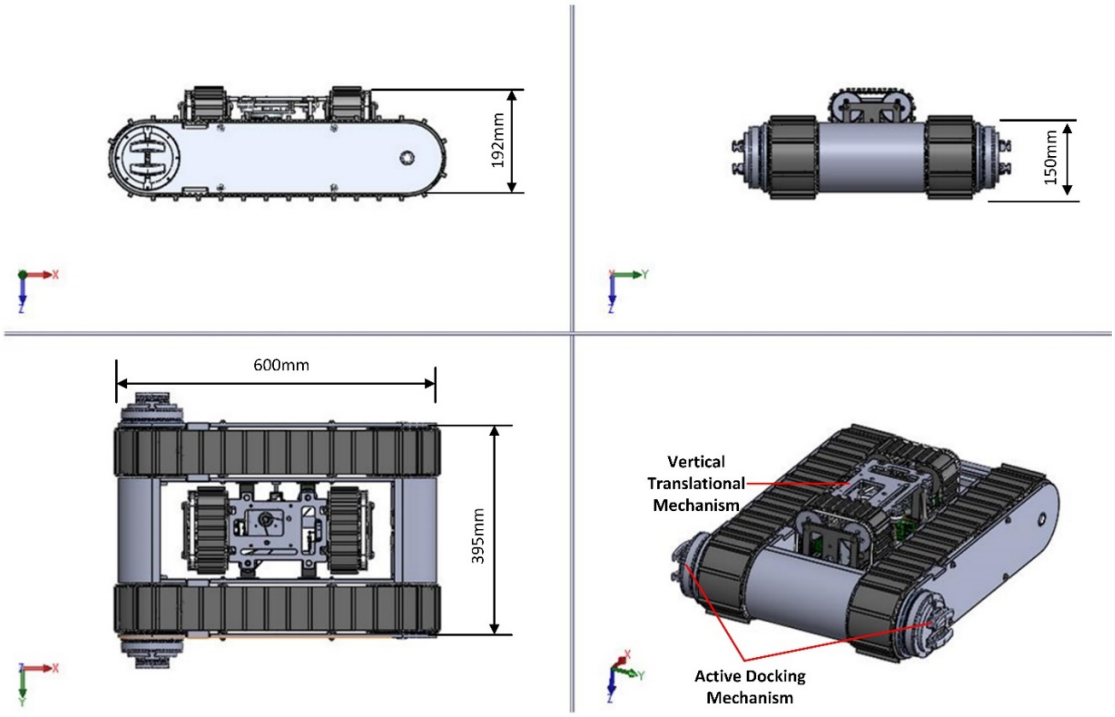

Figure 3.8: 3D model of the Locomotion module

with two linear bearings supporting the linear guide poles to guide the vertical translation. The relative position between the VTM and the longitudinal tracked units is measured by a linear potentiometer attached to the housing of the VTM. The mobility of the module can be switched between longitudinal, along the X-axis, to lateral mobility, along the Y-axis, by lifting the VTM and the attached lateral tracks above the ground. Alternatively, for lateral mobility, the lateral tracked units are deployed down to push against the ground and raise the longitudinal tracks. This allows lateral motion without turning that aims to reduce collisions between robot modules that could potentially be caused by heading changes that would otherwise be required while performing autonomous docking. The lateral tracked units consists of a DC motor and four pulleys, each located at the corners of the frame, as shown in Figure 3.11. A timing belt system is designed to drive the lateral tracks.

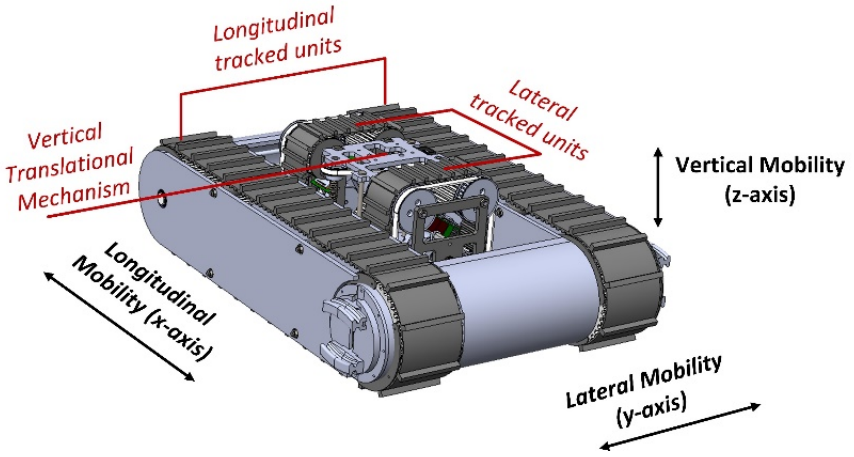

Figure 3.9: Locomotion module

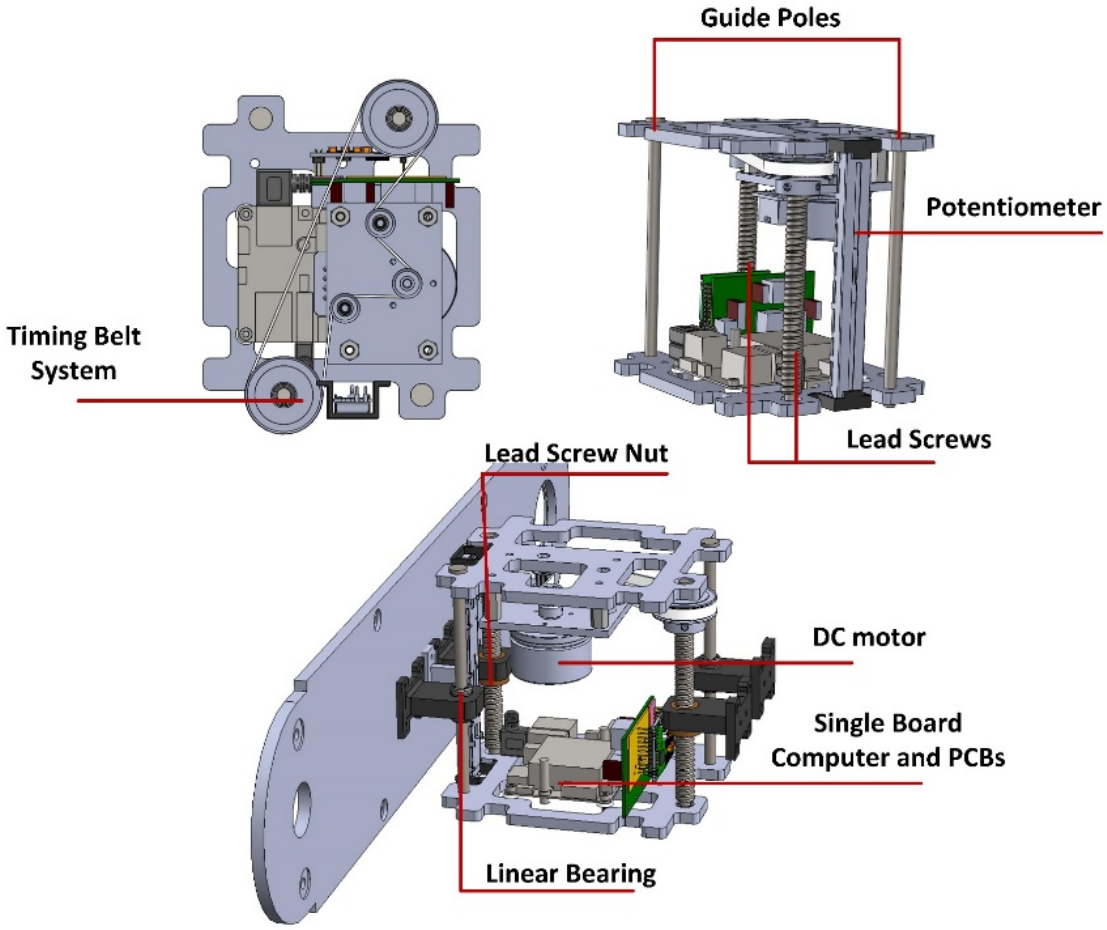

Figure 3.10: Vertical translational mechanism

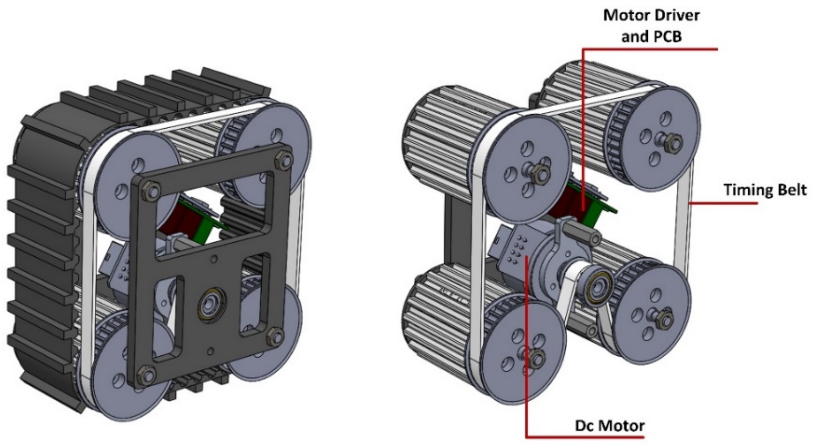

Figure 3.11: Lateral tracked unit

3.2.3 Active Docking Mechanism

The active docking mechanism equipped with two DOFs represents the key to mechanically enabling reconfigurability of the robotic system. Figure 3.12 demonstrates how a worm gear assembly actuates the relative rotation between the housing of the docking mechanism and stationary robot frame. The worm drive was chosen due to its high torque capability and non-backdriveable characteristic, which allows it to easily and safely rotate the locomotion module around the clamping axis of the load carrier when necessary for deploying various configurations. The symmetric translation of the clamping profiles relative to the rotating plate is implemented using a constant lead cam. The constant lead cam was selected for its compact design and non-backdriveable characteristic to prevent undesired decoupling. The clamping profiles act as followers when a high torque geared DC Motor rotates the cam located behind the sliding plate. Depending on the direction of rotation, the clamping profiles either move outward or inward to meet at the center. The clamping profiles are designed with H-grooved profiles to tolerate misalignments in six DOFs. The H-grooved profiles also have the advantage of being genderless and fail-safe if the docking is between two active docking mechanisms. That is to say that either active docking mechanism can

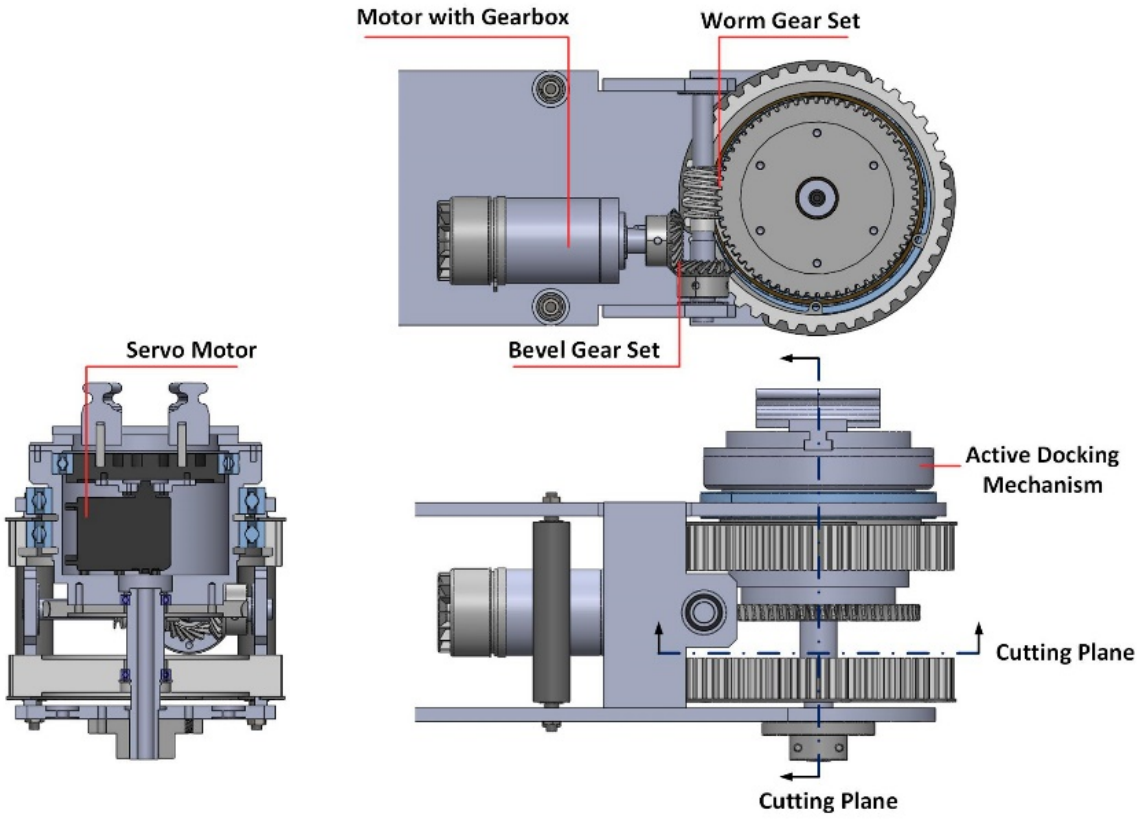

Figure 3.12: Lateral tracked unit

clamp over the other. The fail-safe feature is ensured by there being enough travel still in the mechanism after docking is achieved that the clamping profiles that the interior set can move closer together or the outer set can move further apart to release the two clamps from each other should one side fail.

The clamping profiles are designed with H-grooved profiles to tolerate misalignments in six DOFs. The H-grooved profiles also have the advantage of being genderless and fail-safe if the docking is between two active docking mechanisms. That is to say that either active docking mechanism can clamp over the other. The fail-safe feature is ensured by there being enough travel still in the mechanism after docking is achieved that the clamping profiles that the interior set can move closer together or the outer set can move further apart to release the two clamps from each other should one side fail. The ability of the clamps to

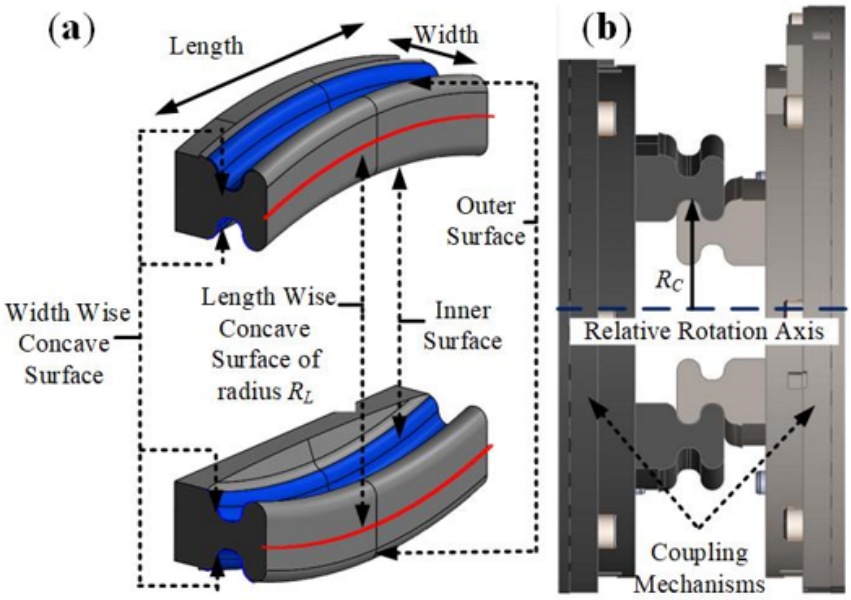

Figure 3.13: H-grooved clamping profiles (a) Isometric view of clamping profiles. (b) Side view of two docked coupling mechanisms [28]

tolerate misalignment in 6 DOF is the largest contributing factor of the mechanical design to enabling the autonomous docking and reconfiguration. The tolerance means there will be a much wider window of success for the path planning algorithm to work with and more forgiveness for misalignment that cannot be avoided due to terrain or other extraneous factors. The clamping profiles used on SMARRT are an updated version of those developed and discussed in [28]. Because only a few of the parameters were changed between the clamps developed previously and those being used in this paper, only the results of new simulations will be discussed here. To achieve the misalignment tolerance the clamping profiles were designed as special H-grooved profiles. Each clamping profile was designed with two widths and one length wise concave surfaces as shown in 3.13.

3.2.4 Analysis of the Docking Mechanism

The docking mechanism is designed to prevent unexpected motion between the modules and withstand shearing and bending forces to ensure robust operation. It was further analyzed, and the results are shown in this section.

Clamping Force

A cam-follower system is used to actuate the translational DOF to open and close the clamps. Based on the clamping force analysis conducted by Wael et. Al [28]. It is desirable to maximize $(1/2)F_N$ in order to maximize the allowable misalignments. This may be achieved by maximizing motor torque, reducing friction between the cam and followers, and minimizing the distance between the followers during docking while still enabling fail-safe operation. For the simulation of the active docking mechanism used on SMARRT, new clamping forces needed to be calculated as the cam profile and motors being used are different than those used by Wael et. Al. Figure 3.14 shows the clamping force vs. separation between the clamping faces when friction between the cam and followers is neglected for the SMARRT active docking mechanism. A clamp separation of 20mm allows for fail-safe coupling between modules. Based on this 120N was used as the applied force for each clamp as a reasonable compromise between the actual predicted clamping forces and values that allowed the simulation to function correctly.

Motion Study

Motion studies were conducted in SolidWorks to verify the docking mechanism's relative rotation and symmetric translation, as shown in Figure 3.15. The constant lead cam driven by a servo motor caused the symmetric linear translation of the H-grooved clamps. The

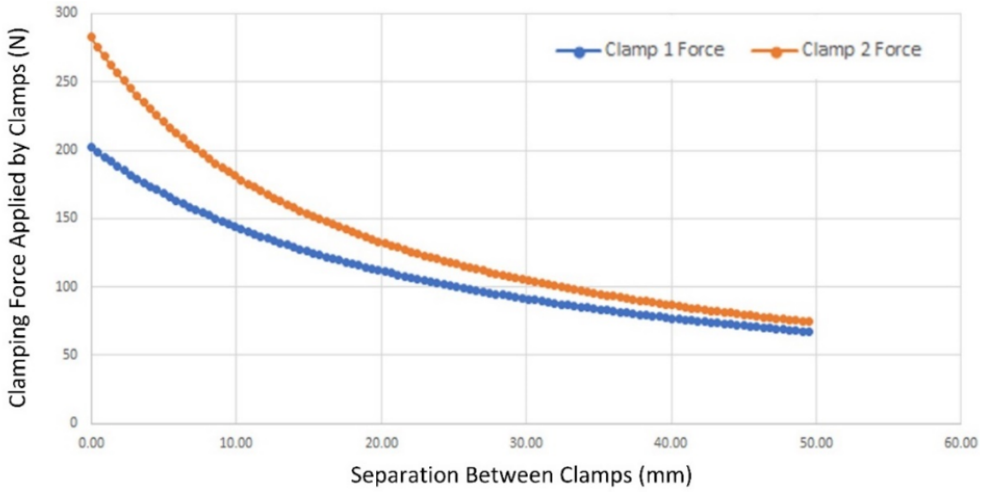

Figure 3.14: Clamping force vs. separation between the clamping faces

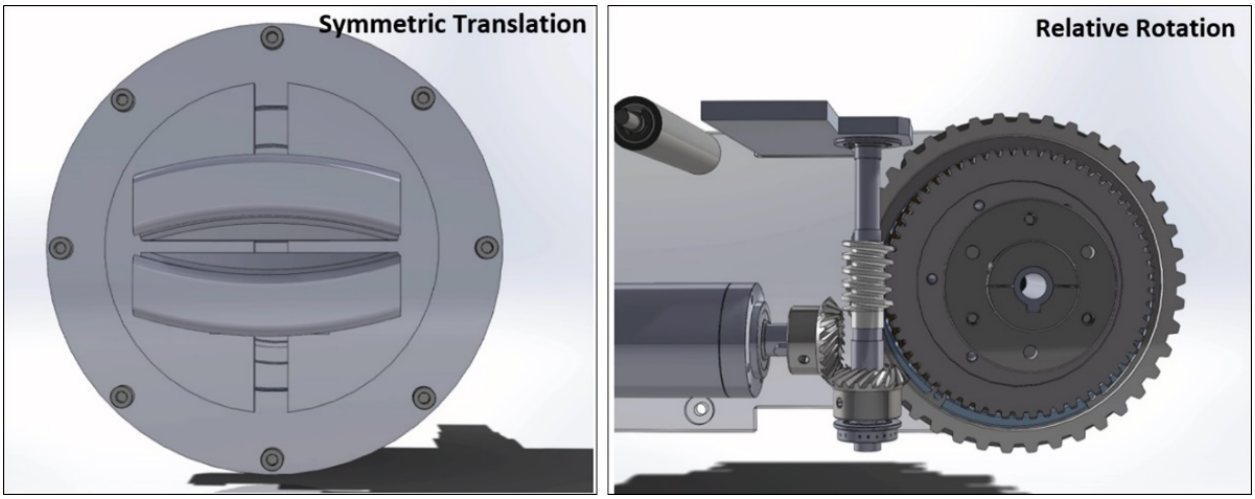

Figure 3.15: Symmetric translation and relative rotation of the docking mechanism

worm gear set derived the rotation of the docking mechanism with a gear ratio of 60:1. The worm gear is rigidly attached to coupler housing. Two bevel gears provided a 1:1, 90 deg transmission, which allowed a DC motor to be positioned lengthwise within the track structure.

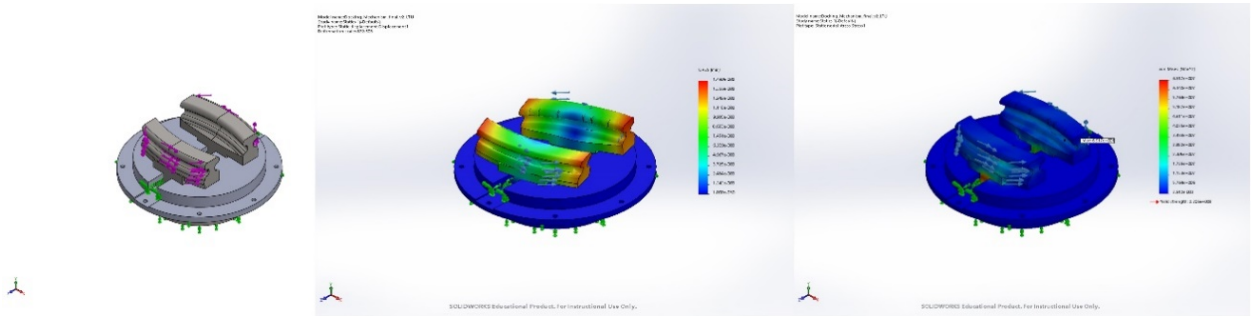

Figure 3.16: Static analysis of the docking mechanism

Static Analysis

A static analysis (Figure 3.16) was conducted on the docking mechanism to show that the clamps can withstand the torque applied from the other module. The analysis was performed under the assumption that the load would be directly applied to the clamping profiles. The material of the mechanism was also assumed to be plain carbon steel with a yield strength of approximately 220 MPa. For simplicity in the stress analysis, the torque applied was 75 Nm and evenly distributed across the outward faces of the H-grooved clamping profile. The fixed constraints were applied across the bottom of the sliding plate that houses the clamping profiles and the bottom of the constant lead cam. This assumption was assumed to be credible because these parts would restrict the loading torque. The maximum stress is 69.71 MPa with a factor of safety of 3.189, and the maximum displacement is equal to 0.0149 mm. Given these values and the strength and rigidity criteria for plain carbon steel, it can be concluded that the stresses induced within the section are within safe limits.

Simulation in Solidworks

A geometric analysis of the clamping mechanism was conducted in order to find the absolute best performance that could be expected of the module, and then several simulations were conducted in Solidworks. The Solidworks simulations help better understand what reason-

able expectations for real world performance are as well as give insight as to what factors impact that performance. In simulation, the clamping force was applied directly to an active set of clamps that interacted with a passive set of clamp profiles that were attached to a large box used as a simple approximation of a second robot. Simulations were conducted to evaluate misalignment due to a rotation about the z-axis, a translation along the x-axis, and a translation along the y-axis. For each of these misalignments, three different surface interface conditions between the module and ground as well as between the clamps were modelled: (1) No Friction, (2) Low Friction: $_s = 0.08$, $_k = 0.05$, (3) High Friction: $_s = 0.3$, $_k = 0.25$. For each interface condition three different module masses, 10kg, 15kg, and 20kg were also simulated. This design of experiments expands upon the previous work [28] that demonstrated how the shape of clamping profiles impacts the maximum allowable misalignment and now looks at how surface interface conditions and module mass impact real-world performance. The geometric analysis shows that the clamps should be able to tolerate a maximum translational misalignment of 5.05mm along X, 30mm along Y, and a yaw misalignment of 9.56°. The results of the different simulations are shown in Tables 3.1 and 3.2 below. However, the key takeaway from the simulations is that any increase in resistance to motion dramatically reduces the allowable misalignment. Translation along the X-axis is the least sensitive misalignment with only a small decrease in allowable values being seen when clamp-to-clamp friction is increased. Translation along the Y-axis is sensitive to all factors but is most heavily influenced by increases in clamp-to-clamp friction. Yaw is largely insensitive to changes in clamp-to-clamp friction unlike the other two directions of misalignment tested. However, yaw is very sensitive to increases in module-to-ground friction while not being greatly affected by changes to friction between the clamps. These results suggest that designs going forward should pay specific attention to the friction coefficients of the clamps. It must also be ensured that the motor driving the clamps has enough torque to overcome module to ground friction since this factor cannot be mitigated while still achieving other

Table 3.1: Simulation Summary for High Friction Clamps with Varying Module-to-Ground Friction

Directions & Conditions / Module Mass (kg)	Max Allowable Misalignment (mm)								
	x-axis			y-axis			z-axis		
	No Friction	*Low Friction*	*High Friction*	*No Friction*	*Low Friction*	*High Friction*	*No Friction*	*Low Friction*	*High Friction*
10	*4*	*3.75*	*3.75*	*3.75*	*3.25*	*3.5*	*8.5*	*6*	*3*
15	*3.75*	*3.75*	*3.75*	*3.75*	*2.75*	*2.75*	*8.5*	*5.25*	*1.75*
20	*4*	*4*	*3.75*	*2.25*	*0.25*	*0.25*	*8.5*	*4.5*	*1*

Table 3.2: Simulation Summary for No Friction Module-to-Ground Condition and Varying Clamp Frictions

Directions & Conditions / Module Mass (kg)	Max Allowable Misalignment (mm)					
	x-axis		y-axis		yaw	
	No Friction (Clamps)	*Low Friction (Clamps)*	*No Friction (Clamps)*	*Low Friction (Clamps)*	*No Friction (Clamps)*	*Low Friction (Clamps)*
10	*4.75*	*4.75*	*29*	*23*	*8*	*8.5*
15	*4.75*	*4.75*	*29*	*24*	*8.25*	*8.75*
20	*5*	*4.75*	*29*	*23*	*8.5*	*9*

design goals.

3.3 Overview of the SMARRT System

The electronic hardware architecture is presented in Figure 3.17. The components of each robot module are classified into three main subsystems: Sensing, control, and actuation. The control system mainly contains a single-board computer and a microcontroller in charge of sending commands, acquiring sensor data, and communicating with the other robots. The load carrier is the supervisor of the whole robotic system. Robot Operating System (ROS) is utilized to help the load carrier publish all high-level commands which the locomotion modules subscribe to. The sensing system of each module has an IMU and GPS receiver to determine the robot poses for localization. The load carrier has a ZED camera and four

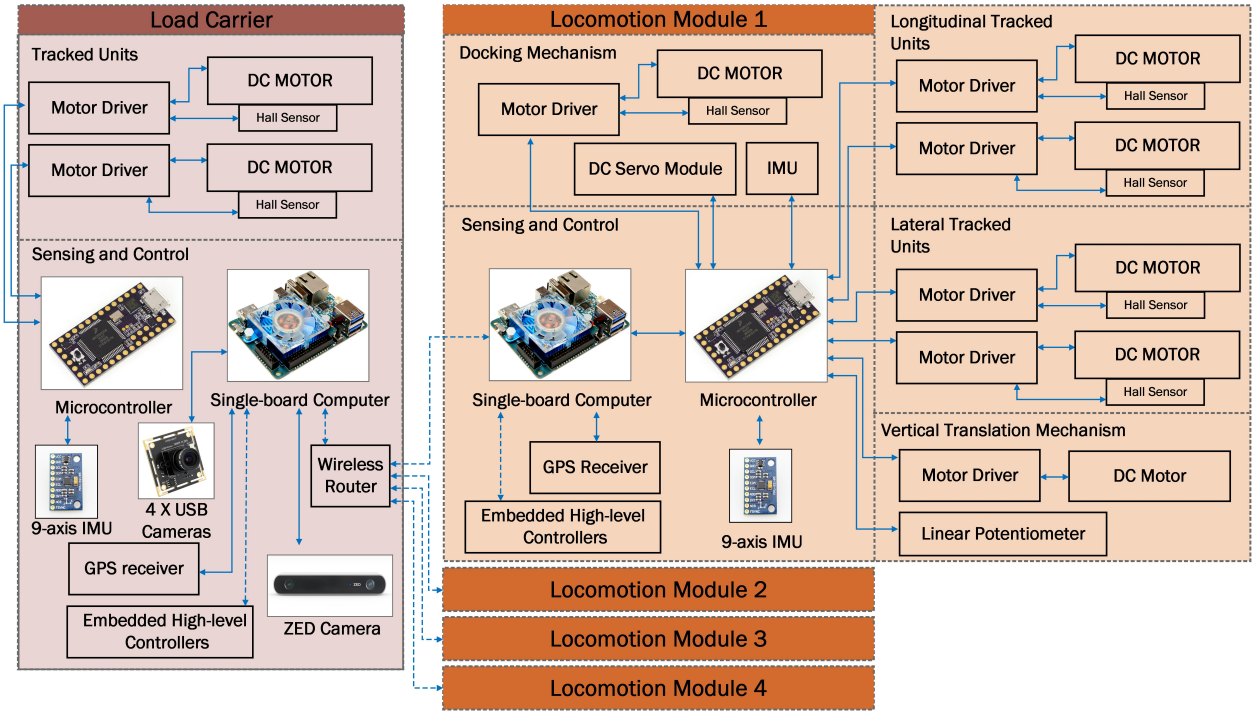

Figure 3.17: Electronic Schematic of SMARRT System

USB cameras for terrain classification and autonomous docking. For the actuation system, a high-torque DC motor is used to drive each of the belts. Another DC motor drives the VTM of the locomotion module to provide vertical mobility. A potentiometer is attached to monitor the position of the VTM and gives feedback to a PID controller for precise height control.

3.4 ROS-based Control Architecture

The overall ROS-based architecture for the control of the SMARRT system is presented in Figure 3.18. The onboard computer of the load carrier is running the ROS master, configuration planner, object tracking, and motion planner nodes. The object tracking node calculates the relative poses of all the locomotion modules relative to the load carrier according to the

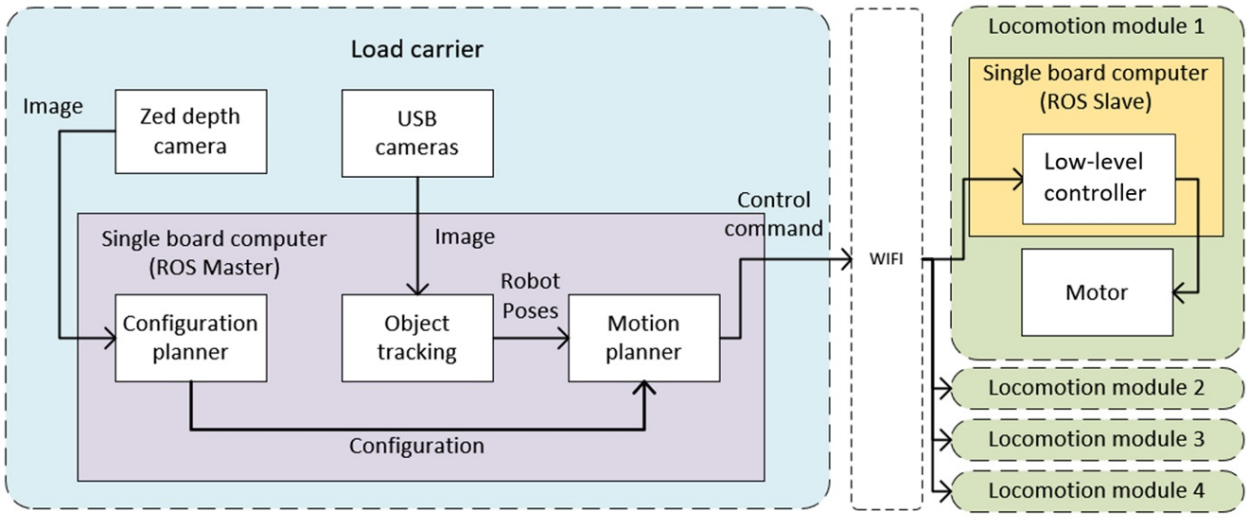

Figure 3.18: Autonomous docking architecture of SMARRT system

images provided by the surround-view camera. The configuration planner determines the required configuration pattern with the target passive docking points for the locomotion modules to align. The motion planner chooses the proper locomotion modules according to the target passive docking points from the configuration planner and the relative poses from the object tracking node to generate the velocity commands for the chosen locomotion modules. A low-level controller is designed and implemented on the single board computer of each locomotion module to subscribe to the commands from the load carrier and generate control signals to the DC motors.

3.5 Robot Kinematics of the SMARRT Modules

It is necessary to study the robot kinematics to determine the relationship between the velocities of the robot and the speed of each track to bridge the command from the high-level controllers to the low-level controllers. The load carrier has the same kinematics as the

locomotion module in the tracked mode, as described by the equation below:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{3.1}$$

where v and ω are the linear and angular velocity of the locomotion module, respectively. (x, y) is the current position of the robot with respect to the world coordinations. θ is the angle between the heading of the robot and the x-axis of the world frame. When the locomotion module is operated in the lateral movement mode, the heading of the robot has an additional $\frac{\pi}{2}$ with respect to the motions in the previous mode. The kinematic equations become:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta + \frac{\pi}{2}) & 0 \\ \sin(\theta + \frac{\pi}{2}) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{3.2}$$

A skid-steer robot model is considered to determine the velocities of the right and left tracks for the load carrier and the locomotion module in tracked mode. To avoid repetition, the locomotion module in the longitudinal tracked mode is taken as an example to present the kinematic relations. As shown in 3.19, the geometric center of the module is assumed to coincide with its center of gravity. The body coordinate frame $\{\mathrm{x}, \mathrm{y}\}$ is located at the geometric center of the locomotion module. Any planar movement of a rigid body can be regarded as a rotation about one point. The location of that point can be defined as the Instantaneous Center of Rotation (ICR). The locomotion module during a steady-state turning maneuver can be considered as a rigid body that is rotated about an ICR. The location of the ICR of the locomotion module relative to the ground is represented in the

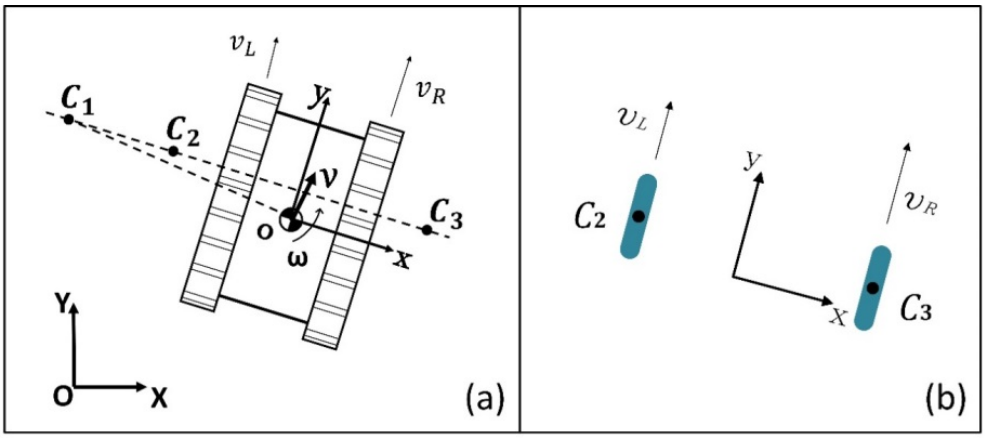

Figure 3.19: Schematic figures of the locomotion module: (a) in the longitudinal tracked mobility mode with ICR locations, (b) geometric equivalences between tacked and wheeled robot

local coordinate frame, and the x coordinate can be calculated as,

$$x_{c_1} = -\frac{v_y}{\omega}, \tag{3.3}$$

where v_y and ω are the velocity along the y axis of the local frame of the robot and the angular velocity of the locomotion module, respectively. According to Kennedy's Theorem, the three instantaneous centers shared by three rigid bodies in relative motion to one another all lie on the same straight line. The locomotion module and the two tracks can be considered as rigid bodies, respectively. The $C_2 = (x_{c2}, y_{c2})$ and $C_3 = (x_{c3}, y_{c3})$ are ICRs of the left and the right units, as shown in Figure 3.19. The three ICRs have the same y coordinates given by,

$$y_{c_1} = y_{c_2} = y_{c_3} = \frac{v_x}{\omega}. \tag{3.4}$$

The x coordinates of C_2 and $C3$ can be calculated as follows,

$$\begin{aligned} x_{c_2} &= \frac{v_L - v_y}{\omega} \\ x_{c_3} &= \frac{v_R - v_y}{\omega} \end{aligned}, \tag{3.5}$$

where v_L and v_R are the linear velocity of the left unit and the right unit with respect to the local frame. The direct kinematics can be derived from equations 3.3–3.5:

$$\begin{aligned} v_x &= \frac{v_R - v_L}{x_{c_3} - x_{c_2}} y_{c_1} \\ v_y &= \frac{v_R + v_L}{2} - \frac{v_R - v_L}{x_{c_3} - x_{c_2}} \frac{x_{c_2} + x_{c_3}}{2}. \\ \omega &= \frac{v_R - v_L}{x_{c_3} - x_{c_2}} \end{aligned} \tag{3.6}$$

The inverse kinematics for the locomotion module can be represented as follows,

$$(v_L, v_R) = f(v, \omega), \tag{3.7}$$

where $v = (v_x, v_y)$ is the linear velocity of the robot, v_x and v_y are the components of v along the X and Y axes, respectively, ω is the angular velocity of the robot, v_L and v_R are the linear velocity of the left and right tracked or wheeled mobility unit with respect to the local frame. Based on the calculated ICRs, the linear velocities of the robot can be calculated as follows,

$$\begin{aligned} v_L &= -\left|x_{c2}\right| \omega + v_y \\ v_R &= \left|x_{c3}\right| \omega + v_y \end{aligned} \tag{3.8}$$

In the above model, the local x coordinates x_{c2} and x_{c3} were estimated via experiments in CoppeliaSim simulation. Known values of v_L and v_R were given to the locomotion robot model in simulation, and the resulting ω and v_y were acquired using the onboard IMU. The value of x_{c2} and x_{c3} were determined by curve fitting of the recorded data. The estimated values are as follows: in the longitudinal tracked mobility mode, $|x_{c2}| = 0.212$ m, and $|x_{c3}| = 0.212$ m, in the lateral tracked mobility mode, $|x_{c2}| = 0.192$ m, and $|x_{c3}| = 0.192$ m. It should be noted that this kinematic model can be considered as an equivalence between the tacked and ideal wheel robots, where the ideal wheels are located at the ICRs of the

tracks, as shown in Figure 3.19(b). The track ICRs are always located outside the track centerlines due to slippage. The center of mass of the robot significantly affects the track ICRs. If it is closer to one side, then that side's track will slip less on the account of contact pressure, and its ICR will be closer to the robot. In this work, the geometric center of the module is assumed to coincide with its center of gravity. As a result, the values of the x coordinates of the left and right track's ICRs are the same. In conclusion, the linear velocity v and angular velocity ω of the robot modules of the proposed modular self-reconfigurable robotic system can be converted to the speed of each tracked or wheeled unit according to the following equation:

$$\begin{aligned} v_L &= -d_1\omega + v \\ v_R &= d_2\omega + v \\ \omega &= \frac{v_R - v_L}{d_2 + d_1} \\ v &= \frac{d_1 v_R + d_2 v_L}{d_1 + d_2} \end{aligned} \quad , \tag{3.9}$$

where d_1, d_2 are the distances from the geometric center of the robot to the location of the equivalent ideal wheels.

Chapter 4

SIMULATION ANALYSIS OF THE SMARRT SYSTEM USING PHYSICS ENGINE

To demonstrate and test the overall functionality and feasibility of the SMARRT system, simulations in a robot simulator named CoppeliaSim were developed as presented in this chapter.

4.1 Simulation Model of Load Carrier

Figure 4.1 shows the simulation model in CoppeliaSim. The simulation of chain-like tracks was computationally intensive and unstable for complex dynamic elements. Therefore, a 5-wheel approximation is used for the simulation since the autonomous docking process is assumed to be taking place on a flat surface. The model uses the skid-steer wheel control and imitates the skid-steer behavior correctly. It can respond to the velocity command from ROS. There are four cameras set up around the load carrier, and the field of view (FOV) of each camera is 135 degrees. The wide range of view allows other nodes in ROS to determine the position and orientation of other locomotion modules and thus lead to autonomous docking. Figure 4.2 demonstrates the ROS topic published and subscribed by the load carrier model.

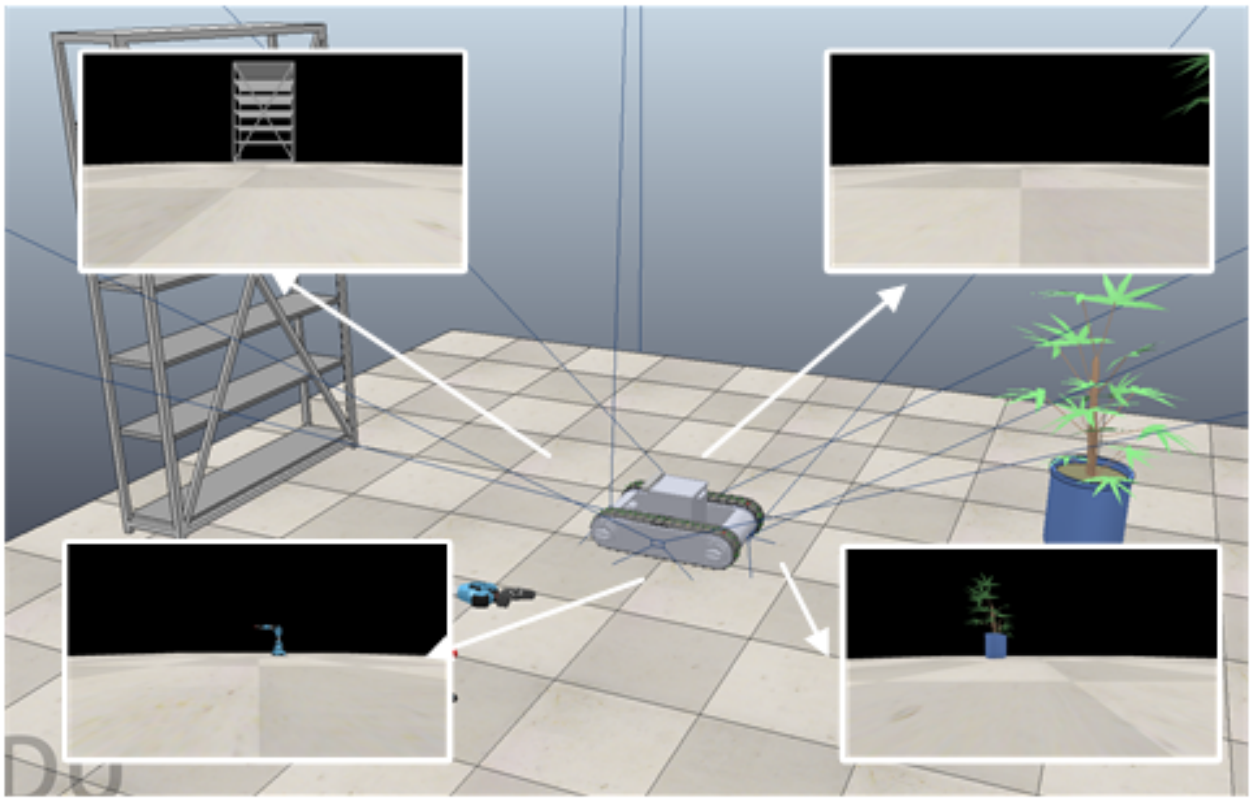

Figure 4.1: CoppeliaSim model of the load carrier

4.2 Simulation Model of Locomotion Module

Figure 4.3 shows the locomotion module in CoppeliaSim. Due to the same concern related to computing power, the locomotion module uses the 6-wheels approximation for the track and 2-wheels approximation for the vertical translational mechanism. Figure 4.4 shows the ROS topics related to the locomotion module.

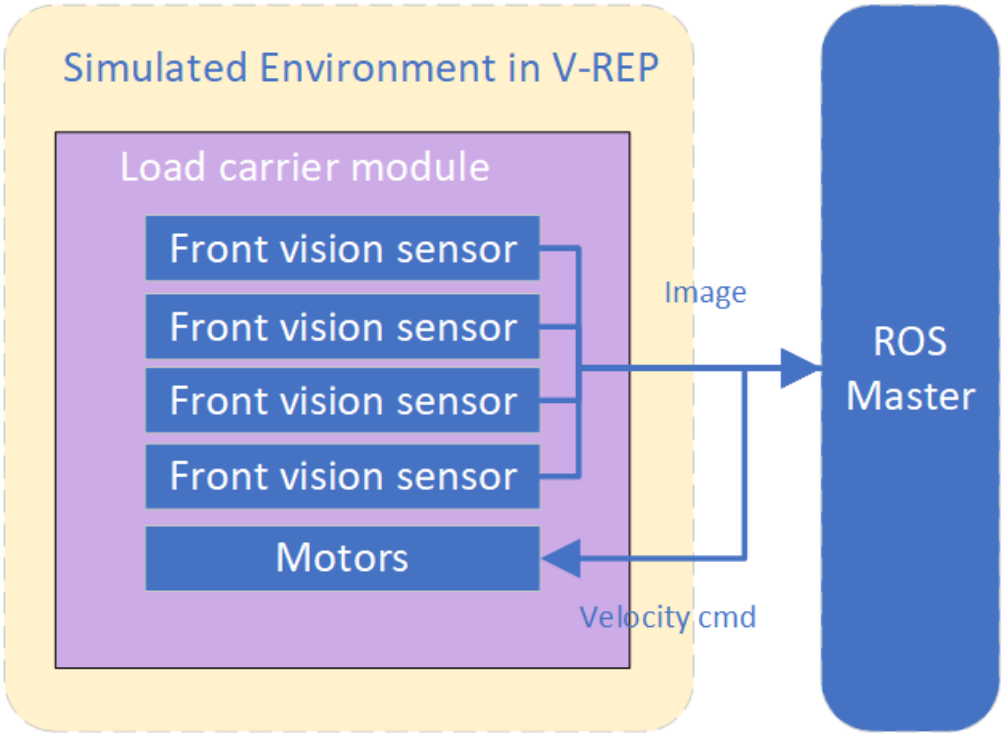

Figure 4.2: CoppeliaSim model of the load carrier and ROS topic

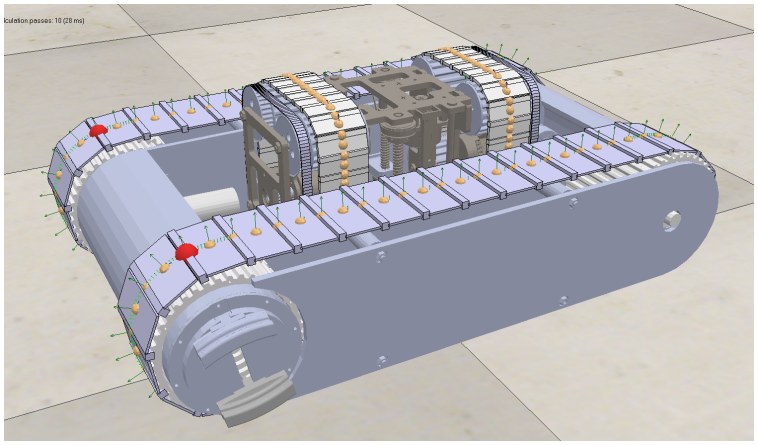

Figure 4.3: CoppeliaSim model of the locomotion module

4.2.1 Simulation Model of Docking Mechanism

The active docking mechanism is modeled in CoppeliaSim, as shown in Figure 4.5. The model is able to interact with ROS Melodic in Ubuntu 18. The pose of each part of the docking mechanism and simulation time are published out to track the motion of the clamps. The

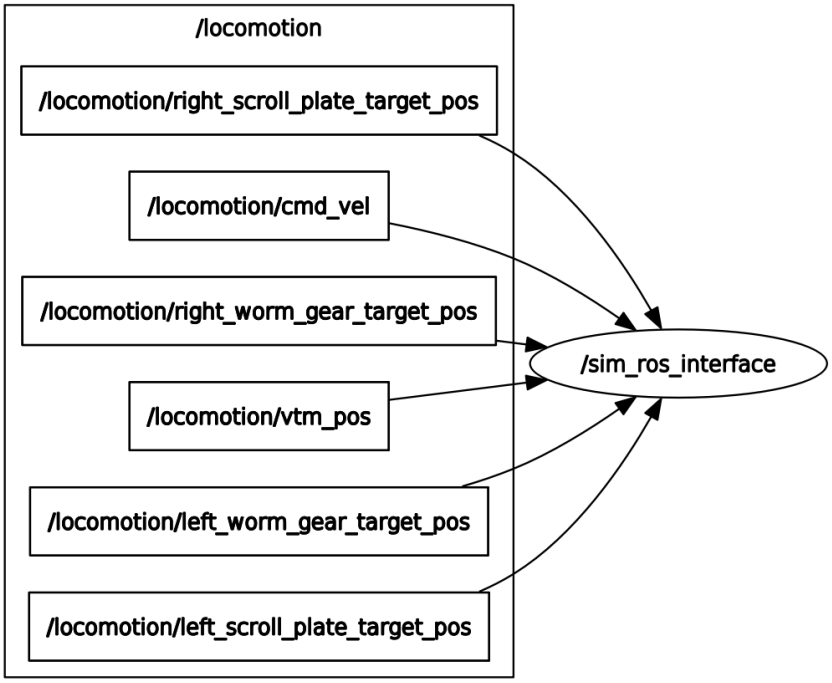

Figure 4.4: CoppeliaSim model of the locomotion module and ROS topic

servo motor attached to the scroll plate subscribes to the topic */scroll_plate_target_pos* and allows the input from 0 to 720 degrees corresponding to the translational motion of the clamps. The DC motor with worm gear subscribes to */worm_gear_target_pos* and allows the input from 0 to 360 degrees. This is the preliminary model of the docking mechanism, and further development is required to analyze the motion and force/torque.

4.3 Simulation of the Environment

The test environment in the CoppeliaSim simulator was exported from a Unity scene with muddy terrain, ditch, rocky terrain, high hill, and a narrow corridor. A load carrier and two locomotion modules were simulated to test the performances of the robot modules in different configurations. Figure 4.6 demonstrates the environment in CoppeliaSim with different terrain types and the navigation plan, including the starting points and the positions

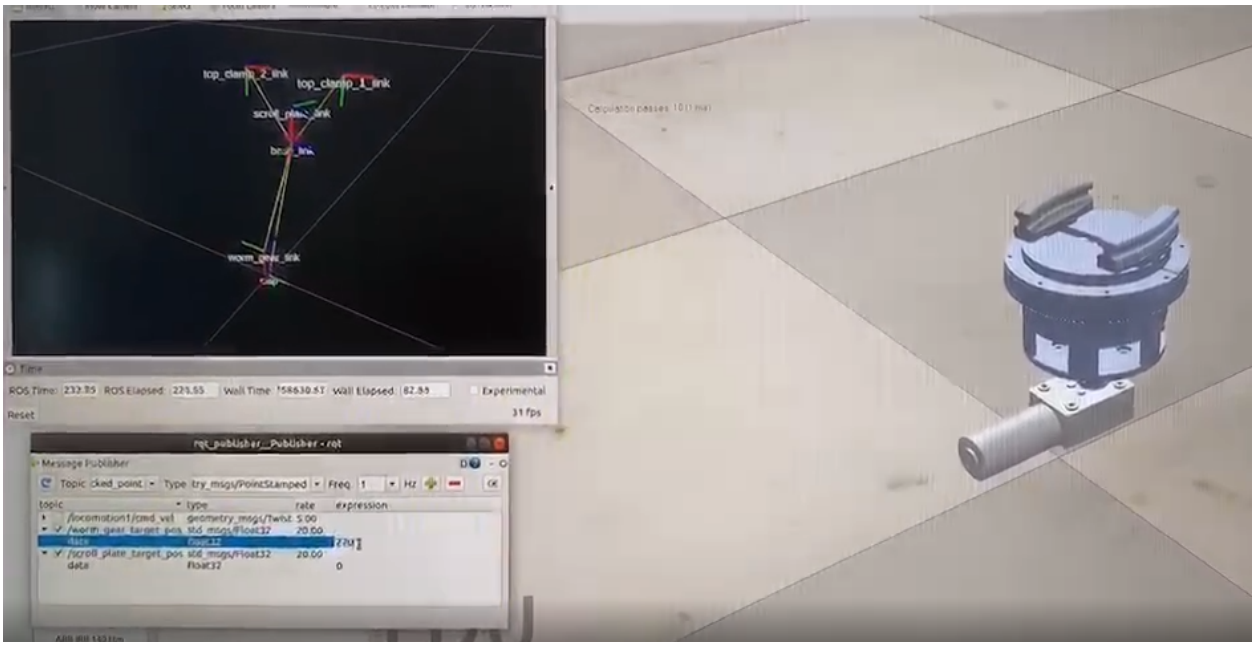

Figure 4.5: CoppeliaSim model of the locomotion module

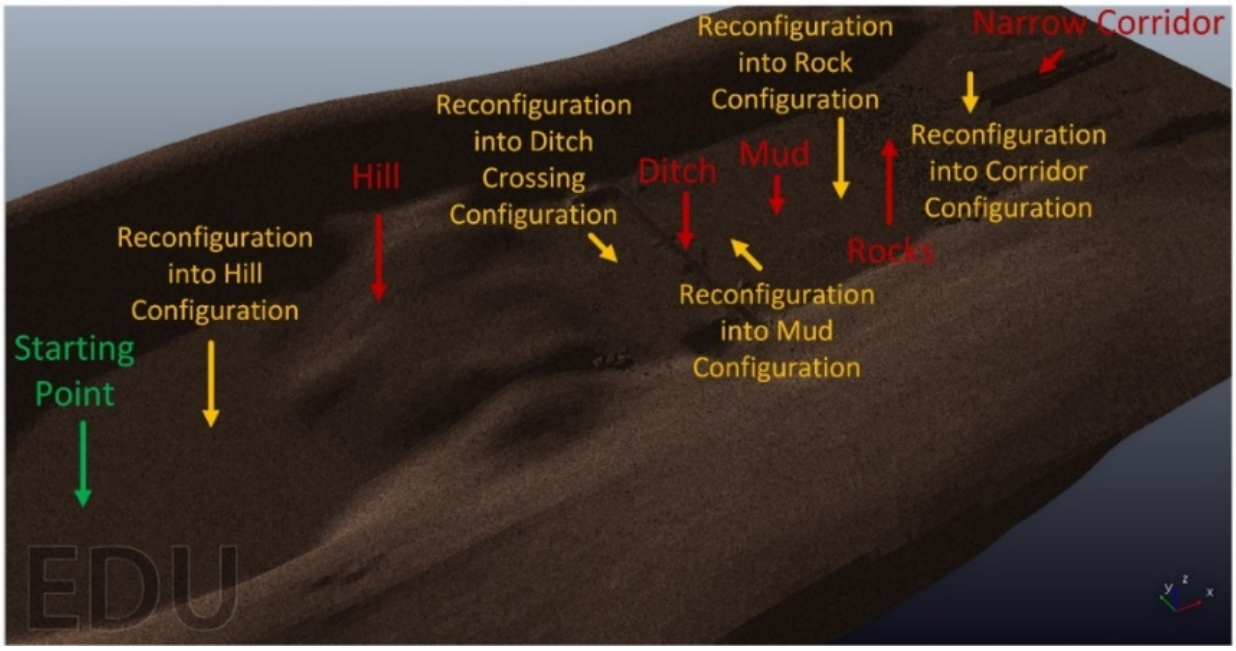

Figure 4.6: Environment in CoppeliaSim and navigation plan

to reconfigure into required configurations. Currently, the robot modules in the simulation were controlled by the user. In the tests, the three robot models were rearranged into several

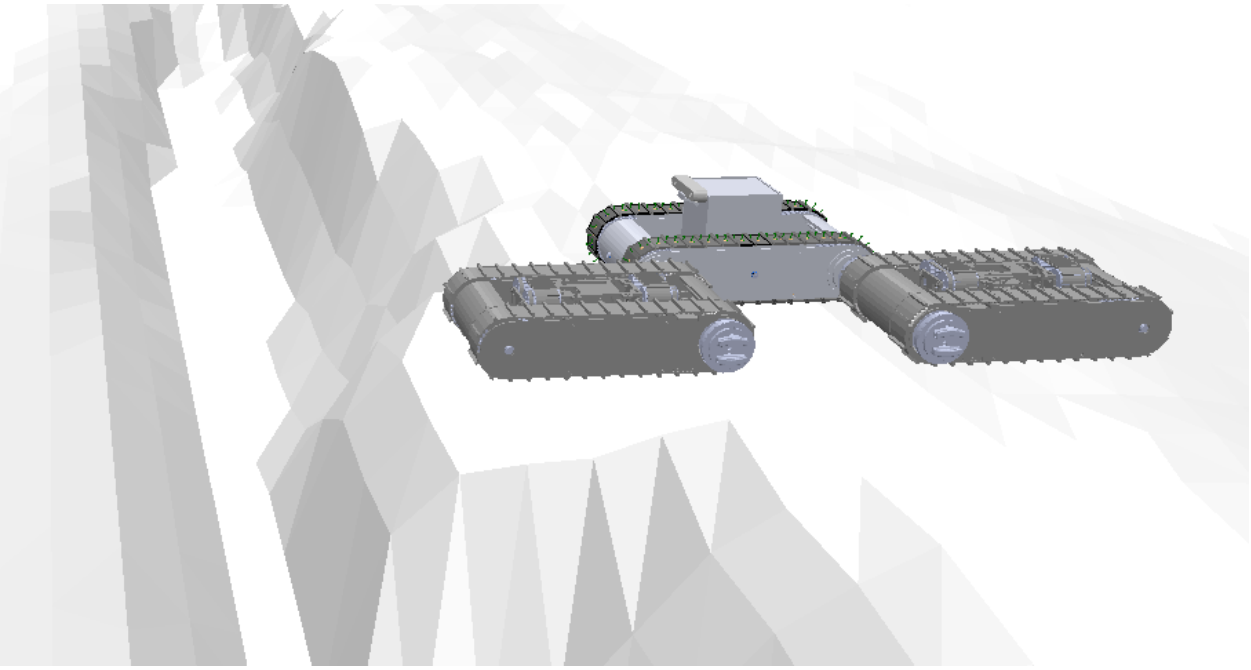

Figure 4.7: SMARRT robot system in a 3-module configuration to traverse a ditch

different configuration types, for example, the SMARRT robot system should reconfigure into a 3-module configuration to cross a ditch, as shown in Figure 4.7. The large configuration is simulated in CoppeliaSim, and the docking mechanism between the load carrier and the locomotion was simplified to a rigid connection with a force sensor to reduce the simulation time. In the future, a threshold will be set to the force sensor, such that the link between the docking mechanisms will break when it exceeds the physical limit in order to test whether the connection is reliable. A brief analysis was performed in order to find the maximum width of the ditch. As shown in Figure 4.8, the large configuration's Center of Gravity (COG) and the front or rear wheel shouldn't be in the ditch to avoid tipping into the ditch. Thus, the maximum width of the ditch can be determined as,

$$L_max = min\ (L_f, L_r), \tag{4.1}$$

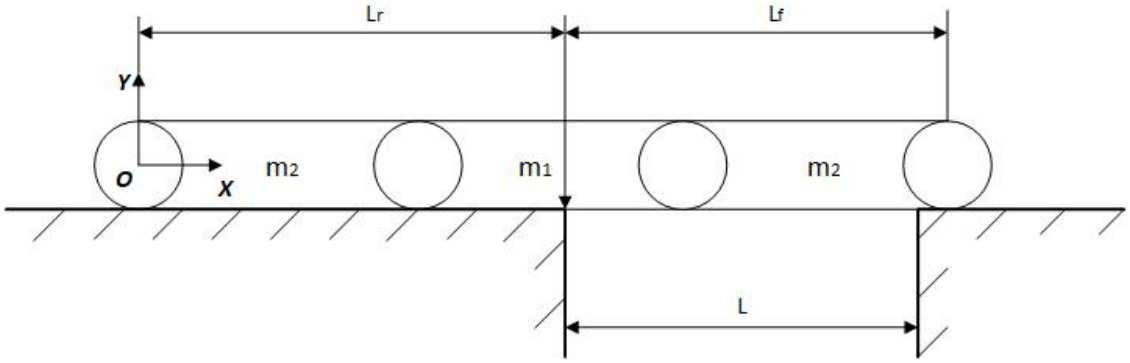

Figure 4.8: Diagrams to show the 3-module configuration crossing over ditches (front view)

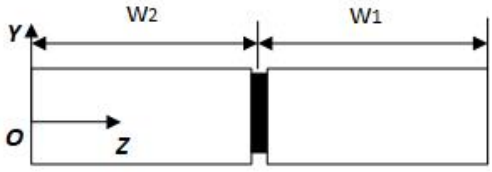

Figure 4.9: Side view of the 3-module configuration

where L_f and L_r are the distances from the COG position to the front wheel and rear wheel, respectively. In this case, $L_f = L_r = 69$ cm. This configuration is not symmetric. The lateral rollover needs to be taken into consideration. To ensure that the robot does not roll over laterally, the COG position in the lateral direction should be smaller than w_2, as shown in Figure 4.9. The configuration is not symmetric; therefore, the robot will roll to the side because of the weight of the load carrier. In the simulation, it was successfully demonstrated that the SMARRT robot system can traverse a 65cm ditch with the 3-module configuration. According to tests in the CoppeliaSim simulator, the weight of the load carrier can be up to 18kg when the weight of the locomotion module is set to 10kg. The large configuration will flip over if the load carrier is too heavy.

Preliminary tests of other large configurations on various terrains were done. The results showed that the proposed robotic modules were able to navigate on these terrains with different configurations. Further tests will be conducted to analyze the performance and limitations of the robots. Figure 4.10 shows the other configurations for various terrain

types. The robots reconfigured into a diagonal line to pass over the rocky terrain (Figure 4.10a). This is to maximize the balance and avoid unexpected situations such as rollover or getting stuck on top of a large rock. To cross a ditch, the robot module formed a crab-like shape to maximize the length. This large configuration's COG should fall on the side of the two locomotion modules to avoid rollover (Figure 4.10b). The muddy terrain was simulated with a small coefficient of friction. A closed-form configuration, as shown in Figure 4.10c, was selected to pass this area to distribute the traction and avoid slipping. A triangular configuration was used to climb up and down the hill for better stability and traction (Figure 4.10d). The large configuration decoupled and headed to the narrow corridor, as shown in (Figure 4.10e). A local path planner [59] was developed to improve the autonomy level and to avoid collisions.

4.4 Simulation of the Docking Process

The docking process was simulated in CoppeliaSim. All the robot modules were initialized within a short distance from one another. The mechanical structure of the locomotion module allowed lateral motion without turning. The first step was to minimize the error in y and γ. The locomotion module should operate in the longitudinal tracked mode during this process, as shown in Figure 4.11. The active docking mechanism was aligned with the passive docking mechanism in y and γ (Figure 4.11b). Then, the locomotion module should change to lateral tracked mode to activate the lateral motion (Figure 4.11c). After minimizing the error in the x-direction, the locomotion module should change back to the longitudinal tracked mode and minimize the error in the z-direction at the same time (Figure 4.11d).

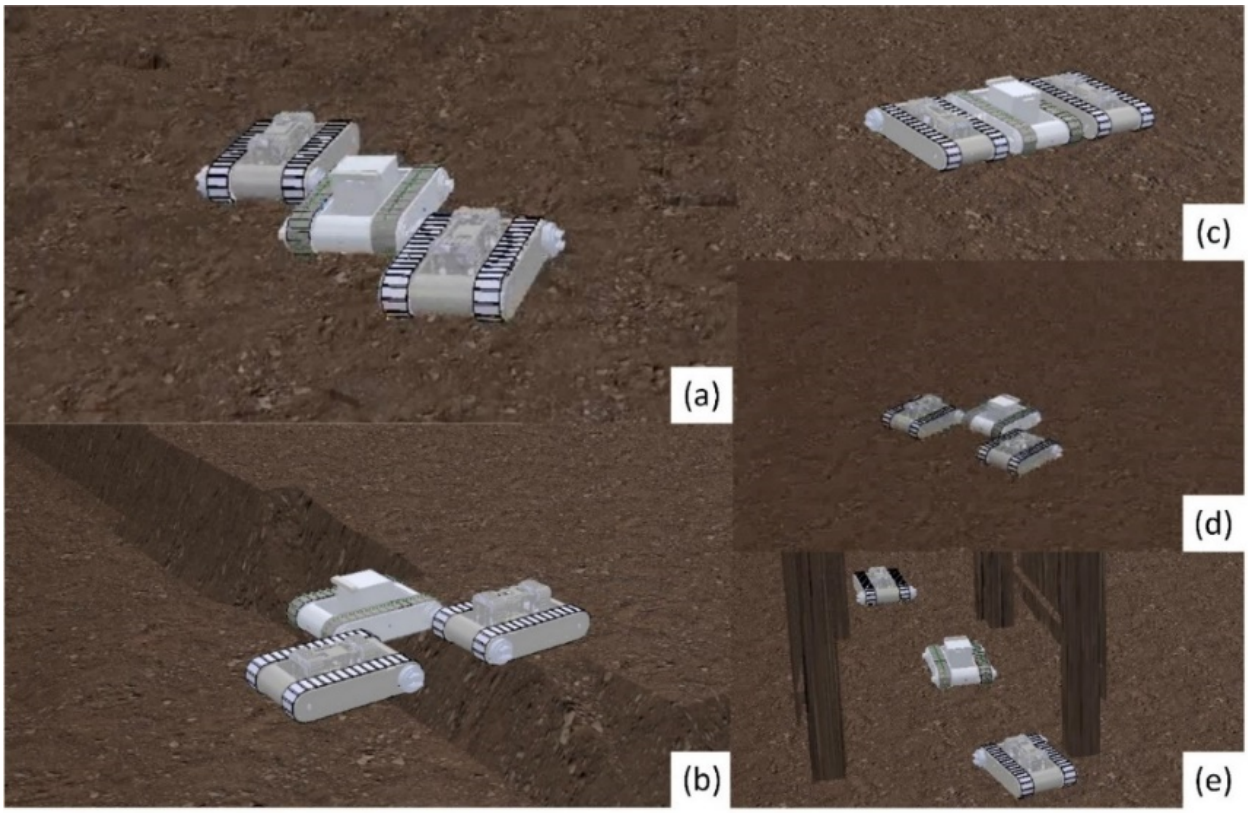

Figure 4.10: Different configurations for different terrain. a) Diagonal form to pass the rocky terrain; b) Crab-like form to cross a ditch; c) Closed-form to pass the muddy terrain. d)Triangle form to drag up the load carrier e) Decoupled to pass a narrow corridor

4.5 Torque and Force Analysis Simulation

Force sensors were attached to the docking points of the large configuration to measure the forces and torques during the operation, as shown in Figure 4.12.

4.6 Robot Localization Simulation

To track the poses of the locomotion module, Apriltags were attached to the locomotion modules for localization. A ROS package *apriltag_ros* was applied to publish the transforms between the camera frame and the tag frame. This method was tested in the CoppeliaSim

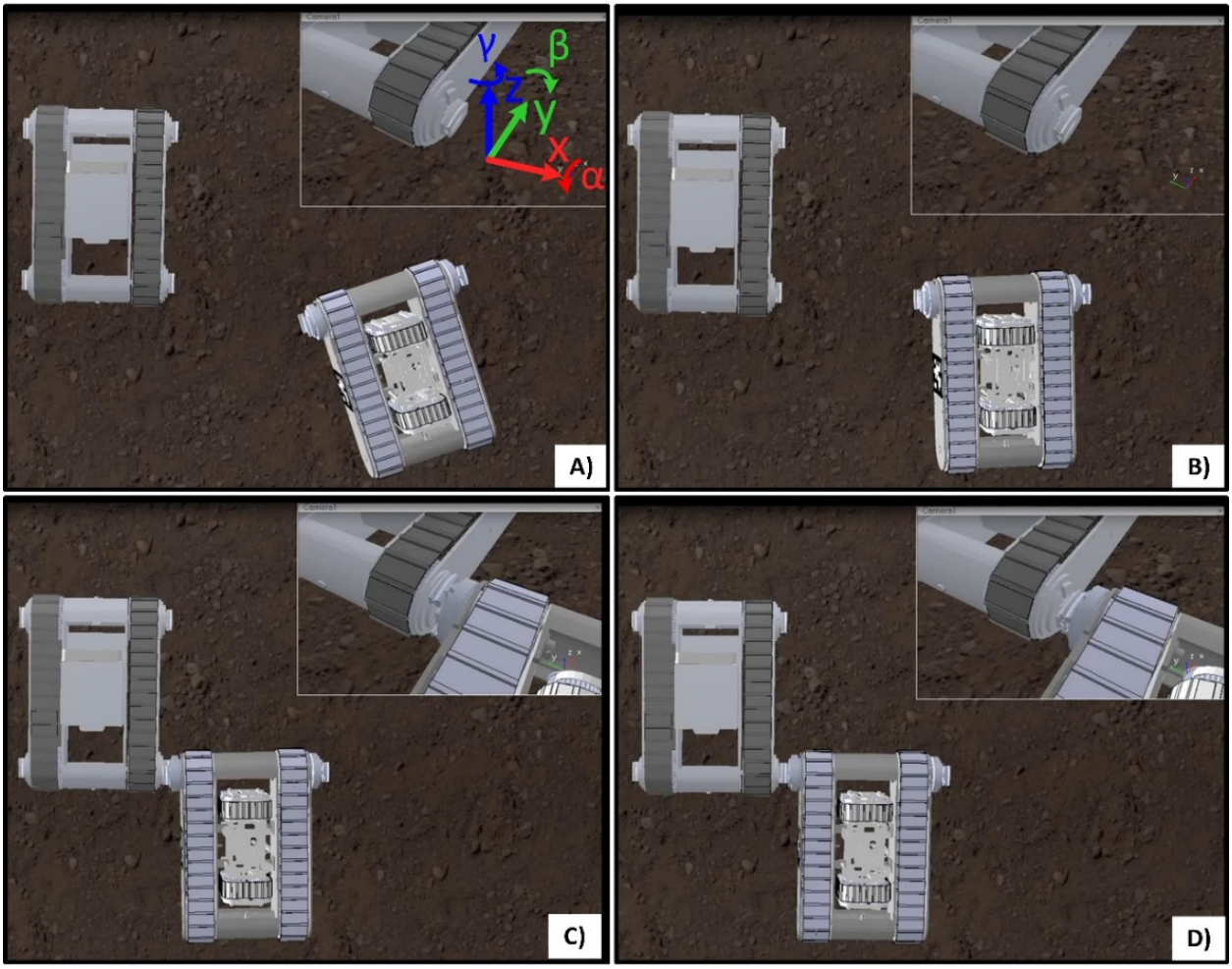

Figure 4.11: Docking process in CoppeliaSim. a) Minimize the error in x and gamma; b) Minimize the error in y; c) Minimize the error in z; d) Close the clamps

simulator. AprilTags with different IDs were placed on the locomotion for localization purposes, as shown in Figure 4.13. Figure 4.14 shows the transform information of the frames to track the positions of the robotic modules. This is the first step to apply the controller developed in Matlab to the robot models in CoppeliaSim.

In the CoppeliaSim simulator, the 4 USB cameras were simulated to form a surround-view to detect the tags for localization. Four *apriltag_ros* nodes run in parallel to continuously detect the tags from all four video streams. The detected tags with numbers and the relative poses were published out. Two example scenes were demonstrated to show the feasibility of

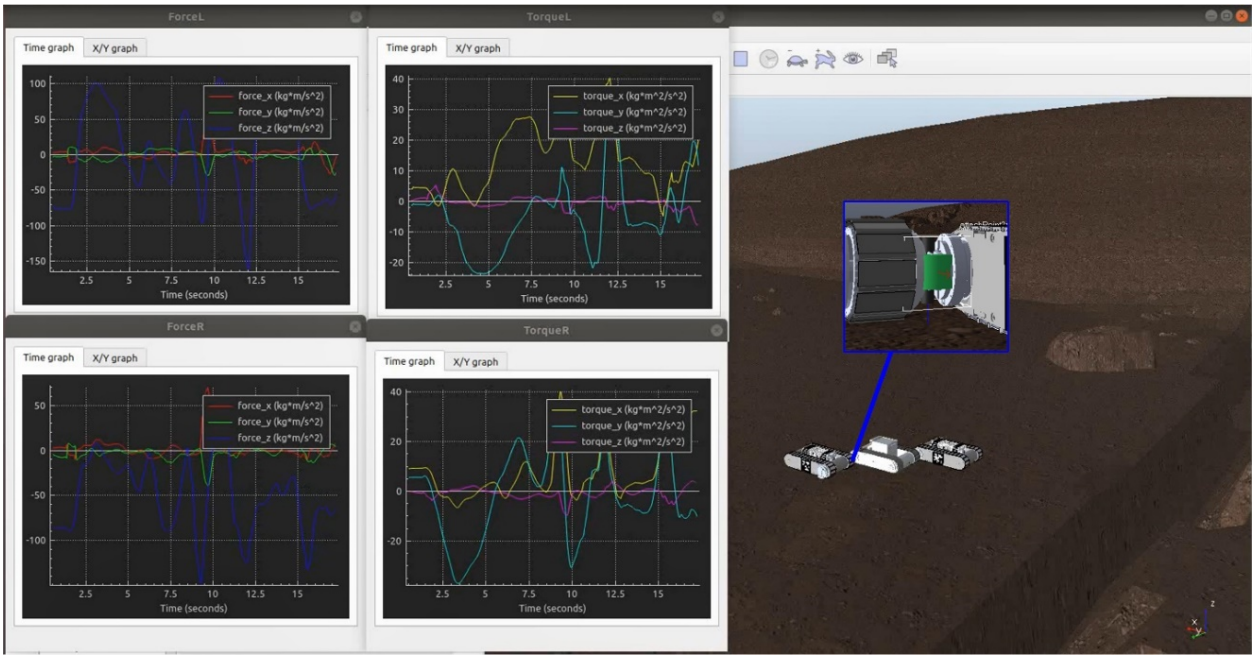

Figure 4.12: Force sensor attached to the docking point to measure the force and torque applied to the docking point during operation.

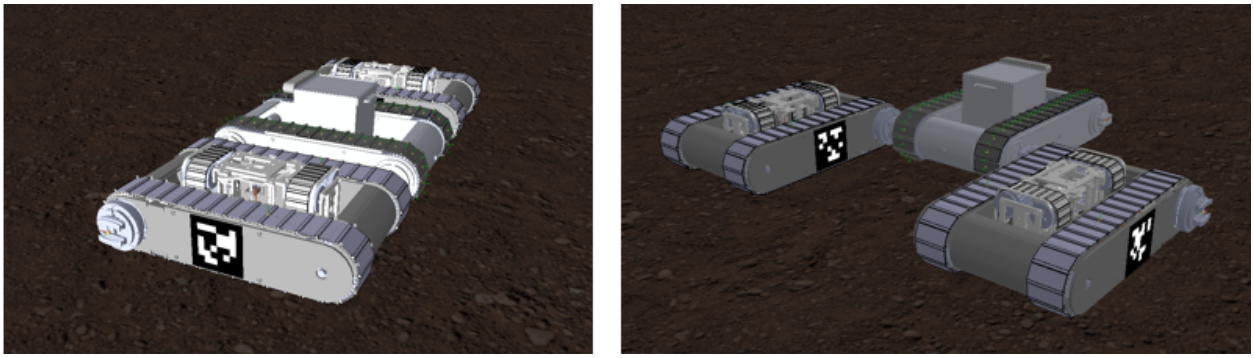

Figure 4.13: Load carrier and the locomotion module with apriltags in CoppeliaSim

the proposed robot localization method. As shown in Figure 4.15a, the pose of locomotion module 0 can be decided according to the detected Tag 1 from the front and left cameras; The back camera detected tag 2 to determine the pose of the locomotion module 1. Figure 4.15b shows another scene with a different robot's initial poses. Tag 0 and Tag 3 were detected to calculate the robot poses.

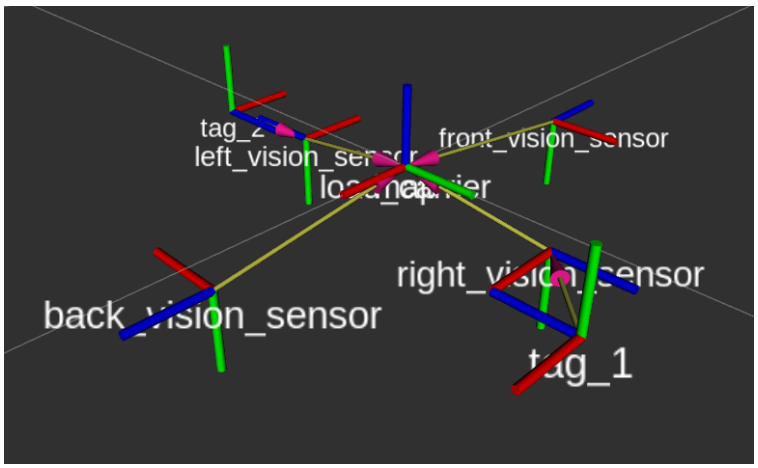

Figure 4.14: Local frames of the load carrier model and the locomotion module

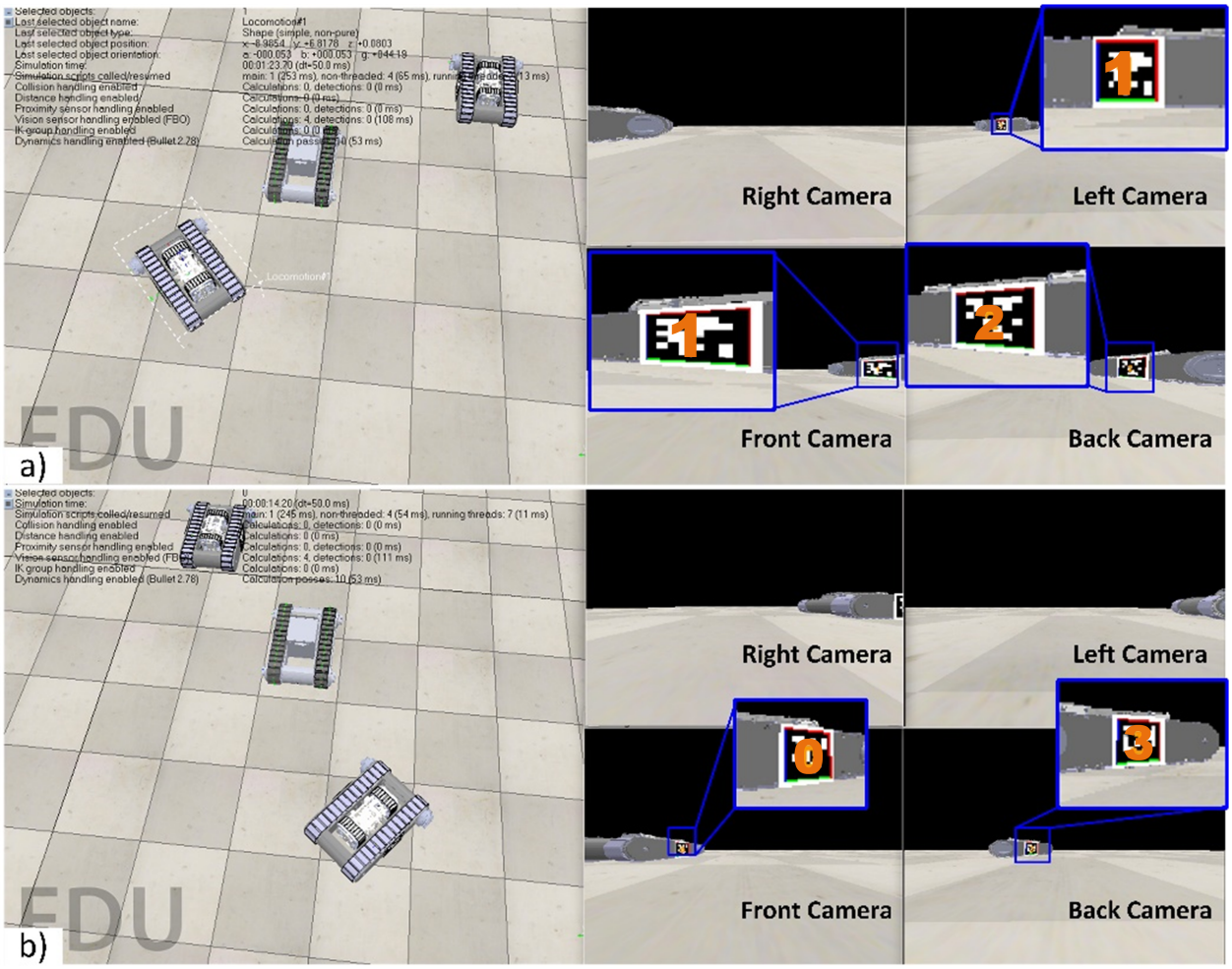

Figure 4.15: Surround-view implementation in V-rep and detected tags

Chapter 5

MOTION CONTROL OF THE SMARRT MODULE TOWARDS SELF-RECONFIGURATION

One of the core features of the proposed robotic system is self-reconfiguration capability. When the reconfiguration is required to perform complex tasks such as humanoid configuration to enable higher functionality or length extended configuration to cross a ditch, efficient and feasible algorithms need to be developed to deploy each robotic module to the desired position for achieving the reconfiguration. During the reconfiguration process, the target coupling mechanism of each module should align with each other autonomously. Assume that the load carrier of the SMARRT system detected a ditch and sent out a reconfiguration requirement. The locomotion modules that received the messages should navigate to the target positions with the desired poses, decided according to the location of the load carrier, from an initial position relatively far away.

An accurate localization method is necessary to let the robot realize where it is when performing navigation tasks. In general, popular localization methods have measurement errors that are allowable in a long-term navigation task. Some challenging tasks, such as autonomous alignment, require more accurate localization for precision control. In this case, an alternative sensory system, localization method, or controllers may be required to maneuver the

robot to the desired pose with a small acceptable error range after the robot gets close to the target. In the present case, after performing the proposed long-distance navigation, the robot should be within a close distance from the target. Efficient algorithms and controllers need to be developed for the locomotion module approaching the finial pose while adapting to the novel multi-directional locomotion module and special task requirements.

5.1 Introduction of Pose Stabilization Control

Due to the limitation of the hardware, especially the accuracy of the sensory equipment, such as GPS or Pozyx system, used to perform the localization task for determining the current locations of the robot modules, it is not possible to deploy the robot to the target position and achieve autonomous docking only with the global path and motion planners. This is because the misalignments allowed by the docking mechanisms is smaller than the error between the expected target position and the actual position the robot finally arrives. More accurate sensors and localization method, as well as a precision control algorithm are required to maneuver the robot module to the desired poses when it comes close to the target docking mechanism. This subsection presents the strategies used to realize the precision control.

5.2 Problem Statement

It is assumed that the load carrier will be stationary during the docking process. One of the locomotion modules in the lateral tracked mode should generally approach the desired pose to align the two docking mechanisms. As shown in Figure 5.1, a desired pose of the locomotion module should be determined according to the pose of the load carrier that ensures the

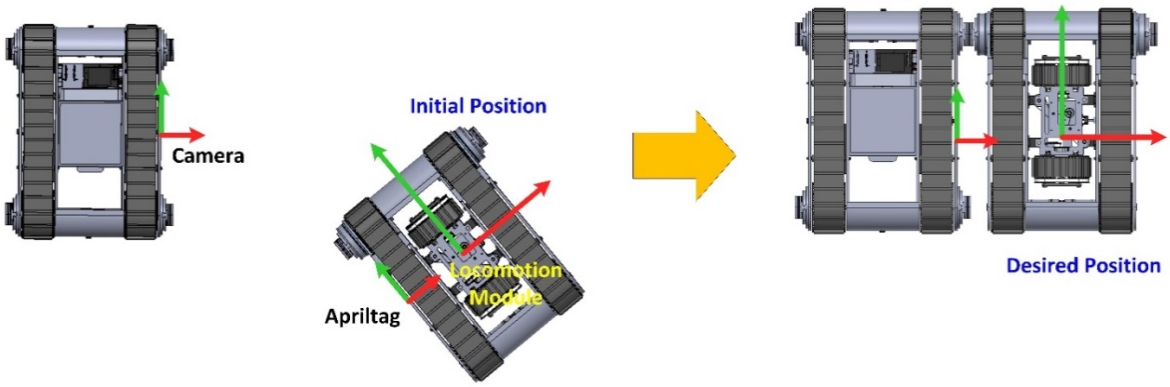

Figure 5.1: Initial and desired positions of the locomotion module

two docking mechanisms align and couple together after the locomotion module reaches the target position. This can be defined as a pose stabilization problems. The controller should take the pose differences between the current position and the desired position of the locomotion module as input and generate proper velocity commands for the locomotion module. In other words, the objective of the controller is to find a control $u = (v(t), \omega(t))$ to satisfy $\lim_{t \to \infty} e(t) = 0$.

5.3 Controller Design

The objective of the control problem is to minimize the error between an initial pose and the desired pose, including the orientation. Since the target system is nonlinear, has non-holonomic constraints, is underactuated and the theoretical limitations on point stabilization problems such that the system is not able to be globally asymptotically stable with a smooth time-invariant controller [55, 74], the design of the controller is challenging. In this work, we propose a Lyapunov function-based controller with a transit goal generator to solve the problem. It is necessary to study the robot kinematics to determine the relationship between the velocities of the robot and the speed of each track to bridge the command from the high-

level controllers to the low-level controllers. The load carrier has the same kinematics as the locomotion module in the longitudinal tracked mode, as described by the equation below:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{5.1}$$

where v and ω are the linear and angular velocities of the locomotion module, respectively. (x, y) is the current position of the robot with respect to the world coordinate frame. θ is the angle between the heading of the robot and the x-axis of the world frame. A transient goal is generated for the robot to avoid the situation where the robot goes directly to the goal position without updating the heading orientation. The transient goal generator gives the robot other target positions until the robot can head to the goal position with the desired orientation. The overall control process can be represented as shown in Figure 5.2. The transient goal generator creates a temporary goal according to the desired state

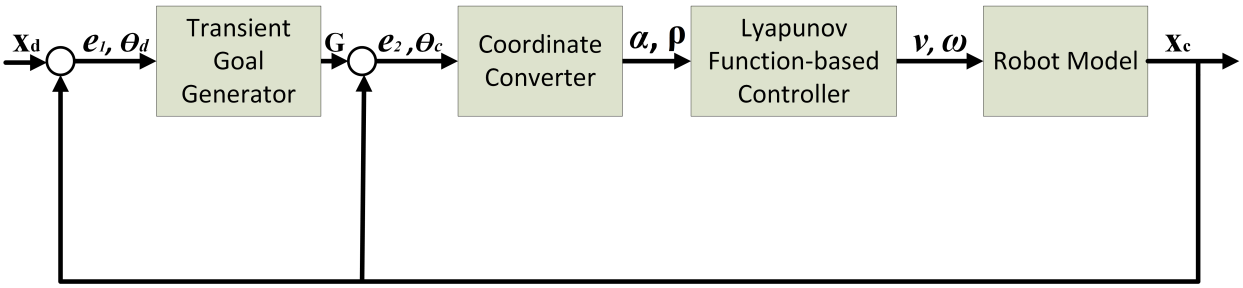

Figure 5.2: Control diagram of the proposed controller

$\mathbf{x}_d = [\ x_d \quad y_d \quad \theta_d\ ]^T$, and current pose $\mathbf{x}_c = [\ x_d \quad y_c \quad \theta_c\ ]^T$ of the robot following the equation below:

$$\mathbf{G} = \begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \|\mathbf{e}_1\| \cdot \begin{bmatrix} \cos(\beta + \gamma) \\ \sin(\beta + \gamma) \end{bmatrix}, \tag{5.2}$$

where $\mathbf{e}_1 = \begin{bmatrix} x_d - x & y_d - y \end{bmatrix}^{\mathrm{T}}$, $\beta = \tan^{-1}\left(\frac{y_d - y}{x_d - x}\right)$ and $\gamma{=}\beta{-}\theta_d$. Figure 5.3 shows the location of the transient goal, as the robot gets closer to the desired position, the transient goal tends to coincide with the desired position. The difference between the coordinates of the transient goal and the current position of the robot, $\mathbf{e}_2 = \begin{bmatrix} x_g - x_c & y_g - y_c \end{bmatrix}^{\mathrm{T}}$, as well as the heading angle θ_c of the robot will be fed in to a coordinate converter which aims to represent the robot poses in the Polar coordinate system as $\rho = \|e_2\|$ and $\alpha = \varphi - \theta_c$, where $\varphi = \tan^{-1}\left(\frac{y_g - y_c}{x_g - x_c}\right)$. The outputs ρ and α are the inputs to the Lyapunov Function-based controller. The kinematics of the robot should be converted into polar coordinates as well for developing the control laws:

$$\begin{cases} \dot{\rho} = -v\cos\alpha \\ \dot{\varphi} = v\frac{\sin\alpha}{\rho} \\ \dot{\theta}_c = \omega \end{cases} \tag{5.3}$$

The control law was designed as,

$$\begin{aligned} v &= k_1\rho\cos\alpha \\ \omega &= k_1\sin\alpha\cos\alpha + k_2\alpha \end{aligned}, \tag{5.4}$$

where k_1 and k_2 are the gains to be analyzed and tuned to get a stable system. The control laws came from a positive definite quadratic form of the Lyapunov function: $\mathrm{V} = \mathrm{V}_1 + \mathrm{V}_2 = \frac{1}{2}\rho^2 + \frac{1}{2}\alpha^2$. The time derivative can be calculated as:

$$\dot{V} = \dot{V}_1 + \dot{V}_2 = \rho\dot{\rho} + \alpha\dot{\alpha} = \rho(-v\cos\alpha) + \alpha(v\frac{\sin\alpha}{\rho} - \omega) \tag{5.5}$$

If a continuous function $V : \mathbf{U} \subseteq \mathbb{R}^n \to \mathbb{R}$ is positive definite such that $V(x) \to \infty$ as $\|x\| \to \infty$ and its derivative $-\dot{V}$ is positive definite, then the origin of the system is globally

asymptotically stable. In order to get a stable system, $-\dot{V}$ must be positive definite, let $V_1 = -k_1\rho^2\cos^2\alpha$ and $v = k_1\rho\cos\alpha$, where $k_1 > 0$. Substitute v into equation 5.5 and obtain $\dot{V}_2 = \alpha(k_1\cos\alpha\sin\alpha - \omega)$, where $\omega = k_1\sin\alpha\cos\alpha + k_2\alpha$ and $k_2 > 0$. Finally, the system can be proved to be globally asymptotically stable at the origin.

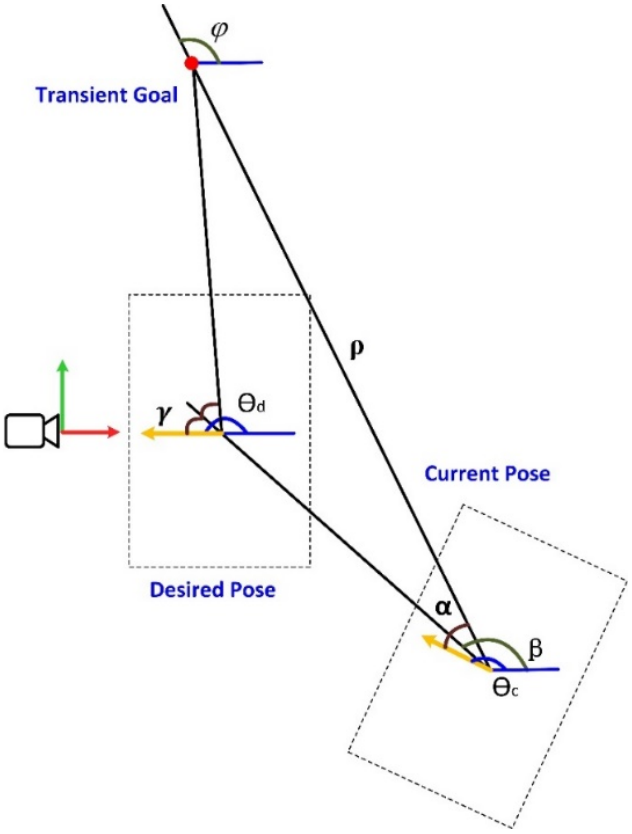

Figure 5.3: The robot poses and parameters

5.4 Simulation in MATLAB

The pose stabilization controller was implemented using Simulink in MATLAB to test the feasibility and tune the parameters of the controller. The Simulink module is presented in Figure 5.4.

Some trajectories of the robot were plotted and presented in Figure 5.5 to demonstrate the

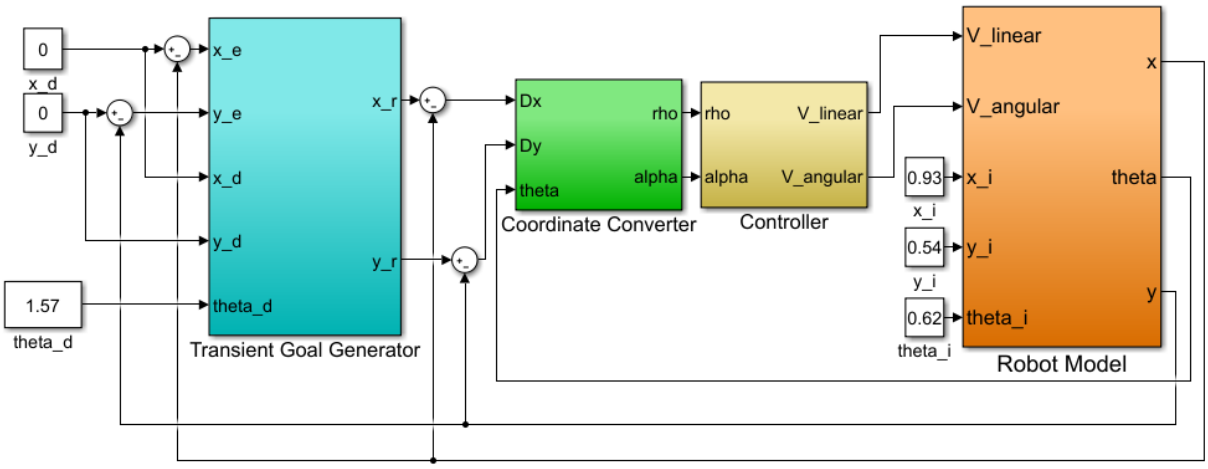

Figure 5.4: Simulink model of the overall control process

performance of the controller. The desired position is located at the origin of the world frame. Four random initial positions were selected from different quadrants within 1.5 m of the target. It should be pointed out that the controller aims to steer the robot to a transient goal until the robot's heading is the same as the desired pose. Since the robot allows backward motion, the desired heading of the robot should be determined according to the initial position of the robot. In this case, when the initial position of the robot is located in Quadrant I and IV, the desired pose should be π rad. Otherwise, the desired heading of the robot should be 0 rad. The time intervals between every two triangle marks on the trajectories are the same. From the results, the robot successfully achieved the desired pose in each run.

5.5 Real-world Validation

A real-world experiment was conducted to prove the feasibility of the proposed controller and the docking process. The precondition of this autonomous alignment is a reliable sensing

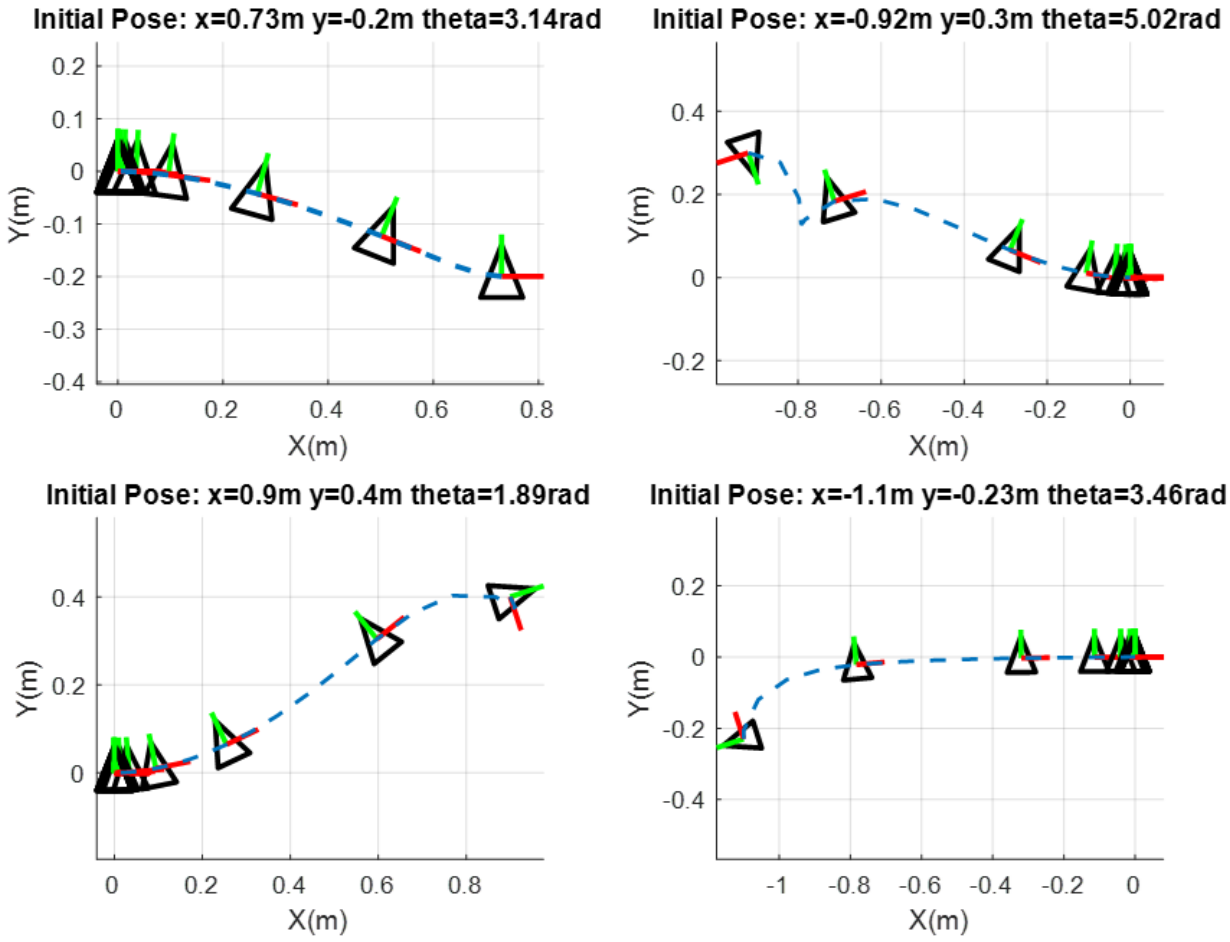

Figure 5.5: Trajectories of the robot model from the simulation in MATLAB

system to track the poses of the locomotion module. Apriltags [75] were utilized for localizing the locomotion module during the experiment. As shown in Figure 5.6, an Apriltag was placed on the track frame of a prototype locomotion module, and a USB camera was set up to track the tag. The first step of the experiment was to align the passive docking mechanism attached to the locomotion module to the active docking mechanism alongside the camera (Figure 5.6).

The ROS package *apriltag_ros* was implemented to publish the transforms between the camera frame and the tag frame. In order to accommodate the kinematic analysis of the

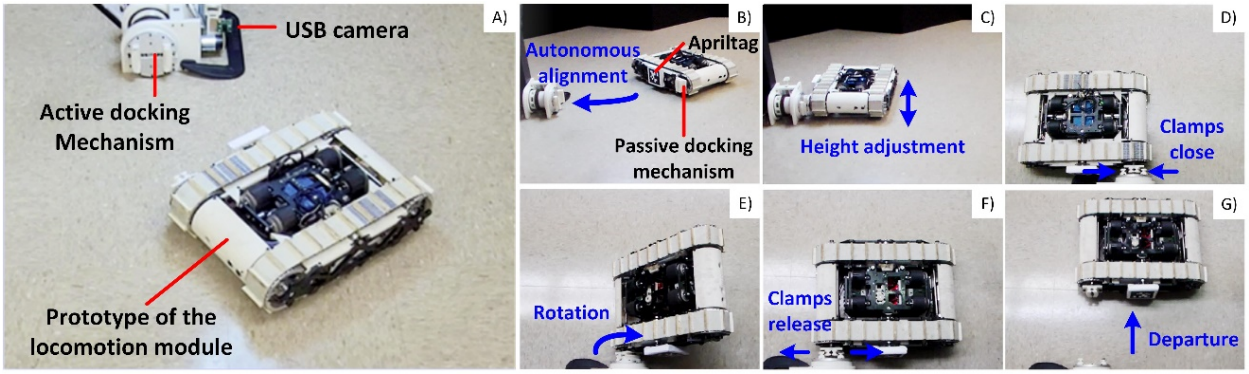

Figure 5.6: Experimental setup and the overall process

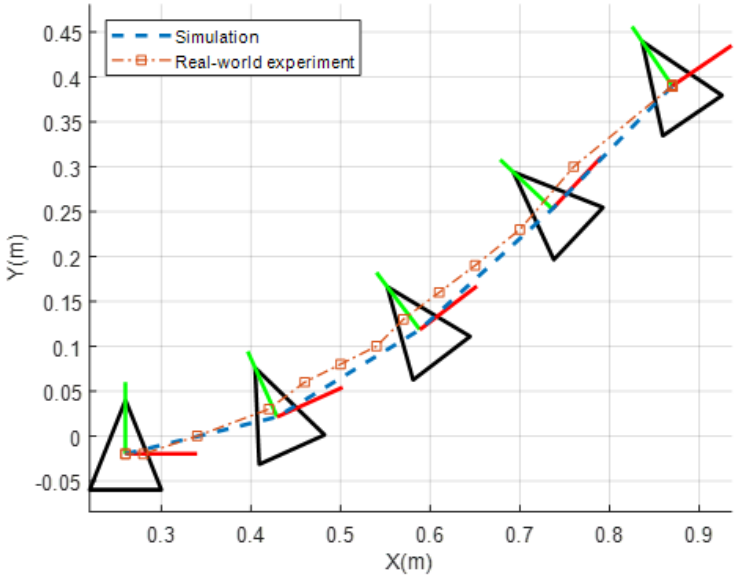

Figure 5.7: Comparison between the simulation and real-world experiment

locomotion module, the output relative pose was converted from the relative pose of the camera frame to the local frame of the locomotion module. The Apriltag should be in the FOV of the camera at all times when performing the autonomous alignment. After the prototype arrived at the desired position, the VTM adjusted the height of the passive docking mechanism for the alignment in the z-axis. The clamps of the active docking mechanism closed and rotated the locomotion module 180 degrees. Finally, the robot departed after the clamps were released. The overall process is shown in Figure 5.6b−g. The trajectory

of the autonomous alignment process was recorded and compared with the results from the simulations, as shown in Figure 5.7. The trajectory from the real-world experiment is a little different from the simulation results due to disturbances such as the measurement error of the localization and slippage of the robot, but it was still able to achieve the desired pose. This proves that the proposed controller is able to avoid disturbances as it is globally asymptotically stable.

Chapter 6

TRAJECTORY OPTIMIZATION AND TRACKING OF THE SMARRT MODULE

A trajectory optimization algorithm was developed to help the robot determine when to switch the locomotion modes and find the fastest path to the destination with the desired pose. After the trajectory is planned, a Predictive Model Control (MPC) based trajectory tracking controller takes charge of planning the motion for the locomotion module.

This chapter presents a pose stabilization controller, trajectory optimization method, 2D Simulation, and real-world experiments that were conducted to test the pose stabilization controller.

6.1 Trajectory Optimization

The pose stabilization controller presented in the previous section successfully maneuvered the locomotion module to the desired pose with allowable misalignments for completing the reconfiguration task. The prerequisite of the proposed controller is that the locomotion module arrived at a close point from the final destination and switched to the wheeled/lateral

tracked locomotion mode to perform the autonomous alignment task. This section presents an optimal solution to the autonomous alignment problem using the direct collocation-based method to plan the fastest path from the nearby point to the target position and determine the best timing to switch the locomotion modes.

The special mechanical structure of locomotion module brings in a new path planning problem, that is, to find the time-optimal (or shortest path) trajectories for a multi-directional vehicle with bounded curvatures. The curvature constraints are added to avoid the slippery motion of the differential drive mobile robots. It turns out that this new path planning problem exists widely in modular and reconfigurable robotic systems [76]. For this kind of system, the system dynamics and its moving mode change every time when its configuration is changed [77]. In a broader background, similar problems are also found in aerospace engineering, such as a rocket/spacecraft with multiple directional propulsion systems [78] or the optimal interception/rendezvous problem of two moving objects [79]. Note that some systems allow activating multiple modes simultaneously (e.g., a satellite activates multiple jets for orbit correction) while some can only choose one mode (e.g., the STORM mobile robot can choose either the tracked mode or the wheeled mode). In mathematics, all these systems belong to the hybrid/switching system and the optimal control of such systems is called switched/hybrid optimal control problem (SOCP/HOCP) [80, 81]. However, existing solution to this problem are either on theoretical interest (which is very general and too broad) or on numerical algorithms. Neither could be directly applied on our new problem for explicitly characterizing the optimal path of a multi-directional vehicle with bounded curvatures.

6.1.1 Related Work

Dubins [82] first explored the shortest path problem of a unidirectional car (a car that can only move forward, or Dubins car, or unidirectional car) with maximum steering constraints. The method that he used is mainly geometric reasoning, which is specific to this unique problem. Reeds and Shepp [83] extended this work to a bidirectional car (or Reeds-Shepp car) that also allows going backwards. They used another ad hoc approach mixing geometric reasoning and variational analysis to find the solution. As the knowledge of this problem accumulated, mathematicians started to look at this problem from a more profound viewpoint, that is, reformulating the problem into an optimal control problem. This way, more advanced mathematical tools from optimal control theory could be used. Boissonnat et al [84] and Sussman et al [85] firstly applied the Pontryagin's Maximum Principle (PMP) [86] to reformulate and solve this problem. Since this viewpoint is more systematic and insightful, it was soon extended to other similar problems (such as for the differential drive [87, 88, 89] vehicle, for the omni-directional vehicle [90], and for vehicles with environmental constraints [91, 92]).

However, when the multiple moving modes were introduced, the nature of the problem changes significantly. First of all, the system is no more a continuous dynamics. The multiple moving modes make the system become a *switching/hybrid system* with externally forced switching (EFS). Independent switching variables have to be used to describe the different modes. This is the fundamental difference from the existing research on unidirectional [82], bidirectional steering [83, 93], bidirectional differential drive [87, 88, 89], and omni-directional vehicles [90]. Note that the omni-directional vehicle essentially belongs to the continuous dynamics category although it looks more dexterous than a multi-directional vehicle. Second, since the different moving mode usually have different moving directions, the optimal path not only consists of "inflexion" points (C^1 but not C^2) and "cusps" (C^0 but not C^1, and

the left curve is tangent to the right curve), but also "corner" points (whose left and right tangent vectors are not parallel). The works on optimal control of switching systems are extensively surveyed in [80]. These works, especially those done by Bengea and DeCarlo [94] as well as Xu and Antsaklis [95], form the concrete theoretical foundations of our results.

6.1.2 Switched Optimal Control Problem (SOCP)

The SOCP problem considered in this dissertation is formulated as a time optimal control problem with fixed endpoints. That is, the cost function to be minimized is defined as

$$T = \int_0^T 1 dt \tag{6.1}$$

where $T \in \mathbb{R}^+$ is the time when the system reaches the end point. There is no penalties on the switching event. The optimal trajectory is constrained by the system dynamics of an ideal car with normalized speed and steering ranges:

$$\dot{\mathbf{x}} = \sum_{i=1}^{n} v_i(t)\mathbf{f}_i(\mathbf{x}(t), \mathbf{u}_i(t)) \tag{6.2}$$

where $v_i \in \{0, 1\}$ is the selector of the i_{th} moving mode. $\mathbf{x} = [x, y, \theta]^T \in \mathbb{R}^2 \times \mathrm{S}^1$ is the state vector that encompasses the car location x and y, as well as its orientation θ. $\mathbf{u}_i = [g_i, s_i]^T \in [-1, 1] \times [-1, 1]$ is the control input for the i_{th} mode, where the symbol "g" stands for "gear" (go forward or backward) and "s" stands for "steering" (go left or right).

$\mathbf{f}_i$ is given as

$$\begin{aligned}\mathbf{f}_i(\mathbf{x}, \mathbf{u}_i) &= \begin{bmatrix} g_i \cos(\theta + d_i) \\ g_i \sin(\theta + d_i) \\ s_i / R_i \end{bmatrix} \\ &= g_i \begin{bmatrix} \cos(\theta + d_i) \\ \sin(\theta + d_i) \\ 0 \end{bmatrix} + s_i \begin{bmatrix} 0 \\ 0 \\ 1/R_i \end{bmatrix} \\ &= g_i \mathbf{a}_i(\mathbf{x}) + s_i \mathbf{b}_i(\mathbf{x})\end{aligned} \tag{6.3}$$

where $d_i \in [0, \pi)$ is a constant scalar, representing the heading direction of the i_{th} mode. $R_i \in \mathbb{R}^+$ is a constant scalar to define the curvature bound.

Since the vehicle is allowed to go backwards, the optimal path from the start configuration to the end configuration is the same as the path from the end configuration to the start configuration. This characteristic is known as the *symmetric* property for the Reeds-Shepp car and we extend the same sense to our problem. Therefore, due to the symmetric property and without loss of generality, the car is initialized at the origin with zero degree of orientation. The initial and terminal states are given as

$$\mathbf{x}(0) = \mathbf{0}, \quad \mathbf{x}(T) = [x_f \ y_f \ \theta_f]^T \tag{6.4}$$

Therefore, the optimal control problem is to determine a map $\mathbf{u}(t) : \mathbb{R} \longrightarrow \Omega$ that minimizes Eq. (6.1) subject to Eqs. (6.2-6.4). Ω is the admissible control set:

$$\begin{aligned}\Omega = \{\mathbf{u} = [\ g_1 \quad s_1 \quad v_1 \quad ... \quad g_n \quad s_n \quad v_n \ ]^T \\ : g_i \in [-1, 1], \ s_i \in [-1, 1], \ v_i \in \{0, 1\}\}\end{aligned}$$

An important special case for the SOCP is that only one mode is allowed at the same time, such as in the case of the STORM robot. This case is defined by an additional constraint as shown in Eq. (6.5) and the problems are called a special SOCP (SSOCP).

$$\sum_{i=1}^{n} v_i^2 = 1 \tag{6.5}$$

6.1.3 Controllability and Existence

The controllability of our problem is straightforward and given as bellow:

Theorem 6.1 (Controlability)**.** *The SOCP and the SSOCP described in section 6.1.2 are controllable.*

Proof. Note that the controllability of the SSOCP implies the controllability of the SOCP since an admissible trajectory for the SSOCP is always an admissible trajectory of the SOCP. It is well known that the Reeds-Shepp car problem is controllable (see section 7 of [85]). Therefore, let $v_1 \equiv 1$ and all the other modes be turned off. Then the multi-directional vehicle becomes a Reeds-Shepp car and thus can reach all the points in the state space $(\mathbb{R}^2 \times \mathbb{S}^1)$, i.e. SSOCP is controllable. □

With the known controllability, the existence of the solution could be proved using Theorem 6 by Sussmann and Tang [85], which provides a sufficient condition for a system with control linear form.

Theorem 6.2 (SOCP Existence)**.** *The SOCP described in section 6.1.2 has a solution.*

Proof. Based on Theorem 6 of Sussmann and Tang [85], it needs to be shown that the SOCP

meets all the conditions. Expanding Eq. (6.2) yields

$$\begin{aligned}\dot{\mathbf{x}} &= \sum_{i=1}^{n} v_i g_i \mathbf{a}_i(\mathbf{x}) + v_i s_i \mathbf{b}_i(\mathbf{x}) \\ &= \sum_{i=1}^{n} u_{i,a} \mathbf{a}_i(\mathbf{x}) + u_{i,b} \mathbf{b}_i(\mathbf{x})\end{aligned} \tag{6.6}$$

Apparently, the new control input $u_{i,a}$ and $u_{i,b} \in [-1, 1]$ which is a compact convex set and any value in $[-1, 1]$ could be decomposed into an admissible value of v_i (choosing from $\{0, 1\}$) and an admissible value of g_i or s_i (choosing from $[-1, 1]$). Since $\mathbf{a}_i$ and $\mathbf{b}_i$ are differentiable vectors in $\mathbb{R}^3$ and are bounded, the system is complete. Theorem 6.1 shows that the problem is controllable. Therefore, the SOCP has a solution. □

Remark 6.1. Note that we do not need to convexify the value set of v_i to prove the existence due to the specific control linear form of Eq. (6.3) and the existing convexity of the g_i and s_i value sets. This is a stronger result than that in Bengea and DeCarlo [94]. Due to the same reason, Theorem 6.2 is not restricted to a two-switched system.

However, this proof does not apply to SSOCP since the additional constraint in Eq. (6.5) breaks the convexity of the control set as well as the tangent vector space (which is one of the critical conditions for most existence theorems, see chapter 9-16 of Cesari [96]). For this case, the existence of a solution has to be determined by examining the compactness of the reachable set and the lower semicontinuity of the cost function, which is usually difficult to verify/falsify. To avoid this, we use the existing results from the optimal control theory for hybrid systems [81, 97]. However, the strongest statement we can get for our problem is for the cases with a predefined mode-switching sequence (the original SSOCP allows arbitrary and infinitely many mode-switching). Although the statement is weaker than the original SSOCP setting, it covers most practical cases, including the motivating problem of the STORM robot. The main theory that we use is Corollary 3.6 of Goebel [97], whose original

statement is quite long because it aims to cover a broad class of cost functions. For the time-optimal problem, the statement could be much simplified and is presented here.

Theorem 6.3 (Corollary 3.6 of Goebel, 2019)**.** *For a hybrid dynamical system:*

$$\begin{cases} (x,u) \in C_x \times U & \dot{x} = f_0(x) + \sum_{l=1}^{m} f_l(x) u_l \quad (6.7) \\ (x,u) \in D_x \times U & x^+ = G(x,u) \quad (6.8) \end{cases}$$

defined on a compact hybrid time domain $E = \bigcup_{j=j_a}^{j_b} [t_j, t_{j+1}] \times \{j\} \subset \mathbb{R}^2$ *and to minimize the cost function of* $\sum_{j=j_a}^{j_b} \int_{t_j}^{t_{j+1}} 1 dt$ *(time optimal), where* $x \in \mathbb{R}^n$ *is the state,* $u \in U \subset \mathbb{R}^m$ *is the control,* $\phi : \mathbb{R}^+ \to \mathbb{R}^n$ *is one trajectory of the hybrid system,* $j_a \leq j_b$ *are integers, if the following conditions are satisfied:*

1. f_l *is continuous and* $\exists M > 0, \quad \forall x, \quad \forall l = 0, 1, ..., m,\ f_l(x) \leq M(1 + \|x\|)$
2. C_x *is closed and* U *is compact and convex*
3. G *is locally bounded,*
4. *The set* Θ *of compact hybrid time domain is bounded and is closed with respect to set convergence*
5. *The value function* $V(t_c, j_c) = \inf\{\sum_{j=j_c}^{j_b} \int_{t_j}^{t_{j+1}} 1 dt \ | t_{j_c} = t_c\}$ *is finite*

then the optimal solution exists.

Directly applying this theorem to our problem gives the following result.

Theorem 6.4 (SSOCP Existence)**.** *A SSOCP with predefined mode-switching sequence has a solution.*

Proof. Firstly, a predefined mode-switching sequence for the SSOCP automatically defines a compact hybrid time domain with nonempty interior for each time interval $I_j = [t_j, t_{j+1}]$. Every mode-switching event correponds to one $j_a \leq j \leq j_b$. Define a new state as $z = (\mathbf{x}, i_j)$ where $i_j(t) \in I_d = \{1, 2, ..., n\}$ represents the selected mode, then the SSOCP could be formulated into the hybrid dynamical system framework as in Eqs. (6.7)-(6.8). Note that the $u_1, u_2 \in [-1, 1]$ do not depend on the mode index i_j.

$$
\begin{aligned}
&C_z = \mathbb{R}^2 \times \mathrm{S}^1 \times I_d, \quad U = [-1, 1]^2 \\
&\dot{z} = (f_1(z)u_1 + f_2(z)u_2, 0), \quad u_1 = g_i, \quad u_2 = s_i \\
&f_1(z) = \mathbf{a}_{i_j}(\mathbf{x}), \quad f_2(z) = \mathbf{b}_{i_j}(\mathbf{x}) \\
&D_z = \mathbb{R}^2 \times \mathrm{S}^1 \times I_d, \quad z^+ = (\mathbf{x}, i_{j+1})
\end{aligned}
$$

It is easy to verify that conditions (1)-(3) are satisfied. For a predefined mode-switching case, the hybrid time domain is bounded and closed. The value function is finite too. Therefore, based on Theorem 6.3, the optimal control for the above hybrid dynamical system exists, i.e. the SSOCP has a solution. □

6.1.4 Problem Statement

In our cases, the hybrid tracked-wheeled feature of the locomotion module introduces discontinuous system dynamics. For example, the longitudinal locomotion mode, lateral locomotion mode, and the motion of the VTM are intrinsically discrete. Therefore, previous motion planning for the autonomous docking task was stage-based that the overall task is divided into several sub-tasks:

1. rough navigation to the desired position using tracks,

2. fine adjustment of the robot gesture in lateral locomotion mode,
3. fine adjustment of the height of the VTM to align the docking mechanism in the Z direction,
4. and fine adjustment of the clamps' gesture.

To discover an optimal solution for the locomotion module to approach the target and choose the best position to switch the dynamics, the problem can be defined as a SSOCP and a direct collocation-based optimization method is proposed to solve this problem.

The continuous-time problem is supposed to be converted into a nonlinear program by discretizing the trajectory into collocation points. The continuous system dynamics should be converted into a set of constraints and applied to the state and control at the collocation points. Namely, the trajectory of a continuous-time problem is supposed to be discretized into a finite set of decision variables by representing the continuous states by their values at specific collocation points in time and the dynamics of the target system should be switched at a certain point.

The proposed trajectory optimization methods should plan an optimal path from the initial pose to the desired pose and determine the position for the locomotion module to switch the dynamics simultaneously. As shown in Figure 6.1, the locomotion module approaches to the desired position for reconfiguration in the longitudinal tracked mode. After it arrives at the point determined by the proposed direct collocation based path planner, the Vertical Translational Mechanism (VTM) will lower down the lateral tracked unit to change the locomotion mode. Two different approaches were proposed to determine the position to switch the dynamics. When optimizing the trajectories, the mobile robot is considered as an ideal agent moving on flat ground, and there is no slippage between the robot and the ground. The robot dynamics are fully described by the following two dynamics, which correspond

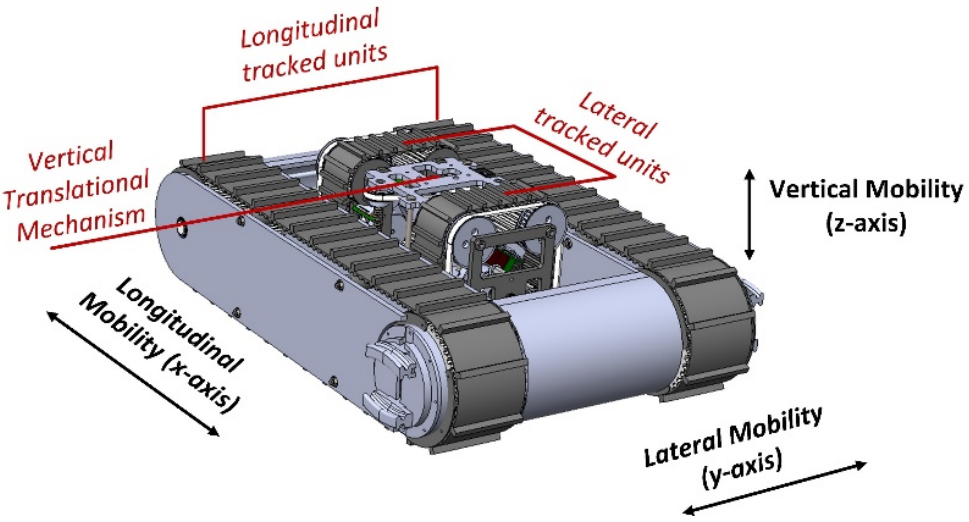

Figure 6.1: 3D model of the Locomotion Module

to the track and wheel modes, respectively. The disturbances from uneven terrain and the robot's slipperiness will be taken into consideration when tracking the trajectory.

6.1.5 Problem Formulation

The trajectory optimization aims to discover an optimal path according to an objective function that mathematically describes what the best solution should be. In general, an objective function can be defined in Bolza from:

$$J(t_0, t_f, x(t), u(t)) = \int_{t_0}^{t_f} l(t, x(t), u(t))dt + g(t_0, t_f, x(t_0), x(t_f)), \tag{6.9}$$

where the state $\mathbf{x} \in \mathbb{R}^n$, control $\mathbf{u} \in \mathbb{R}^m$, as well as the initial and final time (t_0, t_f) are considered as the decision variables. The integral of $l : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ defines the running cost, and the function $g : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defines the boundary objective. If $l \equiv 0$, the problem is said to be in Mayer form. If $g \equiv 0$, it becomes a Lagrange problem. In our case, the best solution is defined as the fastest path from the initial point to the final

position and the objective function is in lagrange form:

$$J = \int_{t_0}^{t_f} 1dt, \tag{6.10}$$

The locomotion module is a hybrid system with discontinuous dynamics. Generally, these control systems have two types of switching behaviors [98, 99]. One is the autonomous or uncontrolled switch without the influence from a separate switching mechanism. The second type of switch can be controlled directly, as the switch between the longitudinal tracked mode and vertical tracked mode through the VTM of the locomotion module. In our case, the switch of the system will happened only once during the entire autonomous alignment process. The state variables for the hybrid system are: $\mathbf{x}^{\mathrm{T}} = [x, y, \theta]^{\mathrm{T}}$, where $[x, y]$ represents the position of the locomotion module in $\{XOY\}$ plane; θ is the direction of the heading of the locomotion module. The control inputs are: $\mathbf{u}^{\mathrm{T}} = [v, \omega]^{\mathrm{T}}$, where v and ω are the angular and linear speeds of the locomotion module, respectively. Before switching, the system dynamics of the locomotion module in the longitudinal tracked mode can be formulated as:

$$f_1 = \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6.11}$$

After switching, the system dynamics of the locomotion module in the lateral tracked mode becomes:

$$f_2 = \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta + \frac{\pi}{2}) & 0 \\ \sin(\theta + \frac{\pi}{2}) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{6.12}$$

Then the problem can be formulated as:

$$
\begin{aligned}
&\text{minimize } J \\
&\text{subject to:} \\
&\text{dynamics: } \dot{\mathbf{x}}(t) = \begin{cases} f_1(\mathbf{x}(t), \mathbf{u}(t)) & t \leqslant t_s \\ f_2(\mathbf{x}(t), \mathbf{u}(t)) & t > t_s \end{cases} \\
&\text{boundary constraints: } \mathbf{x}(0) = \mathbf{x}_0,\ \mathbf{x}(t_f) = \mathbf{x}_f \\
&\text{path constraints: } -\mathbf{x}_{\max} \leqslant \mathbf{x} \leqslant \mathbf{x}_{\max},\ -\mathbf{u}_{\max} \leqslant \mathbf{u} \leqslant \mathbf{u}_{\max}
\end{aligned}
\tag{6.13}
$$

6.1.6 Proposed Solution

The first step to solve the optimization problem is to define the collocation points by discretizing the trajectory into a finite set of decision variables according to following steps:

1. Partition the time interval T into M+N subintervals to compute a piecewise constant control sequence $\{\mathbf{u}^1{}_1, \mathbf{u}^1{}_2, \cdots, \mathbf{u}^1{}_M\}$ and $\{\mathbf{u}^2{}_{M+1}, \mathbf{u}^2{}_{M+2}, \cdots, \mathbf{u}^2{}_{M+N}\}$, then the corresponding states at the collocation points are defined as $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{M+N}\}$.

2. Convert the system dynamics into collocation constraints.

   (a) The dynamics can be constructed in integral form:

$$
\begin{cases} \int_{t_k}^{t_{k+1}} \dot{x} dt = \int_{t_k}^{t_{k+1}} f_1 dt & k = \{1, 2, \cdots, M-1\} \\ \int_{t_k}^{t_{k+1}} \dot{x} dt = \int_{t_k}^{t_{k+1}} f_2 dt & k = \{M, \cdots, M+N-1\} \end{cases}, \tag{6.14}
$$

   (b) and then the discretized system dynamics can be approximated using trapezoidal quadrature. The approximation is applied between every pair of collocation points

as:

$$\begin{cases} x_{k+1} - x_k = \frac{1}{2}h_k({f^1}_{k+1} + {f^1}_k) & k = \{1, 2, \cdots, M-1\} \\ x_{k+1} - x_k = \frac{1}{2}h_k({f^2}_{k+1} + {f^2}_k) & k = \{M, \cdots, M+N-1\} \end{cases}, \tag{6.15}$$

where $h_k = t_{k+1} - t_k$ and t_k is the time at the collocation point k. Then the system dynamics at each collocation point becomes:

$$\begin{cases} {f^1}_k = f_1(t_k, x_k, u_k) & k = \{1, 2, \cdots, M\} \\ {f^2}_k = f_2(t_k, x_k, u_k) & k = \{M+1, \cdots M+N\} \end{cases}, \tag{6.16}$$

where t_k, x_k, u_k are the decision variables in the nonlinear program.

3. Enforce the path constrains at each collocation point: $-\mathbf{x}_{\max} \leqslant \mathbf{x}_k \leqslant \mathbf{x}_{\max}$, $-\mathbf{u}_{\max} \leqslant \mathbf{u}_k \leqslant \mathbf{u}_{\max}$, and the boundary constraints should be enforced at the first and the last collocation points.

4. Find the approximate objective function using trapezoid rule:

$$\int_{t_0}^{t_f} l(t, x(t), u(t)) \approx \sum_{0}^{M+N-1} \frac{1}{2}h_k(l_k + l_{k+1}) \tag{6.17}$$

By following the procedures above, the switching of the dynamics is enforced at the specific collocation point M. However, a further analysis of the timing to change the dynamics is required. Two different ways were utilized to determine the switching point. An iterative way is to set a fixed time interval and choose the switching point at different collocation points generally and solve the nonlinear program with different switching points to obtain $M+N-1$ paths. The fastest path will be selected as the final solution. An optimal method is to define different time intervals Δt_1 and Δt_2. In this method, the objective function

becomes:

$$\int_{t_0}^{t_f} l(t, x(t), u(t)) \approx \sum_{0}^{M-1} \frac{1}{2}\Delta t_1(l_i + l_{i+1}) + \sum_{M}^{M+N-1} \frac{1}{2}\Delta t_2(l_j + l_{j+1}) \tag{6.18}$$

where $M \in \mathbb{Z} : 0 \leqslant M \leqslant N-2$, and $N \in \mathbb{Z} : N \geqslant M+1$ are predefined constant values.

6.1.7 Implementation in Matlab and Results

The proposed direct collocation-based optimization method for solving an optimal path with a proper switching position was implemented in Matlab using the f_{mincon} solver. As mentioned in 6.1.6, two different approaches were developed to solve the problems. Figures 6.2-6.4 demonstrate the results from the first iterative method in which fifteen collocation points were defined and the switching point was enforced at the first collocation point to the fourteenth points successively. This is to ensure that the robot module will be in the lateral tracked mode in the end.

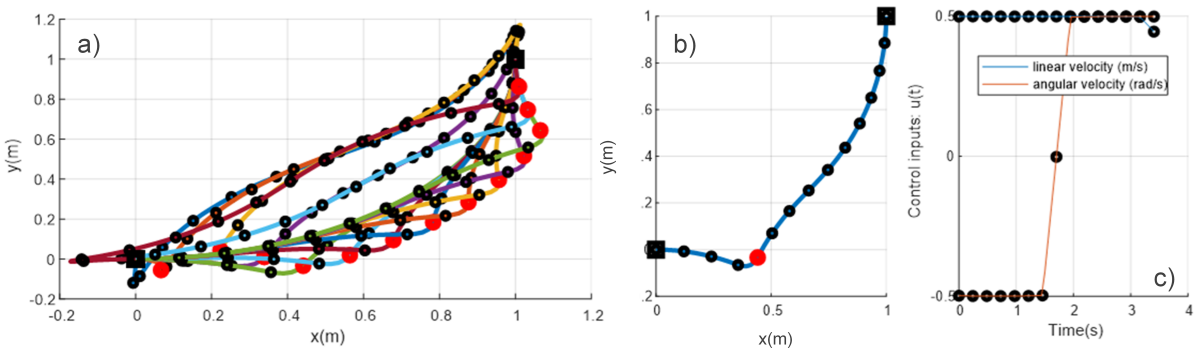

Figure 6.2: Solutions with the boundary constraints, $\mathbf{x_0}$=(0 m, 0 m, 0 rad), $\mathbf{x_f}$=(1 m, 1 m, 0 rad) and path control constraints: $\mathbf{u_{max}}$=(0.5 m/s, 0.5 rad/s), a) all the planned paths with switching points (red dots); b) the fastest path c) and control signals were presented

Different boundary and path constraints were selected to test the proposed method. Figure 6.2a shows all the optimal paths with the same boundary constraints, $\mathbf{x_0}$=(0 m, 0 m, 0 rad),

$\mathbf{x_f}$=(1 m, 1 m, 0 rad), control constraints: $\mathbf{u_{max}}$=(0.5 m/s, 0.5 rad/s), and the same dynamic constraints. The red dots indicate the switching points. Figure 6.2b shows the fastest path and and Figure 6.2c presents the control signals to achieve the best trajectory. Similarly, Figure 6.3 and Figure 6.4 demonstrate the other solutions with different boundary conditions and maximum velocities.

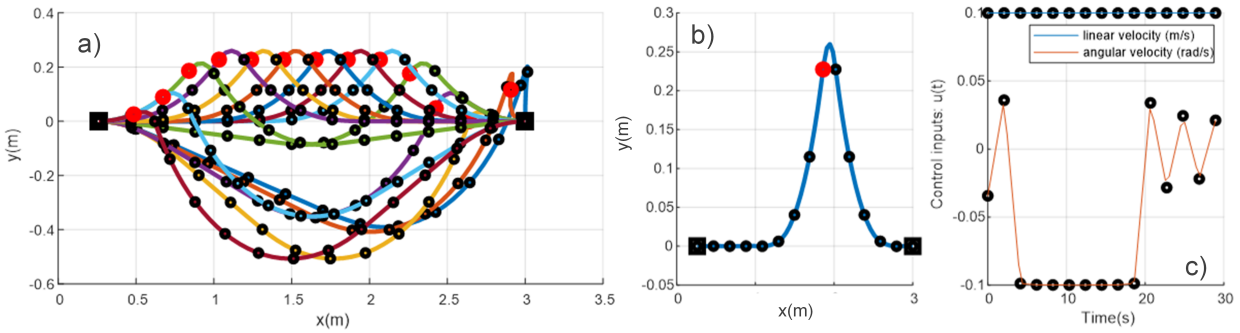

Figure 6.3: Solutions with the boundary constraints, $\mathbf{x_0}$=(3 m, 0 m, $\pi\, rad$), x_f=(0.26 m, 0 m, 0.5π rad), and path control constraints: $\mathbf{u_{max}}$=(0.1 m/s, 0.1 rad/s), a) all the planned paths with switching points (red dots); b) the fastest path c) and control signals were presented

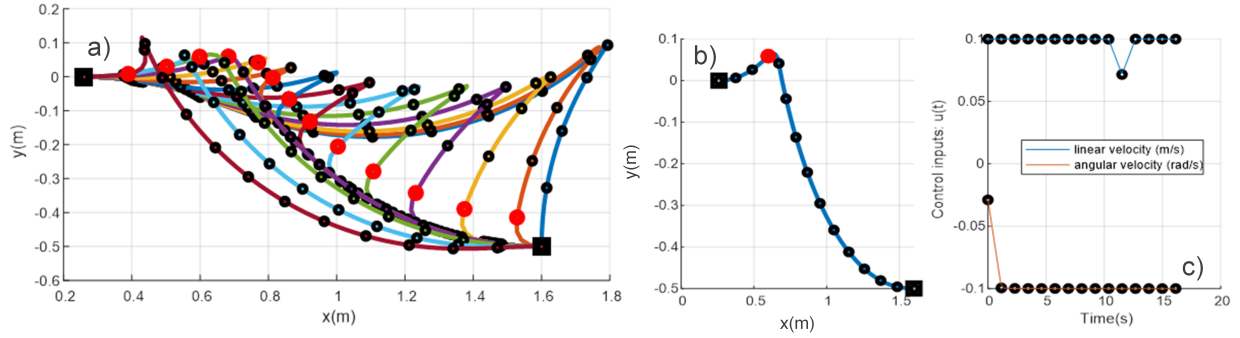

Figure 6.4: Solutions with the boundary constraints, $\mathbf{x_0}$=(1.6 m, -0.5 m, $\pi\, rad$),x_f=(0.26 m, 0 m, 0.5π rad) and path control constraints: $\mathbf{u_{max}}$=(0.1 m/s, 0.1 rad/s), a) all the planned paths with switching points (red dots); b) the fastest path c) and control signals were presented

The fastest path from each run was compared to the result from the other various time interval based-method with the same constraints, as shown in Figure 6.5. From the comparison,

the optimal trajectories from different methods coincide with slight differences. In conclusion, the results from all the tests successfully discover the best trajectory that satisfied all the constraints and a switching point to change the dynamics, which proves the feasibility of the proposed direct collocation based optimization for path planning of the hybrid system.

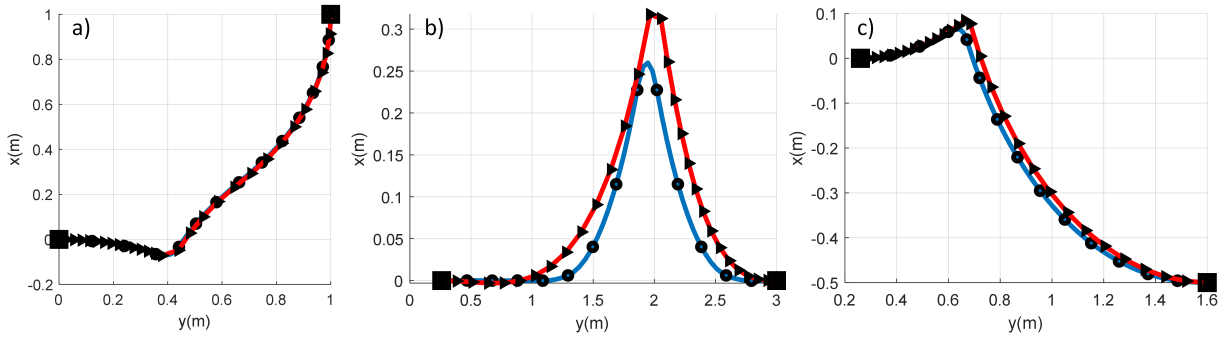

Figure 6.5: Comparisons between results from different methods: a) $\mathbf{x_0}$=(0 m, 0 m, 0 rad), $\mathbf{x_f}$=(1 m, 1 m, 0 rad) and $\mathbf{u_{max}}$=(0.5 m/s, 0.5 rad/s), b) $\mathbf{x_0}$=(3 m, 0 m, π rad), $\mathbf{x_f}$=(0.26 m, 0 m, 0.5π rad) and $\mathbf{u_{max}}$=(0.1 m/s, 0.1 rad/s) c) $\mathbf{x_0}$=(1.6 m, -0.5 m, πrad), $\mathbf{x_f}$=(0.26 m, 0 m, 0.5πrad) and $\mathbf{u_{max}}$=(0.1 m/s, 0.1 rad/s)

6.2 Model Predictive Control based Trajectory Tracking

After determining the optimal trajectory and the position to check the dynamics, a trajectory tracking controller should be developed for the robot to track the planned path. Although a sequence of control signals has been obtained along with the optimal trajectory, the open-loop control commands cannot ensure that the robot is at the desired position. It tends to deviate from the planned path due to the slippage of the differential-drive tracks, delays in communication, uncertainty in calibration when mapping the robot's linear and angular velocities to the direct commands for each actuation motor. The optimal trajectory is planned for the locomotion module to complete the autonomous alignment mission that

requires the robot to stabilize at a proper pose to align the coupling mechanisms from each side for docking and reconfiguration. The millimeter-sized allowable misalignment range put forward the requirement for the high accuracy of the robot's final pose. Therefore, an effective controller should be developed for the locomotion module to track the planned path and arrive at the final pose with as little measurement error as possible. This section presents a Model Predictive Control (MPC) based trajectory tracking controller developed for the locomotion module to respond to the challenges and achieve better performance when aligning with the target docking mechanism.

6.2.1 Problem Formulation

The locomotion module has two locomotion modes and a vertical translational DOF, which allows the switching of the system dynamics. The kinematics of a hybrid system is formulated as:

$$\dot{\mathbf{x}} = \sum_{i=1}^{n} v_i(t)\mathrm{f}_i(\mathbf{x}(t), \mathbf{u}(t)) \tag{6.19}$$

where $v_i \in \{0, 1\}$ is a selector that indicates that the robot is in the i_{th} ($i \in \mathbb{Z}$) moving mode and $n \in \mathbb{Z}$ is the number of possible kinematic modes. Only one mode is allowed at the same time, so there exists only one i that makes v_i =1. In our case, the system is altered between two locomotion modes and the reference trajectory $\mathbf{x_r} = [\mathbf{x_{r_1}}, \mathbf{x_{r_2}}]$, where $\mathbf{x_{r_1}} = [\ \mathbf{x}_{r_1} \quad \mathbf{y}_{r_1} \quad \theta_{r_1}\ ]^T \in \mathbb{R}^{3\times M}$ is the optimal trajectory before switching the locomotion mode and $\mathbf{x_{r_2}} = [\ \mathbf{x}_{r_2} \quad \mathbf{y}_{r_2} \quad \theta_{r_2}\ ]^T \in \mathbb{R}^{3\times(N-M)}$ is the reference trajectory after switching. The corresponding reference controls are represented as $\mathbf{u_r} = [\mathbf{u_{r_1}}, \mathbf{u_{r_2}}]$, where $\mathbf{u}_{r_1} = [\ v_{r_1} \quad \omega_{r_1}\ ]^T \in \mathbb{R}^{2\times M}$ and $\mathbf{u}_{r_2} = [\ v_{r_2} \quad \omega_{r_2}\ ]^T \in \mathbb{R}^{2\times(N-M)}$

The reference trajectory and control signals were obtained from the direct collocation-based trajectory optimizer as described in 6.1.

A discrete-time dynamics can be obtained using Euler's approximation as:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{A}\mathbf{u}(k)\Delta T, \tag{6.20}$$

where A will change from

$$A = \begin{bmatrix} \cos\theta(k) & 0 \\ \sin\theta(k) & 0 \\ 0 & 1 \end{bmatrix}$$

to

$$A = \begin{bmatrix} \cos(\theta(k) + \frac{\pi}{2}) & 0 \\ \sin(\theta(k) + \frac{\pi}{2}) & 0 \\ 0 & 1 \end{bmatrix},$$

if the locomotion module arrives at the switching point. Besides, a Taylor series around the point (x_r, u_r) is utilized to discard the high order of the error model, presented as:

$$\dot{x} = f(x_r, u_r) + \left.\frac{\partial f(x,u)}{\partial x}\right|_{\substack{x=x_r \\ u=u_r}} (x - x_r) + \left.\frac{\partial f(x,u)}{\partial u}\right|_{\substack{x=x_r \\ u=u_r}} (u - u_r), \tag{6.21}$$

Accordingly, the discrete-time error model can be formulated as:

$$\mathbf{e}(k+1) = \mathbf{B}(k)\mathbf{e}(k) + \mathbf{C}(k)\mathbf{e}_u(k) \tag{6.22}$$

where

$$\mathbf{B}(k) = \begin{bmatrix} 1 & 0 & -v_r(k)\sin\theta_r(k)\Delta T \\ 0 & 1 & v_r(k)\cos\theta_r(k)\Delta T \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{C}(k) = \begin{bmatrix} \sin\theta_r(k)\Delta T & 0 \\ \cos\theta_r(k)\Delta T & 0 \\ 0 & \Delta T \end{bmatrix},$$

when the locomotion module is moving in the longitudinal tracked mode. After switching the locomotion modes,

$$\mathbf{B}(k) = \begin{bmatrix} 1 & 0 & -v_r(k)\sin(\theta_r(k)+\frac{\pi}{2})\Delta T \\ 0 & 1 & v_r(k)\cos(\theta_r(k)+\frac{\pi}{2})\Delta T \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{C}(k) = \begin{bmatrix} \sin(\theta_r(k)+\frac{\pi}{2})\Delta T & 0 \\ \cos(\theta_r(k)+\frac{\pi}{2})\Delta T & 0 \\ 0 & \Delta T \end{bmatrix}.$$

6.2.2 Proposed Solution

A MPC-based method was proposed to solve the trajectory tracking problem. The basic concept of the MPC strategy can be defined in Figure 6.6. A prediction horizon N should be defined to predict the future outputs at each time within the defined horizon. The predicted outputs $y\,(t+k\,|t)$ for $k = 1...N$ are based on the predicted control signals $u\,(t+k\,|t)$, $k = 0...N$ - 1 [29]. This optimal control sequence is calculated by solving an open-loop optimization problem for the prediction horizon at each time step according to a well-defined objective function and constraints. The first value of the computed control sequence will be taken into account. At the next time step, the new system state will be obtained and new control input will be calculated following the same process. In general, Linear Model

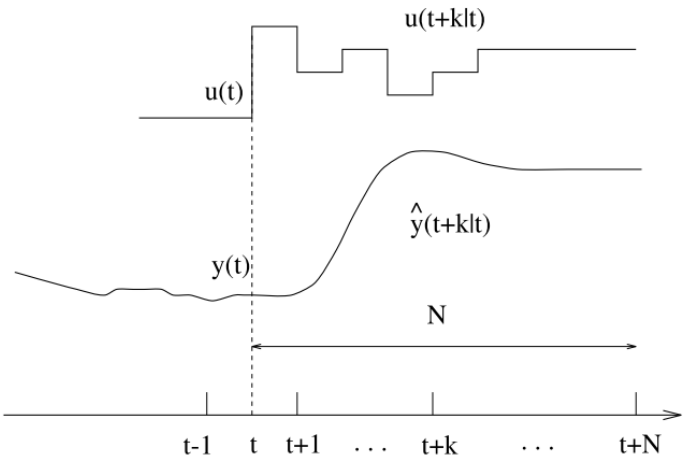

Figure 6.6: MPC strategy [29]

Predictive Control (LMPC) is popular to be applied to solve the trajectory tracking problem. An error model should be obtained to track the differences between the desired trajectory and the current states.

According to Equation 6.22, the predicted states and corresponding inputs within a certain horizon can be fed into an open-loop optimizer to calculate a sequence of the optimized control signals according to a well-defined cost function:

$$\begin{aligned} &\text{minimize } \mathrm{J} = \mathrm{Q_f}(\mathrm{x_T})^2 + \sum_{0}^{\mathrm{T\text{-}1}} [\mathrm{Q}(\mathrm{x_t})^2 + \mathrm{R}(\mathrm{u_t})^2] + \mathrm{R_d}(\mathrm{u_{t+1}}\text{-}\mathrm{u_t})^2 \\ &\text{subject to: error model } \mathbf{e} \text{ and constraints } \text{ - } \mathbf{u}_{\max} \leqslant \mathbf{u} \leqslant \mathbf{u}_{\max}, \end{aligned} \tag{6.23}$$

where Q_f represents the final cost, Q and R are the running costs of the states and controls, respectively. R_d shows the smoothness of the control signals.

6.2.3 Results and Conclusion

The proposed method was implemented using Python. The open-loop optimization problem used an open source Python-embedded modeling language for convex optimization, named CVXPY [100, 101]. The robot motions were predicted and simulated using the kinematic model as described in 3.5. When solving the problem, the dynamics equation changed after the robot arrived at a nearby point (within 0.02m) to the switching point or the index of the target reference point is larger than the index of the switching point. Figure 6.7 shows two examples of the resulting tracking path in green and planned path in red. From the results, it is obvious that the MPC-based trajectory tracking controller tracked the planned path and arrived at the final position successfully. However, at the switching point, the robot was not able to track the exact path because equations for predicting the future motions were not changed in time when it was within the prediction horizon N.

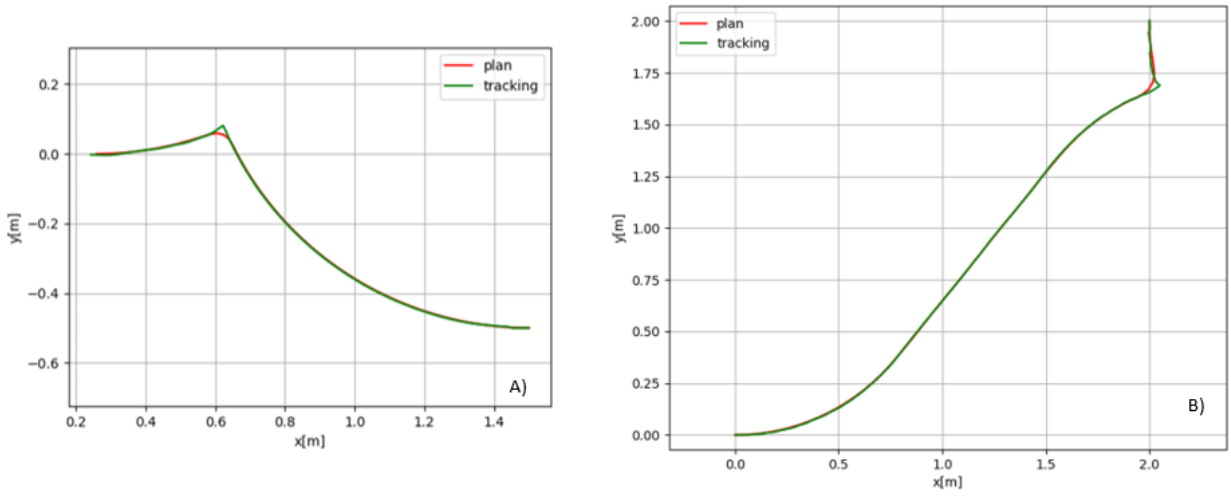

Figure 6.7: Solutions with different reference trajectories

In conclusion, the MPC-base trajectory tracking controller was initially proved to be feasible for tracking the trajectory with changed dynamics. Improvements are required for obtaining a smooth curve around the switching point. Read-world test will be conducted in the future

for further analysis of the controller.

Chapter 7

CONCLUSION AND FUTURE WORK

This chapter concludes the dissertation with a summary of the current work as well as a forward looking discussion into prospective research.

7.1 Summary

This dissertation focused on the development and evaluation of an novel modular robotic system. The mechanical and electrical design of SMARRT, a new reconfigurable modular robotic system, was described in detail. Special attention was paid to simulation testing conducted on the docking mechanism, which is one of the key mechanical advancements of this system that facilitates autonomous docking. Following the physical design, review details were provided on simulations conducted in CoppeliaSim, which validated the ability of the robot system to adapt to a variety of terrains through reconfiguration. The control algorithm and robot localization method, which represent the software part of the autonomous reconfiguration process, were then presented in detail. Navigation strategies were reviewed, and new methodologies were developed for deploying the robot modules autonomously, especially for maneuvering a robot to the desired pose for reconfiguring into different configuration types for completing various missions and improving the autonomy while navigating challenging

terrains. A Lyapunov function-based pose stabilization controller was then developed and validated in MATLAB simulation to complement the autonomous navigation task. This is to align the docking mechanism of the locomotion module to the target docking mechanism for docking. This autonomous alignment process can not be achieved via long-distance navigation because of the small range of the allowed misalignment. Both autonomous docking capabilities were validated in a physical experiment where a prototype of the SMARRT locomotion module located a standalone docking mechanism, drove up to it, docked, flipped, and then drove away. After solving the autonomous alignment problem at a closed range, an optimization approach was proposed to help the locomotion module find the best position to switch the dynamics and an optimal trajectory for autonomous alignment.

7.2 Future Work

Having proved the function of the overall mechanical design and controller, future work will primarily be focused on increasing efficiency for both. From the mechanical design side, there is room for improvement in terms of packaging space, smooth operation, and general durability of the active docking mechanism. Once realized, these improvements will make the active docking mechanism an attractive solution for reconfigurable robotic systems at large as well as potentially other industrial applications where genderless fail-safe couplings are advantageous. From the software side, efficient algorithms and controllers will be developed for the locomotion module approaching the final pose while adapting to the novel multi-directional locomotion module and special task requirements. In addition, the current locomotion module is able to adjust its pose during the autonomous alignment process, along x, y, z and yaw directions using the differential drived tracks. The pitch can be adjusted via the rotation of the docking mechanism. However, the differences in roll

direction is not able to be eliminated. As a result, the current experiments were conducted assuming that the reconfiguration process should be performed on relatively flat terrain. And the misalignment should be within the maximum allowable misalignment range of the docking mechanism. The reconfiguration capability can be improved for more challenging situations by designing a VTM that can rotate the tracks to adjust the roll.

Bibliography

[1] L. Parker, *Multiple Mobile Robot Systems*, pp. 921–941. 2008.

[2] M. J. Matarić, "Issues and approaches in the design of collective autonomous agents," *Robotics and Autonomous Systems*, vol. 16, no. 2-4, pp. 321–331, 1995.

[3] L. Parker, "Alliance: an architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[4] L. E. Parker, "Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance," 2000.

[5] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," *Industrial Robot: An International Journal*, vol. 43, no. 6, pp. 583–587, 2016.

[6] P. Baker and M. Canessa, "Warehouse design: A structured approach," *European Journal of Operational Research*, vol. 193, no. 2, pp. 425–436, 2009.

[7] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and R. Marinier III, "Progress toward multi-robot reconnaissance and the magic 2010 competition," *Journal of Field Robotics*, vol. 29, pp. 762–792, 2012.

[8] R. Reid, A. Cann, C. Meiklejohn, L. Poli, A. Boeing, and T. Braunl, "Cooperative multi-robot navigation, exploration, mapping and object detection with ROS," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1083–1088, IEEE, 2013.

[9] A. Howard, L. Parker, and G. Sukhatme, "The SDR Experience: Experiments with a Large-Scale Heterogeneous Mobile Robot Team," vol. 21, 2004.

[10] B. M. Yamauchi, "PackBot: a versatile platform for military robotics," in *Proc.SPIE*, vol. 5422, 2004.

[11] P. Wells and D. Deguire, "TALON: a universal unmanned ground vehicle platform, enabling the mission to be the focus," in *Proc.SPIE*, vol. 5804, 2005.

[12] S. H. Munkeby, D. Jones, G. Bugg, and K. Smith, "Applications for the MATILDA robotic platform," in *Proc.SPIE*, vol. 4715, 2002.

[13] J. W. Purvis and P. R. Klarer, "Ratler: Robotic all-terrain lunar exploration rover," 1992.

[14] T. Kot and P. Novak, "Application of virtual reality in teleoperation of the military mobile robotic system TAROS," *International Journal of Advanced Robotic Systems*, vol. 15, p. 172988141775154, 2018.

[15] M. Dimastrogiovanni, F. Cordes, and G. Reina, "Terrain Estimation for Planetary Exploration Robots," 2020.

[16] W. Saab, P. Racioppo, and P. Ben-Tzvi, "A review of coupling mechanism designs for modular reconfigurable robots," *Robotica*, vol. 37, no. 2, pp. 378–403, 2019.

[17] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, "Design of the atron lattice-based self-reconfigurable robot," *Auton. Robots*, vol. 21, p. 165–183, Sept. 2006.

[18] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *Mechatronics, IEEE/ASME Transactions on*, vol. 7, pp. 431 – 441, 2003.

[19] A. Spröwitz, P. Laprade, S. Bonardi, M. Mayer, R. Moeckel, P.-A. Mudry, and A. J. Ijspeert, "Roombots—towards decentralized reconfiguration with self-reconfiguring

modular robotic metamodules," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1126–1132, 2010.

[20] J. Liu, S. Ma, Y. Wang, and B. Li, "Network-based reconfiguration routes for a self-reconfigurable robot," *Science in China Series F: Information Sciences*, vol. 51, pp. 1532–1546, 2008.

[21] G. Jing, T. Tosun, M. Yim, and H. Kress-Gazit, "Accomplishing high-level tasks with modular robots," *CoRR*, vol. abs/1712.02299, 2017.

[22] R. Gross, E. Tuci, M. Dorigo, M. Bonani, and F. Mondada, "Object transport by modular robots that self-assemble," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2558–2564, 2006.

[23] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, and C. Taylor, "Towards robotic self-reassembly after explosion," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2767–2772, 2007.

[24] Z. Guanghua, D. Zhicheng, and W. Wei, "Realization of a modular reconfigurable robot for rough terrain," in *2006 International Conference on Mechatronics and Automation*, pp. 289–294, 2006.

[25] F. Vieira, A. Medeiros, P. Alsina, and A. Jr, "Position and orientation control of a two-wheeled differentially driven nonholonomic mobile robot.," pp. 256–262, 2004.

[26] T. T. Hoang, P. M. Duong, N. T. T. Van, and T. Q. Vinh, "Stabilization control of the differential mobile robot using lyapunov function and extended kalman filter," *ArXiv*, vol. abs/1707.05458, 2017.

[27] M. KALYONCU and F. DEMİRBAŞ, "Differential Drive Mobile Robot Trajectory

Tracking With Using Pid and Kinematic Based Backstepping Controller," *Selcuk University Journal of Engineering ,Science and Technology*, vol. 5, no. 1, pp. 1–15, 2017.

[28] W. Saab and P. Ben-Tzvi, "A Genderless Coupling Mechanism with 6-DOF Misalignment Capability for Modular Self-Reconfigurable Robots," *Journal of Mechanisms and Robotics*, vol. 8, no. c, pp. 1–9, 2016.

[29] K. Patan, *Model predictive control*, vol. 197. 2019.

[30] P. Moubarak and P. Ben-Tzvi, "Modular and reconfigurable mobile robotics," *Robotics and Autonomous Systems*, vol. 60, no. 12, pp. 1648–1663, 2012.

[31] P. Moubarak and P. Ben-Tzvi, "Modular and reconfigurable mobile robotics," *Robotics and Autonomous Systems*, vol. 60, no. 12, pp. 1648–1663, 2012.

[32] K. M. Hossain, C. A. Nelson, and P. Dasgupta, "Enumeration of Configurations and Their Kinematics for ModRED II Modular Robots," *Journal of Mechanisms and Robotics*, vol. 9, no. 5, p. 054501, 2017.

[33] M. Yao, X. Xiao, C. H. Belke, H. Cui, and J. Paik, "Optimal Distribution of Active Modules in Reconfiguration Planning of Modular Robots," *Journal of Mechanisms and Robotics*, vol. 11, no. 1, p. 011017, 2019.

[34] S. S. Sohal, B. Sebastian, and P. Ben-Tzvi, "Autonomous Docking of Hybrid-Wheeled Modular Robots With an Integrated Active Genderless Docking Mechanism," *Journal of Mechanisms and Robotics*, vol. 14, no. 1, p. 011010, 2022.

[35] M. Jorgensen, E. Ostergaard, and H. Lund, "Modular ATRON: modules for a self-reconfigurable robot," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, pp. 2068–2073, 2004.

[36] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, "Design of the ATRON lattice-based self-reconfigurable robot," *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, 2006.

[37] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-TRAN: self-reconfigurable modular robotic system," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[38] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Distributed Self-Reconfiguration of M-TRAN III Modular Robotic System," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 373–386, 2008.

[39] A. Sproewitz, P. Laprade, and R. Moeckel, "Roombots — Towards Decentralized Reconfiguration with Self-Reconfiguring Modular Robotic Metamodules," no. iii, pp. 1126–1132, 2010.

[40] A. Sproewitz, A. Billard, P. Dillenbourg, and A. J. Ijspeert, "Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture," in *2009 IEEE International Conference on Robotics and Automation*, pp. 4259–4264, 2009.

[41] A. Sprowitz, S. Pouya, S. Bonardi, J. Van Den Kieboom, R. Mockel, A. Billard, P. Dillenbourg, and A. Jan Ijspeert, "Roombots: Reconfigurable Robots for Adaptive Furniture," *IEEE Computational Intelligence Magazine*, vol. 5, no. 3, pp. 20–32, 2010.

[42] R. Gross, E. Tuci, M. Dorigo, M. Bonani, and F. Mondada, "Object transport by modular robots that self-assemble," in *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pp. 2558–2564, 2006.

[43] F. Mondada, L. Gambardella, D. Floreano, S. Nolfi, J. Deneubourg, and M. Dorigo, "The cooperation of swarm-bots - Physical interactions in collective robotics," *IEEE Robotics & Automation Magazine*, vol. 12, no. 2, pp. 21–28, 2005.

[44] R. Gro, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous Self-Assembly in Swarm-Bots," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1115–1130, 2006.

[45] Z. Guanghua, D. Zhicheng, and W. Wei, "Realization of a Modular Reconfigurable Robot for Rough Terrain," in *2006 International Conference on Mechatronics and Automation*, pp. 289–294, 2006.

[46] H. X. Zhang, S. Chen, W. Wang, J. W. Zhang, and G. H. Zong, "Runtime reconfiguration of a modular mobile robot with serial and parallel mechanisms," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2999–3004, 2007.

[47] H. Zhang, Z. Deng, W. Wang, J. Zhang, and G. Zong, "Locomotion Capabilities of a Novel Reconfigurable Robot with 3 DOF Active Joints for Rugged Terrain," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5588–5593, IEEE, 2006.

[48] W. Wang, W. Yu, and H. Zhang, "JL-2: A Mobile Multi-robot System with Docking and Manipulating Capabilities," *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 9, 2010.

[49] K. Malu, J. Majumdar, and Sandeep, "Kinematics, localization and control of differential drive mobile robot," *Global Journal of Research In Engineering*, vol. 14, 2014.

[50] "Lyapunov stability," Accessed on: Nov. 3, 2020. [Online], Available: https://en.wikipedia.org/wiki/Lyapunov_stability.

[51] F. H. Clarke, Y. S. Ledyaev, and R. J. Stern, "Asymptotic Stability and Smooth Lyapunov Functions," *Journal of Differential Equations*, vol. 149, no. 1, pp. 69–114, 1998.

[52] J. Hespanha, "Internal or Lyapunov stability," *Linear Systems Theory*, pp. 87–107, 2018.

[53] S. Gunnar, "Lyapunov functions and stability problems Definitions and main theorems," pp. 1–11, 2013.

[54] Z. Li and S. S. Sastry, "Lyapunov stability theory," *Extensions of Linear-Quadratic Control Theory*, pp. 39–59, 2005.

[55] A. De Luca, G. Oriolo, and M. Vendittelli, "Control of Wheeled Mobile Robots: An Experimental Overview," pp. 181–226, 2001.

[56] P. M. Moubarak, E. J. Alvarez, and P. Ben-Tzvi, "Reconfiguring a modular robot into a humanoid formation: A multi-body dynamic perspective on motion scheduling for modules and their assemblies," in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 687–692, 2013.

[57] P. M. Moubarak, P. Ben-Tzvi, Z. Ma, and E. J. Alvarez, "An active coupling mechanism with three modes of operation for modular mobile robotics," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5489–5494, 2013.

[58] P. Kumar, W. Saab, and P. Ben-Tzvi, "Design of a Multi-Directional Hybrid-Locomotion Modular Robot With Feedforward Stability Control," in *Volume 5B: 41st Mechanisms and Robotics Conference*, p. V05BT08A010, 2017.

[59] S. Feng, B. Sebastian, and P. Ben-Tzvi, "A Collision Avoidance Method Based on Deep Reinforcement Learning," *Robotics*, vol. 10, no. 2, p. 73, 2021.

[60] B. Sebastian and P. Ben-Tzvi, "Physics Based Path Planning for Autonomous Tracked Vehicle in Challenging Terrain," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 95, no. 2, pp. 511–526, 2019.

[61] B. Sebastian and P. Ben-Tzvi, "Active Disturbance Rejection Control for Handling Slip in Tracked Vehicle Locomotion," *Journal of Mechanisms and Robotics*, vol. 11, no. 2, p. 021003, 2018.

[62] H. Umari and S. Mukhopadhyay, "Autonomous Robotic Exploration Based on Multiple Rapidly-exploring Randomized Trees," pp. 1396–1402, 2017.

[63] "Gazebo," Accessed on: Jan. 24, 2017. [Online], Available: http://gazebosim.org/.

[64] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1396–1402, 2017.

[65] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[66] K. Krinkin, A. Filatov, A. y. Filatov, A. Huletski, and D. Kartashov, "Evaluation of modern laser based indoor slam algorithms," in *2018 22nd Conference of Open Innovations Association (FRUCT)*, pp. 101–106, 2018.

[67] X. Zhang, J. Lai, D. Xu, H. Li, and M. Fu, "2d lidar-based slam and path planning for indoor rescue using mobile robots," *Journal of Advanced Transportation*, vol. 2020, p. 1–14, 2020.

[68] M. Filipenko and I. Afanasyev, "Comparison of various slam systems for mobile robot in an indoor environment," in *2018 International Conference on Intelligent Systems (IS)*, pp. 400–407, 2018.

[69] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar

slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[70] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[71] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-ptam: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27–42, 2017.

[72] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *CoRR*, vol. abs/1607.02565, 2016.

[73] "Zed stereo camera."

[74] G. Campion and G. Bastint, "Nonholonomic Mechanical Systems," 1991.

[75] E. Olson, "AprilTag: A robust and flexible visual fiducial system," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3400–3407, 2011.

[76] P. Moubarak and P. Ben-Tzvi, "Modular and reconfigurable mobile robotics," *Robotics and autonomous systems*, vol. 60, no. 12, pp. 1648–1663, 2012.

[77] Y. Shi, M. R. Elara, A. V. Le, V. Prabakaran, and K. L. Wood, "Path tracking control of self-reconfigurable robot htetro with four differential drive units," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3998–4005, 2020.

[78] Y. Guo, J.-h. Guo, X. Liu, A.-j. Li, and C.-q. Wang, "Finite-time blended control for air-to-air missile with lateral thrusters and aerodynamic surfaces," *Aerospace Science and Technology*, vol. 97, p. 105638, 2020.

[79] Y. Zheng, Z. Chen, X. Shao, and W. Zhao, "Time-optimal guidance for intercepting moving targets by dubins vehicles," *Automatica*, vol. 128, p. 109557, 2021.

[80] F. Zhu and P. J. Antsaklis, "Optimal control of hybrid switched systems: A brief survey," *Discrete Event Dynamic Systems*, vol. 25, no. 3, pp. 345–364, 2015.

[81] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 31–45, 1998.

[82] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[83] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[84] J.-D. Boissonnat, A. Cérézo, and J. Leblond, "Shortest paths of bounded curvature in the plane," *Journal of Intelligent and Robotic Systems*, vol. 11, no. 1, pp. 5–20, 1994.

[85] H. J. Sussmann and G. Tang, "Shortest paths for the reeds-shepp car: a worked out example of the use of geometric techniques in nonlinear optimal control," *Technical Report SYNCON 91-10*, 1991.

[86] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*. New York: Interscience, 1962.

[87] D. B. Reister and F. G. Pin, "Time-optimal trajectories for mobile robots with two independently driven wheels," *The International Journal of Robotics Research*, vol. 13, no. 1, pp. 38–54, 1994.

[88] D. J. Balkcom and M. T. Mason, "Time optimal trajectories for bounded velocity differential drive vehicles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 199–217, 2002.

[89] H. Chitsaz, S. M. LaValle, D. J. Balkcom, and M. T. Mason, "Minimum wheel-rotation paths for differential-drive mobile robots," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 66–80, 2009.

[90] D. J. Balkcom, P. A. Kavathekar, and M. T. Mason, "Time-optimal trajectories for an omni-directional vehicle," *The International Journal of Robotics Research*, vol. 25, no. 10, pp. 985–999, 2006.

[91] B. Jha, Z. Chen, and T. Shima, "On shortest dubins path via a circular boundary," *Automatica*, vol. 121, p. 109192, 2020.

[92] K. B. Kim and B. K. Kim, "Minimum-time trajectory for three-wheeled omnidirectional mobile robots following a bounded-curvature path with a referenced heading profile," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 800–808, 2011.

[93] P. Soueres and J.-D. Boissonnat, "Optimal trajectories for nonholonomic mobile robots," in *Robot motion planning and control*, pp. 93–170, Springer, 1998.

[94] S. C. Bengea and R. A. DeCarlo, "Optimal control of switching systems," *automatica*, vol. 41, no. 1, pp. 11–27, 2005.

[95] X. Xu and P. J. Antsaklis, "Optimal control of switched systems based on parameterization of the switching instants," *IEEE transactions on automatic control*, vol. 49, no. 1, pp. 2–16, 2004.

[96] L. Cesari, *Optimization—Theory and Applications: Problems with Ordinary Differential Equations*. Berlin: Springer Science & Business Media, 1983.

[97] R. Goebel, "Existence of optimal controls on hybrid time domains," *Nonlinear Analysis: Hybrid Systems*, vol. 31, pp. 153–165, 2019.

[98] H. G. Bock, C. Kirches, A. Meyer, and A. Potschka, "Numerical solution of optimal control problems with explicit and implicit switches," *Optimization Methods and Software*, vol. 33, no. 3, pp. 450–474, 2018.

[99] F. Zhu and P. J. Antsaklis, "Optimal control of hybrid switched systems: A brief survey," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 25, no. 3, pp. 345–364, 2015.

[100] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[101] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.