Improving Robot Capabilities Through Reconfigurability

Sha Yi

CMU-RI-TR-24-16

May 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Katia Sycara, *Co-chair*
Zeynep Temel, *Co-chair*
Aaron Johnson
Nikolaus Correll, *University of Colorado at Boulder*

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

To my parents, my partner, and my cat.

Abstract

Advancements in robot capabilities are often achieved through integrating more hardware components. These hardware additions often lead to systems with high power consumption, fragility, and difficulties in control and maintenance. However, is this approach the only path to enhancing robot functionality?

In this thesis, I introduce the PuzzleBots, a modular multi-robot system with passive mechanisms. Leveraging the inherent agility of individual locomotion, robots can collaborate to assemble into functional structures, reconfigure, and adapt to different environments. We show that we can enhance the physical capabilities of robot systems without significantly complicating the hardware design. We first utilize the environment's structural features and forces. By using gravity as an activation force, we can implement passive mechanisms as connections between robots, without the need for additional power. By incorporating compliance within the robot assembly to improve traction, coupled robots can navigate challenging terrains more effectively. We then introduce our modular multi-robot systems, where the collective performance surpasses the capabilities of any single robot. By employing gravity as an activation force, we utilize passive mechanisms as connections between robots, without the need for additional power. Furthermore, we incorporate compliance within the robot assembly to improve traction, enabling coupled robots to navigate challenging terrains more effectively. We also utilize heterogeneity by combining different types of robots, where each one of them has its own strengths and weaknesses. Thirdly, we present our distributed model predictive control framework, which facilitates precise, real-time control over this highly constrained multi-robot system.

In summary, by utilizing the environment, coordinating an assembly of multiple robots, and controlling them efficiently, we can improve robot capabilities without complicating the hardware. We show the potential for simpler and more sustainable robot designs by showcasing the effectiveness of the PuzzleBot system, which uses fewer active components. I hope to encourage future works about the use of passive mechanisms and simple shapes to create efficient and functional robots.

Acknowledgments

I am most grateful to my advisors Katia Sycara and Zeynep Temel for helping my to start this project out of nowhere at the start of my PhD. I am also very thankful to Katia as a mentor during my Master's study and for her effort in introducing me to research when I just came to CMU. Without her help and guidance, I might not have chosen to pursue an academic career. When I started this project coming from a planning and control background as a first-year PhD, Zeynep was extremely supportive and guided me in designing and making robots. The pandemic had a significant influence on research related to hardware, or projects that need lab machines and equipment. Zeynep and Katia managed to coordinate with the university and helped me to get access to equipment to keep research going, while also keeping me safe during the time. To fully support this new project I started, Katia and Zeynep also nominated and supported me for the CMU presidential fellowship. Without their support inside and outside of research, this thesis would not be possible.

I am also thankful to my thesis committee members - Aaron Johnson and Nikolaus Correll, for their time and feedback on my thesis. Outside of the thesis, their support and suggestions for my career are invaluable. Both of their works have also been extremely inspiring to my current and future research directions.

I would also like to express my gratitude to my collaborator Ryan St. Pierre who provided help in designing experiments and making our robots more robust. And also Andreas Kolling for hosting my internship at Amazon Robotics.

Many thanks to CMU faculty members who inspired and supported me in different ways - Zac Manchester, John Dolan, Micheal Kaess, and Chris Atkeson.

I am also extremely grateful to my former and current lab mates at the AART lab and Zoomlab: Wenhao, Changjoo, Jaskaran, Sasanka, Vignesh, Akshat, Sumit, Yifan, Fan, Yunfei, Sophie, Ankur, Dana, Simon, Joe, Yunfei, Xueting, AJ, Huao, Micheal, Zilin, Sarvesh, Spensor, Shashwat, Chris, Sidney, Jennifer, Erin, James, Avi, Pragna, Chiheb, David.

I would also like to thank Rachel, John, and Howie for having me at the

RI Summer Scholar program during my undergrad. Without this valuable opportunity, I would not have had the chance to come to CMU.

Also many thanks to the staff and engineers in RI for supporting me in the manufacturing and administrative process: Suzanne, BJ, Kaitlyn, Ashlyn, Carloz, Keyla, and Tim from the machine shop.

Last but not least, thanks to all my friends I met in Pittsburgh for supporting me inside and outside of research. All the great hot pot nights and Ascend climbs had made my PhD life more fun. My family, partner, and my cat for supporting me along the way.

Contents

List of Figures

Chapter 1

Introduction

Commercial robotics has made significant advancements in both capabilities and agility. For instance, we now have quadrupeds adeptly navigating unstructured terrains and humanoid robots performing backflips. Specialized robotic platforms, with different locomotion techniques and unique morphologies, unlock new capabilities that traditionally designed robots are incapable of. This enhancement in mobility and flexibility is frequently realized by introducing additional degrees of freedom to the system, necessitating the incorporation of extra actuators. Nevertheless, such novel mechanisms are challenging to model and control due to their distinctive dynamics, actuation and localization noise, or inherent under-actuation. While precise actuators and sensors with high frequency and resolution can reduce the uncertainties in controlling the system, they come at a significant energy cost that may not be necessary. For example, to achieve agile motion and superior perception capability, we can build robots with 500 Hz motors covered with full-body cameras, but do we really need this for our applications?

Our goal is to design energy-efficient novel robotic systems and develop robust control and planning strategies to react and interact with challenging and dynamic environments. This thesis demonstrates one step towards this goal through a reconfigurable robotic swarm system, the *PuzzleBots*. Inspired by the cooperative abilities of ants to form functional structures in complex environments, I have designed the first passive coupling in modular multi-robot systems and developed control frameworks for the self-assembly of functional structures in response to environments. Such struc-

tures can form bridges across gaps, traverse unstructured and challenging terrains, and perform collective tasks, powered solely by individual locomotion. Applications include disaster response, exploration in energy-constrained environments, and collaborative transport. However, creating dynamic, energy-efficient, and robust functional structures like those observed in natural systems remains challenging for robots. To facilitate these behaviors, I also developed scalable real-time control frameworks for our multi-robot system. The works spanned electronics and mechanical design, controller development, simulation modeling based on hardware experiments, and implementation on the actual robot hardware system.

Existing modular robot platforms that employ dynamic coupling and decoupling methods focus mainly on the coupling mechanism itself. These existing magnetic or active mechanical structures suffer from high energy consumption, limited mobility, and lack of flexibility in the assembly structure. Instead, the Puzzlebots utilize the power and agility from locomotion, supplemented by environmental energy sources such as gravity. This unique strategy facilitates formation without the need for additional power or electronics. Modeling and controlling passive structures pose considerable challenges, primarily due to their inherent compliance and the lack of direct access to states. The complexity escalates significantly in multi-robot systems, where the states and action space grow exponentially with the number of robots and inter-robot constraints. In this thesis, these challenges were successfully addressed, leading to the development of large-scale autonomous robot swarms. The system exhibits the capability to form dynamic assemblies, adapt and conform to environmental structures, and execute collective tasks with high efficiency and precision.

Power-efficient Passive Coupling Mechanism

Despite the wide use of magnets in facilitating coupling behaviors between robots, the robust load-bearing capability demanded by magnetic forces requires significant energy input. Similar limitations hold for active mechanical coupling mechanisms despite their relatively complicated design. Moreover, many modular robots often compromise their independent mobility to accommodate these coupling mechanisms. Passive coupling mechanisms, on the other hand, do not require additional power

to maintain the coupling status between robots. The connection between robots is formed utilizing the individual mobility of each robot module, which is notably underexplored in prior research. In our initial work [75], I designed a modular mobile robot system, the *PuzzleBots*, with a fully passive coupling mechanism, enabling the robots to couple and decouple dynamically with locomotion from the wheels. To the best of our knowledge, it is the first fully passive coupling mechanism on modular robots without magnets, enabling robots to couple and decouple dynamically. I also utilized redundancy in connections to provide additional robustness against environment disturbances [76]. A robot coupled on the side to lock the connection gives robustness and high rigidity for an assembled line configuration, which is essential for a functional structure such as a bridge. By utilizing the inherent agility provided by the robot locomotion itself, robots can form configurations with different morphologies without compromising the power allocated to mobility. I have shown that our modular system can form a chain-like or mesh-like bridge across a gap larger than the body length of a robot, and flexibly decouple to autonomously navigate the environment.

Real-Time Control for Constrained Multi-Robot Systems

The assembled structure of a multi-robot system unlocks unique capabilities beyond a single robot. Nevertheless, controlling a custom robot platform is challenging due to the uncertainty in localization and actuation. These complexities are further compounded on physically coupled robots due to the unmodeled contact dynamics and constraints. Operating such systems in real-time within unstructured environments further complicates the task. In our previous works [36, 74], we introduced a quadratic programming (QP) based approach that formulated connectivity as linear constraints. This approach was scalable to a large number of robots, enabling real-time operations and adaptability in diverse environments. In our subsequent work [76] for the PuzzleBots, I introduced a heterogeneous system of robots with varying levels of agility and ground traction. The unified framework of the *connection-pair oriented configuration control* algorithm is designed to model coupling geometry and status. I incorporated the physical coupling of connection pairs into the linear constraints of the QP-based controller. This approach allowed for real-time operations and efficient

responses to environmental structures. This reactive controller also further explored the agility of each individual robot and gave it further flexibility in changing the rigidity of the connected assembly.

Flexible Assembly and Environment Adaptation

Although PuzzleBots in [76] possess the ability to dynamically configure into various assemblies, the resultant structure remains rigid after the formation. While this proves advantageous for load-bearing applications over gaps, it constrains their mobility and renders them ineffective on different types of terrains. To broaden their adaptability to diverse landscapes, such as uneven terrains and areas with varying height drops, I have devised a flexible *soft anchor* connection mechanism [77]. The assembled structure is flexible enough to form curved configurations compliant with environment structures and rigid bridges across gaps. This soft coupling mechanism extends compliance beyond two-dimensional planar motion, which significantly enhances the adaptability of the PuzzleBots. However, the inherent complexity of soft mechanisms poses challenges in accurate modeling and precise control. To address these challenges, I iterated the design and control process by developing a simulation of the soft anchor model. I relaxed the previous restrictive constraints in [76] based on the data collected from hardware experiments, and incorporated the soft connection into linear constraints with a Model Predictive Control (MPC) framework. This enabled fast computation and yielded more optimal control inputs for our densely constrained non-holonomic multi-robot systems. We further developed a distributed MPC framework [56, 78] for real-time scalable computation. We show that the inherited compliance within a soft assembly of multiple robots provides increased traction. Multiple robots can traverse challenging terrains that a single robot is not capable of.

Chapter 2

PuzzleBots: Physical Coupling in Robot Swarms

2.1 Introduction

Collaborative swarm behaviors have been widely observed in nature. Ants have shown the ability to create functional structures like bridges by joining together and collaboratively performing tasks in a complex environment [17, 45]. Robots face similar challenges when they operate in uncertain environmental conditions. For instance, gaps and holes may block the navigation of robots, particularly those having small characteristic lengths. In such scenarios, the ability of robots can be extended using physical coupling to form a functional swarm system and continue performing the designated tasks.

Large groups of robots have been shown to improve the efficiency and robustness of task performances [36, 41, 44]. It has also been shown that physically coupled structures of modular robots [14, 71] can navigate in confined spaces and go over small gaps. In modular robots whose modules are initially coupled [71, 79], each module has limited capability to navigate around the environment. Compared to a multi-robot system where there is no physical connection between robots, coupled modular robots are also less robust to module failure, meaning that if one of the modules fails during execution, the entire system may be at risk. Most modular

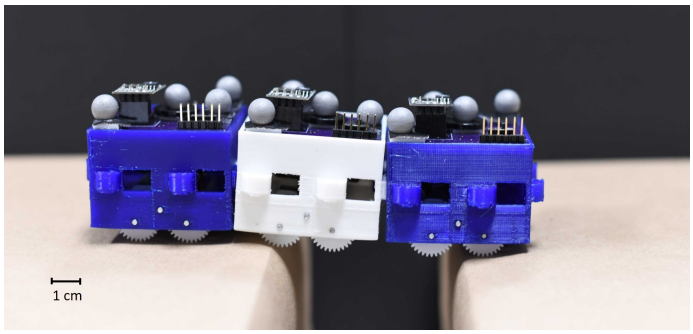


Figure 2.1: Three robots collaborate to cross a gap between two platforms.

robots address this issue by adding complicated coupling mechanisms [48, 65] that consumes additional power, which is already limited on a small module.

Based on the above limitations, our goal with the PuzzleBots system is to build a robotic swarms system where 1) individual robots can dynamically couple and decouple with each other, 2) the coupling mechanism consumes minimum energy so that the main tasks of each robot are not influenced, 3) there is sufficient mobility and controllability of each individual robot to navigate within the environment, and 4) the fabrication of the robot is easy and cost is low so that it is possible to manufacture robots in larger quantity as a swarm system, even outside of the lab environment, i.e. at the task site. We will study the coupling mechanism based on passive connections - no additional components or power involved to perform the coupling behavior. To the best of our knowledge, this is the first work that utilizes passive connections, instead of active connections [14, 48, 49, 65], to form functional structures without sacrificing the mobility of each robot and minimizing energy consumption of coupling mechanism.

The outline of the paper is as follows. In Section 2.2, we give an overview of the related works in robotic swarm systems and modular robots. Section 2.3 presents the detailed methods of our hardware and software platforms, including the mechanical design of the coupling mechanism, electronics of PuzzleBot, and the controller of the robots. In Section 2.4, we present experiments of characterizing the knobs for coupling and the results of the coupled-system performing gap-crossing motility. Finally, in Section 2.5, we conclude our results and discuss future work.

2.2 Related Works

Robotics systems that involve interaction between multiple robots are able to demonstrate broad, dynamic, and collective behaviors [5, 16, 44, 51, 52, 73]. Control algorithms have been extensively studied in swarm systems where robots do not physically interact with each other [5, 7, 41, 44, 52]. Robotic systems that actively leverage physical connections between robots lie mostly within the domain of modular self-reconfigurable robots [80]. Modular robots have shown exceptional performance in their flexibility and versatility to self-reconfigure for different tasks [47, 79]. Modular robots can be classified into two groups by the connection types between the modules. In the first group, robotic modules are connected throughout their execution, and the research focuses on controlling the configuration of the modules with respect to each other [71]. Since the connection is not detachable, the flexibility of this group of robots is limited to a single connected component, and they are not resilient to module failures during execution. The other group includes the ATRON [23], M-TRAN III [26], SlimeBot [57], Lily [19], M-blocks [49], SMORES [65], FreeBot[28], and Swarm-bot [18] in which robotic modules can couple and decouple during execution. The Lily robots rely on external actuation from the fluid to connect with each other. ATRON, M-TRAN III and M-blocks do not require external actuation and flexibly connect with other modules using magnetic forces. Each module with the ATRON and M-TRAN III systems has limited mobility on their own. While the M-blocks modules can move by flipping along one of their axes, mobility of individual modules are limited compared with a standard wheeled robots, for example [44]. SlimeBot, SMORES, FreeBot, and Swarm-bot modules are able to move around the environment independently. The SlimeBots connections are loosely couple and the main purpose is to communicate between robots, thus cannot bear any load. SMORES and FreeBot utilize magnetic forces for connection. The SMORES connection can bear the load of six robots. However, if the modules are misaligned, it can only support the weight of one module. Electromagnetic connections may also consumes high power [28] with high loads. Permanent magnets do not consumer power, but will require additional power when separating the magnets. Swarm-bot has independent grippers and complex connection mechanism, thus may not be able to carry load multiple times of its weight. Our proposed system aims to overcome the challenges of

the above examples; it consists of individual low-cost robots that can perform as a swarm system while having the capability to physically couple with each other.

2.3 Methods

The goal of PuzzleBots robotic swarm system is to demonstrate inter-robot collaboration by physically coupling with each other to form flexible, functional structures using a large number of robots. Therefore, the design considerations are as follows:

- Each robot is equipped with a coupling mechanism that enables dynamic coupling and decoupling behaviors with multiple robots.
- To accommodate each robot's task performance, the coupling mechanism should consume minimum energy during execution.
- Sufficient mobility and controllability are required so that each robot can navigate and complete tasks in the environment on its own.
- To make it financially viable to build a system with a large number of robots, the cost of each robot should be kept as low as possible.

This section will introduce our first design of the PuzzleBots prototype that fulfills the requirements mentioned above.

2.3.1 Robot Design

Figure 2.2 shows our first generation of PuzzleBots. Each robot weighs 62 g, including battery and four motion trackers to be used in the Vicon motion-tracking System[1]. A robot can carry a weight of 400 g, more than six times its own weight. Robots are equipped with on-board power, actuation, communication, and computation components. The coupling mechanism is inspired by the jigsaw puzzle. The body of the robot is 3D printed with thermoplastic polyurethane (TPU), consisting of eight knobs and holes equally distributed along the four sides of the robot, as shown in Figure 2.3. There are two hooks on the outer side of each knob, one on the top and one on the bottom. Detailed explanation about the working principle will be provided in Section 2.3.2. Each robot is 50 mm in width, 50 mm in depth, and 35

[1] https://www.vicon.com/

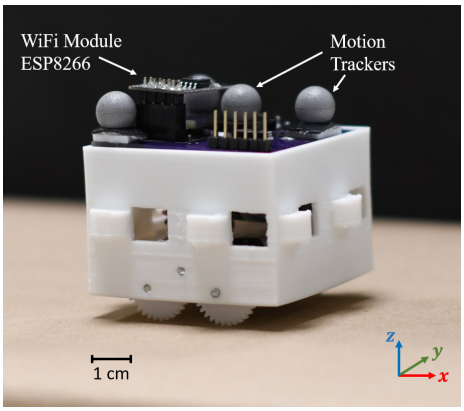


Figure 2.2: A PuzzleBot with motion trackers in Vicon system. The xyz axis of the robot body frame points front, left, and up, correspondingly.

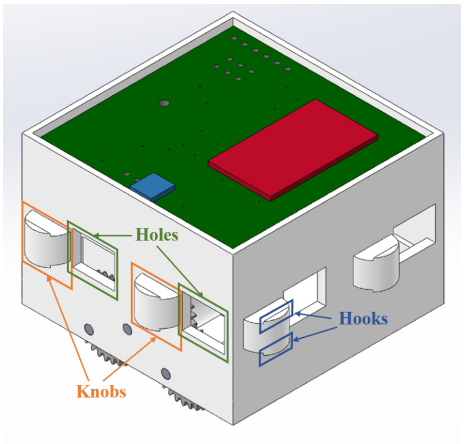


Figure 2.3: Design of the assembled robot. Each side of the robot body consists of two knobs and holes. There are hooks on top and bottom of each knob.

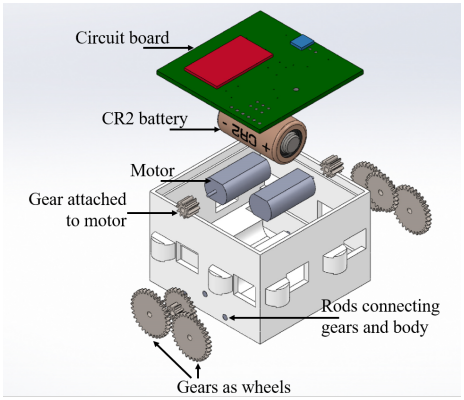


Figure 2.4: An exploded view of the PuzzleBot with mechanical and electrical parts.

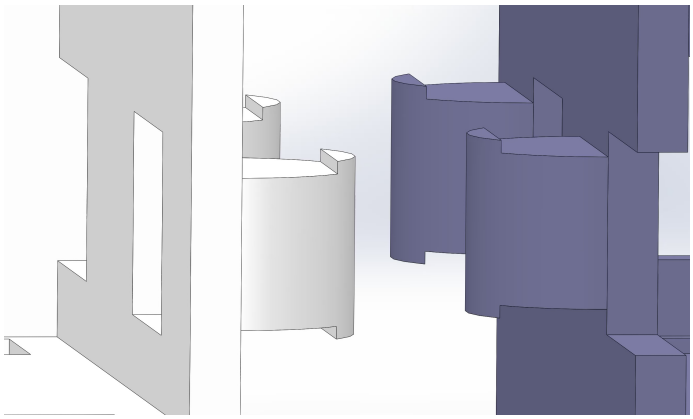


Figure 2.5: Coupling mechanism between two robots (white and purple): The initial state where two robots are separated from each other.

mm in height, excluding the knobs. The mechanical and electrical components are placed inside the robot as seen in Figure 2.4, where each component will be explained in Section 2.3.3 and Section 2.3.4.

As cost and time play vital roles in building systems with a large number of robots. We limit the cost by choosing commercially available parts. Each robot costs around US$33.8, including - printing cost of the body (US$3), CR2 battery (US$2.45), two DC motors (US$3.64 each), ESP8266 WiFi Module (US$6.95), gears and rods (US$1.6), double-sided circuit board (US$5.7), and all other on-board electronics (approximate US$6.8). Prices of small parts are computed based on purchasing quantity of 10-15 since purchasing in bulk may reduce the price. The time for 3D printing a robot chassis takes 4 hours and the assembly time for each robot takes approximately 30 minutes.

2.3.2 Coupling Mechanism Design

The body of each robot is 3D printed with NinjaFlex Cheetah TPU with Tensile Modulus 26Mpa. The knobs and holes shown in Figure 2.3 are printed with the body as a single structure. The coupling mechanism works as the knobs on one piece can fit in the hole of the other puzzle piece. Similarly, the knobs on the robot body are designed to fit in the hole of other robots. As shown in Figure 2.5, initially, the two robots are separated from each other. The figures are zoomed in to focus on the

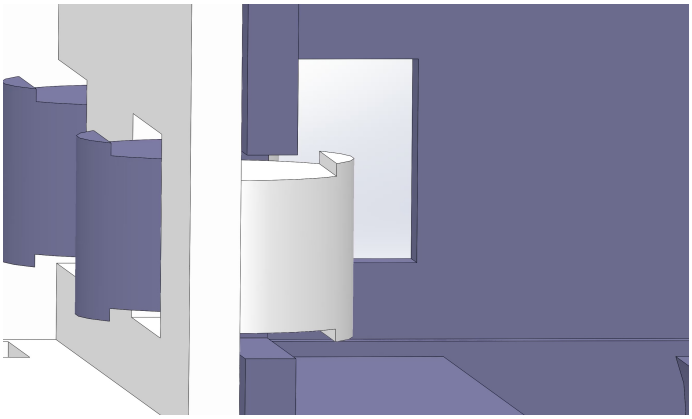


Figure 2.6: The purple robot can insert its knobs into the holes of the white robot without additional force.

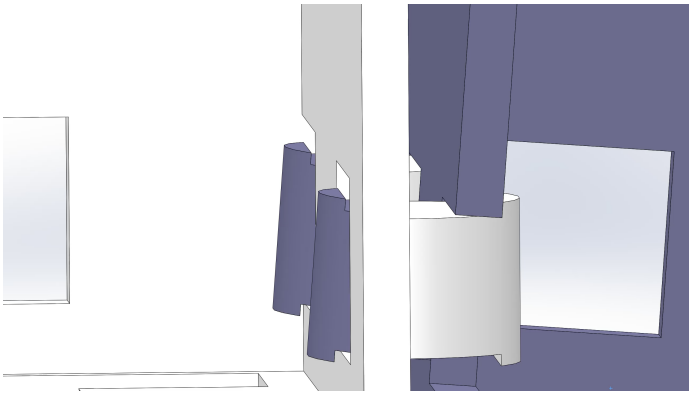


Figure 2.7: When the purple robot tilts with gravity when it is coming to a gap, the coupling mechanism is activated with hooks on the knobs blocking the movement.

coupling mechanism for clarification. We are assuming that the two robots are on a flat surface so that the knobs and holes are aligned in the vertical direction. Two robots can move towards each other, as shown in Figure 2.6. Since the maximum height of the knob, including the top and bottom hooks, is less than the height of the hole, ideally, one robot can slide its knobs into the other robot without any additional force. During manufacturing, the flexible TPU material is used to provide a tight fit between the holes and knobs. As robots move towards a gap, the robot that first leaves the platform will tilt due to gravitational force, resulting in the configuration shown in Figure 2.7. Both friction and the top and bottom hooks will block the movement of the robot falling down the gap. This enables the robots to keep moving to cross the gap towards the other platform, forming a bridge during the process.

The coupling process itself, where knobs are inserted into the holes, does not consume any extra energy other than the energy needed for actuating the robot movement. With this passive coupling mechanism, we can minimize the energy used for coupling compared with other active methods to maximize individual task performances. The connection, once the knobs are fully inserted, are able to hold the weight of 389 robots. The characterization of the knobs used for coupling will be discussed in Section 2.4.1.

2.3.3 Electronics

The electronics design accommodates the requirements introduced in Section 2.3.1 of on-board power, actuation, communication, and computation. As shown in Figure **??**, the circuit board is printed double-sided with a WiFi module, on-off switch, programming pins on the top, and all other components on the bottom. The whole circuit is powered by a 3V CR2 battery, with a capacity of 850mAh. The battery is available in both rechargeable and non-rechargeable versions. We are using the non-rechargeable ones in the paper for simplicity. The microcontroller unit (MCU) is an 8-bits ATMega328P that operates between 1.8V to 5.5V with an 8M oscillator. It consists of six Pulse-width modulation (PWM) channels, which is convenient to control the motors. The robot communicates with an external computer via the ESP8266 WiFi module. ESP8266 is a low-cost, open-source, small (14.4×24.7 mm) WiFi module that supports standard TCP/IP protocol and 2.4GHz WiFi connection

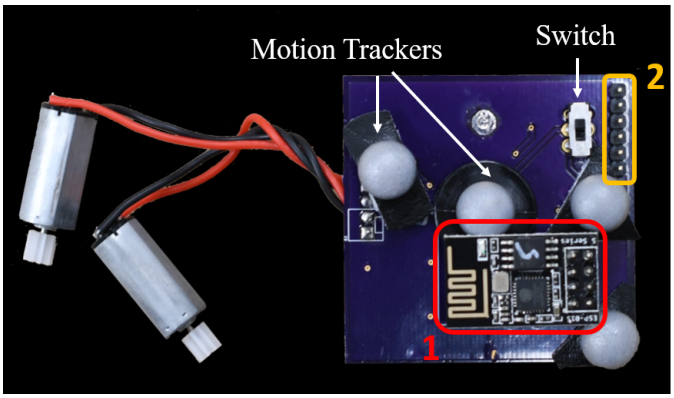


Figure 2.8: Top view of the circuit board: 1) WiFi module ESP8266, 2) Programming pins.

while consumes 215 mA current with maximum usage. The integrated AT command interface enables easy communication with the microcontroller. Officially it operates at 3.3V, however with our experiments, it is able to operate normally with voltage as low as 2.8V, thus fitting in our system with the 3V battery supply. The DC motors operate at a voltage from 1V to 4V. These motors can provide a torque of 0.9 mNm with 370 mA current. We can control the velocity of the motor via PWM. The DRV8833 dual H-bridge is a motor driver that has four PWM inputs and four outputs. Each of the PWM inputs is connected with the corresponding MCU PWM output pins, and the four outputs are connected with the two motors (two inputs on each motor). This enables us to control the direction of the current going through the motor, enabling the motors to rotate forward and backward. The DRV8833 operates between 2.7V to 10.8V, with a peak current of 1A per H-bridge.

2.3.4 Mechanical Structures and Controls

We are able to control the velocity of the left and right sets of gears by providing pulse-width modulation (PWM) signals to the two motors. In this section, we will present the methods of controlling the velocity of the robots within our designed mechanical structures.

As shown in Figure 2.10, each motor controls one side of the gear sets. The gear g_1 is attached to the motor. The gears g_2 and g_3 form a set of double reduction gears. The two side gears are identical, both referred to as g_4; they also serve as wheels

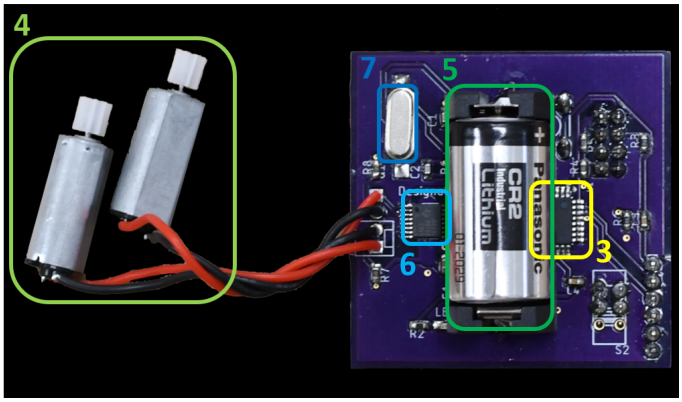


Figure 2.9: Bottom view of the circuit board: 3) Microcontroller unit ATMega328P, 4) DC Motors, 5) CR2 battery, 6) DRV8833 dual H-bridge motor driver, 7) 8M oscillator.

for the robot. This enables larger friction between the ground, and also simplifies the design. We denote the linear velocity of gear g_i, $i = \{1, 2, 3, 4\}$, as v_i, angular velocity as ω_i, the number of teeth as z_i, and reference diameter as d_i. All gears have the same module coefficient $M = \frac{d_i}{z_i}$. With the configuration in Figure 2.10, we have $v_1 = v_2$, $\omega_2 = \omega_3$, $v_3 = v_4$, and $\frac{v_i}{\omega_i} = \frac{d_i}{2}$. Since PWM signal controls the angular velocity of the motor ω_1, $v_4 = \omega_1 M \frac{z_1 z_3}{2 z_2}$.

Two side gears on each side are identical, and each side is controlled by an individual motor. For simplicity, we model our robot as a differential drive model. Recall that in Figure 2.2, the forward direction of the robot is aligned with x axis. Thus, we provide forward velocity v_x and angular velocity ω via WiFi to the robot. With differential drive model, we are able to calculate the velocity needed on the left v_l and right v_r as [27]

$$v_r = \frac{2v_x + \omega L}{2R}, \; v_l = \frac{2v_x - \omega L}{2R} \tag{2.1}$$

where L is the length between the wheels and R is the radius of the wheels, i.e. gear g_4. The output PWM signal from the MCU controls the rotational speed of each individual motor denoting as ω_r on the right and ω_l on the left. By combining the equations of the gear sets and the differential drive, we have

$$\omega_r = \frac{z_2(2v_x + \omega L)}{M z_1 z_3 R}, \; \omega_l = \frac{z_2(2v_x - \omega L)}{M z_1 z_3 R} \tag{2.2}$$

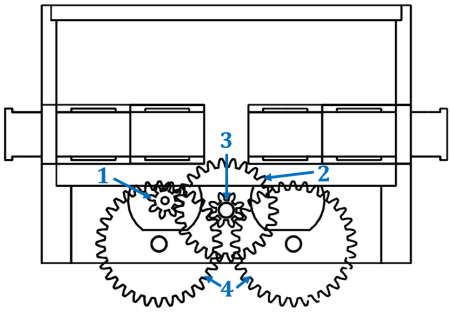


Figure 2.10: Side view of the robot with double reduction gear. The first pair consists of gear g_1 and gear g_2 where gear g_1 is attached to the motor and g_2 is on the center rod. The second part of the double reduction gear consists of g_3 and g_4, where g_3 is attached to g_2. The two identical side gears are both referred to as g_4. The two g_4 also serve as wheels.

In our design, we have $z_1 = 8,\ z_2 = 26,\ z_3 = 8,\ M = 0.5,\ L = 40\ mm, R = 17\ mm$. During implementation, by substituting these values, we are able to control the robot accordingly.

2.3.5 Swarms Coupling Behaviors

The algorithms used for the robots to couple and decouple are one-dimensional rendezvous and anti-rendezvous behaviors for swarms. Rendezvous swarms behavior is a consensus algorithm where each robot communicates with its neighbors to move towards a direction that will eventually gather everyone together [40, 41]. Consider our system of N robots on a one-dimensional line, we denote the position of robot i as $x_i \in \mathbb{R}$ with control input $\dot{x}_i = u_i$, where $i = \{1, \ldots, N\}$. In our system setup, all robots are able to communicate with each other via a central computer. Therefore, we can simplify the rendezvous controller as

$$\dot{x}_i = \frac{1}{N} \sum_{j \neq i} (x_j - x_i) \tag{2.3}$$

As a result, the robots will move towards each other until they are physically coupled. Once they are successfully coupled ($\min ||x_i - x_j|| = \text{robot body length}, \forall i, j \in 1, \ldots, N, i \neq j$), they can perform other behaviors as one connected component.

Similarly, the robots are able to decouple with each other via 1D anti-rendezvous:

$$\dot{x}_i = -\frac{1}{N}\sum_{j \neq i}(x_j - x_i) \tag{2.4}$$

As a result, the robots will move to decouple with each other and perform individual tasks later on.

However, due to actuation uncertainty, the robots might not stay precisely aligned during this 1D rendezvous behavior. In actual experiments, we utilize the environment to reduce these uncertainties, e.g., having one robot stay against a wall.

2.4 Experiments

We characterized the coupling knobs for maximum performance of the coupling mechanism. We then performed experiments of up to nine robots for the gap-crossing behavior with different environmental parameters. We also present frames from the video where robots couple, cross a gap, decouple, and visit individual goals.

2.4.1 Characterizing the Coupling Knobs

The knobs and holes are the core part of the coupling mechanism, and the dimensions determine the performance of connections. It is ideal to have the height of the knobs (including the hooks) to match precisely the height of the holes. However, 3D printed surfaces, especially surfaces that need support underneath, are generally not smooth. Although TPU is a flexible material, having a tight fit will require more torque from the actuators. Thus, by experimenting with different parameters, we design the height of the knobs to be 1mm less than that of the holes.

The size of the hook on the knob is also an essential parameter for coupling. Two possible coupling status with different hook width is shown in Figure 2.11 and 2.12. In Figure 2.11, the hook width is 1 mm and the hooks can lock the movement of the robot on the right when it is tilted. When the robots are moving towards the gap, the larger the tilting angle, the lower the front of the robot will be, and the smaller the gap they will be able to cross. The larger the hook width gives a smaller tilting angle, and thus better gap-crossing performance. However, as shown in Figure 2.12,

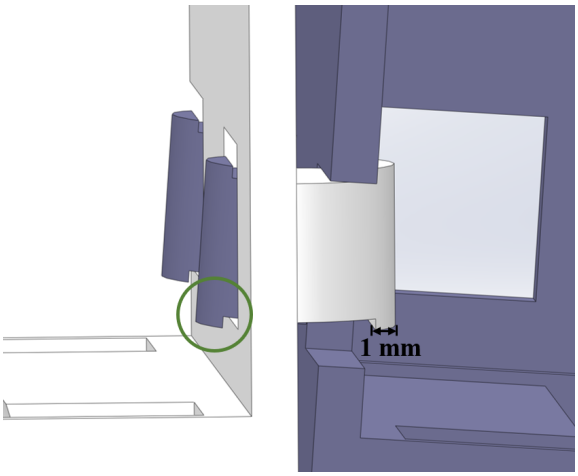


Figure 2.11: A successful coupling example with hook width 1mm when the robot is tilted due to gravity. The hooks are able to block the movement of the robot's body.

when the hook width is too large (still smaller than *knob radius* − *wall thickness*), the hooks may fail to lock the movement.

As shown in Figure 2.13, we measure the tilting angles of robots when they move towards a gap. The experiments are done with three robots, and the measurement is performed when the second robot is just about to leave the platform. This is the moment when the tilting angles are the largest. Due to gravity and the coupling, the remaining two will also tilt when the first robot tilts. We measure their angles θ_1, θ_2, θ_3 with respect to the horizontal plane. The 1 mm hook width gives the largest angle, while the 1.5 mm hook width gives the smallest tilting angle. Therefore, in the remaining experiments, all robot knobs have a 1.5 mm hook for maximum performance.

2.4.2 System Experiments

Gap-crossing Performance

The length of each robot is 50 mm. We analyze the performances of the gap-crossing behavior with different variables: number of robots (1, 2, 3, 6, 9), length of the gap (10 mm to 100 mm), heading angle (0° to 50°), height difference between the two platforms (0 mm, 6 mm, 12 mm, the starting platform is higher than the target platform). We perform five runs for each combination and recorded the number of

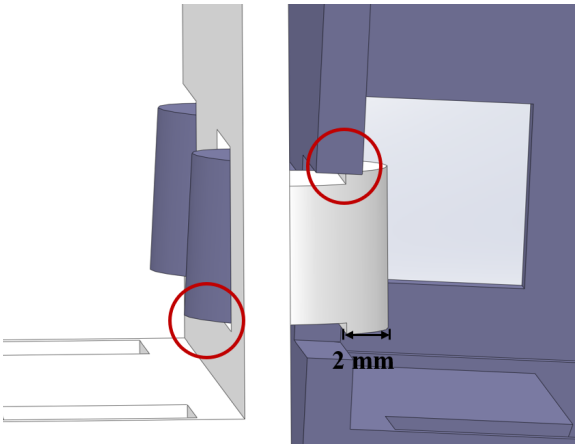


Figure 2.12: An unsuccessful coupling example with hook width 2 mm. Since the hook width is too large, it relies only on friction between surfaces, not the hooks, to block the movement of the robot body.

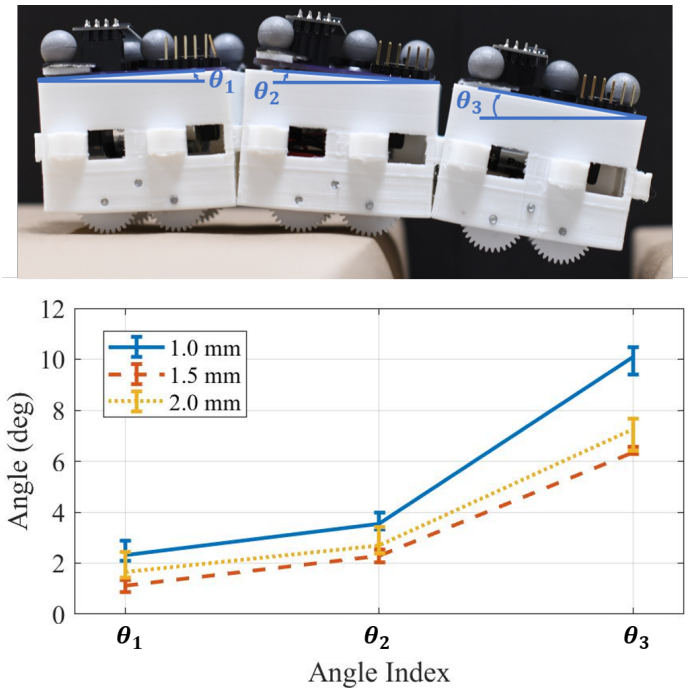


Figure 2.13: The tilting angles of robots, with different sets of hook width (1.0, 1.5, 2.0 mm) when they move towards a gap. The measurement is performed when the whole assembly is just about to move off the original platform.

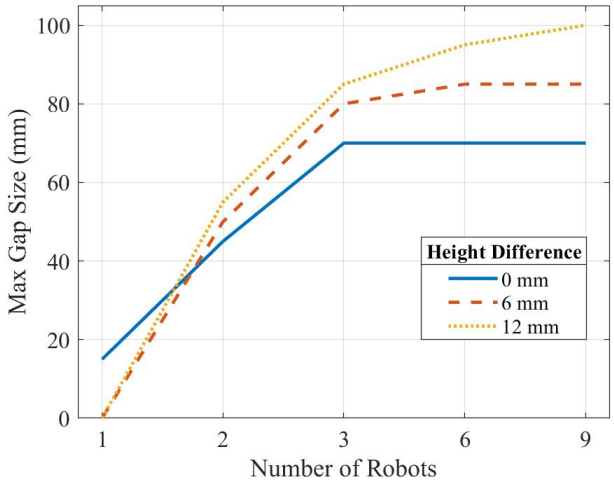


Figure 2.14: Maximum gap size that the robots are able to cross (regardless of heading angle) versus the number of robots.

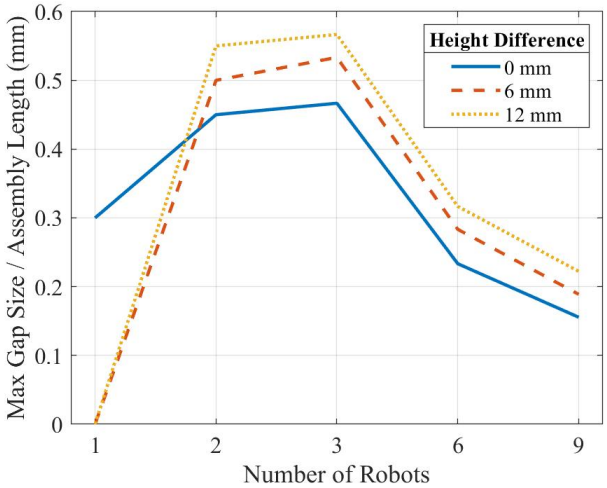


Figure 2.15: Ratio of the maximum gap size with respect to the length of the whole assembly (*robot number* $\times$ *body length*) versus the number of robots.

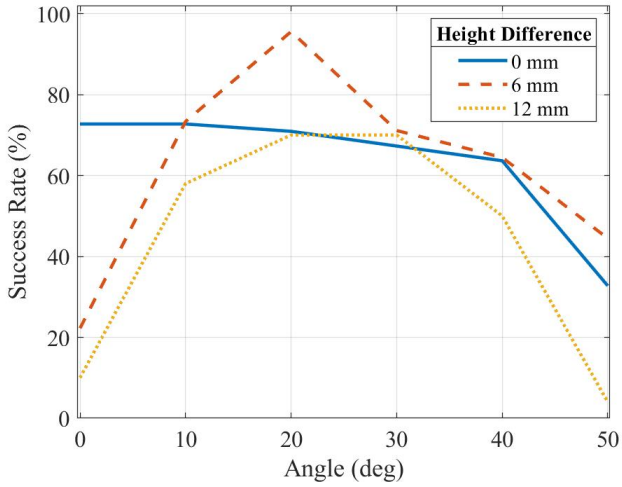


Figure 2.16: Success rate of all experiments under the same heading angle.

successful runs, i.e. all of the robots crossed the gap while staying coupled.

We present the result of the maximum gap sizes different number of robots can cross (2.14), the ratio of maximum gap size with the length of the whole robot assembly (2.15), and the relation between success rate and the heading angle of the robots (2.16). In Figure 2.14, we consider a gap size and height difference that the robots can cross when the success rate is higher than 50%. We can see that as the number of robots increases, they can cross over a larger gap. However, with a gap between the platforms of the same height, increasing the number of robots does not increase performance. The major bottleneck is with the tilting angle mentioned in Section 2.4.1. The tilting angle does not increase or decrease with the change of robot number. This bottleneck persists with a larger height difference, but the height difference in platforms compensate for the height drop in the robot assembly. This results in better performances as the robot number increases with larger platform height difference. In Figure 2.15, the ratio of the maximum gap size and the length of the assembly shows the effectiveness of increasing the number of robots. However, although the robot assembly can cross larger gaps with more robots, i.e., longer assembly length, the significance of increasing the number of robots decreases after the three robots setting. In our experiments with different heading angles, the edge of each platform aligns with the y-axis of the world frame. Thus, the robot with a heading angle of 0° is perpendicular to the platform edge. Figure 2.16 shows the

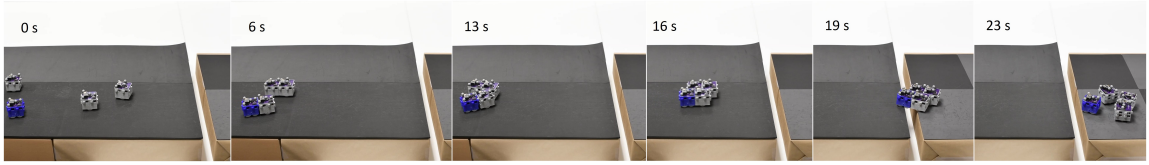


Figure 2.17: These video frame sequences show that three initially separated robots are able to couple with each other, go cross a gap (60 mm), decouple, and go to their individual goal locations (white squares). In the first frame, the robots start on platform on the right. They will need to cross the gap and reach the left platform. The blue arrow shows their goal direction, and yellow arrows show their current direction of motion of each robot.

success rate of all number of robots, given the specific heading angles. Failure cases include 1) robots fail to proceed once reaching the opposite platform due to tilting angle; 2) connections broke due to impact when robots reach the other platform; 3) robot tumbled when reaching the other platform (only for single robot case). With the same height platforms, the success rate gradually drops as the angle increases because of the traversing distance across the gap increases. However, with a larger height difference, angles around 20° give better performances. Since the whole robot assembly will only fall off when the center of mass of the assembly leaves the original platform, having an angle with the platform edge will increase the length protruding the platform. This enables the diagonal of the robots to reach the other side first. This did not give better performances with platforms of the same height because of the height drop of the robots due to the tilting angle.

Combined Behaviors

To demonstrate the ability of the PuzzleBots to assembly and disassembly autonomously, we present a sequence of video frames of our hardware system, as shown in Figure 2.17. The original video is included as a supplement. Three robots initially separated are located on the right platform. The right platform is 6 mm higher than the left. The gap between the two platforms is 60 mm wide. There is a wall on the right, aligned with the platform on the right. The left and middle robots run the 1D rendezvous controller, as described in Section 2.3.4, while the right robot runs into the wall. The robots can couple with each other against the wall. As soon as the minimum distance between robots reaches their body length, they

will move towards the gap. With the coupling mechanism, they can cross the gap and reach the other platform. All three run the anti-rendezvous controller when they have successfully crossed the gap. Once they decouple with each other, the robots will move towards their individual goal locations.

2.5 Conclusion and Discussion

In this paper, we have introduced the PuzzleBots, a robotic swarm system where robots can couple with each other to form functional structures without additional energy for coupling, while maintaining individual mobility for completing different tasks. We utilize knobs and holes on the robot body to perform the coupling mechanism. We show with hardware experiments that the robots can cross gaps approximately half the size of the whole assembly and can couple and decouple autonomously based on task requirements.

While we show that the robots can couple in the front and back, the first step of our future work tries to realize horizontal coupling mechanism. Robots coupling in the left and right to form a mesh-like structure may further extend the performance of the gap-crossing behavior. Although we have trials that show the possibility of pushing into each other from the side to couple, controlling this behavior is currently under investigation. Note that this is the first version of Puzzlebots. Future development may include on-board sensors, rechargeable batteries, improved wheel design, and a further decentralized system without a central computer to bring the system to real-world applications.

Furthermore, our work utilizes the environment to reduce actuation uncertainty during execution. However, a systematic way of when and how to best make use of the environment remains an open question. Additionally, we are also interested in the 3D coupling, where more flexibility can be introduced to form 3D structures like ropes or nets. We hope this contribution can provide benefits to robotic applications in uncertain and complex environments.

Chapter 3

Configuration Control for Physical Coupling of Heterogeneous Robot Swarms

3.1 Introduction

In unstructured environments, ants form and adapt functional structures dynamically in response to obstacles, gaps, and holes [45]. Inspired by these animals, robot swarms perform collective behaviors and accomplish complex tasks that a single robot is not capable of. In this paper, we build on our previous work that introduced PuzzleBots [75] - a reconfigurable robot swarm system with a passive coupling mechanism, and present extensions on structural enhancement configuration control.

Existing robot swarms, or Multi-Robot System (MRS) platforms [44, 51] have demonstrated collective and decentralized collaboration, but the robots do not physically interact with each other - physical abilities of the robots remain the same as a single robot. In the modular robot systems, individual units are equipped with active mechanical structures to couple and form various structures [18, 19, 28, 48, 65]. The most common method for dynamic coupling is performed by magnetic forces [28, 48, 55, 65]. This may consume high energy during the coupling or decoupling process, and also has limited load-carrying capabilities. In addition, the magnets are

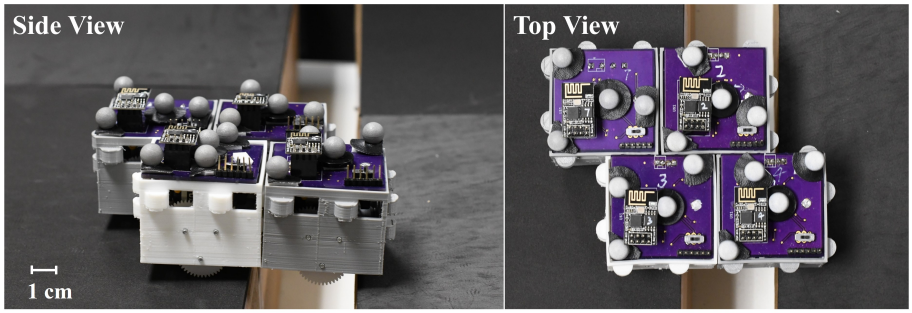


Figure 3.1: Four robots form a mesh configuration to cross a gap between two platforms.

directional, limiting the formation of the robots and introducing complexity in controlling and planning algorithms. In most modular robot systems, each connection is connected via a single contact point/face. Single connection is more fragile compared with multiple connections when encountering complex environments [35]. Reconfiguration algorithms for pre-connected modular robot systems focus on graph topology reconfiguration [20, 30, 32]. Modular robots that have limited mobility reconfigure based on motion primitives [67] or grid-based setup [11, 53]. These methods restrict the formation and are ineffective with systems where robots have individual mobility and no strong connection with each other.

In PuzzleBots [75], we presented a passive coupling mechanism where each robot has knobs and holes. PuzzleBots can couple with each other by pushing the knobs into the holes of the other robot when initially aligned, cross a gap, and decouple to perform individual tasks. The passive coupling mechanism does not consume any additional power compared to active coupling mechanisms. While the initial design with four active wheels helps the robots climb onto a platform, it limits their mobility to perform precise motions. In this paper, we focus on the following challenges: 1) improve the existing hardware platform to achieve precise motion while maintaining the gap-crossing capability, 2) reduce fragility of single-connection assembly, 3) develop a planning and control algorithm to achieve a given configuration when robots do not start from aligned positions.

We provide solutions to the challenges mentioned above by a heterogeneous robot swarm system and a connection-pair-oriented configuration control algorithm. In this paper, we assume the target configuration is given by the user and the goal

of the robots is to align and form this predefined configuration. To improve the gap-crossing performance, we introduce a heterogeneous system containing pilot robots and non-pilot robots. Pilot robots have the same design as [75] that helps climbing onto platforms. Non-pilot robots have caster wheels that enables precise control in both linear velocity and angular velocity. This heterogeneous system utilize the advantages of both designs while minimizing the drawbacks. Based on this, we propose a *connection-pair oriented configuration control algorithm* with which robots can form given configurations from unaligned positions. Due to the passive coupling mechanism, the connection between robots in [75] can be fragile and sensitive to disturbances. Borrowing the k-connectivity concept from graph theory, we introduce the mesh assembly shown in Figure 3.1, where robots can couple in two dimensions to strengthen the connection pairs formed in one direction. Experiments show that the mesh configuration helps maintain a stable assembly formation and increases the strength of the connection.

The outline of the paper is as follows. First, we discuss the problem setup of what the robots are expected to achieve in Section 3.2. Then, in Section 3.3, we present our connection-pair-oriented configuration control algorithm that drives the robots to a given coupled configuration. Next, in Section 3.4, we present our heterogeneous system consisting of pilot robots and non-pilot robots. Finally, experiments of up to nine robots with line and mesh formation to cross gaps of different sizes, as well as calibration and a behavior sequence demonstration, are presented in Section 3.5.

3.2 Problem Formulation

Consider a set of N robots on a 2D plane. Denote the poses of the robots as $\mathbf{p} = [p_1, p_2, \ldots, p_N] \in \mathbb{R}^{3\times N}$, where each robot pose consists of its coordinates in x and y axis, and its heading angle, i.e. $p_i = [x_i, y_i, \theta_i]$. The control of the robots is based on unicycle model where the control input $\mathbf{u} \in \mathbb{R}^{2\times N}$ consists of linear velocities and angular velocities, i.e. $u_i = [v_i, \omega_i]$. The transition equation [27] is defined as

$$\dot{p_i} = \begin{bmatrix} \dot{x_i} \\ \dot{y_i} \\ \dot{\theta_i} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = J_i \cdot u_i \ . \tag{3.1}$$

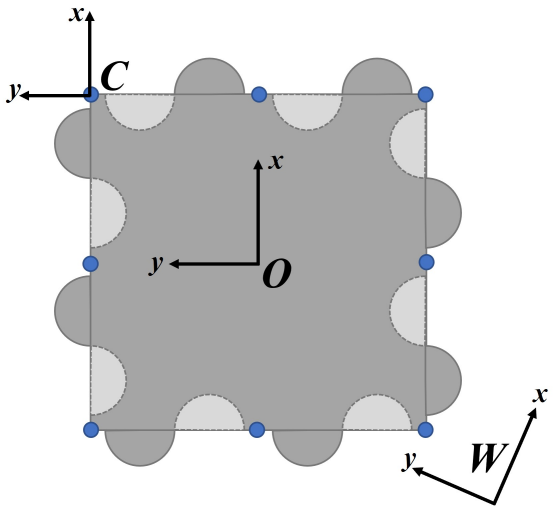


Figure 3.2: PuzzleBot with eight connection points shown as blue dots and the robot frame (O), world frame (W), and connection point frame (C).

Our heterogeneous system consists of two type of robots - pilot and non-pilot robots. The two type of robots are different in wheel design but follow the same dynamics in Equation (5.1). Details will be introduced in Section 3.4. All robots have the same body with two knobs and two holes on each side to provide passive coupling behavior [75]. The knobs on one robot can be inserted into the holes of another robot to couple. To parametrize each coupling pair, we define eight connection points on the robot body as shown in Figure 3.2. Define the connection point set of robot i as $\mathcal{C}_i$. Robot i and robot j are coupled when $\mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset$, forming one or more *connection pairs*. Each connection pair uniquely defines a coupling configuration, while each coupling configuration may have multiple connection pairs, as shown in Figure 3.3. Define an *assembly* as a group of successfully coupled robots. An assembly consists of two or more robots and the relationship of their connection pairs.

The goal configuration $\mathbf{p}_g = [p_{g_1}, p_{g_2}, \ldots, p_{g_N}] \in \mathbb{R}^{3\times N}$ consists of N relative poses on the 2D plane, i.e. $G \cdot \mathbf{p}_g^T$ is the same goal configuration as $\mathbf{p}_g$ where $G \in SE(2)$ is a rigid transformation on the 2D plane. To form functional assembly structures, robots in the goal configuration are coupled, i.e. for each robot i in $\mathbf{p}_g$, there exist a robot j whose $\mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset$. This coupling constraint on the goal configuration is particularly challenging since a simple go-to-goal controller cannot guarantee the coupling behavior or the alignment of the connection pairs. We will introduce our

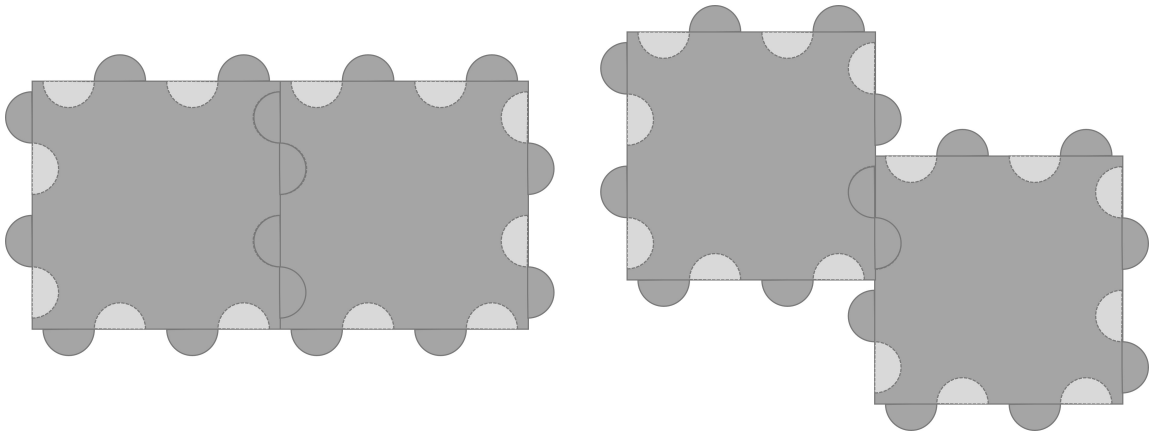


Figure 3.3: Two possible connection configurations for two robots.

connection pair oriented configuration control algorithm to solve this problem in section 3.3.

3.3 Methodology

In this section, we will present three aspects of the configuration control based on connection pairs. Section 3.3.1 introduces a PID based controller that aligns a single pair of connection points. Then the formation of a two-robot assembly, which serves as a basis for multiple connection pair alignment, is defined in section 3.2. Section 3.3.2 introduces an optimization-based control for an assembly to reach individual goals while maintaining in-assembly connection pairs. The last section 3.3.3 presents our connection pair based configuration control algorithm. Heterogeneity of the system will be discussed in section 3.4.

3.3.1 Single Connection Pair Alignment

As shown in Figure 3.2, the robot body frame is defined as O, and the contact frame of a connection point is defined as C. The O frame and C frame have the same orientation. Note that the C frame is a general representation of a contact frame but not a specific connection point on a robot. Given a target connection pair alignment C_i and C_j of the robots i and j respectively, where $i, j = 1, 2, \ldots, N$, denote the position of C_i in x and y axis in world frame W as $[c_{x_i}, c_{y_i}]$. The heading angle of C_i

is θ_i, aligned with the robot frame. For robot i, the controller for connection pair alignment is

$$u_i = J_i^+ \begin{bmatrix} \Delta c_x \\ \Delta c_y \\ \Delta\theta \end{bmatrix} = J_i^+ \begin{bmatrix} c_{x_j} - c_{x_i} \\ c_{y_j} - c_{y_i} \\ \arctan \frac{\Delta c_y}{\Delta c_x} - \theta_i + \theta_{bias} \end{bmatrix}, \tag{3.2}$$

where J_i^+ is the pseudo-inverse of J_i in Equation (5.1). $\Delta\theta$ is wrapped into $(-\frac{\pi}{2}, \frac{\pi}{2}]$. θ_{bias} is an angle bias added when the connection pair will lead the robots to align side-by-side. This will help the robot to push its knob into the hole of the other robot. Similarly, u_j is computed accordingly as in Equation (3.2). Note that not any given set of connection pair can be aligned due to local minima with this Jacobian pseudo-inverse method. For example, for the two robots in Figure 3.3, it will be infeasible if the connection point on the left robot is on the left side of the robot body. In our setting, we assume robots only start from feasible positions with respect to a given connection pair.

3.3.2 Connection Pair Maintenance

Once one or more connection pairs are aligned in the system, we study the motion of an assembly - how to reach another configuration while maintaining current connection pairs within an assembly. Consider a given target control input $\hat{\mathbf{u}}$ for an assembly. The assembly consists of one or multiple connection pair(s) already aligned. We aim to find the control $\mathbf{u}$ that minimizes $||\mathbf{u} - \hat{\mathbf{u}}||^2$ while maintaining the connection pairs within the assembly. We formulate this problem as a quadratic programming (QP) problem with linear constraints. Define the homogeneous transformation from the world frame W to the contact frame C of robot i to be g_{wc}. We have $g_{wc} = g_{wo}g_{oc}$, where g_{oc} is constant for a given connection point. Denote

$$g_{oc} = \begin{bmatrix} 1 & 0 & d_{x_c} \\ 0 & 1 & d_{y_c} \\ 0 & 0 & 1 \end{bmatrix},$$

and when applying $u = [v_i, \omega_i]$ over time Δt,

$$g_{wo} = \begin{bmatrix} \cos(\theta_i + \omega_i \Delta t) & -\sin(\theta_i + \omega_i \Delta t) & x_i + v_i \Delta t \cos\theta_i \\ \sin(\theta_i + \omega_i \Delta t) & \cos(\theta_i + \omega_i \Delta t) & y_i + v_i \Delta t \sin\theta_i \\ 0 & 0 & 1 \end{bmatrix} .$$

By calculating $g_{wc} = g_{wo} g_{oc}$ and extracting the translational component of C, the position vector $[c_{x_i}, c_{y_i}]^T$ becomes

$$\begin{bmatrix} x_i + v_i \Delta t \cos\theta_i + d_{x_c} \cos(\theta_i + \omega_i \Delta t) - d_{y_c} \sin(\theta_i + \omega_i \Delta t) \\ y_i + v_i \Delta t \sin\theta_i + d_{x_c} \sin(\theta_i + \omega_i \Delta t) + d_{y_c} \cos(\theta_i + \omega_i \Delta t) \end{bmatrix} .$$

Consider Δt as a very small value, we may perform Taylor expansion around θ_i to linearize the above equation as

$$\cos(\theta_i + \omega_i \Delta t) \approx \cos\theta_i - \omega_i \Delta t \sin\theta_i \tag{3.3}$$

$$\sin(\theta_i + \omega_i \Delta t) \approx \sin\theta_i + \omega_i \Delta t \cos\theta_i \ . \tag{3.4}$$

To simplify the notation, denote $\cos\theta_i$ as c_i and $\sin\theta_i$ as s_i, the position vector becomes

$$\begin{bmatrix} c_{x_i} \\ c_{y_i} \end{bmatrix} = \begin{bmatrix} c_i & -d_{x_c} s_i - d_{y_c} c_i \\ s_i & d_{x_c} c_i - d_{y_c} s_i \end{bmatrix} u_i \Delta t + \begin{bmatrix} x_i + d_{x_c} c_i - d_{y_c} s_i \\ y_i + d_{x_c} s_i + d_{y_c} c_i \end{bmatrix} . \tag{3.5}$$

The connection pair constraint on robot i and j is defined as

$$-\epsilon \leq \begin{bmatrix} c_{x_i} \\ c_{y_i} \end{bmatrix} - \begin{bmatrix} c_{x_j} \\ c_{y_j} \end{bmatrix} \leq \epsilon \ . \tag{3.6}$$

We may stack all the connection pair constraints in Equation (3.6) in an assembly in one equation $A\mathbf{u} \leq b$. The connection pair maintenance problem then becomes

$$\mathbf{u}^* = \arg\min ||\mathbf{u} - \hat{\mathbf{u}}||^2 \tag{3.7}$$

$$\text{s.t. } A\mathbf{u} \leq b \ . \tag{3.8}$$

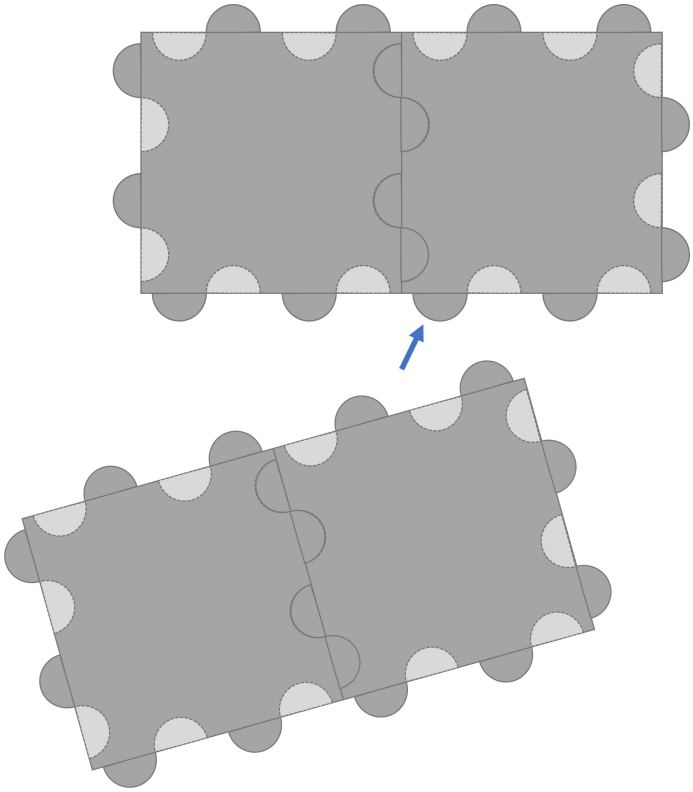


Figure 3.4: Two convex assemblies couple to form a larger assembly.

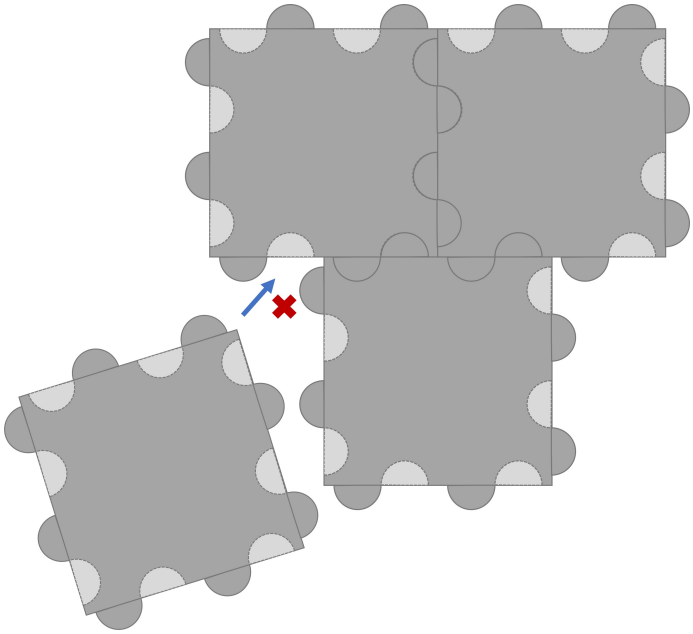


Figure 3.5: One robot cannot couple with a concave assembly.

3.3.3 Connection Pair Based Configuration Control

As defined in section 3.2, given a goal configuration $\mathbf{p}_g$, we aim to provide a solution for the robots to reach this goal configuration. During the process of assembling into the goal configuration, a sequential constraint exists - concave assemblies may not be able to assemble into one larger assembly due to the knobs blocking each other. For example, in Figure 3.4, two convex assemblies, each formed by two robots, are able to couple and form a larger assembly of four robots. However, in Figure 3.5, one robot tries to couple with a concave assembly of three robots, but the knobs block its motion. In the algorithm to be introduced in Section 3.3.3, we plan to avoid forming concave assemblies by prioritizing convex assemblies first.

Algorithm 1 Find Existing Connection Pairs

Input: $\mathbf{p}$: input poses
Output: *pair_dict*: connection pairs
Initialize: *pair_dict*={}

```
function findExistPairs(p)
    G ← constructGraph(p) based on distance
    edge_set ← getMinimumSpanningTree(G)
    for vertex i and j in edge_set do
        [C_i, C_j] ← findMinDistancePair(C_i, C_j)
        pair_dict[(i, j)] = [C_i, C_j]
    end for
    return pair_dict
end function
```

The connection pair assignment is shown in Algorithm 2. First, we align the center of the input goal configuration with the current robot poses as

$$\mathbf{p}'_g = \mathbf{p}_g - \overline{\mathbf{p}_g} + \overline{\mathbf{p}}, \tag{3.9}$$

where $\overline{\mathbf{p}}$ is the mean of $\mathbf{p}$. We then find the connection pair assignment based on the goal configuration $\mathbf{p}'_g$ as shown in Algorithm 1. In Algorithm 1, we find the connection pairs based on an existing configuration. This input robot poses $\mathbf{p}$ assumes the robots are already aligned. First, we construct a fully connected graph $\mathcal{G}$ where the vertices are the location points in $\mathbf{p}$ and the edge weights are the distances between the vertices. We then find the Minimum Spanning Tree (MST) based on this

distance-induced graph. This provides information on the minimum connection pairs to monitor in order to maintain the current configuration $\mathbf{p}$. Then for each edge, we loop through all combinations of connection points on the two vertices of this edge to find the one with minimum distance. With this information in Algorithm 2, we obtain the connection pairs needed to form the goal configuration $\mathbf{p}_g$. The pairs are then sorted based on the distance between the robot poses in $\mathbf{p}_g$. Notice that the concave assembly is only formed with mesh configuration, and line formation can only form a convex assembly. Two robots in mesh configuration always have a distance of $\sqrt{2}$ times the body length between each other, which is larger than that of the distance in the line configuration. By sorting the distance between robot poses, we can guarantee that convex assemblies are always formed before the concave assemblies. We then calculate the distance matrix between $\mathbf{p}'_g$ and $\mathbf{p}$. Each element in the distance matrix d_{ij} is calculated as $d_{ij} = ||p'_{g_i} - p_j||^2$. Hungarian algorithm [25] is then used to find the minimum sum of distance assignment between the shifted goal configuration and the current robot poses. Finally, the resulting assignment is mapped to the sorted goal configuration pairs.

Algorithm 2 Connection Pair Assignment

Input: $\mathbf{p}_g$: goal configuration, $\mathbf{p}$: robot poses
Output: *pair_dict*: connection pair assignment

```
function AssignConnectionPairs(p_g, p)
    p'_g ← alignCenter(p_g, p)
    goal_pair_dict ← findExistPairs(p'_g)
    sorted_pair_dict ← sortPairs(goal_pair_dict, p'_g) based on distance
    dist_matrix ← getDistanceMatrix(p'_g, p)
    id_assign ← HungarianAlgorithm(dist_matrix)
    pair_dict ← updatePairAssignment(sorted_pair_dict, id_assign)
    return pair_dict
end function
```

The configuration control algorithm is shown in Algorithm 3. It takes in a goal configuration and outputs the control input for the robots to execute. During the process, we maintain the information of 1) busy vector where $busy[i]$ represents if robot i is currently aligning an active connection pair, 2) already connected pairs, denoted as $\mathcal{C}_{conn} = \{(i,j) : (c_i, c_j), \ldots | c_i \in \mathcal{C}_i, c_j \in \mathcal{C}_j, \mathcal{C}_i \cap \mathcal{C}_j = (c_i, c_j)\}$; 3) the active connection pairs $\mathcal{C}_{exec}$ that are currently being executed. First, we obtain

the goal connection pairs to be executed, denoted as $\mathcal{C}_{goal}$ with Algorithm 2. To maintain a dynamic connection between the robots, we will updated the connection pairs in $\mathcal{C}_{conn}$ with Algorithm 1. Then we check if any new connection pair can be executed, as in the single pair alignment in Section 3.3.1 and the target control input obtained in this step is denoted as $\hat{\mathbf{u}}[busy]$. Note that, only the robots that have active connection pairs to execute are assigned with target control input. Therefore, for already connected pairs, e.g. $(i, j) \in \mathcal{C}_{conn}$, if robot i is $busy$, the target control input for robot j becomes $\hat{u}_j = \hat{u}_i$ to mimic the motion of the leading robot i in this connection pair. To limit the influence of uncertainties in the hardware actuation, we incorporate a connection bias in the control signal to further maintain the already connected pairs. For each robot i, the connection bias is

$$u_{connection\_bias}[i] = \sum_j J_i^+(p_j - p_i), \text{for all}(i, j) \in \mathcal{C}_{conn} \ .$$

The optimal control input $\mathbf{u}^*$ is then obtained from Equation (3.7). Finally, we find the already aligned connection pairs in the current execution set $\mathcal{C}_{exec}$. The connected pairs are removed, and the robots return to non-busy status. Depending on the structure of the configuration, the best case run time of this algorithm is $O(\log N)$ while the worst case is $O(N)$.

3.4 Hardware Setup

Each PuzzleBot is equipped with onboard power, computation, communication, and actuation. The circuit design remains the same as in the previous version of the PuzzleBots [75]. Each robot is equipped with four trackers for indoor localization via the Vicon motion tracking system. The robots, Vicon, and a central computer are connected within the same WiFi network. The central computer computes the command velocity and sends it to each robot to execute.

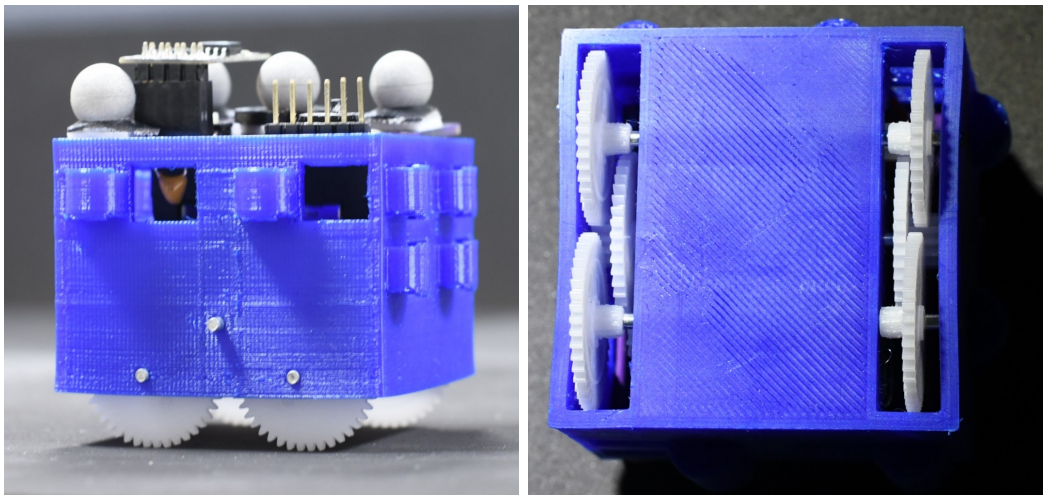
In the previous version of the PuzzleBots, each robot has two wheels on each side, four wheels in total. The two wheels are controlled by one motor via a double reduction gear set. This four-wheeled setup successfully demonstrated the ability to climb onto a platform when crossing a gap. However, their mobility is limited

Algorithm 3 Connection Pair Based Configuration Control

Input: $\mathbf{p}_g$: goal configuration, $\mathbf{p}$: robot poses, ϵ: threshold
Output: $\mathbf{u}^*$: control input
Initialize: $busy$ =[False, ...], $\mathcal{C}_{conn} = \{\}$, $\mathcal{C}_{exec} = \{\}$

function ConfigurationControl($\mathbf{p}_g$, $\mathbf{p}$)
  $\mathcal{C}_{goal} \leftarrow$ assignConnectionPairs($\mathbf{p}_g$, $\mathbf{p}$)
  for $pairs$ in $\mathcal{C}_{conn}$ **do**
    update connection pairs $pairs$
  end for
  for (i, j) in $\mathcal{C}_{goal}$ **do**
    if i, j not busy **then**
      $\mathcal{C}_{exec}$.append((i, j))
      $busy[i]$ = True, $busy[j]$ = True
    end if
  end for
  $\hat{\mathbf{u}}[busy] \leftarrow$ alignConnectionPairs($\mathcal{C}_{exec}$, $\mathbf{p}$)
  $\hat{\mathbf{u}}[\sim busy] \leftarrow \hat{\mathbf{u}}[busy]$ for each $\mathcal{C}_{conn}$
  $\hat{\mathbf{u}} = (1 - k_{bias})\hat{\mathbf{u}} + k_{bias}\mathbf{u}_{connection\_bias}$
  $\mathbf{u}^* = \arg\min ||\mathbf{u} - \hat{\mathbf{u}}||^2$, s.t. $A\mathbf{u} \leq b$
  for (i, j) in $\mathcal{C}_{exec}$ **do**
    if distance between connection pairs $< \epsilon$ **then**
      $\mathcal{C}_{exec}$.remove((i, j))
      $busy[i]$ = False, $busy[j]$ = False
      u_i, $u_j \leftarrow 0$
    end if
  end for
  return $\mathbf{u}^*$
end function

due to friction. The robot can freely move forward and backward but has limited rotation ability. This highly restricts the robots' performance of achieving precise poses. Therefore, we modified the design by adding two caster wheels in the front and back while replacing the two wheels on each side with only one. The new robot design is shown in Figure 3.7a. While the caster wheels successfully solve the problem in rotation and enable the robot to do high-precision tasks, the gap-crossing behavior becomes limited. Since the caster wheels are not actuated, the robot cannot climb onto the platform when the caster wheel reaches the other side of the gap. Thus, we propose a heterogeneous robot swarm system that consists of two types of robots:



(a) (b)

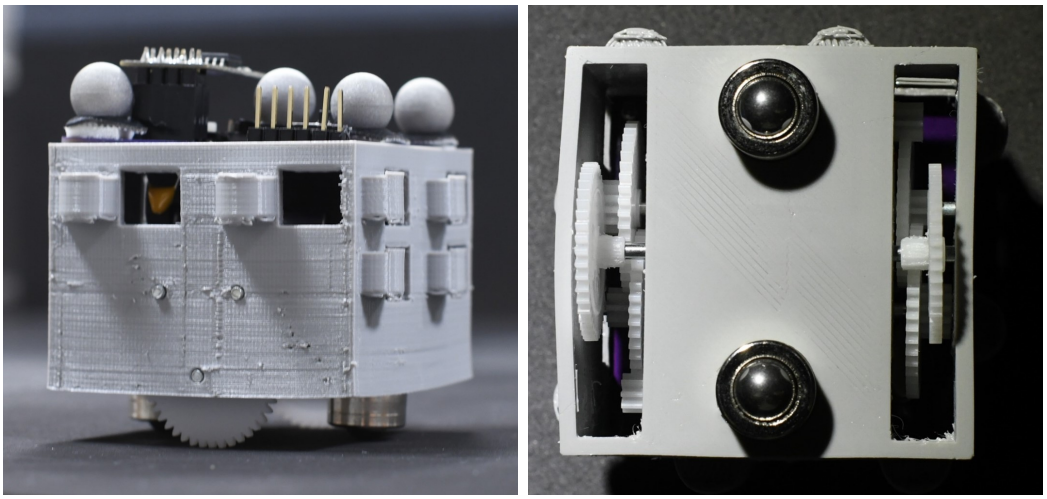
Figure 3.6: Pilot robot: (a) side view, (b) bottom view.

pilot robots and non-pilot robots. The pilot robot is a four-wheeled robot, and the non-pilot robot is one with casters and side wheels. As shown in Figure 3.6 and 3.7, the wheels of the pilot robot will help to climb onto a platform, while the flexibility of the non-pilot robot enables the system to form complex configurations and perform high-precision tasks.

The electronics board on the pilot and non-pilot robot is the same. The control of these two kinds of robots also remains the same, i.e., both follow the differential drive model. The linear velocity is linearly proportional to the left and weight wheel average velocity. The angular velocity is linearly proportional to the velocity difference between the right and left wheels. Denote the left and right motor Pulse-width modulation (PWM) signal as M_r, M_l. The motor rotational speed is linearly proportional to the PWM signal with a fixed load. We parametrize the velocity equation for each robot as

$$v = k_v \frac{M_r + M_l}{2}, \omega = k_\omega (M_r - M_l) \ . \tag{3.10}$$

From experiments, each robot needs a start-up torque to move. Denote the corresponding PWM signal as M_{min}, and a maximum PWM as M_{max}. With a given



(a) (b)

Figure 3.7: Non-pilot robot: (a) side view, (b) bottom view.

control input $u^* = [v^*, \omega^*]$, we compute

$$\arg\min_{M_r, M_l} \mu_v ||k_v \frac{M_r + M_l}{2} - v^*||^2 + \mu_\omega ||k_\omega (M_r - M_l) - \omega^*||^2$$
$$M_{min} \leq |M_r, M_l| \leq M_{max},$$

where μ_v and μ_ω are weight parameters for linear and angular velocities respectively. The difference of controlling the pilot and the non-pilot robot lies in their parameters of the control feasibility region, which is the constraints $A\mathbf{u} \leq b$ in Equation (3.7). We will present the calibration of the feasibility region in Section 3.5.1 as well.

3.5 Experiments and Results

The system consists of several PuzzleBots, a Vicon localization system, and a central computer within the same network. The algorithm is first tested in simulation in CoppeliaSim[46] with the Vortex Studio[1] as the physics engine for concave objects. We conducted three sets of experiments: Section 3.5.1 presents the hardware calibration of the pilot and non-pilot robots. Section 3.5.2 shows a set of screenshots from a

[1] https://www.cm-labs.com/vortex-studio/

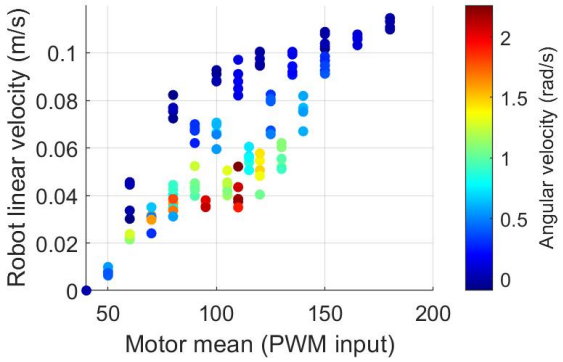


Figure 3.8: Non-pilot robot linear velocity.

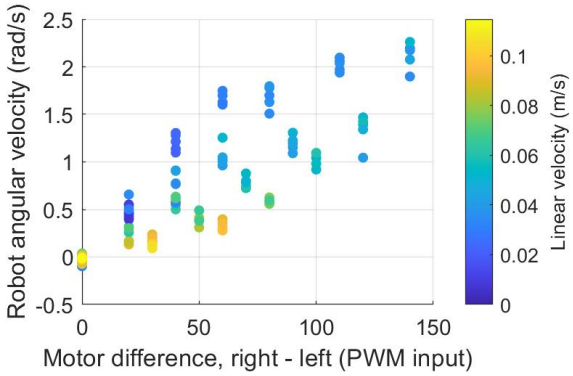


Figure 3.9: Non-pilot robot angular velocity.

video sequence where three non-pilot robots and one pilot robot couple into a mesh configuration, cross a gap and decouple on the other platform. The last Section 3.5.3 presents quantitative results of the gap-crossing performance based on the line and mesh configuration.

3.5.1 Robot Calibration

Robots are commanded with a combination of different motor PWM signals for a period of time, and the Vicon software records the poses. Due to the noise, the velocity obtained directly from two consecutive poses is unusable. Thus, we average the linear and angular velocity across a longer period of time (several seconds). We calculated the mean and difference of the left and right PWM signals. The parameters obtained from the experiments are $k_{v,\text{non-pilot}} = 0.0006$, $k_{\omega,\text{non-pilot}} = 0.0142$, $k_{v,\text{pilot}} = 0.0024$,

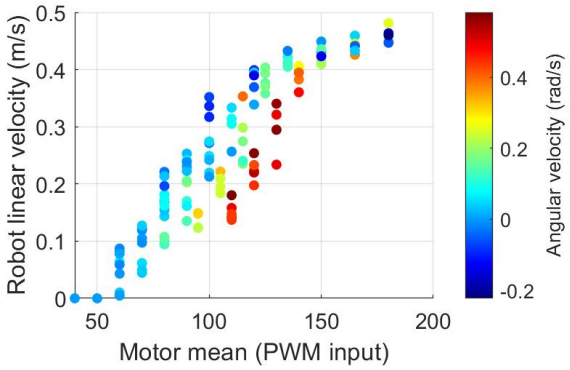


Figure 3.10: Pilot robot linear velocity.

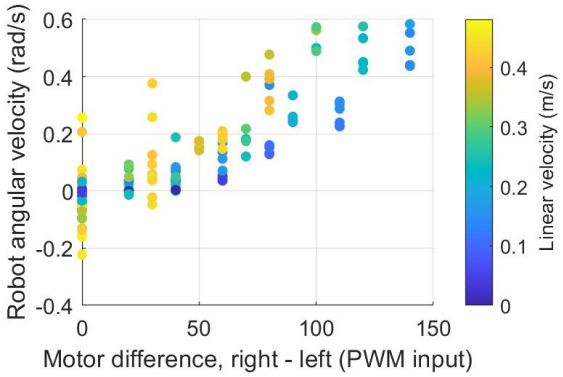


Figure 3.11: Pilot robot angular velocity.

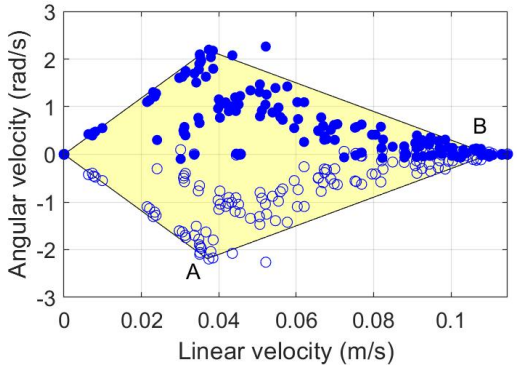


Figure 3.12: Non-pilot robot $v - \omega$ constraints.

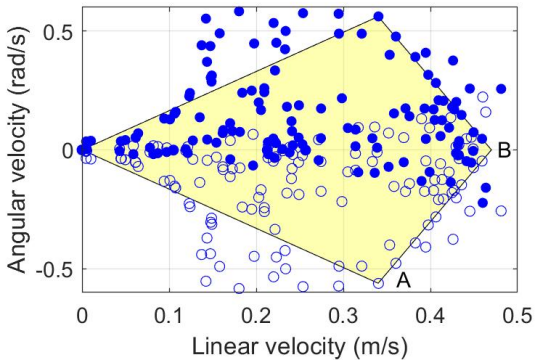


Figure 3.13: Pilot robot $v - \omega$ constraints.

$k_{\omega,\text{pilot}} = 0.0033$. Figure 3.8 and 3.10 show that with the same motor mean, the robot has higher linear velocity when the angular velocity is small. As shown in Figure 3.12 and 3.13, the feasibility region of the non-pilot and pilot when the linear velocity v is positive is shown in the polygon. The points obtained from the experiment are projected along the $\omega = 0$ axis based on symmetry. Similarly, the polygon is projected onto the negative v plane. The point $A = (a_x, a_y)$ and point $B = (b_x, 0)$ are the critical points defining the polygon. We manually label A and B by observation, and the polygon region defined by A and B is recorded. In our experiment, we have $A_{pilot} = (0.34, -0.561)$, $B_{pilot} = (0.47, 0)$, $A_{non-pilot} = (0.037, -2.19)$, $B_{pilot} = (0.11, 0)$. The non-pilot robot is able to rotate in high angular velocity with low linear velocity, while the pilot robot requires a large linear velocity to rotate. The steering distance for the pilot robot to turn is thus larger, making it difficult to perform the coupling behavior, which requires precise rotation. Therefore, a combination of both robots can utilize the flexibility of the non-pilot robot, as well as the climbing wheels of the pilot robot.

3.5.2 Combined Behavior Sequences

As shown in Figure 3.14, the four robots - three non-pilot robots shown in grey and one pilot robot shown in blue, start from unaligned positions on the left platform. The two platforms have a height difference of 5 mm, and the gap size is 40 mm. Due to the large steering distance of pilot robots, they are not given a velocity command until they are coupled with non-pilot robots, which is embedded in the algorithm. In

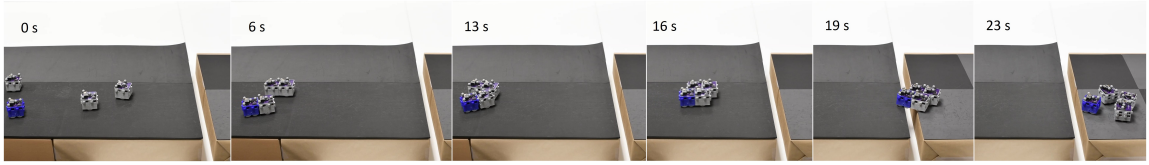


Figure 3.14: Screenshots of four robots coupling to form a mesh configuration, crossing the gap, and decouple.

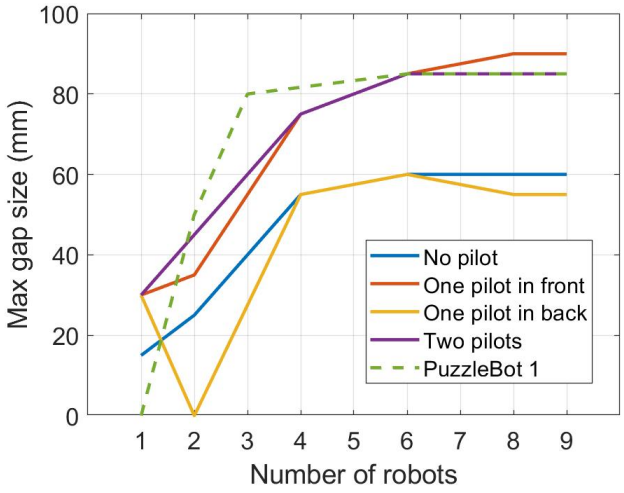


Figure 3.15: Maximum gap size the line configuration assembly can cross against the number of robots.

this case, we see that the robots are able to first form two line assembly based on the connection pair assignment and then come together to form a mesh configuration. They are able to cross the gap and reach the other platform, and eventually decouple with each other.

3.5.3 Gap-crossing Performances

We performed experiments of a various numbers of robots, with line and mesh formation, of different lengths of gaps with 6 mm height difference. The robots are coupled with a heading angle of 20°. These two numbers are chosen based on the experiments in [75] that 6 mm is a medium height difference and 20° is the heading angle that gives the best performance compared with other heading angles in most of the experiments. We tested a system of 1, 2, 4, 6, 8, 9 robots with gap lengths

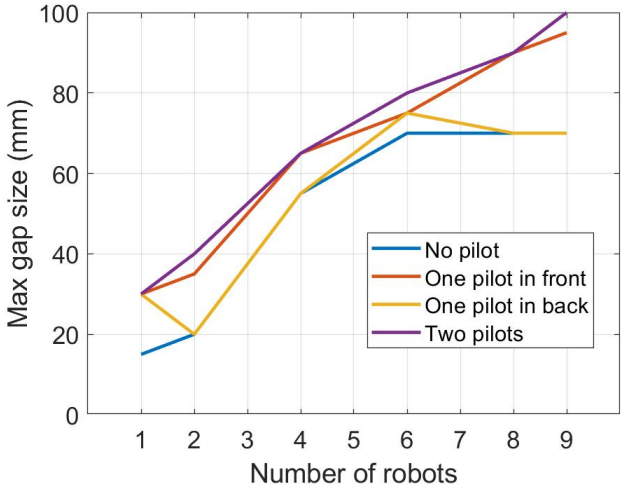


Figure 3.16: Maximum gap size the mesh configuration assembly can cross against the number of robots.

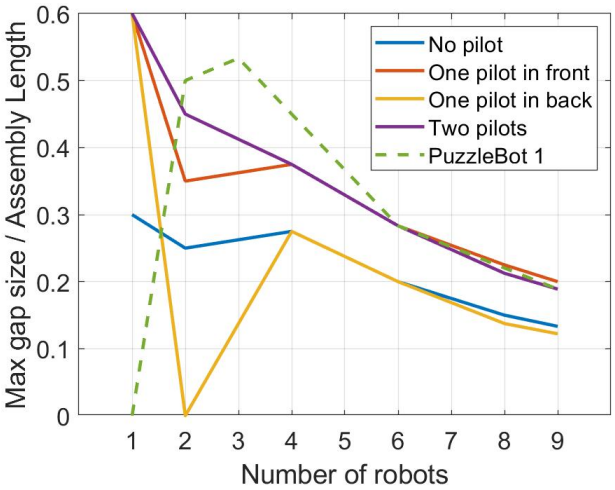


Figure 3.17: Maximum gap size compared with the entire line configuration assembly length, against the number of robots.

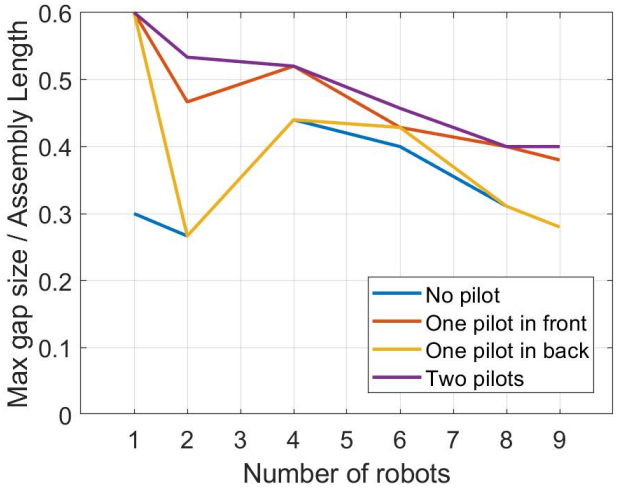


Figure 3.18: Maximum gap size compared with the entire mesh configuration assembly length, against the number of robots.

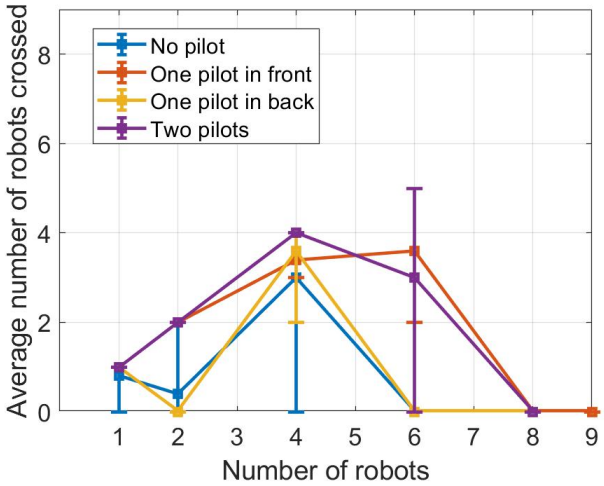


Figure 3.19: Average number of robots that crossed a gap length = 25% length of the line configuration assembly, against the number of robots.

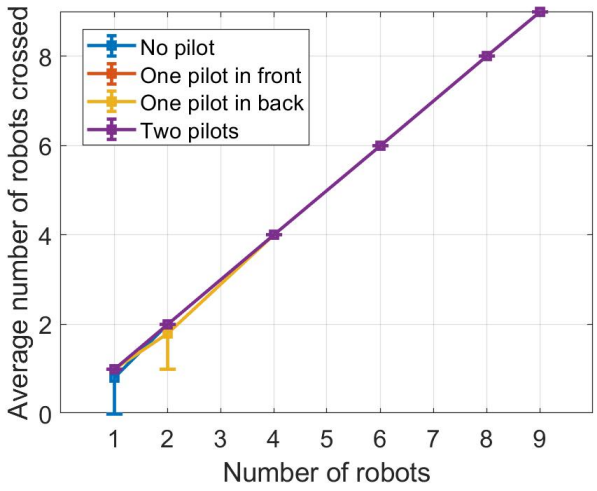


Figure 3.20: Average number of robots that crossed a gap length = 25% length of the mesh configuration assembly, against the number of robots.

ranging from 10 mm to 105 mm, with 5 mm increments. Each experiment is recorded five times. We record the number of robots successfully reach the other platform.

Figure 3.15 and 3.16 show that the robots in either line or mesh formation maintain the gap-crossing ability as in [75], and can cross larger gaps with more than eight robots compared with [75]. The line formation is able to cross larger gaps with a small number of robots compared with the mesh formation. The reason is that the total length of the assembly is longer when the robots form a line compared with a mesh. As seen in Figure 3.17 and 3.18, the robots in mesh formation can cross gaps with length of larger percentage of the assembly length. For example, eight robots with two pilots are able to cross a gap 36% their assembly length, while the same robots in line formation can only cross a gap 21% of its assembly length. Since the coupling mechanism will introduce a small height drop of the robots, forming a mesh configuration will lock and strengthen the connection, thus giving a better performance. Robots with a pilot in the back outperform assembly without a pilot robot since the wheels of the back pilot robot provide additional pushing force for the assembly to cross a gap. A pilot robot in the front performs better compared with a back pilot since the gear wheels of the pilot robot help itself to climb onto the platform. Overall, the assembly with both front and back pilot outperforms other settings. Figure 3.19 and 3.20 show the average number of robots that crossed a given

gap size of 25% length of the entire assembly. The error bars mark the minimum and the maximum number of robots that crossed in this setting. We observe that with the mesh configuration, all robots can cross the gap in most cases. Although in some cases, the system with two pilots line formation has less robots that crossed, the best case of the two-pilot system has more robots that crossed compared with the other settings.

3.6 Conclusions and Future Works

In this paper, we proposed a heterogeneous robot swarm system consists of pilot robots helping climb onto platforms and non-pilot robots providing flexible motions to form functional configurations. Based on our proposed system, we developed the connection-pair-oriented configuration control algorithm, enabling the robots to form various coupling configurations. Our currently approach relies heavily on the motion capture system, limiting the robots to only indoor lab environment. Further studies would include sensor integration, motions on uneven terrains, optimality and scalability analysis on the algorithm, dynamic reconfiguration, and automatic configuration generation based on different tasks and environments.

Chapter 4

Reconfigurable Robot Control Using Flexible Coupling Mechanisms

4.1 Introduction

Robot swarms demonstrated collective behaviors that a single robot could not accomplish [39, 44, 69]. Furthermore, robots that physically connect and interact with each other, i.e., the self-reconfigurable robots, have extended capabilities and behaviors. Applications include reconfiguration for terrain adaptation [21], dynamic infrastructure [37], collaborative transportation [2, 54], search and rescue [70], etc. Our goal is to develop a robot swarm system where robots can dynamically couple to form functional structures, e.g., bridges and ropes, and decouple to navigate around the environment individually.

Many modular robots that can dynamically couple and decouple use magnetic forces, e.g., ATRON [23], M-TRAN III [26], and M-blocks [49]. Since the power and dynamic structures are dedicated to the coupling mechanism, each module has limited mobility compared with a regular mobile robot. This limits their ability to navigate around the environment.

To be able to move around efficiently, the robots need efficient locomotion ap-

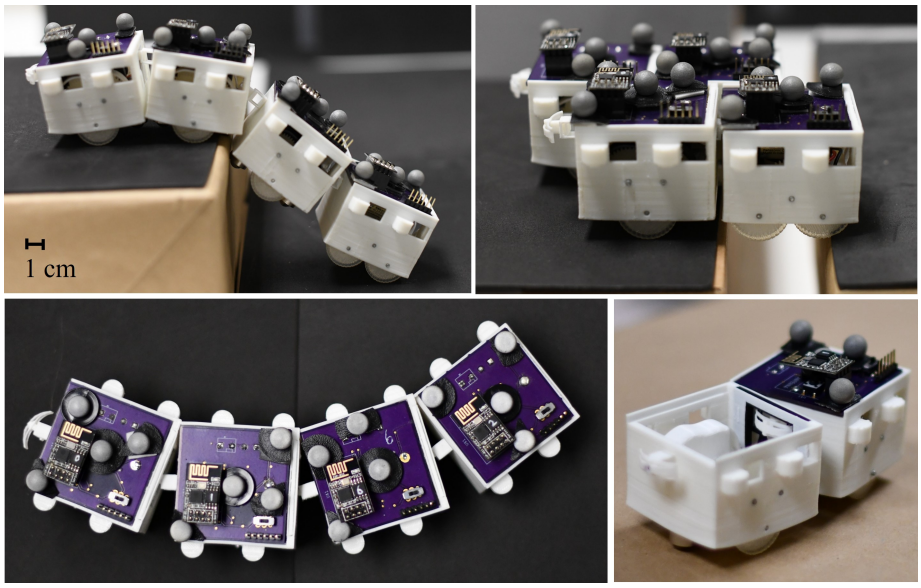


Figure 4.1: Top: Robots can form a flexible chain and go down a step (left) or a rigid structure that crosses a gap between two equal-height platforms (right). Bottom left: Four robots forming an articulated chain. Bottom right: One robot inserts its anchor inside the opening of another one (circuitry removed for visual purposes).

proaches, e.g., wheels or legs. Swarm-bot [18] are wheeled robots with grippers that can grip onto each other to form bridges. The SMORES [65] and FreeBot[28] modules are equipped with wheels and can form 3D structures with magnetic forces. However, these *actuated* coupling mechanisms consume high power when maintaining the coupling status. Moreover, although magnetic connections provide high pulling forces, the shear force is significantly lower. The polarity of magnets also limits the connection to be directional, and the strength of the connection is highly sensitive to misalignment [65]. Compared with the "active" actuated coupling mechanisms, "*passive*" coupling strategies - where no power is dedicated to maintaining the coupling status - are more energy efficient. The ModQuad[54], Roboats [37], SlimeBot [57], and the PuzzleBots [75] utilize passive coupling mechanisms to form and maintain structures. However, with these passive mechanisms, robots form a rigid 2D connection when coupled together, which greatly limits their functionality and mobility.

Although passive coupling mechanisms have the benefits of power efficiency, maintaining the coupled states require precise control of the robot's motion. The PuzzleBots [76] utilize quadratic programming (QP) based configuration control

algorithm that constrains a pair of points between two robots to maintain the coupled state. This point constraint is very sensitive to localization errors and heavily restricts the robot's motion. Control Barrier Functions are commonly used to relax distance-based constraints [7, 81]. However, considering robots as point masses and their interactions as point pairs does not fully capture the geometry of the coupling behaviors. Instead, polytopes are a more precise option to describe geometries. Most studies that focus on the geometry of robots and objects are for collision avoidance [63, 64]. However, to maintain the connection of passive coupling mechanisms, robots need to keep their connection points within polygon shapes instead of avoiding collision with polytopes.

Our goal is to design a passive coupling mechanism that enables robots to form 3D functional structures and control the assembled structure to perform collective behaviors. To form a 3D structure, the coupling mechanism should provide strong enough force to hold the load of multiple robots. While having a strong connection, the robots should still be able to decouple from each other dynamically. It is challenging to balance a strong connection and an easy disconnecting method on a single passive mechanism with no dedicated power. The assembly formed by multiple robots should also be flexible in terms of mobility so that the assembled structure can move in a desirable configuration forming a bridge over a gap or a rope down a stair. Modeling and controlling such a flexible structure is also challenging. Soft objects are difficult to model due to their deformability and the lack of state and force estimation. With a passive coupling mechanism, the robots need precise control over their motion to maintain the connection. Such maintenance constraints should be restrictive enough so that the robots do not decouple easily and relaxed enough so that the assembled structure can move around the environment.

Therefore, we propose a *soft anchor* design that is passive and has a strong holding force for 3D structures while keeping a simple coupling and decoupling behavior between robots. The anchor acts as an unactuated joint when inserted into the opening of another robot. The anchor is asymmetric - the force of pushing it into an opening is small, while the force required to pull it out is large. This unactuated joint formed by the soft anchor also leaves room for robots in an assembled structure to rotate or translate within the limit of the joint, compliant with the robot's motion. We limit our soft anchor design to having only one degree of freedom (yaw rotation),

and its state can be estimated by the relative position and displacement. By obtaining force profiles for our soft anchor, we can model and simulate it as a linkage of rigid components similar to [83]. We maintain the connection between robots by enforcing a polygon-based constraint. This restricts the relative motion between robots to maintain the coupling status while giving some flexibility to the robots to perform rotations or translations.

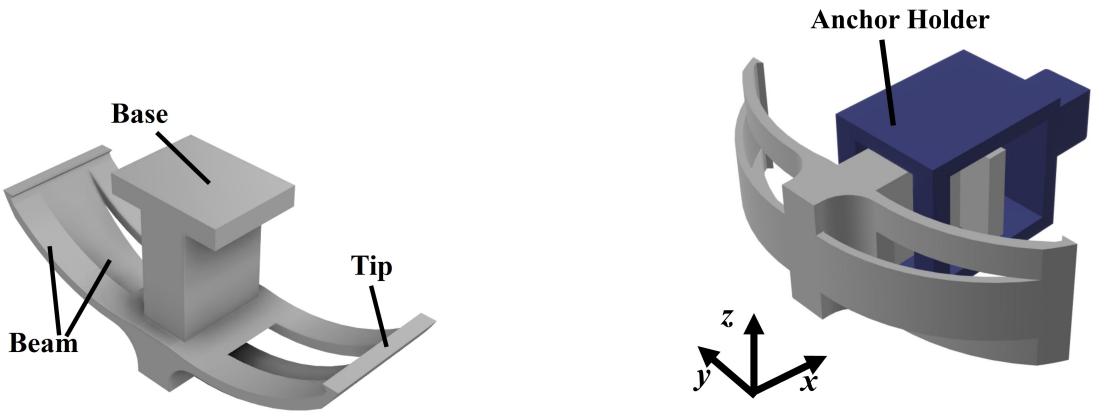
Our contribution lies in three aspects. First, we propose the design of a soft asymmetric anchoring mechanism that allows robots to couple and form flexible or rigid configurations. Second, we present a three-bar linkage system in simulation that models the soft anchor and simulate robot behaviors comparable to the actual hardware system. Finally, based on our designed mechanism, we propose a Model Predictive Control (MPC) based framework with polygon constraints to model the precise geometry of the coupling formations.

4.2 Design of Flexible Coupling Mechanisms

In this section, we will first review our robot design. Similar to Yi et al. [76], we have *pilot* robots with four actuated wheels for better ground grip and *non-pilot* robots with two actuated wheels to perform agile motions for coupling and decoupling. Based on the robot model, we propose our flexible passive anchor design that provides coupling and decoupling behavior between robots, as well as a locking behavior for maintaining configurations on unstructured terrain.

4.2.1 Robot Design

The robots, shown in Figure 4.1, are 50 mm in depth (excluding the anchor), 50 mm in width, 45 mm in height (without WiFi and optical markers), and weighs 68 g. There are four markers on the top of the robot for the motion-tracking system (Vicon) to provide high-frequency pose information. Each robot has an ESP8266 Wifi Module that connects to the same network with the Vicon and a central computer. The onboard power is provided with a 3V CR2 battery. Two DC motors execute velocity commands received from the central computer. Design configurations to utilize the new anchor structure are explained in the next section.



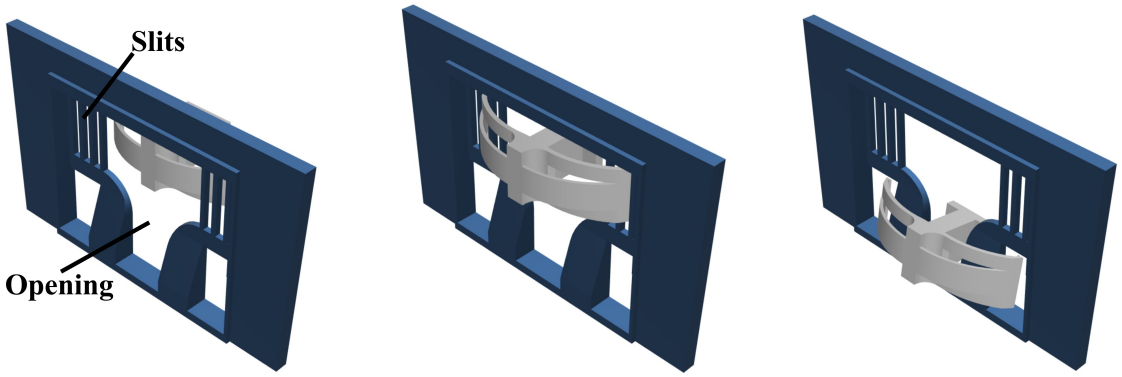
(a) (b)

Figure 4.2: (a) Anchor design. (b) Anchor (gray) with its base inserted in a holder (blue) acting as a floating joint.

4.2.2 Anchor Design

Existing flexible coupling mechanisms that utilize magnets have limited shear force [28, 54, 65], active mechanical connections are power-consuming and complicated to manufacture and integrate with a different platform [18, 37], while the fully passive connections are rigid and not robust with disturbances [75]. Therefore, the design goal of our coupling mechanism includes 1) strong enough forces to hold multiple robots at the same time; 2) a fully passive coupling mechanism so that no additional power is consumed for the coupling process; 3) flexible as an assembled structure while being robust on rough terrains and with uncertainties from actuation and localization.

With the above-mentioned requirements, we propose our anchor coupling design shown in Figure 4.2. Figure 4.2a shows the anchor with two shorter beams on the side and one longer beam in the middle. The beams are curved towards the base of the anchor. The anchor is 3D printed with Thermoplastic Polyurethane (TPU) (NinjaFlex Cheetah). With this design, the forward pushing force of the anchor can be small, and the backward pushing force of the anchor can be large. Characterization and force profiles are presented in Section 4.4.1. As shown in Figure 4.2b, the anchor sits in a rigid base, which is 3D printed with a hard plastic material, Polylactic Acid (Ultimaker Tough PLA). The soft anchor (gray) is placed inside the anchor holder (blue), through which the anchor is connected to the robot's body. The anchor can



(a) (b) (c)

Figure 4.3: (a) Anchor and an opening uncoupled when on the same height. (b) Anchor and opening coupled. The anchor tips are sitting in the slits. (c) Anchor locks the connection when the robot's vertical position changes with gravity due to the irregularities on the ground.

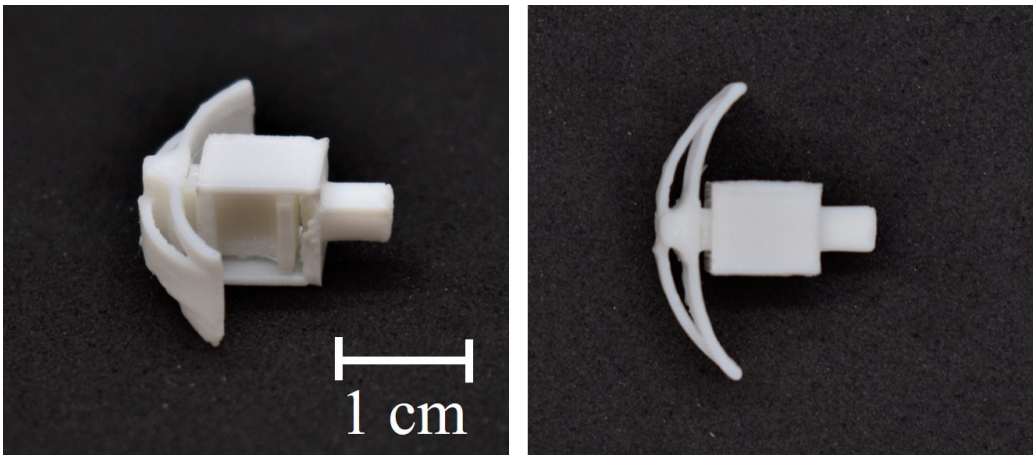


Figure 4.4: 3D printed anchor in collapsed state.

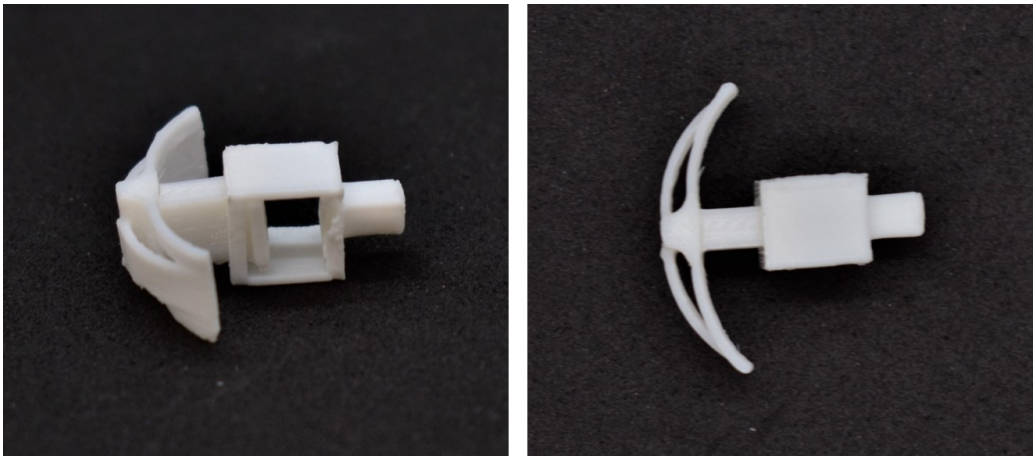


Figure 4.5: 3D printed anchor in extended state. The holder provides translational motion within 5 mm.

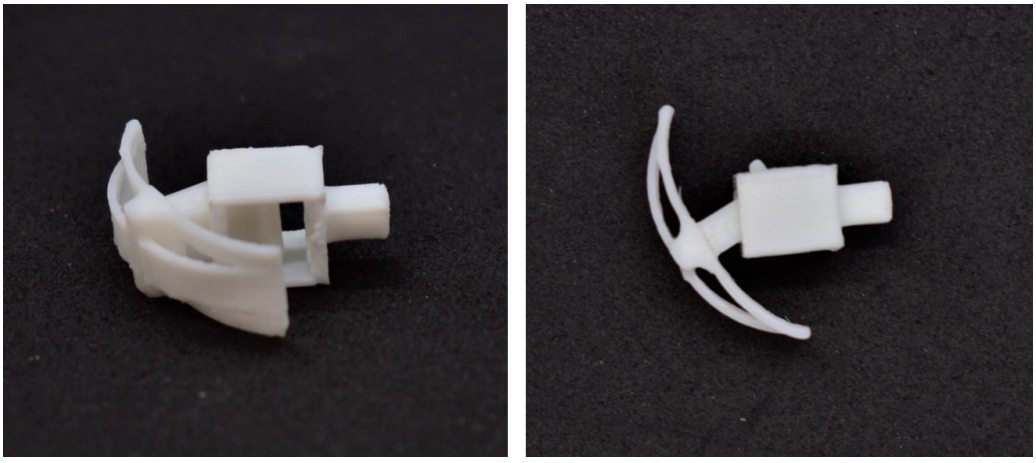


Figure 4.6: 3D printed anchor in rotated state. When not collapsed, the anchor is also free to perform rotations within 0.5 rad.

rotate freely in the yaw direction and moves freely along the x axis inside the holder. The anchor has limited motion in the y direction (< 0.5 mm) or pitch rotation (< 0.1 rad). This floating joint between the anchor and the holder gives flexibility when the robots are coupled together. The anchor and its holder are mounted on the back of the robot, and there is an opening in the front of the robot shown in blue as in Figure 4.3a, which is a state when two robots are not coupled. As shown in Figure 4.3b, one robot can push its anchor inside the opening of another robot to couple. The anchor tips will sit in the slits on the two sides of the opening. With the anchor beam design we introduced, the force of pushing the anchor inside will be significantly smaller than the force provided by the root during forward motion. The force needed to pull the anchor backward is more significant than the wheel actuator force. Thus, the anchor will be easy to push inside the opening for the robots to couple and cannot decouple easily. The robot needs to wiggle to decouple from each other. We show this decoupling controller in Section 4.3.2. By adding different constraints over the robot's motion, we may avoid the wiggling behavior when we would like the robots to couple or stay coupled, and only enable the wiggling behavior if we want the robots to decouple. This provides robustness under uncertainty with actuation and localization in the hardware system. This motion is presented in Section 4.4. When one robot moves toward an edge of a platform, it will drop with gravity. The anchor will drop with the robot and sits in the slot as shown in Figure 4.3c. This locks the robot's movement, and the assembly can form a chain down the platform's edge. This anchor-opening connection type provides additional flexibility for the coupled assembly. The robots can still couple on the side with the knob-hole connection as in [76], which can be necessary when reaching a platform of the same height across a gap.

In summary, the geometry and soft material of the anchor enables flexible coupling and decoupling behavior while providing robustness during execution. We show the hardware experiments in Section 4.4.3 and in our supplementary video material. To achieve this on our non-linear differential drive platform, we will introduce our multi-robot Model Predictive Control (MPC) method with polygon-based constraints in the next section.

4.3 Modeling and Control of Coupling Behaviors

To maintain the coupling status, one way is to maintain a minimal distance between two specific points (connection points) on the robot bodies. This is very restrictive on the robot's motion and not robust against noise or disturbances. In this work, we incorporate the anchor coupling mechanism introduced in the previous section on a unicycle dynamics model. To this end, we utilize Model Predictive Control (MPC) in our multi-robot system along with polygon constraints to maintain the coupling status. In this section, we introduce the basic setup for MPC and the sequential coupling behavior for two robots. Then we propose the polygon-based constraint for maintaining the coupling pairs in an assembly. The actual hardware system consists of a low-frequency central MPC planner and a high-frequency PID controller to be introduced in Section 4.4.3.

4.3.1 Robot Dynamics

Consider a multi-robot system with N robots on a 2D plane. The state of robot i, where $i \in \{1, 2, \ldots, N\}$, is $\mathbf{x}_i = [p_x, p_y, \theta, v, w]^\intercal$. $p_x, p_y \in \mathbb{R}$ are the position in the spacial frame x and y axis respectively. $\theta \in (-\pi, \pi]$ is the heading angle (yaw) of the robot. The v and w are linear and angular velocities, respectively. Control input for robot i is $\mathbf{u}_i = [\dot{v}_i, \dot{w}_i]^\intercal$. Consider robot dynamics as follows (we temporarily drop the robot index i to simplify notations)

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ w \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u} \tag{4.1}$$

In Section 4.3.4, we integrate this dynamic equation with the Euler method for discretization in the MPC setup. Although there are more accurate discretization methods available like RK4 [6], due to our short MPC horizon, there is no noticeable difference between the Euler and RK4 discretization in our experiments. Thus, we choose the Euler discretization method for faster computation.

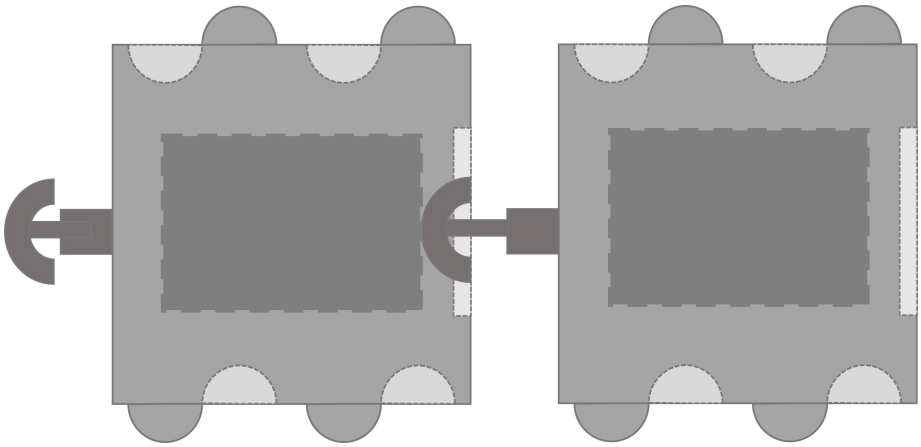


Figure 4.7: Two robots coupled together with an extended anchor. The dark gray area in the middle of the robot is the battery box, which limits the anchor state and enforces the anchor tip to sit and lock with the opening.

4.3.2 Decoupling Behavior

We introduce the decoupling behavior first due to its simplicity. When two robots are coupled, the anchor of one robot sits inside the opening of another robot, as shown in Figure 4.3b. To decouple, the robot with the anchor wiggles, i.e., periodically rotates and moves with a linear velocity. This wiggling behavior is controlled as

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} v_{\text{bias}} \\ w_{\text{max}} \sin(Bt) \end{bmatrix} \tag{4.2}$$

where $2\pi/B$ is the period. The rotation from this behavior loosens the anchor tip from the slits of the opening and makes it free to rotate inside the robot's body. This rotation in the anchor results in the decoupling of robots from each other (see Section 4.4.3).

4.3.3 Connection as Polygon Constraints

Projected Anchor Zero Position

With a passive coupling mechanism, it is essential to maintain the connection between robots by constraining the relative motion inside an assembly. In [76], a pair of

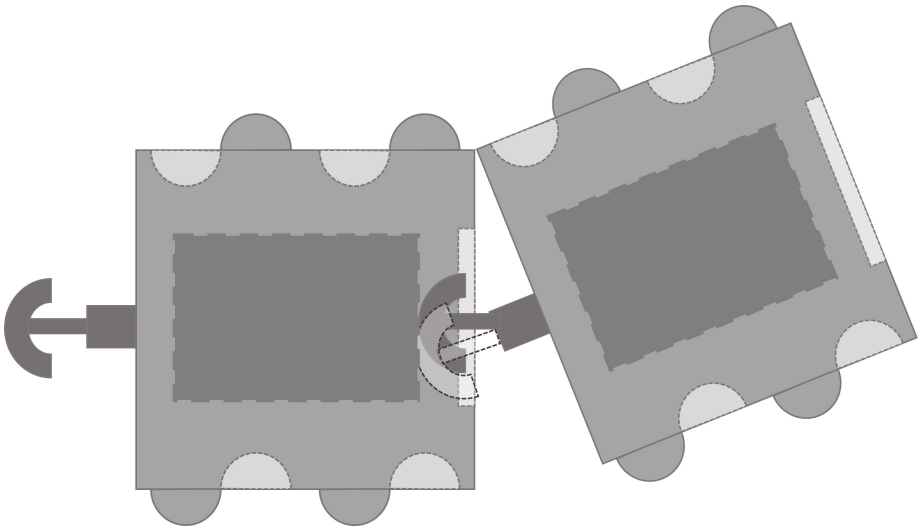


Figure 4.8: One robot rotates as the anchor is coupled. The white shaded anchor shows a projection of the real anchor if the anchor joint is not compliant.

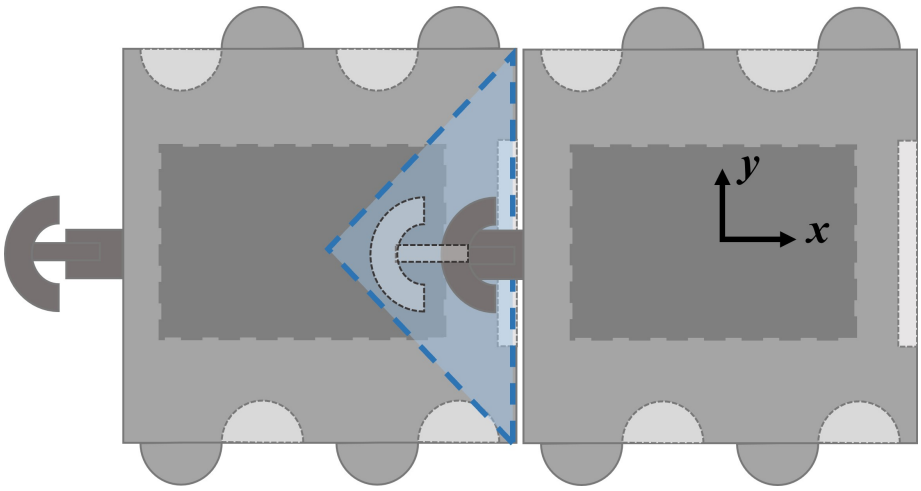


Figure 4.9: The blue shaded area shows the constraint of anchor head location during coupling.

connection points are constrained to always be within a small distance threshold to avoid accidental decoupling. This highly restricts the motion of the robots when they are assembled. With our anchor design, the floating joint, as shown in Figure 4.2b, provides flexibility in the rotation and translation when robots are coupled together. However, since the connection is still passive, we enforce a relaxed polygon constraint for the anchors to maintain the connection.

As shown in Figure 4.7, when two robots couple, the anchor head (shown as light gray) of one robot sits inside the body of the other robot. The battery box inside the robot will block the motion of the anchor head and enforce the anchor tip to sit and lock with the striped opening shown in Figure 4.3b. Note that, in the actual hardware system, there is some room for the anchor to move around. We define the anchor position as shown in Figure 4.7 as the *zero position* when the anchor heading angle aligns with the robot. With the floating joint between the anchor and its holder, the robot can move back and forth within a small distance and rotate within a small angle range. In the state shown in Figure 4.8, the anchor of one robot sits in the opening of another robot while the robots rotate. We can model this state with a projected anchor zero position (shown in light gray). The floating joint is compliant with the robot's motion. Thus, instead of modeling the floating joint status and the corresponding robot positions, we know that the anchor is coupled with the other robot if the projected anchor zero position lies within the shaded polygon region as shown in Figure 4.9. This polygon constraint gives a simple way of modeling the anchor connection and can be directly incorporated into the MPC framework in Section 4.3.4.

Point within Polygon

Before diving into how the polygon constraint is integrated with the robot dynamics, we first need to formulate the constraint of a point inside a polygon. The Point-in-Polygon (PIP) problem is a classic computational geometry problem. Classic solutions for the PIP problem, e.g., ray-casting or winding number algorithms, are challenging to integrate with the MPC framework due to their complexity. This section presents our mathematical derivation of a set of linear constraints for a point to be inside a convex polygon.

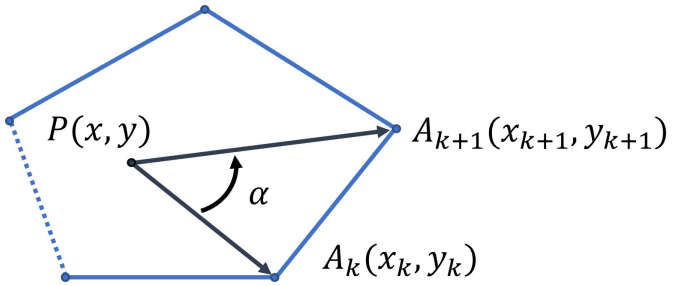


Figure 4.10: A point inside a convex polygon.

As shown in Figure 4.10, a 2D convex polygon is defined by a series of points $A_1, \ldots, A_k, A_{k+1}, \ldots, A_K$, where $A_k \in \mathbb{R}^2$ has coordinate of (x_k, y_k). The numbering of the points increments counterclockwise. By definition of convex polygons, all points within the convex polygon lie on one side of each line segment $A_k A_{k+1}$. In our case of the numbering increments counterclockwise, all points within the convex polygon lie on the left-hand side of $\overrightarrow{A_k A_{k+1}}$. Consider a point $P \in \mathbb{R}^2$ with a coordinate of (x, y) inside the convex polygon, point P should lie on the left-hand side of all $\overrightarrow{A_k A_{k+1}}$, where $k = 1, \ldots, K$, $A_{K+1} = A_1$. With the right-hand rule of cross-product, this is equivalent to

$$\overrightarrow{PA_k} \times \overrightarrow{PA_{k+1}} \geq 0.$$

Expanding this equation gives

$$\begin{aligned} \begin{bmatrix} x_k - x \\ y_k - y \end{bmatrix} \times \begin{bmatrix} x_{k+1} - x \\ y_{k+1} - y \end{bmatrix} &\geq 0 \\ (x_k - x)(y_{k+1} - y) - (y_k - y)(x_{k+1} - x) &\geq 0 \\ \begin{bmatrix} y_{k+1} - y_k & -(x_{k+1} - x_k) \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_k \\ y_k \end{bmatrix} \right) &\leq 0 \end{aligned} \tag{4.3}$$

This Equation (5.9) gives a linear constraint for a point $P = (x, y)$ inside a convex polygon defined by points $A_1, \ldots, A_K$, which can be integrated into the MPC framework. To simplify the notation, we denote this series constraints in Equation (5.9) as $pip(P|A_1, \ldots, A_K) \leq 0$, where the numbering of points A_k, $k = 1, \ldots, K$ increments counterclockwise.

4.3.4 Model Predictive Control Setup

With the non-linear unicycle dynamics, Model Predictive Control plans based on the given non-linear dynamics and constraints and minimizes a given cost function. Given the different target behaviors of the robot swarm, we may define different cost functions for the MPC. We consider the following behaviors: 1) align a set of given connection pairs; 2) go to a specific goal location; 3) fit a set of given velocity commands.

Connection Pair Alignment

Consider a connection pair alignment behavior where the goal is to align a set of connection pair points $\mathcal{C}_{active} = \{(C_i, C_j), \ldots\}$, $i, j \in \{1, \ldots, N\}$, where C_i, C_j are connection points on robot i and j respectively. Each connection point frame C_i is defined by its relative transformation $g_{B_iC_i} \in SE(2)$ to the robot body frame B_i. The homogeneous transformation from the Spatial (world) frame to robot frame B_i is defined as

$$g_{SB_i} = \begin{bmatrix} R(\theta_i) & p_i \\ 0 & 1 \end{bmatrix}, g_{B_iC_i} = \begin{bmatrix} R(\theta_{C_i}) & p_{C_i} \\ 0 & 1 \end{bmatrix} \in SE(2) \tag{4.4}$$

where $R(\theta) \in SO(2)$ is the rotation matrix corresponding to the angle θ and $p_i, p_{C_i} \in \mathbb{R}^2$ is the position vector of the frame. In our setup, p_{C_i} is a predefined constant vector for each connection point. Thus, the cost of performing connection pair alignment is

$$J_c(\mathcal{C}_{active}) = \sum_{(C_i,C_j)\in\mathcal{C}_{active}} \left(\Delta p^\intercal W_p \Delta p + W_\theta \tan^2 \frac{1}{2} \Delta\theta \right) \tag{4.5}$$

where $\Delta p = R(\theta_i)p_{C_i} + p_i - (R(\theta_j)p_{C_j} + p_j)$, $\Delta\theta = \theta_i + \theta_{C_i} - (\theta_j + \theta_{C_j})$, $W_p \in \mathbb{R}^{2\times2}$ is a diagonal weight matrix for the positions, and $W_\theta \in \mathbb{R}$ is a scalar weight for the angle difference. To minimize angle differences, one common method is to take the angle difference directly, which gives undesirable values when the angle is around $-\pi$ or π. Another popular method is taking a modulo with 2π on the angle difference. However, this is not differentiable and significantly influences the performance of the optimization. Using quaternions instead of Euler angles is also a common approach, but it creates a huge computation overhead. Since we are in $SE(2)$ and the angle

difference of $\Delta\theta$ and $\Delta\theta + 2n\pi$, $n \in \mathbb{N}$ are the same, we wrap the angle difference with a tan function that has a period of 2π.

Go to goal

Given a set of goals $\mathcal{G} = \{g_i, \ldots\}$, $g_i \in \mathbb{R}^2$, $i \in \{1, \ldots, N\}$, the cost of the robots going to the goal set is

$$J_g = \sum_{g_i \in \mathcal{G}} (p_i - g_i)^\intercal W_g (p_i - g_i) \tag{4.6}$$

where $W_g \in \mathbb{R}^{2\times 2}$ is a diagonal weight matrix for the goals, and p_i is defined in Equation (4.4). Note that here we do not verify if there are conflicting goals or not. The user or external planner is responsible for validating the goal set.

Fit Command Velocity

Fitting a given command velocity is a simple but essential part of a number of behaviors. For example, an assembly can quickly navigate around the environment with a fixed velocity. One robot can also wiggle with a slight forward/backward velocity bias to decouple from a connected assembly. Consider a given velocity command of $[v_0^*, w_0^*, \ldots, v_N^*, w_N^*]^\intercal$, the cost of robots achieving the given velocities is

$$J_v = [v_0 - v_0^*, \ldots, w_N - w_N^*] W_v [v_0 - v_0^*, \ldots, w_N - w_N^*]^\intercal \tag{4.7}$$

where $W_v \in \mathbb{R}^{2N\times 2N}$ is a diagonal weight matrix.

MPC Setup with Constraints

Consider a set of *already coupled* connection pairs in the system $\mathcal{C}_{conn} = \{(C_i, C_j), \ldots\}$, $i, j \in \{1, \ldots, N\}$ similar to the active connection pair set $\mathcal{C}_{active}$. Regardless of the current behavior, it is essential to maintain the already coupled pairs. For each connection pair, we can determine on which robot the corresponding anchor locates by their relative pose. Without loss of generality and to simplify notations, here we assume the anchor of the connection pair (C_i, C_j) lies on robot i. By utilizing the polygon constraint in Equation (5.9), we maintain each connection pair by adding

the constraint $pip(C_i|R_j, C_j^r, C_j^l)$, where C_j^r and C_j^l denote the front right corner and the front left corner of the robot j respectively.

The MPC computation time depends highly on the number of variables and constraints. Our MPC runs in real-time at every iteration with a fixed time horizon H_m. However, the constraints are only essential for the first few time steps. Thus, we incorporate the constraints only in a shorter time horizon $H_c \leq H_m$. At the time t, the MPC formulation is shown as follows:

$$\min_{x,u} W_f J(t+H_m|t) + W_c \sum_{k=0}^{H_c} J_c(\mathcal{C}_{conn}, t+k|t) + W_m \sum_{k=0}^{H_m-1} J(t+k|t) + W_s u^\intercal u \tag{4.8}$$

$$\text{s.t. } x(t|t) = x(t) \tag{4.9}$$

$$x(t+k+1|t) = x(t+k|t) + f\left(x(t+k|t), u(t+k|t)\right)\Delta t \quad \text{for } k = 0, \ldots, H_m - 1 \tag{4.10}$$

$$pip(C_i(t+k|t)|R_j(t+k|t), C_j^r(t+k|t), C_j^l(t+k|t)) \quad \text{for } k = 1, \ldots, H_c, (C_i, C_j) \in \mathcal{C}_{conn} \tag{4.11}$$

$$x(t+k|t) \in \mathcal{X}, u(t+k|t) \in \mathcal{U}, \ k = 0, \ldots, H_m \tag{4.12}$$

At each time instance t, we compute the above MPC formulation and execute the first control input $u(t|t)$. The cost function in Equation (4.8) includes the cost of the target behavior from the behavior set $J(t+k|t) \in \mathcal{J} = \{J_c, J_g, J_v\}$, the weight of the terminal cost W_f and stage cost W_m for this behavior, a small cost with a weight of W_c for maintaining the already connected pairs $\mathcal{C}_{conn}$, and a smoothness cost with a weight of W_s for the control signal u. The optimization has four sets of constraints. The initial constraint in Equation (4.9) is a hard state constraint so that the optimization starts from the current robots' states. Equation (4.10) is the constraints for dynamics update with the unicycle model in Equation (4.1). As mentioned previously in Section 4.3.1, we utilize the first-order Euler integration instead of a more accurate RK4 method. Due to our short time horizon H_m and the simple dynamics model, the Euler method gives comparable results as RK4 but

with less computation time. The connection pair constraints are incorporated in Equation (4.11) as introduced in section 4.3.3. Note that this constraint is only required within the constraint horizon $H_c \leq H_m$ to speed up the computation of this optimization further. We also incorporate actuation constraints in Equation (4.12). Acceleration constraint set $\mathcal{U} = \{u | u_{\min} \leq u \leq u_{\max}\}$. The velocity limits have a similar "butterfly" shaped pattern as in [76], where the robot cannot stably rotate in place with a zero linear velocity, i.e., a high angular velocity comes with a high linear velocity. A controller based on only the current time instance can enforce the control signal to lie in the positive half-plane given target linear velocity is positive, and vice versa. However, this conditioned statement cannot be directly incorporated into the MPC framework as a constraint. Therefore, we define the feasible velocities in each state as $\mathcal{X} = \{x | v \leq |\frac{w_{\max}}{v_{\max}} w|\}$. This constrains the robot from not rotating in place in this MPC-based setup to further prevent the anchor from decoupling during the process.

With this MPC framework, we can obtain the locally optimal control signal u for each robot at every time instance. We then introduce when the MPC is computed and how it integrates with the current simulation and hardware system.

4.3.5 Connection Pairs and Execution

We first generate a list of connection pairs $\mathcal{C}_{goal} = \{(C_i, C_j), \ldots\}$ to couple based on a given target configuration and the distance-induced graph. Different from [76], which only requires the alignment of individual points, our anchor design requires a sequential coupling process. Thus, we augment each connection pair (C_i, C_j) as in Algorithm 7. For each connection pair (C_i, C_j), where C_i is a connection point on robot i, we initialize its status to be *decoupled*. Once the head of the anchor on one robot aligns with the opening on the other robot, the status will become *head_aligned*. Once the anchor is fully inserted, the status is *head_inserted*. Since the anchor-based connection locates only on the front and back, and the side connections are the knob-hold connection, we can define the type of connection based on the position of C_i and C_j. Unlike the knob-hole connection, the anchor-opening connection is not symmetric. Thus, we take note of which robot the anchor is on for this given connection pair and define the corresponding anchor head position - the C_i' point is

defined at $p_{C_i'} = p_{C_i} + [-l, 0]$ if the anchor locates on robot i, and vice versa for C_j', where l is the length of the anchor.

Algorithm 4 Augment Connection Pairs

Input: $\mathcal{C} = \{(C_i, C_j), \ldots\}$: set of connection pairs
Output: $\tilde{\mathcal{C}}$: augmented set of connection pairs
Initialize: $\tilde{\mathcal{C}}$={}

```
function AUGMENTPAIRS(C)
    for (C_i, C_j) in C do
        (C_i, C_j).status = decoupled
        (C_i, C_j).type = anchor or knob
        (C_i, C_j).anchor_index = getAnchorIndex(C_i, C_j)
        (C_i, C_j).head = (C_i', C_j') based on anchor_index
        C̃.append((C_i, C_j))
    end for
    return C̃
end function
```

During the execution of the connection pair list, we check if the status of each connection pair should be updated or not. As shown in Algorithm 8, based on the current robot poses, we decide the connection status by checking if the connection point, or the anchor head, lies within the polygon formed by the other robot within a small margin ϵ. The lists of active and connected pairs are also updated accordingly.

The connection pair alignment algorithm for the anchor-opening type of connection is shown in Algorithm 9. We obtain the goal connection pairs based on a distance-induced graph from the initial robot poses and store it for future use. We then augment the pairs as in Algorithm 7. We then find the non-conflicting pairs to execute, which are then assigned to the active list $\mathcal{C}_{active}$. We subsequently compute the control signal u from the MPC introduced in Equation (4.8) and update the connection pair status. The control signal acts directly on the robots in simulation. Details of the hardware system can be found in Section 4.4.3.

Algorithm 5 Update Connection Pair Lists

Input: $\mathcal{C}_{conn}$: connected pairs, $\mathcal{C}_{active}$: active pairs, $\tilde{\mathcal{C}}$: augmented pair list, ϵ: threshold
Output: $\mathcal{C}_{conn}$: connected pairs, $\mathcal{C}_{active}$: active pairs

```
1: function UPDATEPAIRS(C_conn, C_active, C̃, ϵ)
2:     for (C_i, C_j) in C_active do
3:         a = (C_i, C_j).anchor_index
4:         if (C_i, C_j).status is decoupled then
5:             if (C_i, C_j).head in polygon(R_¬a, ϵ) then
6:                 (C_i, C_j).status ← head_aligned
7:                 continue
8:             end if
9:         end if
10:        if (C_i, C_j).status is head_aligned then
11:            if C_i in polygon(R_j, ϵ) then
12:                (C_i, C_j).status ← head_inserted
13:                continue
14:            end if
15:            if (C_i, C_j).head not in polygon(R_∼a, ϵ) then
16:                (C_i, C_j).status ← decoupled
17:            end if
18:        end if
19:        if (C_i, C_j).status is head_inserted then
20:            C_active.remove(C_i, C_j)
21:            C_conn.append(C_i, C_j)
22:        end if
23:    end for
24:    return C_conn, C_active
25: end function
```

Algorithm 6 Align Connection Pairs

Input: T: target configuration, x: robot states at time t

Output: u: control input

Initialize: $\mathcal{C}_{conn}$={}, $\mathcal{C}_{active}$={}

function ALIGNCONNECTIONPAIRS(T, x)
    $\mathcal{C}_{goal} \leftarrow$ assignConnectionPairs(T, x) if not assigned
    $\tilde{\mathcal{C}} \leftarrow$ augmentPairs($\mathcal{C}_{goal}$)
    $\mathcal{C}_{active} \leftarrow$ assignActivePair($\tilde{\mathcal{C}}$, x) without conflict
    $u \leftarrow$ MPC(x, $\mathcal{C}_{conn}$, $\mathcal{C}_{active}$)
    $\mathcal{C}_{conn}$, $\mathcal{C}_{active} \leftarrow$ updatePairs($\mathcal{C}_{conn}$, $\mathcal{C}_{active}$, $\tilde{\mathcal{C}}$, ϵ)
    return u
end function

4.4 Results and Experiments

In this section, we first show the characterization of the 3D-printed anchors. The force profile of pushing the anchor front and back is presented. This shows that with the pushing force of each individual robot, the anchor can be pushed into the opening of another robot while it cannot be directly pulled out from the opening. In Section 4.4.2, we simulate the soft anchor with this force profile and test our MPC framework with the coupling behavior. In Section 4.4.3, we discuss in detail the structure of our hardware system and show a sequence of screenshots of robots coupling, decoupling, and going down off a platform as a chain.

4.4.1 Characterization of Anchors

To obtain the force profile of the 3D-printed anchor, we set up our experiment as in Figure 4.11. A linear actuator with a distance encoder is secured on an optical table. A load cell of 500 g is attached to the linear actuator with a 3D-printed PLA holder. An anchor is secured on the other side of the optical table. We collected data on pushing the anchor's beam from the front and back side, which resembles pushing the anchor into an opening as Figure 4.3a, and pulling an anchor when it is locked in the opening as in Figure 4.3b. During the experiment, an Arduino Uno sends position

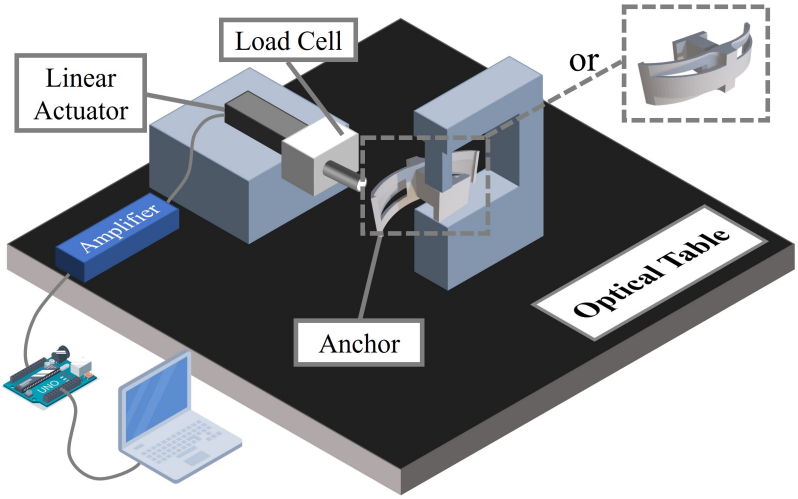


Figure 4.11: The force experiment setup to measure forces for pushing the anchor in the front or back depending on the positioning.

commands with an increment of 0.1 mm to the linear actuator. The signal from the load cell goes through an amplifier and is passed to the Arduino. Five data points are collected at each position, and an average force reading is saved. The experiments are conducted five times on three separately printed anchors respectively.

The result of the experiment is shown in Figure 4.12a. When performing a forward push, which resembles the anchor being pushed into the opening of another robot as in Figure 4.3a, the maximum force is less than 0.2 N. When performing a backward push, the average force needed for a 0.4 mm displacement is 0.6 N. The force drops at around 0.47 mm because of slippage due to the geometry of the anchor, which does not happen on the actual robot. The displacement needed for an anchor to be pulled out is above 3 mm. With the same setup as in Figure 4.11, we measured that the pushing force of a robot at maximum speed has an average of 0.5 N. This will allow the anchor to easily couple, and hard to decouple with a direct pulling force. To measure the holding force when the anchor locks the connection as in Figure 4.3c, we put weights on the robot with the anchor locked. The anchor can hold a load of up to 500 g, which is the weight of seven robots.

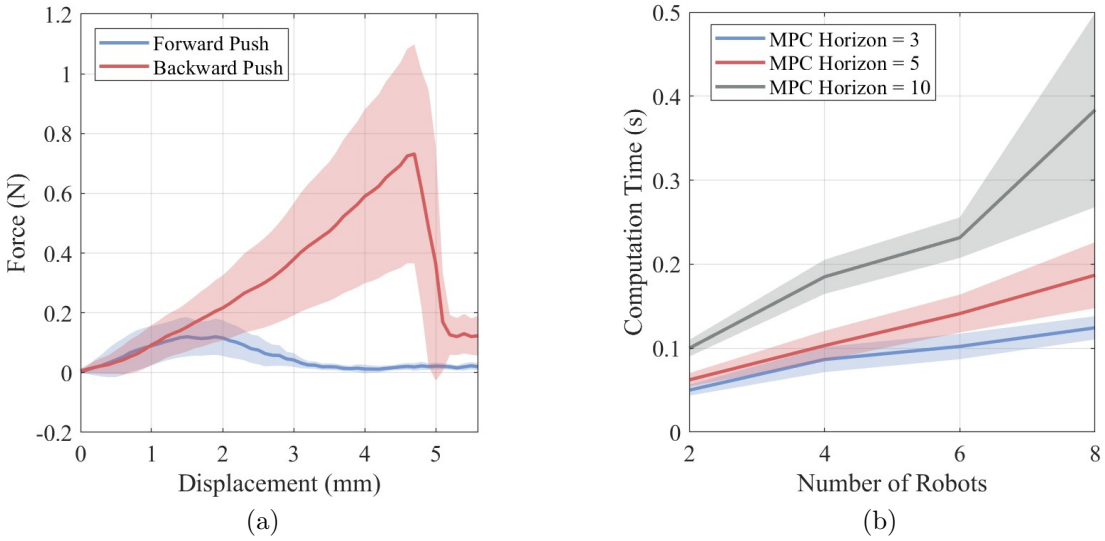
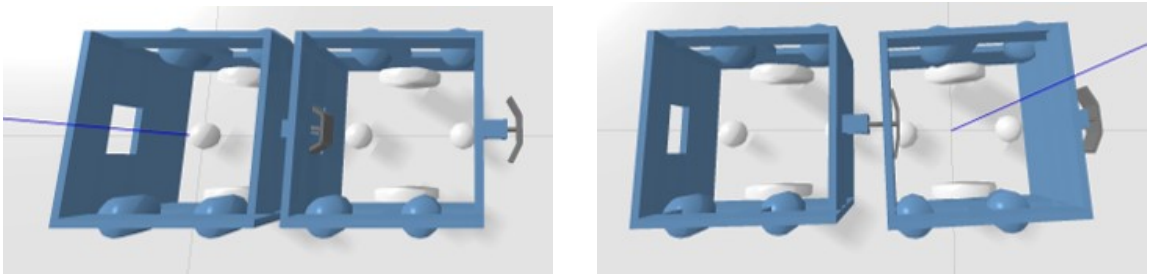


Figure 4.12: (a) Forces of pushing the anchor from the front and back. (b) Computation time versus the number of robots with different MPC horizons. No inter-robot constraints.

4.4.2 Simulation Experiments

We tested our robots with anchor and opening in the Bullet [13] simulation. The robot body is generated as a urdf file with a list of primitive shapes, i.e. box, cylinder, and sphere, which is experimentally more stable and accurate for collision checking compared with a custom mesh file. The robot body, anchor holder, and wheels are generated as one body, while the anchor is kept as a separate body. Since Bullet does not allow a floating joint, the anchor is placed inside the anchor holder without a joint. This multi-body setup enables stable collision checking between the anchor, anchor holder, and the opening from another robot. The soft anchor is simulated as a rigid three-bar linkage where the center bar is fixed to the anchor base, and the left and right bars are connected to the center bar with a rotational joint. The torque of the rotational joint will always try to bring the two side linkages to their resting position, with a maximum force limit corresponding to the current displacement, as shown in Figure 4.12a. The maximum torque from the wheels also has the same limit as the actual hardware system. Figure 4.13 shows the simulation of the robot and its anchor. In Figure 4.13a, the robot is pushing its anchor into the opening of another



(a) (b)

Figure 4.13: Simulation of (a) Anchor pushed into the opening of another robot; (b) One robot pulling the anchor when it is inserted.

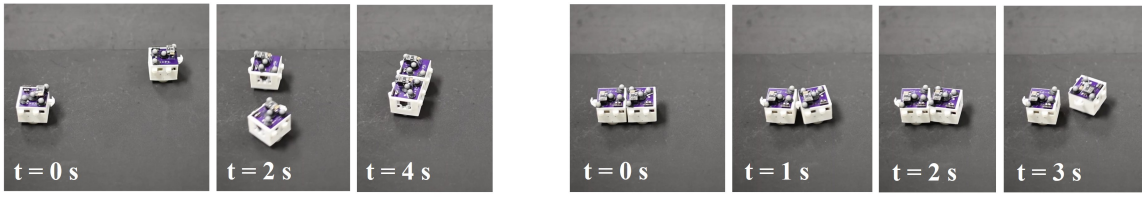


Figure 4.14: Left: Two robots couple with each other; Right: Two coupled robots decouple by wiggling.

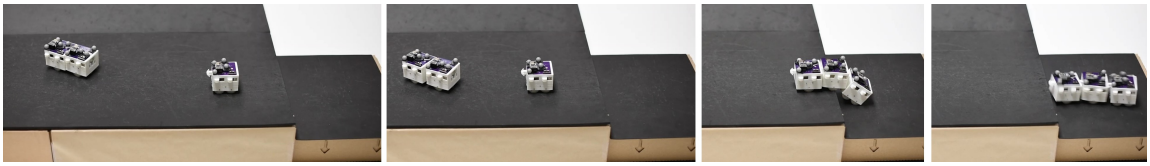


Figure 4.15: Two coupled non-pilot robots move towards the third pilot robot, couple to form an assembly and go down a stair.

robot. The side linkage bends accordingly. In Figure 4.13b, one robot is pulling its anchor when it is inserted. The beams of the anchor are compliant but cannot be pulled out due to the limitation of the wheel torque. These behaviors resemble the actual hardware system with soft anchors.

We tested our algorithm on a desktop with Intel Xeon 3.40 GHz CPU. The MPC framework is written in Python, interfaced with CasADi [3] with the non-linear optimization solver *ip-opt* [72]. The target behavior is to couple as a line with 2, 4, 6, and 8 robots, and the computation time of the optimization is shown in Figure 4.12b. The MPC horizon H_m tested are 3, 5, and 10, with the same constraint horizon $H_c = 3$. We can see that the computation time increases as the number of robot increase, and with a larger MPC horizon, the computation time increases significantly.

4.4.3 Hardware Experiments

The hardware system runs in ROS [59]. The MPC planner node runs at 10 Hz. It takes in the current robot poses, computes the optimal linear and angular acceleration, and outputs the corresponding linear and angular velocity. The optimization running in this MPC node is exactly the same as in the Bullet simulation while having a different interface on the hardware system. A separate high-frequency PID controller node runs at 30 Hz. The controller node takes the Vicon pose, calculates the instant linear and angular velocity within a small time window, and computes the wheel motor values based on the given command velocity from the MPC node.

A series of screenshots from our experiment video is shown in Figure **??**. Figure 4.14 shows two robot couple with each other and decouple by having one robot wiggle with a linear velocity bias. Figure 4.15 shows an assembly of two coupled robots connected with a pilot robot. They successfully formed a chain and went down a stair. More videos of the hardware experiments can be found in our supplementary video material.

4.5 Conclusions and Future Works

In this paper, we presented our soft anchor design that has an asymmetric force profile when pushing from the front and back. This makes the robots easy to couple

while having a sufficient holding force for the connection to carry high loads. When two robots are coupled, the anchor acts as a compliant joint that provides a limited rotation and translation. We also presented our MPC framework with polygon constraints to model the coupling behavior. The polygon constraints capture the geometry of connections. Thus, the assembly is flexible to rotate, move around the environment, and form 3D structures while maintaining their coupling status. The side knobs of the robot enable the robots to form rigid structures. With this setup, the robot swarm can form functional structures in an unstructured environment, e.g., bridges across gaps and ropes down a stair.

The current framework has limitations as follows. The centralized MPC is computationally expensive for more than eight robots. By moving the MPC to decentralized based on assembly segments and performing the computation in parallel, we can improve the scalability of the system. The current anchor system is designed for robots in centimeter-scale. Further experiments will be done to study how the anchor system scales with different sizes of robots. If the robot can be equipped with onboard sensors and creates a map of the unstructured environment, we can also computationally determine the required assembly shape based on the environment structure.

Chapter 5

Reconfigurable Robot Swarms for Terrain Traversal

5.1 Introduction

Collaborative swarms, both in biological and robotic systems, offer an unprecedented ability to collectively traverse unstructured and challenging terrain and form structures with their bodies. One such example from biological systems is ant colonies, which have been studied in scientific literature [17, 45]. Individual ants can link together to form bridges across gaps, create towers, and collectively forage.

Robot swarms routinely work together and perform collective tasks, such as shape formation[44, 52], construction [43], sampling [22], and exploration [9]. During these tasks, these robots act as individual agents, and do not couple together to form a functional structure to accomplish these goals. Modular robots, another type of multi-robot collective, utilize a collection of modules that couple together to form structures. While these assembled structures can configure into different topologies and locomote as a collective unit, each module has a limited ability to perform individual tasks. Bridging together the individual capabilities of robots in swarms along with an ability to couple together, like modular robots, to make functional structures enables new abilities in both swarm and modular robots [10, 23, 31, 71]. Here, we present PuzzleBots (Figure 5.1) - a reconfigurable modular robot system, in which robots can

dynamically form flexible yet strong structures. These robots maintain individual mobility and functionality, while also coupling to form structures like modular robots. PuzzleBots navigates the trade-offs in combining the functionality of swarm robots with structure formation in modular robots.

As shown in Figure 5.2, the development of reconfigurable modular robots presents unique challenges in balancing three interrelated yet often contradictory aspects: reconfigurability, mobility (including precise locomotion), and load-bearing capability with low power consumption. Achieving a high degree of reconfigurability, where robots can dynamically couple and decouple, typically necessitates a more flexible connection. However, maintaining a robust and strong connection with such flexibility can demand substantial power resources. This poses a dilemma: allocating significant power to the reconfiguration process can limit the power available for individual locomotion. Consequently, this limitation may impede the robots' ability to efficiently navigate and explore environments efficiently. Thus, the challenge lies in designing a system that harmonizes these competing requirements to optimize functionality across various terrains and tasks.

In previous works of modular robots that retain their individual mobility, mechanical and magnetic approaches have been explored to provide ways of connections between robots. Magnetic approaches enable dynamic coupling and decoupling between robots [28, 38, 48, 49, 54, 65, 82]. However, active magnetic connections, for example, electromagnets, require high power to operate. Permanent magnets do not require external power but have limited shear force for load-bearing capabilities. The connections formed by magnets are also rigid. Thus, additional actuators and electronics are required to introduce more degrees of freedom to the connections between robots. Mechanical approaches for coupling between robots include grippers [18], hook and loop fasteners [57], sockets [8, 37], and melting plastic [61].

Despite the advances in coupling mechanisms themselves, the range of motions for the connected structure still remains limited. While having individual mobility on each robot, the dexterity provided by these individuals is rarely considered during the development of coupling mechanisms for modular reconfiguration robots. Our previous developments of the PuzzleBots [75, 76, 77] utilize passive mechanisms to form functional structures. By controlling the precise positions and motions of each individual robot, they can form rigid or flexible assembly structures to facilitate

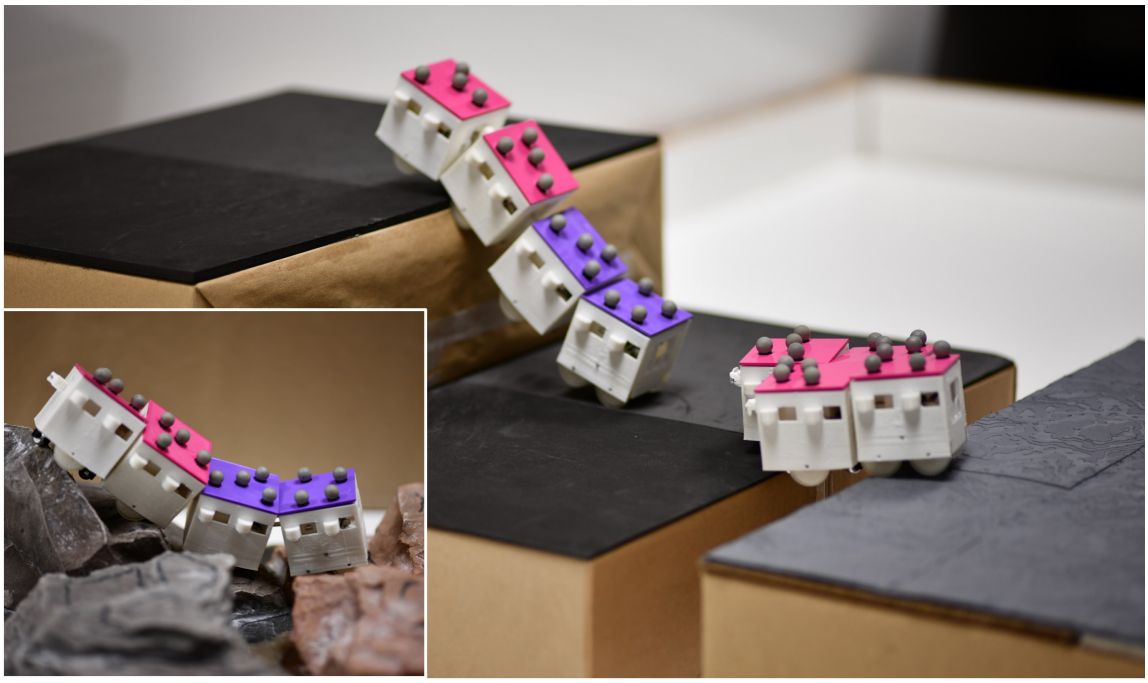


Figure 5.1: **Robots forming a chain and a mesh configuration to traverse challenging terrains.** The inset figure shows robots in an unstructured environment.

different use cases.

While the loosely coupled mechanisms bring flexibility to the formations, they also introduce unique challenges for modeling and controlling the robots. Most planning and control algorithms for modular reconfigurable robots focus on the configuration generation [34], lattice or dodecahedron-based reconfiguration [24, 50, 66], and pre-defined gaits and libraries [29, 42]. These methods are restrictive for flexible connections in [77] since the continuous configuration spaces cannot be fully explored and represented with a limited set of discrete actions. To best describe the geometric relationships between robots, we adopted polygon representations [64] that are commonly used for object avoidance to model our flexible coupling mechanism. In the context of multi-robot non-holonomic systems with non-linear dynamics, the adoption of Model Predictive Control (MPC) is widely used. MPC has proven effective in incorporating constraints related to dynamics, actuation, and safety. Safety constraints within multi-robot MPC formulations typically center around collision avoidance [33, 62, 81], which is a loosely-coupled type of constraint,

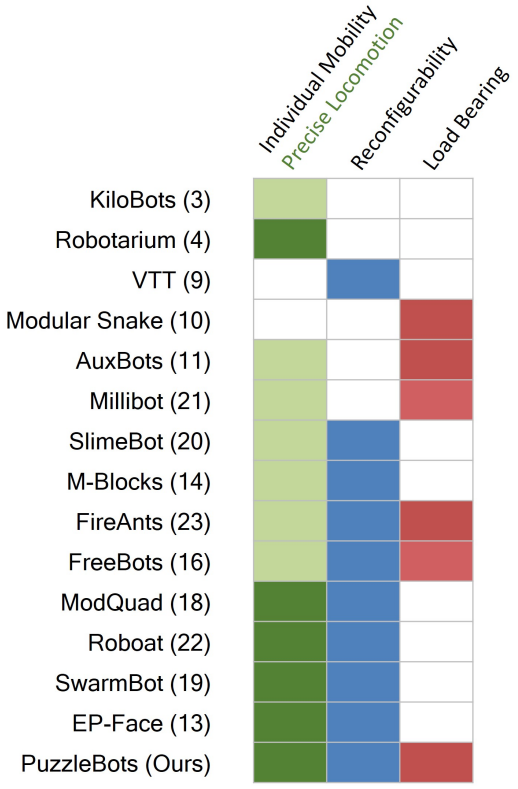


Figure 5.2: **Related works considering three aspects**: individual mobility of each module or robot (dark green if each one has precise locomotion), reconfigurability of changing to different topologies, and load-bearing capability of the connection mechanism.

and activated only when robots approach objects or other robots. On the contrary, maintaining the coupling state between robots is a dense constraint that is constantly activated.

Establishing a robust connection with active mechanisms typically demands high power, while strong passive mechanisms encounter difficulties in either effective coupling or decoupling. Achieving a balance among load-bearing capacity, limited onboard power, and coupling/decoupling flexibility poses a significant challenge. Moreover, achieving precise formation control in large-scale multi-robot systems brings additional challenges in terms of modeling and computation, especially due to the dense inter-robot coupling constraints we mentioned earlier. Leveraging our recent research [75, 76, 77], we present the updated PuzzleBot system. The PuzzleBot utilizes two hybrid passive coupling mechanisms: a rigid connection consisting of knobs and holes, and a flexible connection with an asymmetric anchor mechanism. These passive mechanisms are coupled and decoupled using solely the robot locomotion dexterity, activated by gravity and the environment. A combination of rigid and flexible connections enables sufficient load-bearing capability. On the controller side, the robots are equipped with onboard feedback to fully utilize their locomotion dexterity to facilitate the passive mechanisms. We also present a geometric modeling of the connections and our distributed MPC framework, enabling real-time control of the multi-robot system.

5.2 Results

5.2.1 Robot design overview

As shown in Figure 5.3A, each PuzzleBot is a mobile robot with onboard power, actuation, communication, feedback, and computation. Each robot is 5 cm in width, 5 cm in length and 5.5 cm in height. As shown in Figure 5.3B, each robot is equipped with two DC motors integrated with individual encoders, a WiFi module for communicating with the central computer, a microcontroller, and a 3.7V LiPo battery.

The system architecture is shown in Figure 5.3C. Each module (task assignment, distributed MPC controller, PID controller) is an independent node communicating

via Robotics Operating Systems (ROS). The user selects a desired behavior from an existing behavior library, e.g., line or mesh configuration, trajectory tracking, etc., and the task assignment node computes a series of connection tasks for the controller to execute. These connection tasks may include coupling or decoupling tasks, the maintenance of existing coupling states, as well as the target behavior. The task assignment is activated as needed when called by the distributed MPC node. A Vicon indoor localization system and markers on the robots are used to provide high-frequency feedback on robot poses. The distributed MPC node optimizes the target behavior and connection tasks based on the current robot poses, and sends the command velocities of each individual robot to the PID controller node on the central computer. The PID node computes the desired motor rotational velocities to each robot via TCP packages through WiFi. The PID node is of higher frequency and stabilizes to a given command velocity based on the robot poses from the Vicon system. The robots then stabilize to the target motor velocities with onboard encoder feedback. The modular architecture facilitates the integration of the MPC controller and task assignment nodes with both the hardware controller and the simulation environment. The simulation environment is configured using parameters that have been measured and collected from the hardware components, thereby making the simulated scenarios close to the real-world hardware systems. Detailed information regarding the simulation setup is provided in supplement Section 5.5.4. This design enables efficient testing and execution for various scenarios within a simulator, subsequently allowing for a direct transition to the hardware platform without any additional effort.

To achieve the maximal performance for terrain traversal, we designed the Puzzle-Bots to be a heterogeneous team of robots, consisting of *non-pilot* and *pilot* robots. The non-pilot robot has one actuated wheel on each side, and two ball bearings in the front and back. This design enables high-precision control over the robot's velocities and provides sufficient agility for its tasks. However, since the two ball bearings are not actuated, the climbing ability of the non-pilot robots is limited. The pilot robot has two actuated wheels on each side powered by one motor, which gives maximum climbing capabilities. However, this limits its agility in terms of rotation due to the lateral friction from the four wheels. By creating a heterogeneous team combining the pilot and non-pilot robots, we can utilize the traction and climbing capability of

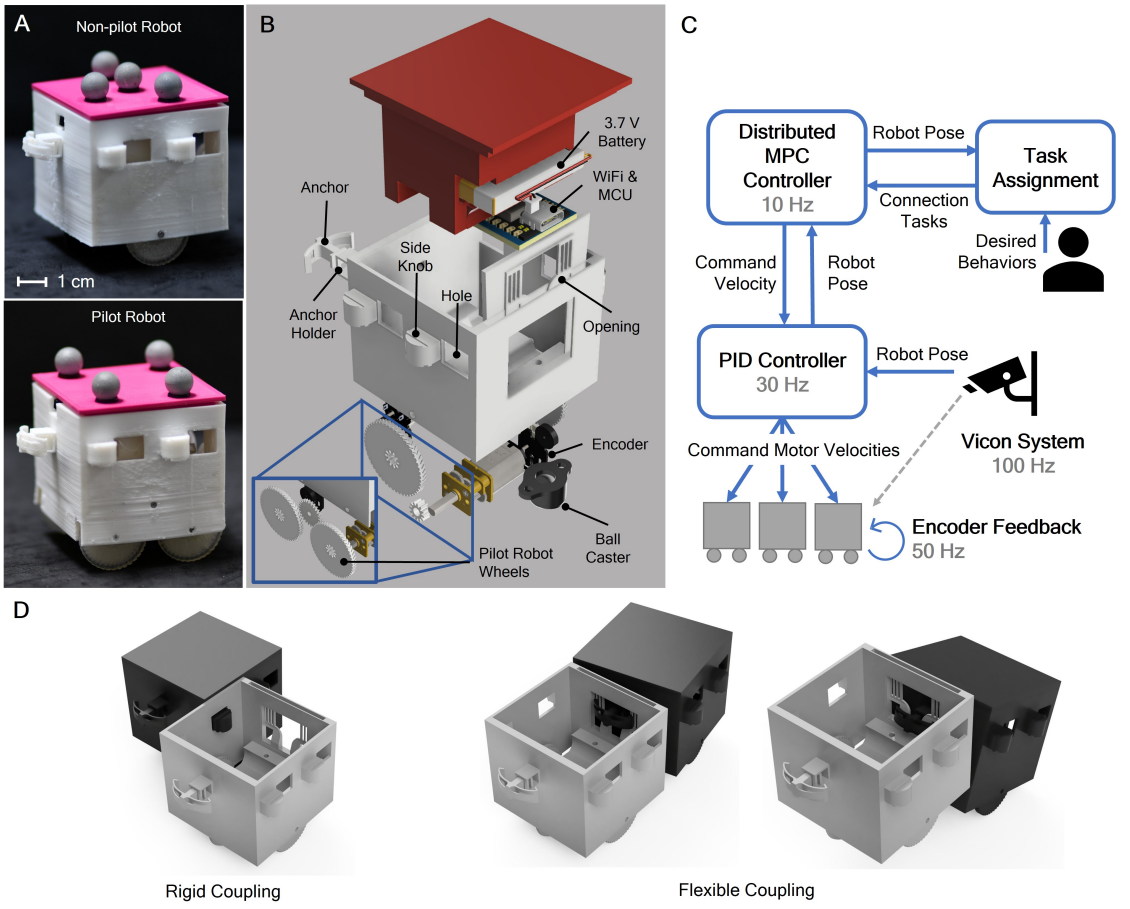


Figure 5.3: **Robot design overview.** (A) *Non-pilot* robot with two actuated wheels (top) and *pilot* robot with four actuated wheels (bottom). (B) An exploded view of the CAD model shows the robot with an onboard battery, communication, and computation module, and actuation. The front and back side consists of a flexible anchor coupling mechanism, while the left and right sides consist of a rigid knob and hole mechanism. (C) The system architecture includes task assignment, distributed model predictive control, and lower-level PID controller, and an external Vicon localization system. (D) Rigid (left) and flexible (middle and right) coupling states between two robots. The flexible coupling gives degrees of freedom in the yaw and pitch axis.

the pilot robot, while maintaining the agility of the non-pilot robots to form diverse configurations and perform tasks with high precision.

Our passive coupling mechanism is shown in Figure 5.3D. The robots, coupled with the flexible connection, retain the ability to rotate within a small range of degrees on a flat platform. When the robots traverse a discontinued terrain, e.g., a gap or height drop, the connection is activated by gravity, and the anchor will be locked in place, providing a strong holding force of up to seven robots. Additionally, the robots can also couple on the side with a rigid mechanism consisting of knobs and holes on the robot body, when a stronger rigid connection is required to form a mesh. Our passive coupling mechanism relies solely on the agility from robot locomotion, and is activated with environment interactions. Thus, this asymmetric anchor design makes it easy to push in, but hard to pull out with the forces from robots alone. The details force profile and additional experiments are provided in the supplement section 5.5.3.

5.2.2 Configuration formations

We conducted experiments on two major configurations: *line* and *mesh* configuration. The line configuration is defined as robots coupling in the front and back with the soft anchor mechanism. The mesh configuration is defined as a combination of multiple rows of the line configuration with an offset of half a robot length so that the front-and-back anchor connection is locked in place by the side knobs. This line configuration forms a flexible chain structure, while the mesh configuration forms a rigid structure.

In Figure 5.4A, the coupling process between two robots is presented. Beginning with the robots in unaligned positions, the objective is to insert the back anchor of one robot into the opening of another. This task involves a precise alignment phase, taking approximately five seconds to position the anchor head in line with the opening. Following this, an additional ten seconds is used for the complete insertion of the anchor. As mentioned in Figure 5.10, the force exerted by the robot exceeds the bending force of the anchor. However, the insertion process is not instantaneous. This delay is attributable to the control mechanism, which does not always output the motor velocities at maximum torques. Additionally, the interaction and disruptions at the contact points between the anchor and the opening also contribute to the

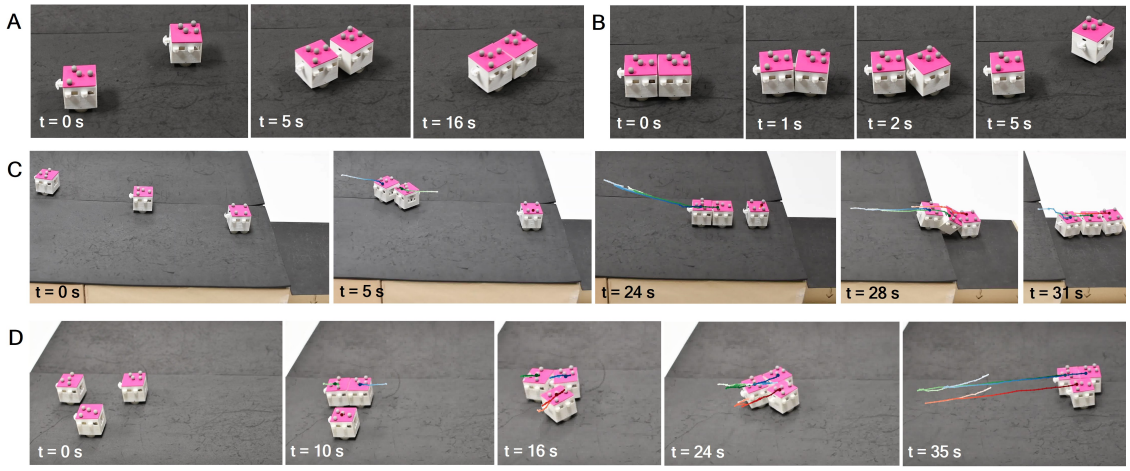


Figure 5.4: **Coupling and decoupling process.** (A) Two robots couple together in a line. (B) An assembly of two robots decouple out of a rigid line. (C) Three robots start on an elevated platform, aligning themselves into a compliant line configuration before navigating a descent in height. (D) Three robots start from misaligned positions and form a mesh configuration. The colored lines show the trajectory from the previous time step.

extended duration of the coupling process. Figure 5.4B illustrates the decoupling procedure of two robots initially joined together, with the anchor of the front robot inserted into the other. The objective of this process is to decouple and prepare for further individual tasks. With the decoupling controller introduced in Section Controller Details, the front robot uses a combination of wiggling and forward motion. This motion results in the disengagement of one leg of the anchor from the other robot within the first second. Subsequently, by the end of two seconds, the entire anchor is detached, resulting in the successful decoupling of the two robots.

As shown in Figure 5.4C, two non-pilot and one pilot robot start from unaligned positions on a 45 mm high elevated platform. The target behaviors are to form a line configuration and move forward. The task assignment generates a series of connection tasks - first, align two non-pilot robots, then move to couple with the pilot robot as a connected component while maintaining the connection, and finally move forward once the pilot robot is successfully coupled. As the robots move forward toward the edge of the platform, the front robot descends under gravity tethering the rest with the soft anchor, securing the front robot in place as part of a connected component.

The front pilot robot with four actuated wheels and the inherent compliance of this line structure ensures effective traction for the wheels from ground contact.

Figure 5.4D shows the process of forming a mesh configuration with three non-pilot robots. The target behavior is to form a mesh configuration with two robots in the first row and one robot in the second row. The task assignment node generates a series of connection tasks - first, align the front-back connection, and then align the side connection while maintaining the already coupled connection. We notice that the side connection takes a significantly longer time to couple. This is due to the fact that the side knob-and-hole connection demands less than one millimeter precision for alignment, in contrast to the soft anchor mechanism, which allows for a broader range of tolerance. Furthermore, low-resolution onboard encoder feedback and noisy localization measurements extend the time required for the robots to adjust and align.

5.2.3 Controller details

The feedback within the system has two parts as shown in Figure 5.3C: 1) the onboard encoder motor velocity feedback and 2) Vicon robot pose feedback. We present the details of both the low-level velocity control and the high-level MPC-based controller with this feedback information.

The onboard velocity stabilization controller utilizes a Proportional-Integral (PI) based configuration, incorporating both proportional and integral gains. Achieving precision control for a centimeter-scale robot is challenging, primarily due to limitations in space and power, which constrain the use of highly accurate sensors. In the case of a standard magnetic wheel encoder, there exists an inverse relationship between reading frequency and resolution, whereby higher reading frequencies result in reduced resolution. Given the need for precision in the coupling behaviors of our system, this limitation must be considered. To balance between fast system response and encoder accuracy, we choose to implement a 50 Hz onboard feedback loop. This selection translates to an encoder resolution of seven, meaning that the system can effectively differentiate seven distinct velocity levels within the 0 to 450 RPM range, or 14 velocities considering the forward and backward motion, i.e. -450 to 450 RPM. To expedite convergence and mitigate oscillations effectively, we utilize three sets of PI values tailored to stabilize different ranges of target velocities. By tuning the motor

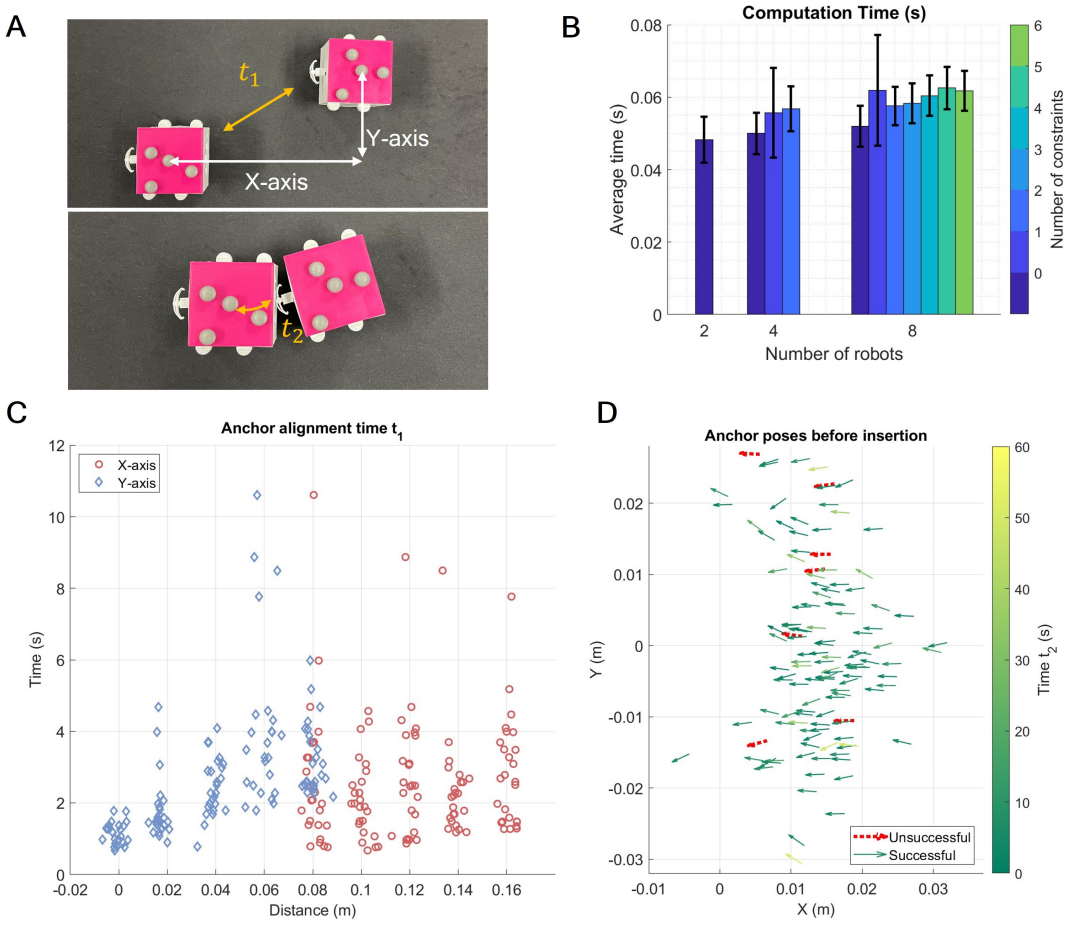


Figure 5.5: **Robot coupling experiments.** (A) Two stages of the coupling behavior: 1) alignment of the anchor head and the opening on another robot, 2) anchor insertion after alignment. (B) The computation time of the distributed MPC optimization for a system of two, four, and eight robots, with different number of constraints based on the coupling status. (C) Time for anchor alignment t_1 with different starting positions in the X and Y axis. (D) Both the successful and unsuccessful anchor poses before insertion (after alignment) with the insertion time t_2.

feedback performance across target velocity differences of 123, 265, and 495 RPM, we obtained the optimal PI parameters for these specific checkpoints. In between these checkpoints, interpolations were used to ensure smooth transitions between these parameter sets. We encountered difficulty in directly assessing the performance of onboard velocity stabilization, mainly due to the primary function of the Vicon system being pose feedback rather than velocity measurement, which is prone to significant noise. Instead, we opted for an indirect evaluation method by measuring the trajectory tracking error of a single robot, yielding an average error of 1.8 mm.

In executing the coupling behavior, our process starts with the generation of a connection pair list based on the desired configuration. Then the robots perform the connection pair alignment tasks in parallel. The optimization of the controller is executed on a central computer within a distributed MPC framework. In this framework, each robot operates within a dedicated thread and treats other robots as numerical counterparts, facilitating scalability to accommodate a large number of robots and constraints. This process generates a set of command velocities, which are then transmitted to each individual robot via WiFi. Subsequently, these command velocities are stabilized with the onboard PI controller. For each connection pair, there are two stages as shown in Figure 5.5A. Two robots first align the anchor head on one robot with the opening of the other robot. Once the anchor head reaches a feasible region, two robots collaborate in inserting the anchor further inside by converging upon each other's positions. We ran 125 experiments of this coupling process with different initial positions, and documented the time spent on the controller optimization, the first alignment stage (t_1), and the second insertion stage (t_2).

In Figure 5.5B, the computation time for running on our Intel i9 CPU with a 3.0GHz processor is illustrated for two, four, and eight robots performing line configuration tasks. The inclusion of one inter-robot constraint occurs once a successful coupling between two robots is achieved, and the corresponding number of constraints is also documented. Our distributed MPC framework demonstrates scalability within a short time duration of 0.08 seconds in all scenarios. There is a minimal increase in computation time observed from two robots to eight robots. In the scenario involving eight robots, although there is an increase in computation time as additional constraints are introduced, it remains sufficiently fast for real-time computation,

which shows the efficiency and effectiveness of our approach.

For the given connection pair task, our MPC controller minimizes the distance between corresponding connection points on each robot's body - specifically, between the anchor head position on one robot and the center of the opening position on another robot. This optimization is subjected to actuation and dynamics constraints, based on a differential drive model. To assess the efficiency of this precise alignment process, we conducted a series of experiments. One robot starts from a fixed position, while the other robot is positioned at approximately 25 different positions with zero heading angle, ranging from 0.08 to 0.16 meters along the x-axis and 0 to 0.08 meters along the y-axis. Each position underwent five runs, resulting in a total of 125 experiments. These positions were transformed into the frame of the other robot to obtain the exact relative positions. In Figure 5.5C, we present the time spent, denoted as t_1, during the anchor head alignment process. We see that 95.2% of the trials successfully achieved anchor head alignment within five seconds. The maximum time recorded for this alignment process was 10.6 seconds. Considering the relatively small initial separation distance, approximately three times the body length maximum, it is noteworthy that the alignment time does not appear to be significantly affected by variations in the x-axis distance. However, a more noticeable increase in alignment time is observed with greater y-axis separation. This can primarily be attributed to the fact that the robots require additional time to align their heading angles in order to face each other.

Once the anchor head alignment has been achieved within a threshold of 3.5 mm, the robots insert the anchor head into the opening by minimizing the distance between them. We record the time spent and capture the anchor holder pose before insertion. Note that due to the passive nature of this structure, we can only access the projected anchor holder pose, as direct access to the anchor head poses is not available. We set a total time limit of 60 seconds ($t_1 + t_2 \leq 60$). As illustrated in Figure 5.5, our experiments achieved a success rate of 94.4%. We did not observe any significant patterns between the insertion time and the initial anchor holder pose. The average completion time for the entire process is 9.38 seconds, with 87.2% of the trials successfully completed within a 30-second time frame, further highlighting the efficiency of our approach.

5.2.4 Terrain traversal experiments

We demonstrate that the Puzzlebot reconfigurable multi-robot system, with passive connections, exhibits enhanced terrain traversal capabilities compared with individual robots with no physical connections. The integration of both rigid and flexible connections enables the robots to navigate across a broad range of terrains, namely rough terrains, stair-like configurations, and terrain gaps. Our experimental evaluations were conducted across three sets of parametrized terrains.

We tested the performance of terrain traversal for a connected assembly versus individual robots on rough surfaces. Apart from the smooth surface we used in Figure 5.4, we generated random surfaces with variances of 1 mm, 2 mm, 3 mm, and 5 mm [1, 58, 60], 3D printed these surfaces, and coated them with paint. We ran our trajectory tracking behavior on 1) one single non-pilot robot, and 2) two non-pilot robots coupled in the front and back with the flexible anchor mechanism. For each surface, robots start from five different locations two times, with a reference trajectory of a straight line centered at the starting point. The resulting trajectories are shown in Figure 5.6A. We see that coupling two robots together introduces small disturbances to the trajectory following behavior. However, it helps the robots traverse surfaces with higher roughness, by creating more contact points with the terrain and pushing and pulling robots to move forward. Figure 5.6B presents the quantitative analysis of boxes showing the 0.25 and 0.75 quantiles of the data, a line inside the boxes showing the median, outlier points showing data outside of the 1.5 interquartile range, and the lines of non-outlier minimum/maximum. In Figure 5.6B(i) of the ratio of average distance, we see that both the individual robot, and the chained robots were able to traverse the given 0.42 m trajectory within 20 seconds in all trails. This observation shows the collective ability of chained robots to navigate and overcome challenging uneven terrains that a single robot cannot traverse. On a notably challenging surface with a variance of 5 mm in Figure 5.6A(v), the chained robots demonstrate a median performance of three times that of a single robot. Figure 5.6B(ii) shows the average linear velocity for each trail. Non-smooth surfaces significantly impact trajectory-tracking behavior by introducing disturbances to the robot's motion. The robot in the chained formation exhibits higher average velocity compared to individual units, particularly on surfaces with larger variances.

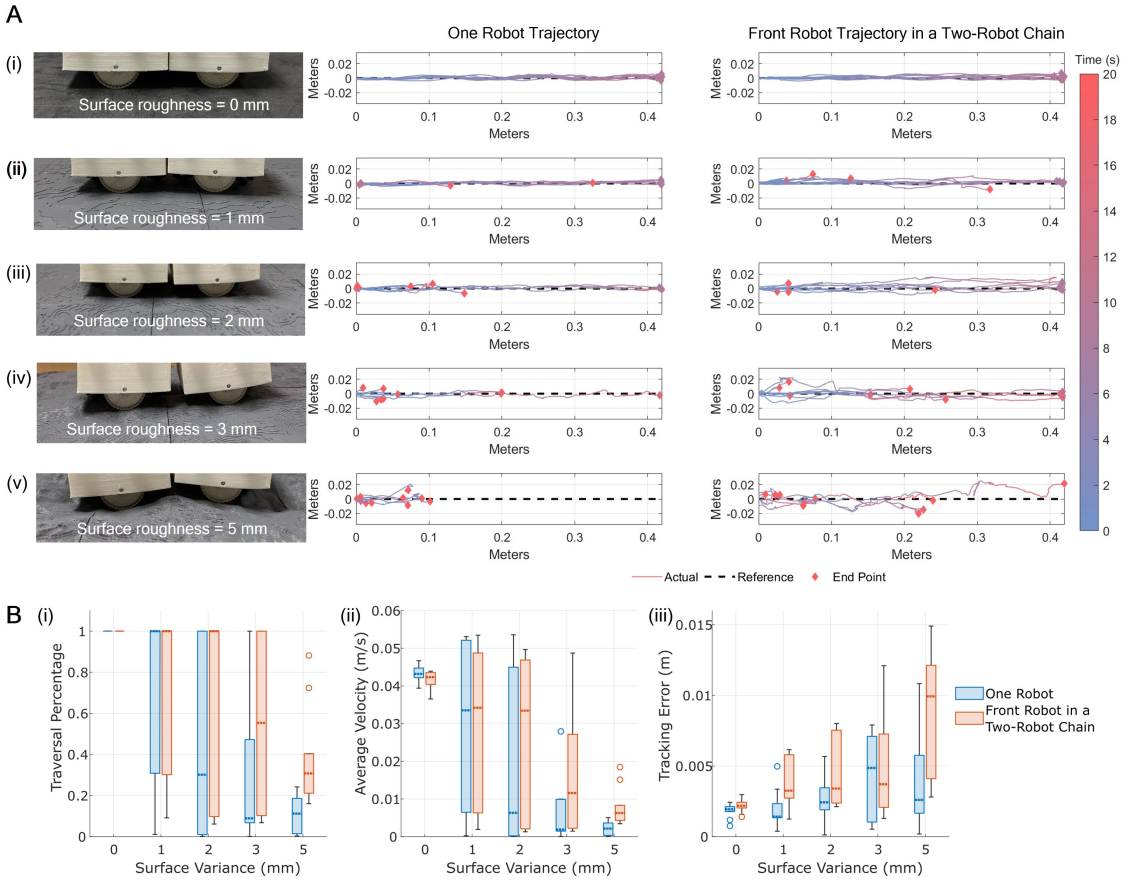


Figure 5.6: **Rough terrain traversal experiments.** (A) Trajectories of 1) one robot, and 2) front robot in the two-robot chain, traversing across surfaces with different variances of roughness within 20 seconds. (B) Quantitative analysis of the trajectories includes (i) the average ratio of the actual distance traversed with respect to a given designated target distance, (ii) the average linear velocity of the robots, and (iii) tracking error.

Note that on the surface with 5 mm roughness, the velocity variance is relatively small as the robots are frequently stuck in ground holes. While chained robots enhance performance in traversing challenging rough terrains, they concurrently introduce disturbances in terms of tracking error when compared to individual robots due to their inter-robot interactions. Figure 5.6B(iii) illustrates that the tracking error of the chained robot surpasses that of a single robot, which becomes more significant on surfaces with higher roughness, since the chained robots frequently have contact with each other during the rise and fall of terrain.

In our experiments, we assessed the performance of the robot assembly on platforms with smooth surfaces under two distinct scenarios: 1) two platforms of identical heights separated by a gap, and 2) two platforms of differing heights with no gap. These tests involved systems of one, two, four, and eight robots. The configurations tested included both line and mesh formations. Additionally, the systems were evaluated with varying numbers of pilot robots: none, one, or two. The heading angles of these pilot robots varied from 0 to 50 degrees, in increments of 10 degrees, to show their impact on system performance under these conditions. The results are shown in Figure 5.7.

Figure 5.7A provides a visual and quantitative representation of the parameters measured during the experiments. The variable w represents the width of the space between two platforms of equal height, a measurement for understanding how wide a gap the robot assembly can traverse. The variable h indicates the height of the obstacle, specifically the height difference between two platforms. The width w is tested in an increment of 10 mm, and h is in an increment of 6 mm. These two variables are essential for assessing the robots' capability to traverse discontinuous terrain, overcoming horizontal and vertical disparities. The heading angle θ is defined as the angle of the robots' forward movement relative to the platform edges, which is later compared with the assembly diagonal angle. Lastly, α denotes the diagonal angle of a given robot configuration. This angle corresponds to the longest part of the robotic assembly, providing insight into the geometry properties of the assembly and its terrain traversal performances.

Figure 5.7B illustrates bar charts showing the maximum gap width and the maximum descending height that a robot assembly can traverse an average of more than 80% of the robot successfully completed the task, independent of heading angle

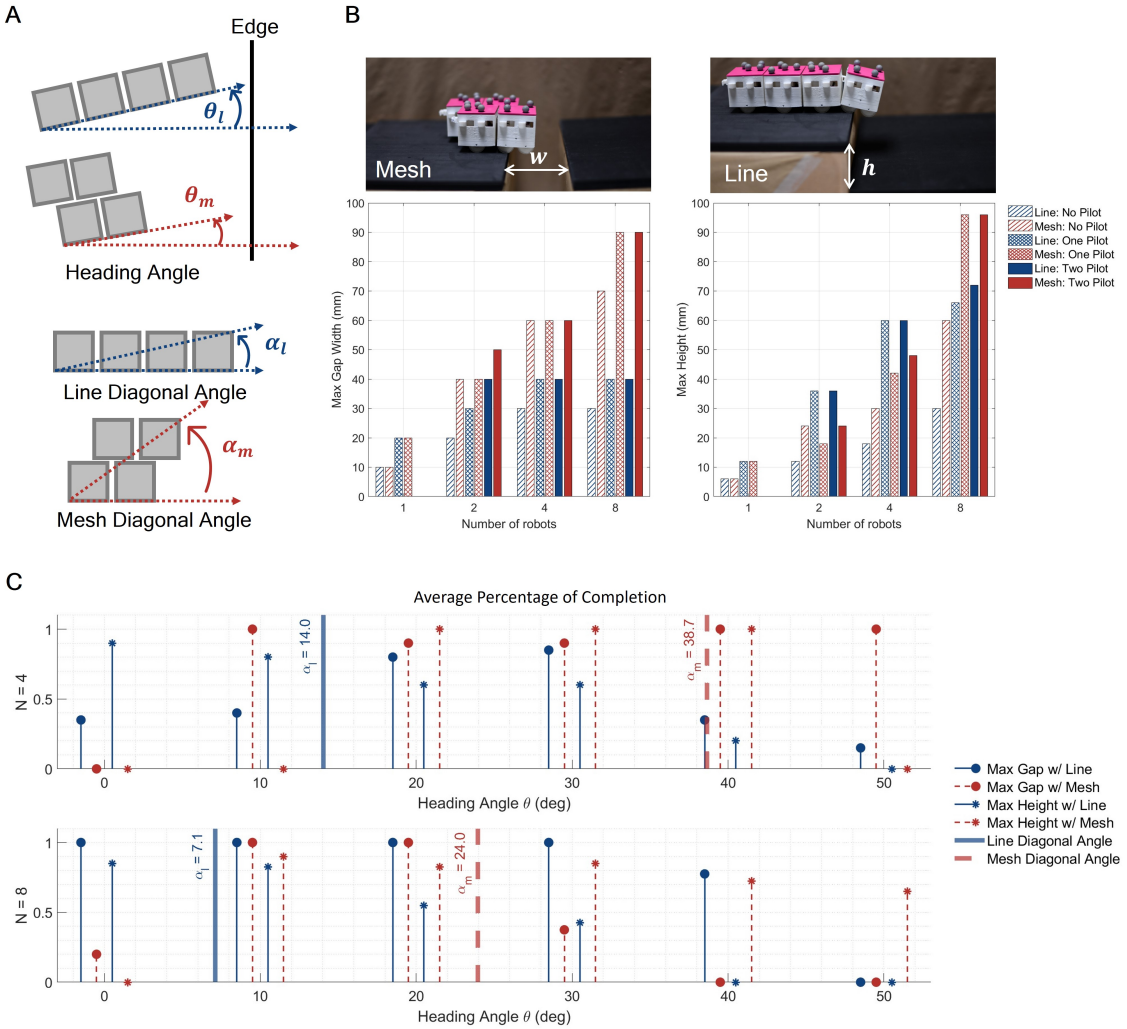


Figure 5.7: **Gap crossing and obstacle descending experiments.** (A) Dimensions related to the experiments of gap crossing and obstacle descending: w is the width between two platforms of the same height, h is the height of the obstacle, θ is the heading angle, and α is the diagonal angle of a given configuration. (B) The maximum gap width, and maximum height a given assembly can traverse with a percentage of completion $> 80\%$. (C) The average percentage of completion of line and mesh configurations for gap-crossing and obstacle-descending tasks with respect to various heading angles.

variations. Since it is necessary for the front of the assembly to reach a platform before the center of mass of the entire structure moves off a platform and loses balance, an assembly is capable of crossing a gap or descending from a height that is up to half the length of the assembly itself. This limitation arises because the assembly's front wheels must make contact with the opposing platform or ground prior to the assembly's center of mass falling off from the edge. This set of experiments shows the capability of traversing different types of discontinuous terrains, and the impact of different assembly configurations on this traversal capability.

For gap traversal experiments, the data demonstrates that the mesh configuration consistently surpasses the line configuration in performance. In detail, a mesh configuration with a four-robot assembly enhances performance by 50%, and this enhancement escalates to 125% with an eight-robot assembly relative to a line formation with two pilot robots. Considering that pilot robots are equipped with four actuated wheels in contrast to the two on non-pilot robots, a pilot robot has the capability to traverse a 20 mm gap, representing 40% of its own body length, which is double the traversal capability of a non-pilot robot. Introducing one pilot robot into a system with non-pilot robots boosts gap-crossing capability by 50% for two robots, and by 40% in a four-robot assembly. Due to the flexible connections in the line formation, the robot assembly tends to bend significantly downward. Therefore, having front pilot robot wheels helps the assembly climb onto the other platform. Nonetheless, the benefit from the pilot robot is not significant for the mesh configuration, with no enhancement for assemblies of four robots or fewer. This is attributed to the rigid connections within the mesh formation, preventing substantial downward bending of the assembly. However, with larger assemblies, such as those with eight robots, the inclusion of one pilot robot leads to a performance improvement of 45%. This improvement is due to the fact that with a greater number of robots, the mesh assembly still experiences a slight downward deflection.

For obstacle-descending behavior, data indicates that with two and four-robot assemblies, the line formation surpasses the mesh formation by 25% to 100%. The reason is that the flexible connections within the line formation offer compliance with the environment structure, ensuring that the front robot maintains ground contact before the entire assembly falls off from the edge. In contrast, for eight-robot assemblies, the mesh formation outperforms the line formation by 33%. This issue

is related to the length of the chained assembly. As it becomes elongated, the front robot may become perpendicular to the ground, preventing the wheels from making contact. The inherent rigidity of the mesh formation ensures that the front robot is more likely to have contact with the ground, especially when the height difference is less than half the length of the assembly.

Figure 5.7C presents the average percentage of robots successfully landing on the other platform (or ground) for both line and mesh formations, with configurations of four and eight robots, each equipped with two pilot robots. The experiments evaluate these formations across various heading angles θ, ranging from 0 to 50 degrees in increments of 10 degrees, on the maximum gap width (or maximum height) for the given configuration. The purpose of this set of experiments is to demonstrate the impact of the projected distance of the robot assembly relative to discontinuities in the terrain, within parameterized environments. This highlights the critical role of spatial alignment and positioning of the robotic assembly in successfully navigating uneven or discontinuous landscapes. We calculate the angle α of the diagonal line - the longest possible line - in a given assembly with respect to its heading angle θ. Theoretically, optimal performance of the assembly is achieved when $\theta = \alpha$, as this alignment results in the maximum projected distance over a discontinuity. This shows the importance of the angular configuration of the assembly in enhancing its ability to effectively traverse discontinuities in the terrain such as gaps and stairs.

The results indicate that in the mesh formation with four robots, optimal performance is achieved at a heading angle (θ) of 40° for both gap crossing and obstacle descending tasks. This is closely aligned with the mesh diagonal angle (α) of 38.7°. In the case of an eight-robot assembly for gap crossing, the best performance is observed at heading angles of 10° and 20°, approximating the diagonal angle α of 24.0°. However, for line formations, the compliance of the assembly leads to a less significant correlation between the diagonal angle and performance. Particularly in obstacle descending behavior, line formations exhibit enhanced performance at smaller heading angles. This can be attributed to the actuation of two front wheels in direct contact with the platform when the assembly starts perpendicular to the edge. Such an orientation gives stronger pulling force and increased stability upon landing, which is crucial for successful descent.

5.3 Discussion

In summary, we present PuzzleBots, a reconfigurable modular robotic system characterized by hybrid passive coupling mechanisms. The system is uniquely designed with a side rigid knob-hole connection, providing robust load-bearing capabilities. Simultaneously, the front and back of each robot are equipped with flexible anchor mechanisms, offering flexibility to the assembled structure. This design facilitates compliance with environment geometries. As these are passive mechanisms, they do not require dedicated power for coupling behaviors, enhancing the system's adaptability and ease of integration with other systems. The PuzzleBots leverage the intrinsic agility of each robot's locomotion to form a diverse range of functional structures. This approach maintains the individual mobility of each unit while also preserving the flexibility of the assembled structure. To effectively control these robots, we developed a distributed model predictive control framework. This framework is both scalable and computationally efficient with an increasing number of robots. We model the inter-robot coupling constraints as polygons derived from the data collected, which further improves the flexibility and scalability of the framework. We demonstrate the ability of the PuzzleBots to dynamically couple and collaboratively navigate rough and discontinuous terrains, which would be impossible for a single robot to traverse. Additionally, we demonstrated the system's capability to decouple and execute individual tasks, showcasing their versatility and adaptability in different tasks.

The primary challenges in developing reconfigurable modular robot systems capable of traversing uneven and discontinuous terrains lie in the flexibility, robustness, and load-bearing capabilities of the coupling mechanisms. These systems must achieve these capabilities while also maintaining low power consumption and preserving the individual mobility of each module. Balancing these factors is crucial for creating a functional and efficient system that can locomote in a variety of environmental structures. Previous works have demonstrated activate mechanisms that involve magnetic [65] and mechanical [18, 37] components, as well as heating [61]. Passive connections mainly involve permanent magnets [28, 48, 54] and hook and loop fasteners [57]. In these designs, stronger connections are generally achieved with higher power during the coupled status or the decoupling process. On the contrary, we utilize the

robot's inherent agility from locomotion, accompanied by environment forces (i.e. gravity) to activate the mechanism. Through our experiments, we have shown that this passive mechanism can be robustly coupled and decoupled with our controller from robot locomotion. Rigid or compliant structures can be formed autonomously with robots starting from unaligned positions. The assembly structures can traverse a wide range of rough and discontinuous terrains that are not achievable with a single robot.

Our research has shown the effectiveness and potential of passive mechanisms in enhancing the physical capabilities of individual robots. However, as the number of robots in the system increases, the complexity of modeling inter-robot relationships using polygon constraints becomes inaccurate. This challenge is primarily due to the limited knowledge of the contact states between robots. In systems utilizing passive mechanisms, direct access to these contact states is not available, particularly with soft components, without the integration of sensors. While adding additional sensors could provide tactile information, this approach contradicts the principles of a passive mechanism. Our solution is to infer the contact states indirectly from the robots' responses and states. We utilize feedback from onboard encoders to localize inter-robot contact states. By estimating these contact states, we gain an estimate of the assembly's dynamics. This approach paves the way for scaling up the system to include a larger number of robots and to form more diverse and complex structures, while maintaining the core benefits of a passive mechanism.

5.4 Materials and Methods

In this section, we present the modeling of our mobile robot systems and the inter-robot connections, controllers for robot behaviors, including coupling and decoupling, connection pair maintenance, and trajectory following, and the distributed Model Predictive Control (MPC) for the multi-robot systems.

5.4.1 Problem Statement and Modeling

While the robots navigate through a 3D environment, the initial configuration of different formations are still within a 2D plane. Hence, initially we consider a group of

N robots situated in a 2D environment. We can represent the positions of robot i as $\mathbf{x}_i = [x_i, y_i, \theta_i]^\intercal \in \mathbb{R}^3$, $i = 1, 2, \ldots, N$, where each robot's pose includes its coordinates along the x and y axes on the 2D plane, as well as its orientation angle. The robots' behavior is modeled by a unicycle model, in which the control inputs $\mathbf{u}_i \in \mathbb{R}^2$ comprise both linear velocities and angular velocities, represented as $\mathbf{u}_i = [v_i, \omega_i]^\intercal \in \mathbb{R}^2$. The mathematical description of a differential drive robot is defined in [27]:

$$\dot{\mathbf{x}}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \tag{5.1}$$

In a discrete time-varying system, we denote the state and control input for robot i at the timestamp t to be $x(t)$ and $u(t)$, respectively. We use a first-order integration. Thus, the discrete robot dynamics can be represented as

$$x(t + \Delta t) = f(x(t), u(t)) = x(t) + \begin{bmatrix} \cos\theta_i(t) & 0 \\ \sin\theta_i(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i(t) \\ \omega_i(t) \end{bmatrix} \Delta t \tag{5.2}$$

The system we developed is composed of two types of robots, as introduced in Figure 5.3 - the pilot and non-pilot types. We model the non-pilot robot as in Equation 5.2. Since the pilot robots tend to be less precise in turning, we use them mainly for straight, back-and-forth movements. The control for the pilot robot is simplified only to control its linear velocity.

Robot coupling and connection pairs

To model the coupling relationship between robots, we introduce the definition of *connection points*. As shown in Figure 5.12, a robot i has seven connection points. The connection point frame is denoted as C_i. A *connection pair* is the alignment of two connection points $\{C_i, C_j\}$, $i, j = 1, 2, \ldots, N$. Each connection point on a robot body has a fixed transformation from the current robot body frame to the connection point frame, which is time-varying based on the robot's current pose. We further denote the position of a connection point C_i in the world frame to be

$\mathbf{c}_i(t) = [x_i^c(t), y_i^c(t)]^\intercal \in \mathbb{R}^2$. We define an *assembly* as a group of successfully coupled robots. Given a set of target behaviors, our goal is to couple, decouple, and maintain coupling/decoupling status based on the behavior in a scalable manner with multiple robots.

5.4.2 Robot behaviors

Coupling controller

Assuming robot i and j are initially separated on a 2D plane. Utilizing a Model Predictive Control (MPC) framework at time t, with a time horizon of $H_m \in \mathbb{Z}^+$, the coupling controller of robot i for a target connection pair $\{C_i, C_j\}$ is

$$\min_{x_i, u_i} \; w_f \|\mathbf{c}_i(H_m) - \mathbf{c}_j\|_2^2 + w_c \sum_{k=0}^{H_m-1} \|\mathbf{c}_i(k) - \mathbf{c}_j\|_2^2 \tag{5.3}$$

$$\text{s.t. } x_i(k=0) = x_i(t) \tag{5.4}$$

$$x_i(k+1) = f(x_i(k), u_i(k)) \tag{5.5}$$

$$x_i(k) \in \mathcal{X}, u_i(k) \in \mathcal{U}, \; k = 0, \ldots, H_m \tag{5.6}$$

where $\mathbf{c}_i(k)$ is the position of connection point C_i based on the robot pose $x_i(k)$. The MPC optimization is solved in real-time. As a distributed controller, robot i takes the numerical value of $\mathbf{c}_j$ at the initial time t. We minimize the distance between the two target connection points in Equation (5.3), subject to the initial position constraints (5.4), dynamics constraints (5.5), and actuation limit (5.6). The output $u_i(k=0)$ is then passed to the lower level PID controller and sent to the robot for execution.

Decoupling controller

The decoupling behavior shown in Figure 5.4B can be achieved by a simple motion described below. In a scenario where two robots are coupled, one robot's anchor is inserted into the opening of the other robot. The decoupling process is initiated when the robot containing the anchor performs a wiggle motion. This wiggling consists of

a combination of angular and linear velocities:

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} v_{\text{bias}} \\ w_{\max} sgn(\sin(Bt)) \end{bmatrix} \tag{5.7}$$

where $sgn(\cdot)$ is a sign function that returns the sign of the input number. This equation describes the wiggling behavior with a period of $2\pi/B$. Through this wiggling action, the anchor tip is loosened from the upper slits of the other robot, allowing it to rotate freely. This rotation, along with the anchor tip's compliance, allows the disengagement of the coupled robots.

Maintaining connection pairs

After two robots are coupled, it is essential to maintain the already coupled connection pairs to further form larger assemblies or perform collaborative tasks. Our approach uses a pure kinematic approach, without considering forces, for maintaining connections among the robots. This choice arises from the fact that, with passive coupling mechanisms, we lack direct access to contact status or forces. Although the integration of sensors could potentially provide us with this information, additional limitations arise, including size, compatibility with operating environments, and power constraints. Additionally, the increased electronic complexity introduced by sensors makes it challenging to directly transfer to other systems. Furthermore, introducing contact forces and modes results in significantly slower and less efficient computations. Hence, we rely solely on kinematics, employing polygon-based constraints to model interactions between robots.

Based on the computation concerns mentioned above, we model the inter-robot connections as linear constraints as shown in Figure 5.8A. The first constraint is a point-in-polygon constraint. We collected data on robot poses during coupling and projected anchor head positions under both fully extended and fully retracted conditions, resulting in 1153 data points shown in Figure 5.16. We constructed a bounding polygon enclosing the projected anchor positions, modeling this area as linear constraints [77]. Additionally, we demonstrate that these linear constraints on anchor points also translate to linear constraints in the robot state space in the supplement section 5.5.6. The second constraint is cutting-plane constraint aimed

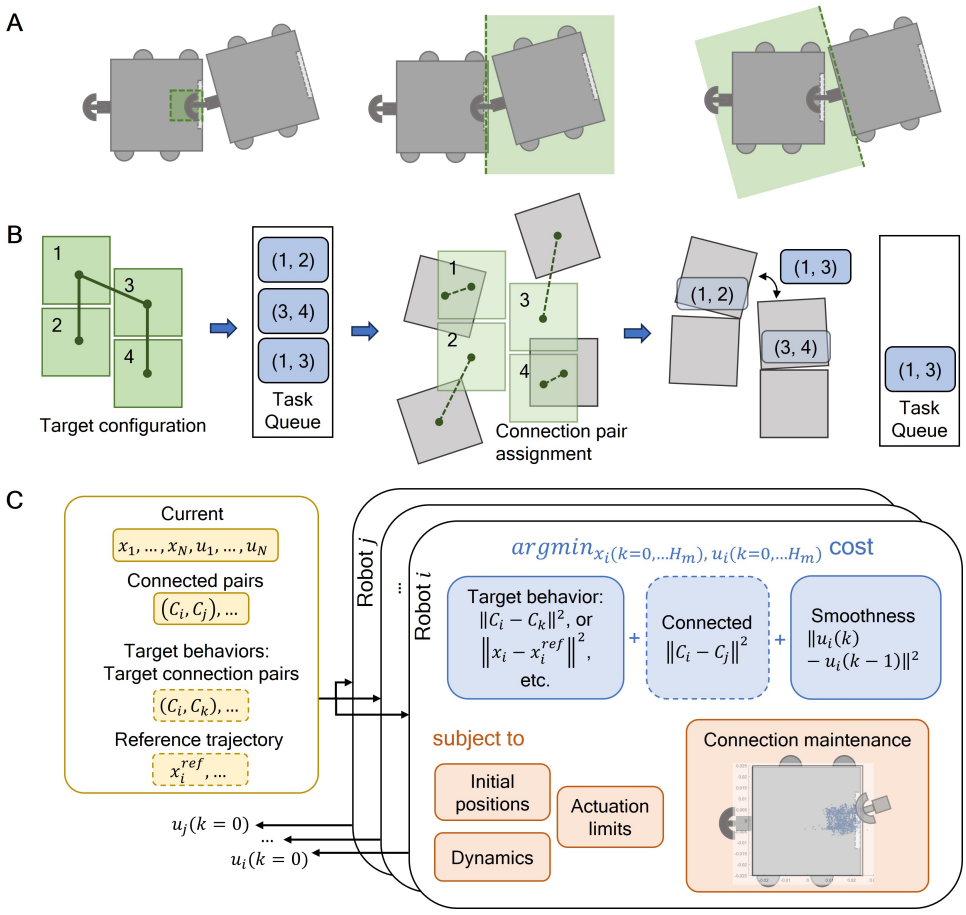


Figure 5.8: **Controller formulations.** (A) Connection maintenance constraints: one polygon constraint, and two cutting plane constraints. (B) To represent the formation of a target configuration, we create a minimum spanning tree (MST) based on a distance-induced graph derived from the robots' poses. Each MST edge corresponds to a connection pair. These pairs are sorted to create a task queue. Tasks are assigned by matching robot poses to the target configuration, and then executed in parallel from the queue. Upon aligning a pair, robots continue with other tasks, maintaining established connections. (C) This distributed MPC framework takes in the current robot states and control inputs, already connected pairs to maintain, and the relevant target behavior variables. Each robot has its own dedicated process that minimizes a specific cost based on the inputs, subject to the initial position, dynamic constraints, actuation limits, and connection maintenance polygon constraints.

at preventing the overlap of robot body geometries. Such overlaps can lead to failure experiments on the hardware, including instances where rotating robots exert excessive force, causing unintended flipping or tipping of adjacent robots. For a pair of robots coupled with an anchor, the cutting planes are defined by a line positioned either in the front or back, with half-planes oriented outward towards the other robot, as illustrated in Figure 5.8A. Notably, this formulation is also linear concerning the robot states. For detailed mathematical derivation, please refer to the supplementary section 5.5.6.

5.4.3 Connection-pair assignment

When robots start from unaligned positions, the goal behavior can be to form a given configuration. As shown in Figure 5.8B, we assign connection pairs relative to a given target configuration. Treating each robot as a vertex, we construct a fully connected graph based on the distances between them. Subsequently, a minimum spanning tree (MST) is derived from this distance-induced graph. For each edge within the MST, connection pairs are generated from the two vertices at its ends and appended to a task queue. This queue is then sorted to prioritize line movements before coupling operations. This is due to the limitations imposed by the robot's mobility in a mesh configuration—wherein lateral movements are possible while rotation is not. Following the generation of the task queue, we match the current robot positions with the given target configuration based on distance with Hungarian algorithm [25]. Subsequently, the tasks are mapped and assigned to the actual robots. Each task corresponds to two connection points on both robots, which is further augmented based on the type of connection, i.e. anchor-opening or knob-hole connection. Details of connection points and their augmentation are discussed in supplement section 5.5.5. Then the robots execute the coupling tasks in parallel. Upon completion of a coupling task, the robot assembly will continue with other non-conflict tasks while maintaining the already established connections with the above constraints.

5.4.4 Distributed Model Predictive Control

To enhance the scalability of our algorithm to accommodate a large number of robots, we introduce a distributed model predictive control (MPC) framework, as shown in

Figure 5.8C. This framework operates on a central computer in a distributed fashion. Initialization of the optimizer starts N processes, where N corresponds to the total number of robots. Within each process, the MPC optimization is computed utilizing *ip-opt* [68], integrated with a casadi interface [4]. The MPC formulation takes the following inputs. The first one is the current robot states x_i, and the current control input u_i, where $i = 1, \ldots, N$. In the process for robot i, we create the state and control variables for a fixed MPC time horizon H_m, denoted as $x_i(k = 0), \ldots, x_i(k = H_m)$. The states for other robots in the system are passed as numerical copies, and assumed to be stationary throughout the time horizon. The second input is the already established connection pairs $(C_i, C_j), \ldots$, which are incorporated into both the cost and constraints to facilitate connection maintenance. The third set of input is information regarding target behaviors. This includes ongoing target connection pairs in the case of configuration formation behavior, or a reference trajectory for trajectory tracking behavior.

Within each MPC iteration, the cost is formulated as a weighted sum of the errors associated with the given target behavior, established connection errors, and a control input difference for smoothness. The target behavior errors are also separated with a smaller intermediate stage cost for $k < H_m$, and a large final cost for the last time step $k = H_m$. By assigning a small stage cost, MPC prioritizes other factors, such as smoothness and connection maintenance, to ensure smooth and efficient control over the robot in real-time. On the other hand, assigning a large final cost encourages the controller to reach our desired behavior by the end of H_m. This helps in regulating the system towards the desired long-term behavior while minimizing other costs. The constraints of the MPC include initial state constraints, dynamics constraints, and actuation limits, as expressed in Equations (5.4)-(5.6), alongside the connection maintenance constraints presented in Figure 5.8A. Incorporating connection maintenance both as soft constraints within the cost function and as hard constraints serves the dual purpose of ensuring connections without breakage, while also providing preferences regarding the positioning of anchor heads or knobs relative to the robot body. This approach enhances the robustness of the system for constraint violations, which may arise from numerical errors or uncertainties in hardware actuation. Upon completion of optimization, the main process collects the first control input $u_i(k = 0)$ from the results obtained across all distributed processes.

Subsequently, this control input is forwarded to the PID controller node to execute, stabilize, and send down to the actual robots. This current control input is stored locally to be passed to the MPC in the next iteration. More details on the algorithm and formulation of the distributed MPC are in the supplement material section 5.5.8.

5.5 Supplementary Materials

5.5.1 Electrical design

A custom 2-layer printed circuit board (PCB) was designed to accommodate all the electronics needed for the robot. The board contains a microcontroller development board (Seeed Studio XIAO ESP32-C3), equipped with Wi-Fi for communication, on top of the board 5.9(A) with a switch for the power. On the bottom 5.9(B), a dual H-bridge motor driver (DRV8833, Texas Instruments) is used for controlling the motors, a voltage regulator (TPS7A0533PDBZR, Texas Instruments) for power distribution, 6-pin JST SH connectors for connecting the motors with encoders, and a battery connector for 3.7 V 1000 mA h lithium polymer battery (ASR00012, TinyCircuits) to power the robot. The robot consumes 0.1 A at 3.7 V on average, hence can run for approximately 10 hours with a fully charged battery.

5.5.2 Rigid and flexible connections

For a reconfigurable multi-robot system, the functionality of an assembly depends on the strength and shape of the robot connection and configuration. In magnetic and other active mechanical structures, connection strength requires high power consumption. For a passive connection, where no power is dedicated to the coupling mechanism, strong connections may require high power to couple or decouple. To enable strong connections from the limited onboard power, we designed an asymmetric flexible anchor mechanism along with rigid connections. By combining two different connection mechanisms and assembling them in different directions, we can achieve structures with high strength.

The design objectives for the flexible anchor are threefold: firstly, to ensure it is sufficiently pliable for robots to establish coupling through locomotion alone; secondly,

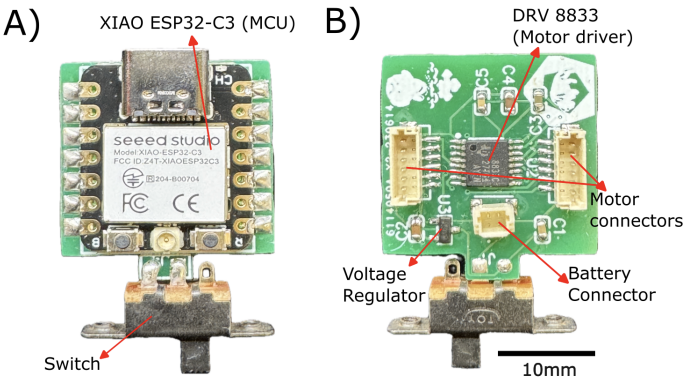


Figure 5.9: Pictures of the top view(A) and bottom view(B) of the populated PCB.

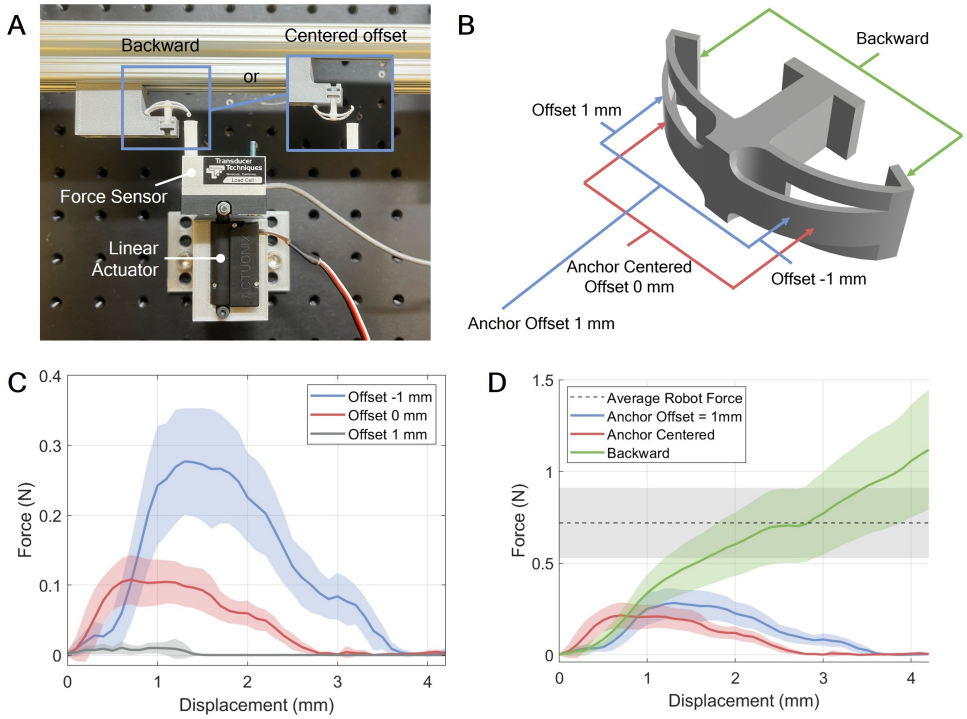


Figure 5.10: (A) Force experiment setup of pushing from the front or back. (B) The different pushing positions for the force profile. (C) Forces with different offset distances from the centered position versus displacement. (D) Forces of pushing from the anchor in the centered position, offset position, and backward positions versus displacement.

to possess the strength to support the collective weight of multiple robots, while still being able to decouple; and thirdly, to offer a degree of rotational flexibility, enabling connected components to undergo slight rotations. The design is shown in Figure 5.10A and B. We tested the force-displacement profiles of the anchor from different positions. Figure 5.10A and B show the experiment setup of pushing the anchor when it is aligned with the opening on the other robot (anchor centered offset 0 mm), deviated by 1mm from the center (offset +/- 1mm), and also from the back of the anchor. Figure 5.10C shows the forces of these corresponding pushing positions based on the displacement from its resting position. Since the sensor is pushing in parallel with the center of the anchor, the forces drop once it passes the critical displacement position. Since the anchor has two legs, Figure 5.10D shows the total forces required from the other robot to achieve a given displacement. The average force one mobile robot can provide is 0.7189 N, which is significantly higher than the forces required to push an anchor inside the other robot (both anchor is centered position and 1 mm offset position). The backward force, i.e., the forces needed for the robot to pull out the anchor directly, is significantly greater than the forces that one single robot can provide. This shows that the robot can easily couple with another robot, while it cannot pull out the anchor directly. Instead, the robots can decouple by locomoting with a specific motion pattern, which will be introduced in the method section.

5.5.3 Additional force experiments

As illustrated in Figure 5.11, we conducted measurements on the load experienced by the rigid side knobs. Despite being designed as a rigid structure, deformation still occurs. This is mainly due to the whole robot body, including the chassis and the knobs, being fabricated using Thermoplastic Polyurethane (TPU), which allows for a buckling effect that facilitates coupling between robots, as discussed in [76]. The force sensor reaches its limit at approximately 3.2N. The pilot robot weighs 104.2 g, while the non-pilot is heavier due to the metal ball bearings, weighing 109.7 g. Combining the forces from both knobs, the robot can support a weight equivalent to around seven times the body weight of one robot. While the actual load might exceed this value, the measured capacity already meets the requirements for all our

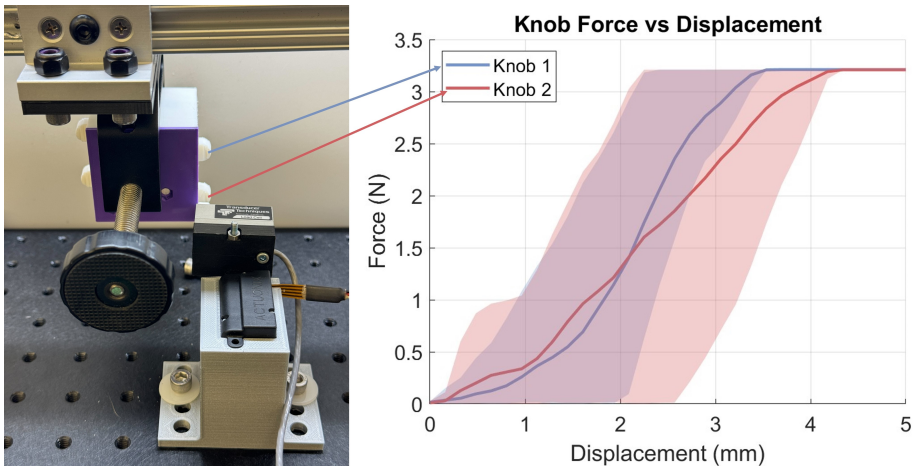


Figure 5.11: Force plot of the two side knobs.

intended robot formations.

5.5.4 Simulation setup

In this research, we evaluated our robots using the Bullet physics engine [13]. We generate the robot's structure based on our mesh and output a URDF (Unified Robot Description Format) file, which outlines the robot using only the primitive shapes - boxes, cylinders, and spheres. Experimentally, this method has proven to be more reliable and precise for contact and collisions compared to using original mesh files. The robot main body is modeled as a rigid body without self-collision checking. For the two side wheels of the non-pilot robot, we connect the wheels to the main body using a revolute joint. The effort and joint torques are setup from the force experiments as in Figure 5.10.

To mimic the flexibility of the anchor, it is modeled as a three-bar linkage, where the middle bar is attached to the base of the anchor, and the side bars are connected to the middle bar with rotating joints. These joints simulate the force shown in Figure 5.10.

5.5.5 Connection pair augmentation

As shown in Figure 5.12, we define six connection points on the robot body for initial task assignment. To obtain the task queue as shown in Figure 5.8, the planner iterate

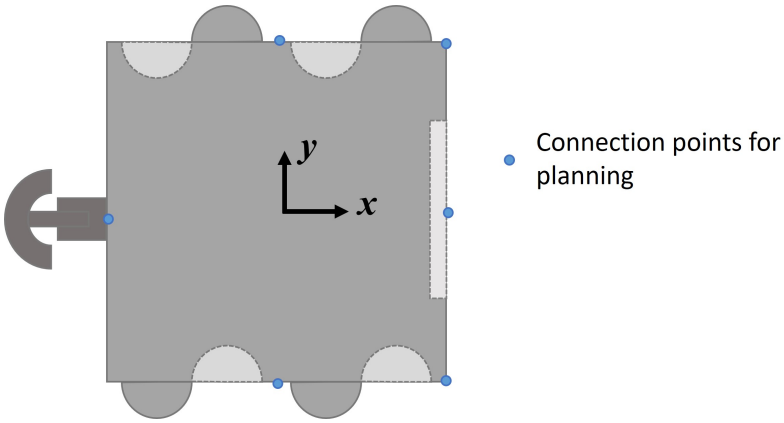


Figure 5.12: Connection points on a robot for computing the connection pair task queue in Figure 5.8A.

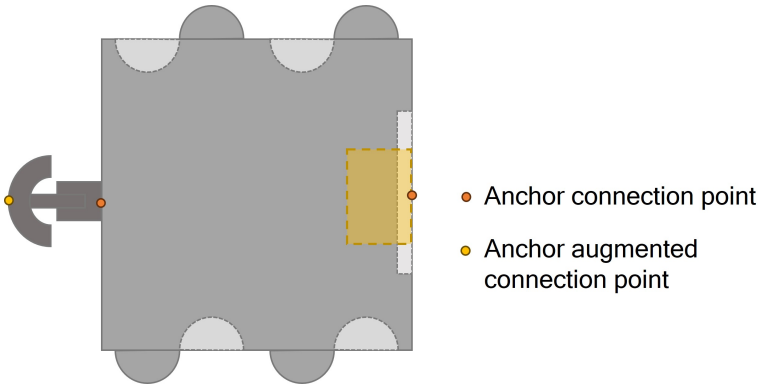


Figure 5.13: Anchor connection points for planning, the augmented anchor connection point, and the polygon constraints for coupling and connection maintenance.

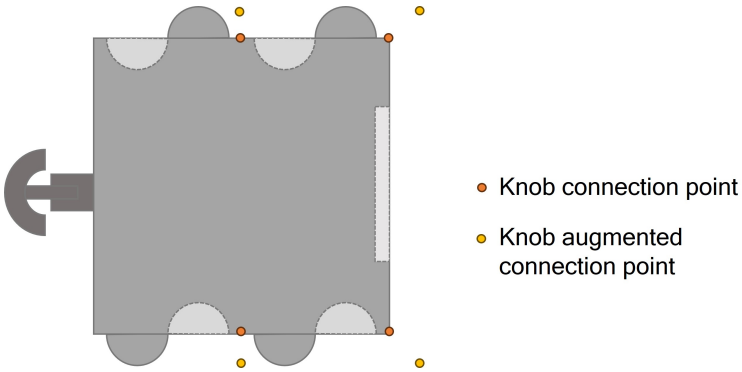


Figure 5.14: Knob connection points for planning, and the augmented connection points for coupling and connection maintenance.

Algorithm 7 Augment Connection Pairs

Input: $\mathcal{C} = \{(C_i, C_j), \ldots\}$: set of connection pairs
Output: $\tilde{\mathcal{C}}$: augmented set of connection pairs
Initialize: $\tilde{\mathcal{C}}$={}

```
function AUGMENTPAIRS(C)
    for (C_i, C_j) in C do
        (C_i, C_j).status = decoupled
        (C_i, C_j).type = anchor or knob
        (C_i, C_j).anchor_index = getAnchorIndex(C_i, C_j)
        (C_i, C_j).head = (C'_i, C'_j) based on anchor_index
        C̃.append((C_i, C_j))
    end for
    return C̃
end function
```

though all the possible connection pairs for a given target configuration, and generate the set of tasks based on minimum total distance with the Hungarian algorithm [25].

Contrary to the typical collision-avoidance behavior on most robotic systems, our objective is controlled collisions that allow the robots to couple with each other. For collision avoidance in a relatively uncluttered environment, the geometry of the robots does not need to be precisely considered; simplifying their shapes into disks or other basic convex forms suffices. This simplification allows for efficient and effective navigation without detailed consideration of each robot's geometry structure. However, when aiming for collision-seeking behavior to enable coupling, the specific geometry of the robots becomes crucial. Given the inherent concave nature of the robot bodies in our design, simple convex approximations are inadequate for this purpose. To address this, we introduce augmented points of alignment that are out of the minimum convex boundary enclosing the robots. These points enable accurate alignment between robots, avoiding unintended collisions.

After obtaining the goal connection pairs based on a distance-induced graph, we augment the pairs as in Algorithm 7. In Algorithm 7, we iterate through the set of all connection pairs and augment them based on characteristics such as current anchor status and type. Then the augmented set of connection pairs is returned.

While iterating through the connection pair list, we assess whether the status of each connection pair needs to be updated. As illustrated in Algorithm 8, we

Algorithm 8 Update Connection Pair Lists

Input: $\mathcal{C}_{conn}$: connected pairs, $\mathcal{C}_{active}$: active pairs, $\tilde{\mathcal{C}}$: augmented pair list, ϵ: threshold
Output: $\mathcal{C}_{conn}$: connected pairs, $\mathcal{C}_{active}$: active pairs

```
function UPDATEPAIRS(C_conn, C_active, C̃, ϵ)
    for (C_i, C_j) in C̃ do
        a = (C_i, C_j).anchor_index
        b = (C_i, C_j).body_index
        if (C_i, C_j).status is decoupled then
            if R_a.head in polygon(R_b, ϵ) then
                (C_i, C_j).status ← head_aligned
            end if
        end if
        if (C_i, C_j).status is head_aligned then
            if R_a.head in polygon(R_b, ϵ) then
                (C_i, C_j).status ← head_inserted
            end if
            if R_a.head not in polygon(R_b, ϵ) then
                (C_i, C_j).status ← decoupled
            end if
        end if
        if (C_i, C_j).status is head_inserted then
            C_active.remove((C_i, C_j))
            C_conn.append((C_i, C_j))
        end if
    end for
    return C_conn, C_active
end function
```

determine the connection status, considering the current positions of the robots, by verifying if the connection point (the anchor head) falls within the polygon of other robot, with a slight tolerance margin denoted by ϵ. Subsequently, C_{conn}, which is the set of already connected pairs and C_{active}, which is the set of active connection pairs that are not yet coupled, are modified.

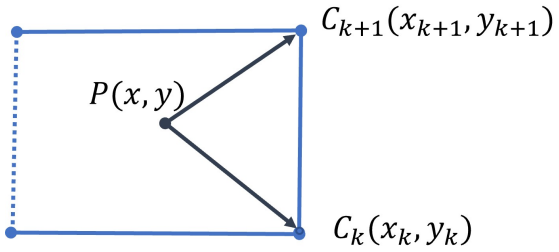


Figure 5.15: Point in polygon.

5.5.6 Coupling constraints

Before introducing the polygon constraints with the dynamics of robots, we first address the problem of determining whether a point resides within a polygon. The Point-in-Polygon (PIP) problem is a classic problem within computational geometry. Traditional methods for solving the PIP problem, such as ray-casting or the winding number approach, are computationally heavy to integrate into a real-time control frameworks.

This section introduces our approach to mathematically deriving the linear inequality constraints that ensure a point is within a convex polygon. This method offers a more direct and computationally efficient way to integrate spatial constraints into the MPC framework, enabling real-time decision-making and control for robotic systems.

As shown in Figure 5.15, a convex polygon on a 2D plane is defined by a series of points $C_1, \ldots, C_k, C_{k+1}, \ldots, C_K$. The coordinate of each point $C_k \in \mathbb{R}^2$ is (x_k, y_k). A convex polygon is characterized by the property that all interior points lie on the same side of each boundary line segment [15], for example C_kC_{k+1}. For simplicity and without loss of generality, we assign numbers to the vertices of our convex polygon in an incremental order, counter-clockwise along the boundary points. Thus, all the points inside the polygon lie on the left-hand side of $\overrightarrow{C_kC_{k+1}}$. Consider a point $P = (x, y) \in \mathbb{R}^2$ inside this convex polygon. It is on the left-hand side of the vector $\overrightarrow{C_kC_{k+1}}$. According to the right-hand rule of cross-product, we have

$$\overrightarrow{PC_k} \times \overrightarrow{PC_{k+1}} \geq 0 \tag{5.8}$$

We substitute the coordinate variables into this equation. Then we have

$$\begin{bmatrix} x_k - x \\ y_k - y \end{bmatrix} \times \begin{bmatrix} x_{k+1} - x \\ y_{k+1} - y \end{bmatrix} \geq 0$$
$$(x_k - x)(y_{k+1} - y) - (y_k - y)(x_{k+1} - x) \geq 0$$
$$\begin{bmatrix} y_{k+1} - y_k & -(x_{k+1} - x_k) \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_k \\ y_k \end{bmatrix} \right) \leq 0 \tag{5.9}$$

Denote the augmented connection point frame in Figure 5.13 to be $P_a \in SE(2)$. The homogeneous transformation from the robot i body frame R_i to the connection point frame is

$$g_{R_i P_a} = \begin{bmatrix} 1 & 0 & a_x \\ 0 & 1 & a_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

where a_x and a_y are constant based on the anchor's resting position. On our robot, $a_x = -0.032$ and $a_y = 0$, both in meters. Consider the constraint polygon sit on robot j with its body frame denoted as R_j. We transform the connection point frame into the body frame R_j where the constraint vertices are defined. As defined in the Method, the pose of robot i is (x_i, y_i, θ_i), and similar for robot j. We also denote the world frame to be W.

$$g_{R_j P_a} = g_{R_j W}\ g_{W R_i}\ g_{R_i P_a} \tag{5.11}$$
$$= g_{W R_j}^{-1}\ g_{W R_i}\ g_{R_i P_a} \tag{5.12}$$
$$= \begin{bmatrix} \mathcal{R}(-\theta_j) & -\mathcal{R}(-\theta_j) \begin{bmatrix} x_j \\ y_j \end{bmatrix} \\ 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{R}(\theta_i) & \begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & a_x \\ 0 & 1 & a_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$
$$= \begin{bmatrix} \mathcal{R}(-\theta_j) & -\mathcal{R}(-\theta_j) \begin{bmatrix} x_j \\ y_j \end{bmatrix} \\ 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{R}(\theta_i) & \mathcal{R}(\theta_i) \begin{bmatrix} a_x \\ a_y \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ 0 \quad 0 & 1 \end{bmatrix} \tag{5.14}$$

where the $\mathcal{R}(\theta) \in SO(2)$ denote the rotation matrix from angle θ. We extract the

translation part and obtain

$$\mathcal{R}(-\theta_j)\left(\mathcal{R}(\theta_i)\begin{bmatrix}a_x\\a_y\end{bmatrix}+\begin{bmatrix}x_i\\y_i\end{bmatrix}\right)-\mathcal{R}(-\theta_j)\begin{bmatrix}x_j\\y_j\end{bmatrix} \tag{5.15}$$

For a distributed setup, the states of neighboring robots are numerical constants. Consider robot i to be the ego robot, x_j, y_j, θ_j are constants throughout the MPC planning horizon. We substitute this Equation (5.15) into the coordinates of the point $P = [x, y]^\intercal$ in Equation (5.9), and denote this as:

$$pip(P_a, \overrightarrow{C_k C_{k+1}}) \leq 0 \tag{5.16}$$

Note that despite the non-linearity of the system with respect to the states x_i, y_i, θ_i, this inequality constraint remains differentiable, with analytical expressions for both gradients and Hessians. This is particularly helpful for MPC optimization, given that our chosen optimizer, ip-opt [68], uses the interior point method. The availability of explicit derivatives facilitates the optimizer's efficiency and effectiveness in navigating the solution space, thereby enhancing the overall computational performance of the optimization process.

5.5.7 Modeling constraint geometry

To create the bounding polygon as a constraint to the passive soft anchor as shown in Figure 5.15, we collected data points on the robot poses when robots are coupled together. By driving them with various velocities while keeping the robots coupled, we collected 1153 data points. We then project the anchor head positions P_a at its retracted position and its extended position onto the body frame of the other robot. The result is shown in Figure 5.16. We use this data to create a bounding polygon to constrain the anchor head in the MPC setup.

Apart from the point-in-polygon constraint, we also have the body line cutting plane constraints in Figure 5.17. We constrain the corner points of the robots to be on the other side of the cutting plane defined by the front (or back) of the other

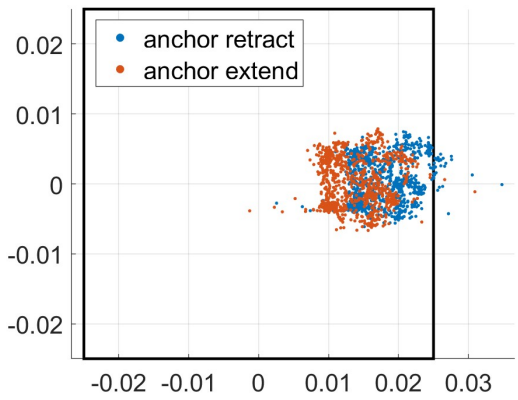


Figure 5.16: Projected anchor head positions when two robots are coupled. The plot shows the projected positions when the anchor is fully retracted and fully extended.

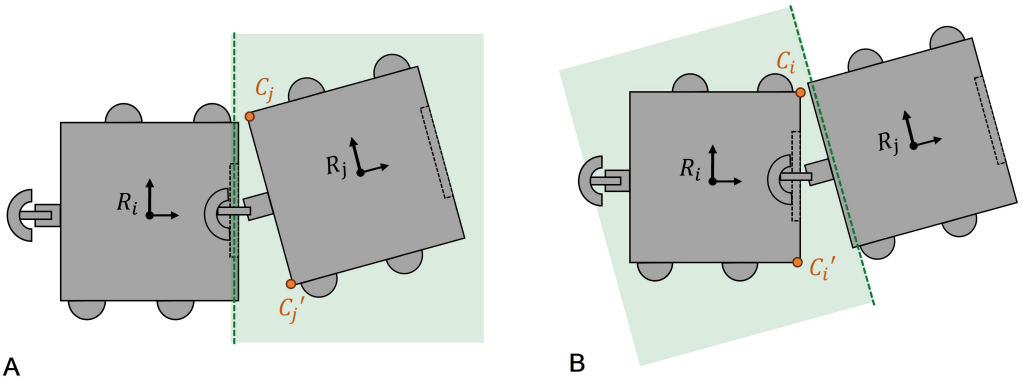


Figure 5.17: The cutting plane constraints for inter-robot connections.

robot body. For Figure 5.17A and Figure 5.17B constraints respectively, we have

$$g^x_{R_iC_j} \geq L/2 \tag{5.17}$$

$$g^x_{R_jC_i} \leq -L/2 \tag{5.18}$$

where $g^x_{R_iC_j}$ represents the translation of corner point C_j in x-axis of the robot body frame R_i, and similarly for the other constraint. L denotes the body length of a robot. We represent these constraints as $line(x_i, x_j) \leq 0$ where $x_i \in \mathbb{R}^3$ is the state of robot i.

5.5.8 Distributed MPC framework details

Algorithm 9 demonstrates a simplified pseudocode of the overall controller algorithm. We first initialize a set u for the control signals and a *cost* variable. Then, we calculate the control signal for each robot in parallel, adding its cost $cost_i$ to the running total *cost* for that time step. A diagram of the MPC is shown in Figure 5.8. The objective function is minimized, and the control signal u_i is obtained and returned, then added to u. This is then passed to either the simulation interface to control the robots in the simulator, or to the lower-level PID controller to be sent to the hardware platform via WiFi.

$$\min_{x_i,u_i} \; w_f\|\mathbf{c}_i(H)-\mathbf{c}_j\|_2^2 + w_c\sum_{k=0}^{H-1}\|\mathbf{c}_i(k)-\mathbf{c}_j\|_2^2 + w_s\sum_{k=0}^{H-1}\|\mathbf{u}_i(k)-\mathbf{u}_i(k+1)\|_2^2 \tag{5.19}$$

$$\text{s.t. } x_i(k=0) = x_i(t) \tag{5.20}$$

$$x_i(k+1) = f(x_i(k),u_i(k)) \tag{5.21}$$

$$x_i(k)\in\mathcal{X}, u_i(k)\in\mathcal{U},\ k=0,\dots,H \tag{5.22}$$

$$pip(P_a(k),\overrightarrow{C_m(k)C_{m+1}(k)})\le 0,\ k=0,\dots,H_c \tag{5.23}$$

$$line(x_i(k),x_j(k))\le 0,\ k=0,\dots,H_c \tag{5.24}$$

The MPC formulation for a couping behavior is shown above. For robot i, the objective in Equation 5.19 includes the intermediate stage cost of the alignment error for the target connection pair between robot i and j with a weight w_c, the final cost weight of this error is denoted as w_f. A smoothness term is also added with a weight of w_s for the control signals. To simplify notation, we denote the MPC horizon to be $H\in\mathbb{Z}^+$.

5.5.9 Costs of other behaviors

For a Model Predictive Control (MPC) framework, the setup of cost and weights are crucial for each different behavior. To ensure the correct operation of a connect

Algorithm 9 Distributed MPC for Coupling Behavior

Input: T: target configuration, x: robot states at time t, $\mathcal{C}_{conn}$: connected pairs, $\mathcal{C}_{active}$: active pairs
Output: u: control input for robots
Initialize: $u = []$, $cost = 0$

```
function CouplePairs(T, x, C_conn, C_active)
    for robot_i in robots do                          ▷ done in parallel
        x_i = state of robot_i
        u_i ← MPC(x_i, C_conn, C_active)
        add u_i to u
        add cost_i to cost
    end for
    return u
end function
```

segment assembly, we consider the robots that have active tasks to operate the *segment leader*. The goal of this segment leader will have the largest weight in the cost function, while all the other robots within this segment will have less weight and copy the goal of this segment leader. During the task assignment, each segment is guaranteed to have only one segment leader, thus guaranteeing there are no conflicting goals for each segment.

For each segment leader in the configuration formations, the MPC formulation follows the same as in Equation (5.19). For a none leader robot within a segment, the objective of the MPC becomes

$$w_f \|\mathbf{x}_i(H) - \mathbf{x}_{leader}\|_2^2 + w_c \sum_{k=0}^{H-1} \|\mathbf{x}_i(k) - \mathbf{x}_{leader}\|_2^2 + w_s \sum_{k=0}^{H-1} \|\mathbf{u}_i(k) - \mathbf{u}_i(k+1)\|_2^2 \tag{5.25}$$

This ensures the connected segment follows the leader for the target behavior. The connection constraints remain the same as in the the leader constraints for configuration formation Equation (5.19). The weights w_f, w_c for the non-leader robots are significantly smaller than the weights for leader robots.

The trajectory following behavior also has a different objective formulation. Consider a given reference trajectory is parametrized by a set of waypoints $\mathbf{x}^{ref}$. Only the leader robot within a segment is considered during the trajectory following behavior.

The objective is

$$w_f \|\mathbf{x}_i(H) - \mathbf{x}^{ref}(H)\|_2^2 + w_c \sum_{k=0}^{H-1} \|\mathbf{x}_i(k) - \mathbf{x}^{ref}(k)\|_2^2 \tag{5.26}$$

The constraints remain the same as in the previous section. The non-leader robots follow the MPC formulation as in Equation (5.25).

5.5.10 Statisical analysis of performance on rough terrain

The mobility characteristics of PuzzleBots were individuals, and connected pairs were compared across rough terrains for three different metrics. These metrics, the tracking error, the percent traversal, and velocity capture the mobility and precise locomotion characteristics of PuzzleBots across terrains with various roughness. Five different surfaces were tested, including a flat terrain, with a surface variance of 0 mm, and then artificial, 3D printed rough terrains with surface variances of 1 mm, 2 mm, 3 mm, and 4 mm.

The locomotion data of the PuzzleBots were collected as described in the Results section. To determine if the mobility of individuals and paired PuzzleBots are different on different terrains, pairwise t-tests were performed for all three metrics on all five terrains (a total of 15 pairwise t-tests were performed). Table 5.1 summarizes the p-value of the pairwise t-tests between individual and linked puzzled bots across terrains and metrics. All tests were performed with the `ttest2` function in MatLab 2021b with a degree of freedom of 18, and resultant p-values were compared to $\alpha = 0.01$, for a confidence interval of 99%.

On flat terrain (a surface variance of 0 mm, individual and linked PuzzleBots did not perform statistically significantly different, and since both individual and linked PuzzleBots completed the entire trajectory (100% for traversal percentage), a pairwise t-test could not be performed. On all terrains with surface variances between 0 mm (flat terrain) and 4 mm, individual and linked Puzzlebots did not perform statistically significantly different in any metric with a 99% confidence interval. However, when the surface variance increases to 5 mm, individual and paired PuzzleBots are statistically significantly different in all three metrics for the pairwise t-tests. Figure 5.6 shows that the paired PuzzleBots outperformed individual PuzzleBots for the traversal

Surface variance (mm)	Tracking error	Percent traversal	Velocity
0	1.13E-01	-	1.36E-01
1	2.06E-02	9.06E-01	9.63E-01
2	8.67E-02	3.22E-01	3.71E-01
3	6.44E-01	5.74E-02	5.94E-02
5	5.56E-03	2.79E-03	2.80E-03

Table 5.1: Experiment analysis for terrain traversal.

percentage and velocity, meaning that paired PuzzleBots could complete longer paths with higher velocities. However, paired PuzzleBots have an increase in the tracking error.

5.6 Limitations

This study has demonstrated the efficacy of leveraging the collective capabilities of multiple robots for navigating challenging terrains. Nonetheless, it is important to acknowledge several limitations inherent to our current approach.

Firstly, in our modeling framework, we model each individual robot with unicycle dynamics. This is simple and accurate, and achieves high precision in trajectory tracking, with accuracy within one millimeter. However, this level of precision is challenging when modeling a connected assembly of multiple robots. The primary challenge lies in our approach to modeling these robots purely from a kinematic perspective, disregarding the contact forces and modes between them. Though this simplification has facilitated an efficient real-time optimization framework, as discussed in the Methods section, it does not accurately capture the complex dynamics of robot interactions. While the modeling error may be negligible for a small assembly of robots, it becomes increasingly significant as the number of robots in the assembly grows. This limits the scalability of our approach for precise control over larger assemblies. Moving forward, we aim to adopt data-driven methodologies to model the interactions among coupled robots more accurately, enhancing both scalability and precision.

Secondly, our use of passive connections, despite offering significant benefits, limits the robots' ability to navigate positive obstacles, such as climbing stairs or overcoming

barriers. The current system is effective for traversing negative obstacles, like gaps and height drops, due to its reliance on gravity and environmental interactions. To address this limitation, future research will explore the integration of minimal actuation into the robots' coupling mechanisms, enabling them to navigate over positive obstacles.

Thirdly, the reliance on indoor Vicon localization systems for precise control and coupling poses another limitation. This dependency constrains the operational environment of the robots to indoor lab settings equipped with such systems. To broaden the applicability and autonomy of the robots, future developments will consider incorporating onboard sensors and other localization technologies, reducing reliance on external systems for navigation and control.

In conclusion, while our findings highlight the potential of collective robot forces in overcoming complex terrains, the outlined limitations underscore the need for continued research and development to enhance the versatility, scalability, and autonomy of robotic assemblies.

Chapter 6

Conclusion and Future Works

In this thesis, we developed the PuzzleBot system utilizing passive mechanisms for modular multi-robot systems. The robots can autonomously couple into functional structures to collaboratively navigate challenging terrains, and decouple to perform individual tasks in parallel.

To conclude, compared with previous works that focus mostly on developing the hardware connection mechanisms, we have shown that we can use the environment to extend the physical capability of each individual robot. By employing gravity as an activation force, we utilize passive mechanisms as connections between robots, without the need for additional power. Furthermore, we incorporate compliance within the robot assembly to improve traction, enabling coupled robots to navigate challenging terrains more effectively. We have demonstrated how modular multi-robot systems significantly enhance the physical capabilities of individual modules. By leveraging a design that incorporates redundancy in inter-robot connections, we have achieved a level of adaptability that allows us to tailor the rigidity of assembled structures to specific environmental structures. We also utilize heterogeneity of combining different types of robots, where each one of them has their own strength and weaknesses. All of these are achieved by leveraging precise and scalable controllers and the inherited agility from locomotion. We utilize a pure geometry and kinematics approach to model both rigid and soft inter-robot connections with passive mechanisms and provide a distributed Model Predictive Control framework for scalable computation. All these improvements are realized with relatively simple hardware designs, showing

that we can enhance robot capabilities without complex or costly mechanisms.

Despite the advantage of simplicity and the ease of transfer of passive mechanisms, modeling and controlling their motions remains challenging. Without sensors or actuators, we do not have direct access to the contact status. For a constrained and contact-dense robot system, reliably modeling contact forces and modes is crucial for controlling such systems. Future works may include inferring contact status from limited onboard feedback, for example, the motor encoder and current feedback. We may also explore data-driven approaches for such systems that are too complicated to model analytically. Specifically for highly constrained multi-robot systems, it is crucial to find a representable state space for such inter-robot contact constraints to be able to generalize to a large number of robots.

The passive coupling mechanism we developed has demonstrated notable flexibility and efficacy, particularly when dealing with negative terrains such as descending obstacles and crossing gaps. To address more diverse terrains, such as navigating over obstacles and stairs, we introduced a simple enhancement with one active motor. This involved the incorporation of one actuator atop the existing passive coupling mechanism [12]. This augmentation enabled active three-dimensional motion capabilities, empowering the robots to ascend onto platforms twice their original height, thereby expanding their capabilities in a wide range of environment geometries. With limited onboard power and actuation on mobile robots, it is important to study the need for actuation - what are the necessary components for target behaviors?

This thesis shows that passive structures and simple designs in robotics are valuable but often overlooked. It challenges the common overuse of actuation in robots with high degrees of freedom. Showcasing how the PuzzleBot system uses fewer active components to achieve its goals may suggest that future robot designs might need less complexity than we currently use. I hope this research encourages others to explore and investigate more about using simple shapes and passive mechanisms in robotics to create efficient and functional robots, which is a direction that could lead to more sustainable and effective designs in the future.

Bibliography

[1] Random Rough Terrain - GitHub Repository. https://github.com/RobotFormAndFunction/random-rough-terrain, 2022. 5.2.4

[2] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research*, 36(9):1000–1021, 2017. 4.1

[3] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. 4.4.2

[4] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019. 5.4.4

[5] Gabriel Arpino, Kyle Morris, Sasanka Nagavalli, and Katia Sycara. Using information invariants to compare swarm algorithms and general multi-robot algorithms. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7682–7687. IEEE, 2018. 2.2

[6] Kendall Atkinson. *An introduction to numerical analysis*. John wiley & sons, 1991. 4.3.1

[7] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015. 2.2, 4.1

[8] H Benjamin Brown, JM Vande Weghe, Curt A Bererton, and Pradeep K Khosla. Millibot trains for enhanced mobility. *IEEE/ASME transactions on mechatronics*, 7(4):452–461, 2002. 5.1

[9] C Cao, H Zhu, Z Ren, H Choset, and J Zhang. Representation granularity enables time-efficient autonomous exploration in large, complex worlds. *Science Robotics*, 8(80):eadf0970, 2023. 5.1

[10] Lillian Chin, Max Burns, Gregory Xie, and Daniela Rus. Flipper-style locomotion through strong expanding modular robots. *IEEE Robotics and Automation*

Letters, 8(2):528–535, 2022. 5.1

[11] Sebastian Claici, John Romanishin, Jeffrey I Lipton, Stephane Bonardi, Kyle W Gilpin, and Daniela Rus. Distributed aggregation for modular robots in the pivoting cube model. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1489–1496. IEEE, 2017. 3.1

[12] James Clinton, Sha Yi, and Zeynep Temel. Enhancing heterogeneous swarm locomotion through simple 1-dof arm mechanisms. *under review*, 2024. 6

[13] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016–2021. 4.4.2, 5.5.4

[14] Jay Davey, Ngai Kwok, and Mark Yim. Emulating self-reconfigurable robots - design of the SMORES system. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4464–4469, 2012. 2.1

[15] Mark De Berg. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000. 5.5.6

[16] Jaydev P Desai, James P Ostrowski, and Vijay Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE transactions on Robotics and Automation*, 17(6):905–908, 2001. 2.2

[17] Jason M Graham, Albert B Kao, Dylana A Wilhelm, and Simon Garnier. Optimal construction of army ant living bridges. *Journal of theoretical biology*, 435:184–198, 2017. 2.1, 5.1

[18] Roderich Groß, Michael Bonani, Francesco Mondada, and Marco Dorigo. Autonomous self-assembly in swarm-bots. *IEEE transactions on robotics*, 22(6): 1115–1130, 2006. 2.2, 3.1, 4.1, 4.2.2, 5.1, 5.3

[19] Bahar Haghighat, Emmanuel Droz, and Alcherio Martinoli. Lily: A miniature floating robotic platform for programmable stochastic self-assembly. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1941–1948. IEEE, 2015. 2.2, 3.1

[20] Feili Hou and Wei-Min Shen. Graph-based optimal reconfiguration planning for self-reconfigurable robots. *Robotics and Autonomous Systems*, 62(7):1047–1059, 2014. 3.1

[21] Ion Ion, Mircea Găvan, Adrian Curaj, Grigore Stamatescu, and Cornel Dinu. Adaptation to rough terrains by using force sensing on the mero modular walking robots. In *Applied Mechanics and Materials*, volume 762, pages 147–154. Trans Tech Publ, 2015. 4.1

[22] Kyle Johnson, Vicente Arroyos, Amélie Ferran, Raul Villanueva, Dennis Yin, Tilboon Elberier, Alberto Aliseda, Sawyer Fuller, Vikram Iyer, and Shyamnath Gollakota. Solar-powered shape-changing origami microfliers. *Science Robotics*,

8(82):eadg4276, 2023. 5.1

[23] Morten Winkler Jorgensen, Esben Hallundbk Ostergaard, and Henrik Hautop 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)*, volume 2, pages 2068–2073. Ieee, 2004. 2.2, 4.1, 5.1

[24] Hiroshi Kawano. Linear heterogeneous reconfiguration of cubic modular robots via simultaneous tunneling and permutation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 332–338. IEEE, 2019. 5.1

[25] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 3.3.3, 5.4.3, 5.5.5

[26] Haruhisa Kurokawa, Kohji Tomita, Akiya Kamimura, Shigeru Kokaji, Takashi Hasuo, and Satoshi Murata. Distributed self-reconfiguration of m-tran iii modular robotic system. *The International Journal of Robotics Research*, 27(3-4):373–386, 2008. 2.2, 4.1

[27] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 2.3.4, 3.2, 5.4.1

[28] Guanqi Liang, Haobo Luo, Ming Li, Huihuan Qian, and Tin Lun Lam. Freebot: A freeform modular self-reconfigurable robot with arbitrary connection point-design and implementation. In *IEEE/RSJ Int. Conf. Intell. Robots Syst., Las Vegas, USA*, 2020. 2.2, 3.1, 4.1, 4.2.2, 5.1, 5.3

[29] Chao Liu and Mark Yim. Configuration recognition with distributed information for modular robots. In *Robotics Research: The 18th International Symposium ISRR*, pages 967–983. Springer, 2020. 5.1

[30] Chao Liu, Michael Whitzer, and Mark Yim. A distributed reconfiguration planning algorithm for modular robots. *IEEE Robotics and Automation Letters*, 4(4):4231–4238, 2019. 3.1

[31] Chao Liu, Sencheng Yu, and Mark Yim. A fast configuration space algorithm for variable topology truss modular robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8260–8266. IEEE, 2020. 5.1

[32] Chao Liu, Sencheng Yu, and Mark Yim. Motion planning for variable topology truss modular robot. In *Proceedings of Robotics: Science and Systems*, 2020. 3.1

[33] Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020. 5.1

[34] Haobo Luo and Tin Lun Lam. Auto-optimizing connection planning method for chain-type modular self-reconfiguration robots. *IEEE Transactions on Robotics*, 39(2):1353–1372, 2022. 5.1

[35] Wenhao Luo and Katia Sycara. Minimum k-connectivity maintenance for robust multi-robot systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7370–7377. IEEE, 2019. 3.1

[36] Wenhao Luo, Sha Yi, and Katia Sycara. Behavior mixing with minimum global and subgroup connectivity maintenance for large-scale multi-robot systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9845–9851. IEEE, 2020. 1, 2.1

[37] Luis A Mateos, Wei Wang, Banti Gheneti, Fabio Duarte, Carlo Ratti, and Daniela Rus. Autonomous latching system for robotic boats. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7933–7939. IEEE, 2019. 4.1, 4.2.2, 5.1, 5.3

[38] Satoshi Murata, Eiichi Yoshida, Akiya Kamimura, Haruhisa Kurokawa, Kohji Tomita, and Shigeru Kokaji. M-tran: Self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4):431–441, 2002. 5.1

[39] Sasanka Nagavalli, Nilanjan Chakraborty, and Katia Sycara. Automated sequencing of swarm behaviors for supervisory control of robotic swarms. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2674–2681. IEEE, 2017. 4.1

[40] Reza Olfati-Saber and Richard M Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533, 2004. 2.3.5

[41] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007. 2.1, 2.2, 2.3.5

[42] Michael Park and Mark Yim. Distributed control and communication fault tolerance for the ckbot. In *2009 ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots*, pages 682–688. IEEE, 2009. 5.1

[43] Kirstin Hagelskjaer Petersen, Radhika Nagpal, and Justin K Werfel. Termes: An autonomous robotic system for three-dimensional collective construction. *Robotics: science and systems VII*, 2011. 5.1

[44] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. IEEE, 2017. 2.1, 2.2, 3.1, 4.1, 5.1

[45] Chris R Reid, Matthew J Lutz, Scott Powell, Albert B Kao, Iain D Couzin, and Simon Garnier. Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. *Proceedings of the National Academy of Sciences*, 112(49): 15113–15118, 2015. 2.1, 3.1, 5.1

[46] 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. www.coppeliarobotics.com. 3.5

[47] David Rollinson and Howie Choset. Pipe network locomotion with a snake robot. *Journal of Field Robotics*, 33(3):322–336, 2016. 2.2

[48] John W. Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295, 2013. 2.1, 3.1, 5.1, 5.3

[49] John W. Romanishin, Kyle Gilpin, Sebastian Claici, and Daniela Rus. 3D M-Blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1925–1932, 2015. 2.1, 2.2, 4.1, 5.1

[50] John W Romanishin, John Mamish, and Daniela Rus. Decentralized control for 3d m-blocks for path following, line formation, and light gradient aggregation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4862–4868. IEEE, 2019. 5.1

[51] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298. IEEE, 2012. 2.2, 3.1

[52] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014. 2.2, 5.1

[53] David Saldana, Bruno Gabrich, Michael Whitzer, Amanda Prorok, Mario FM Campos, Mark Yim, and Vijay Kumar. A decentralized algorithm for assembling structures with modular robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2736–2743. IEEE, 2017. 3.1

[54] David Saldana, Bruno Gabrich, Guanrui Li, Mark Yim, and Vijay Kumar. Modquad: The flying modular structure that self-assembles in midair. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 691–698. IEEE, 2018. 4.1, 4.2.2, 5.1, 5.3

[55] David Saldana, Parakh M Gupta, and Vijay Kumar. Design and control of aerial modules for inflight self-disassembly. *IEEE Robotics and Automation Letters*, 4 (4):3410–3417, 2019. 3.1

[56] Allison J Seo, Sha Yi, and Katia Sycara. Decentralized model predictive control for constrained multi-robot system. *IROS Workshop on in Advances in Multi-Agent Learning*, 2023. 1

[57] Masahiro Shimizu and Akio Ishiguro. An amoeboid modular robot that exhibits real-time adaptive reconfiguration. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1496–1501. IEEE, 2009. 2.2, 4.1, 5.1, 5.3

[58] S Sponberg and RJ Full. Neuromechanical response of musculo-skeletal structures in cockroaches during rapid running on rough terrain. *Journal of Experimental Biology*, 211(3):433–446, 2008. 5.2.4

[59] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. 4.4.3

[60] R St.Pierre and S Bergbreiter. Gait exploration of sub-2 g robots using magnetic actuation. *IEEE Robotics and Automation Letters*, 2(1):34–40, 2017. 5.2.4

[61] Petras Swissler and Michael Rubenstein. Fireant3d: a 3d self-climbing robot towards non-latticed robotic self-assembly. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3340–3347. IEEE, 2020. 5.1, 5.3

[62] Ardalan Tajbakhsh, Lorenz T Biegler, and Aaron M Johnson. Conflict-based model predictive control for scalable multi-robot motion planning. *arXiv preprint arXiv:2303.01619*, 2023. 5.1

[63] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. Duality-based convex optimization for real-time obstacle avoidance between polytopes with control barrier functions. In *2022 American Control Conference (ACC)*, pages 2239–2246. IEEE, 2022. 4.1

[64] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 286–292. IEEE, 2022. 4.1, 5.1

[65] Tarik Tosun, Jay Davey, Chao Liu, and Mark Yim. Design and characterization of the EP-Face connector. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 45–51, 2016. 2.1, 2.2, 3.1, 4.1, 4.2.2, 5.1, 5.3

[66] Yuxiao Tu and Tin Lun Lam. Configuration identification for a freeform modular self-reconfigurable robot-freesn. *IEEE Transactions on Robotics*, 2023. 5.1

[67] Serguei Vassilvitskii, Mark Yim, and John Suh. A complete, local and parallel reconfiguration algorithm for cube style modular robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 1, pages 117–122. IEEE, 2002. 3.1

[68] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathe-*

matical programming, 106:25–57, 2006. 5.4.4, 5.5.6

[69] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172): 754–758, 2014. 4.1

[70] Julian Whitman, Nico Zevallos, Matt Travers, and Howie Choset. Snake robot urban search after the 2017 mexico city earthquake. In *2018 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, pages 1–6. IEEE, 2018. 4.1

[71] Cornell Wright, Aaron Johnson, Aaron Peck, Zachary McCord, Allison Naaktgeboren, Philip Gianfortoni, Manuel Gonzalez-Rivero, Ross Hatton, and Howie Choset. Design of a modular snake robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2609–2614. IEEE, 2007. 2.1, 2.2, 5.1

[72] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. In *Mathematical programming*, pages 25–57, 2006. 4.4.2

[73] Guang-Zhong Yang, Jim Bellingham, Pierre E Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of science robotics. *Science robotics*, 3 (14):eaar7650, 2018. 2.2

[74] Sha Yi, Wenhao Luo, and Katia Sycara. Distributed topology correction for flexible connectivity maintenance in multi-robot systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8874–8880. IEEE, 2021. 1

[75] Sha Yi, Zeynep Temel, and Katia Sycara. Puzzlebots: Physical coupling of robot swarms. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8742–8748. IEEE, 2021. 1, 3.1, 3.1, 3.2, 3.4, 3.5.3, 4.1, 4.2.2, 5.1

[76] Sha Yi, Zeynep Temel, and Katia Sycara. Configuration control for physical coupling of heterogeneous robot swarms. In *2022 IEEE International Conference on Robotics and Automation*. IEEE, 2022. 1, 4.1, 4.2, 4.2.2, 4.3.3, 4.3.4, 4.3.5, 5.1, 5.5.3

[77] Sha Yi, Katia Sycara, and Zeynep Temel. Reconfigurable robot control using flexible coupling mechanisms. In *Robotics Science and Systems (RSS)*, 2023. 1, 5.1, 5.4.2

[78] Sha Yi, Shashwat Singh, Allison Seo, Ryan St. Pierre, Katia Sycara, and Zeynep Temel. Reconfigurable robot swarms for terrain traversal with passive coupling mechanisms. *under review*, 2024. 1

[79] Mark Yim, David G Duff, and Kimon D Roufas. Polybot: a modular reconfigurable robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 514–520. IEEE, 2000. 2.1, 2.2

[80] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007. 2.2

[81] Jun Zeng, Bike Zhang, and Koushil Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *2021 American Control Conference (ACC)*, pages 3882–3889. IEEE, 2021. 4.1, 5.1

[82] Da Zhao and Tin Lun Lam. Snailbot: A continuously dockable modular self-reconfigurable robot using rocker-bogie suspension. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4261–4267. IEEE, 2022. 5.1

[83] Zhiwu Zheng, Prakhar Kumar, Yenan Chen, Hsin Cheng, Sigurd Wagner, Minjie Chen, Naveen Verma, and James C Sturm. Scalable simulation and demonstration of jumping piezoelectric 2-d soft robots. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5199–5204. IEEE, 2022. 4.1