Configuration Control for Physical Coupling of Heterogeneous Robot Swarms

Sha Yi   Zeynep Temel   Katia Sycara

Abstract— In this paper, we present a heterogeneous robot swarm system that can physically couple with each other to form functional structures and dynamically decouple to perform individual tasks. The connection between robots can be formed with a passive coupling mechanism, ensuring minimum energy consumption during coupling and decoupling behavior. The heterogeneity of the system enables the robots to perform structural enhancement configurations based on specific environmental requirements. We propose a *connection-pair oriented configuration control* algorithm to form different assemblies. We show experiments of up to nine robots performing the coupling, gap-crossing, and decoupling behaviors.

I. INTRODUCTION

In unstructured environments, ants form and adapt functional structures dynamically in response to obstacles, gaps, and holes [12]. 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 [20] - 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 [11], [15] 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 [2], [3], [7], [14], [18]. The most common method for dynamic coupling is performed by magnetic forces [7], [14], [17], [18]. This may consume high energy during the coupling or decoupling process, and also has limited load-carrying capabilities. In addition, the magnets are 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 [10]. Reconfiguration algorithms for pre-connected modular robot systems focus on graph topology reconfiguration [4], [8], [9]. Modular robots that have limited mobility reconfigure based on motion primitives [19] or grid-based setup [1], [16].
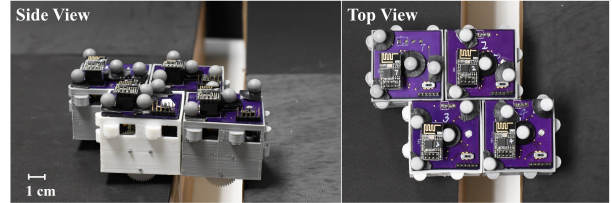
Fig. 1: Four robots form a mesh configuration to cross a gap between two platforms.

These methods restrict the formation and are ineffective with systems where robots have individual mobility and no strong connection with each other.

In PuzzleBots [20], 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 [20] 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 [20] can be fragile and sensitive to disturbances. Borrowing the k-connectivity concept from graph theory, we introduce the mesh assembly shown in Figure 1, where robots can couple in two dimensions to strengthen the
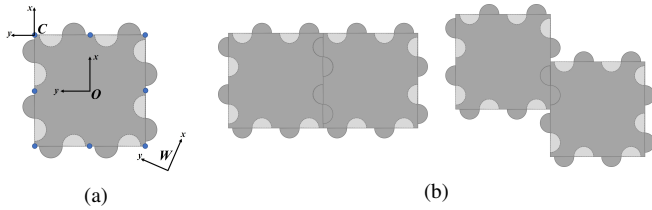
Fig. 2: (a) PuzzleBot with eight connection points shown as blue dots and the robot frame (O), world frame (W), and connection point frame (C); (b) Two possible connection configuration for two robots.

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 II. Then, in Section III, we present our connection-pair-oriented configuration control algorithm that drives the robots to a given coupled configuration. Next, in Section IV, 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 V.

II. 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 [6] is defined as

$$\dot{p}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = J_i \cdot u_i . \quad (1)$$

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 (1). Details will be introduced in Section IV. All robots have the same body with two knobs and two holes on each side to provide passive coupling behavior [20]. 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 2a. 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 2b. 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 *connection pair oriented configuration control* algorithm to solve this problem in section III.

III. METHODOLOGY

In this section, we will present three aspects of the configuration control based on connection pairs. Section III-A 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 II. Section III-B introduces an optimization-based control for an assembly to reach individual goals while maintaining in-assembly connection pairs. The last section III-C presents our connection pair based configuration control algorithm. Heterogeneity of the system will be discussed in section IV.

A. Single Connection Pair Alignment

As shown in Figure 2a, 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

$$u_i = J_i^+ \begin{bmatrix} \Delta c_x \\ \Delta c_y \\ \Delta \theta \end{bmatrix} = J_i^+ \begin{bmatrix} c_{x_j} - c_{x_i} \\ c_{y_j} - c_{y_i} \\ \arctan\frac{\Delta c_y}{\Delta c_x} - \theta_i + \theta_{bias} \end{bmatrix}, \quad (2)$$

where J_i^+ is the pseudo-inverse of J_i in Equation (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 (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 2b, 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.

B. 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
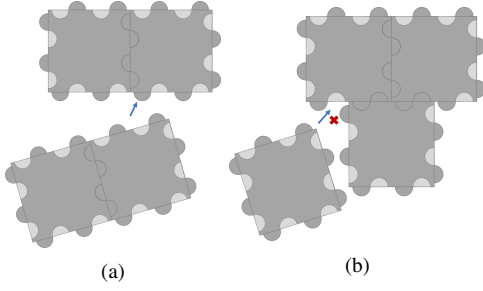
Fig. 3: (a) Two convex assemblies couple to form a larger assembly. (b) One robot cannot couple with a concave assembly.

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

$$\cos(\theta_i + \omega_i\Delta t) \approx \cos\theta_i - \omega_i\Delta t\sin\theta_i \tag{3}$$

$$\sin(\theta_i + \omega_i\Delta t) \approx \sin\theta_i + \omega_i\Delta t\cos\theta_i. \tag{4}$$

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

$$\begin{bmatrix} c_{x_i} \\ c_{y_i} \end{bmatrix} = \begin{bmatrix} c_i & -d_{x_c}s_i - d_{y_c}c_i \\ s_i & d_{x_c}c_i - d_{y_c}s_i \end{bmatrix} u_i\Delta t + \begin{bmatrix} x_i + d_{x_c}c_i - d_{y_c}s_i \\ y_i + d_{x_c}s_i + d_{y_c}c_i \end{bmatrix}. \tag{5}$$

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

$$-\epsilon \le \begin{bmatrix} c_{x_i} \\ c_{y_i} \end{bmatrix} - \begin{bmatrix} c_{x_j} \\ c_{y_j} \end{bmatrix} \le \epsilon. \tag{6}$$

We may stack all the connection pair constraints in Equation (6) in an assembly in one equation $Au \le b$. The connection pair maintenance problem then becomes

$$u^* = \arg\min ||u - \hat{u}||^2 \tag{7}$$

$$\text{s.t. } Au \le b. \tag{8}$$

C. Connection Pair Based Configuration Control

As defined in section II, given a goal configuration 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 3a, two convex assemblies, each formed by two robots, are able to couple and form a larger assembly of four robots. However, in Figure 3b, 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 III-C, we plan to avoid forming concave assemblies by prioritizing convex assemblies first.

Algorithm 1 Find Existing Connection Pairs

Input: p: input poses
Output: $pair\_dict$: connection pairs
Initialize: $pair\_dict=\{\}$
1: **function** FINDEXISTPAIRS(p)
2:     $\mathcal{G} \leftarrow$ constructGraph(p) based on distance
3:     $edge\_set \leftarrow$ getMinimumSpanningTree($\mathcal{G}$)
4:     **for** vertex i and j in $edge\_set$ **do**
5:         $[C_i, C_j] \leftarrow$ findMinDistancePair($\mathcal{C}_i, \mathcal{C}_j$)
6:         $pair\_dict[(i, j)] = [C_i, C_j]$
7:     **return** $pair\_dict$

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

$$p'_g = p_g - \overline{p_g} + \overline{p}, \tag{9}$$

where $\overline{p}$ is the mean of p. We then find the connection pair assignment based on the goal configuration p'_g as shown in Algorithm 1. In Algorithm 1, we find the connection pairs based on an existing configuration. This input robot poses p assumes the robots are already aligned. First, we construct a fully connected graph $\mathcal{G}$ where the vertices are the location points in 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 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 p_g. The pairs are then sorted based on the distance between the robot poses in 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 p'_g and p. Each element in the distance matrix d_{ij} is calculated as $d_{ij} = ||p'_{g_i} - p_j||^2$. Hungarian algorithm [5] 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
1: **function** ASSIGNCONNECTIONPAIRS($\mathbf{p}_g$, $\mathbf{p}$)
2:     $\mathbf{p}'_g \leftarrow$ alignCenter($\mathbf{p}_g$, $\mathbf{p}$)
3:     $goal\_pair\_dict \leftarrow$ findExistPairs($\mathbf{p}'_g$)
4:     $sorted\_pair\_dict \leftarrow$ sortPairs($goal\_pair\_dict$, $\mathbf{p}'_g$) based on distance
5:     $dist\_matrix \leftarrow$ getDistanceMatrix($\mathbf{p}'_g$, $\mathbf{p}$)
6:     $id\_assign \leftarrow$ HungarianAlgorithm($dist\_matrix$)
7:     $pair\_dict \leftarrow$ updatePairAssignment($sorted\_pair\_dict$, id_assign)
8:     **return** $pair\_dict$

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 III-A 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 (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 $O(N)$.

IV. 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 [20]. 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

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, $\ldots$], $\mathcal{C}_{conn} = \{\}$, $\mathcal{C}_{exec} = \{\}$
1: **function** CONFIGURATIONCONTROL($\mathbf{p}_g$, $\mathbf{p}$)
2:     $\mathcal{C}_{goal} \leftarrow$ assignConnectionPairs($\mathbf{p}_g$, $\mathbf{p}$)
3:     **for** $pairs$ in $\mathcal{C}_{conn}$ **do**
4:         update connection pairs $pairs$
5:     **for** (i, j) in $\mathcal{C}_{goal}$ **do**
6:         **if** i, j not busy **then**
7:             $\mathcal{C}_{exec}$.append((i, j))
8:             $busy[i]$ = True, $busy[j]$ = True
9:     $\hat{\mathbf{u}}[busy] \leftarrow$ alignConnectionPairs($\mathcal{C}_{exec}$, $\mathbf{p}$)
10:     $\hat{\mathbf{u}}[\sim busy] \leftarrow \hat{\mathbf{u}}[busy]$ for each $\mathcal{C}_{conn}$
11:     $\hat{\mathbf{u}} = (1 - k_{bias})\hat{\mathbf{u}} + k_{bias}\mathbf{u}_{connection\_bias}$
12:     $\mathbf{u}^* = \arg\min ||\mathbf{u} - \hat{\mathbf{u}}||^2$, s.t. $A\mathbf{u} \leq b$
13:     **for** (i, j) in $\mathcal{C}_{exec}$ **do**
14:         **if** distance between connection pairs $< \epsilon$ **then**
15:             $\mathcal{C}_{exec}$.remove((i, j))
16:             $busy[i]$ = False, $busy[j]$ = False
17:             u_i, $u_j \leftarrow 0$
18:     **return** $\mathbf{u}^*$

---

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 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 4c. 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: pilot robots and non-pilot robots. The pilot robot is a four-wheeled robot, and the non-pilot robot is the one with casters and side wheels. As shown in Figure 4, 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
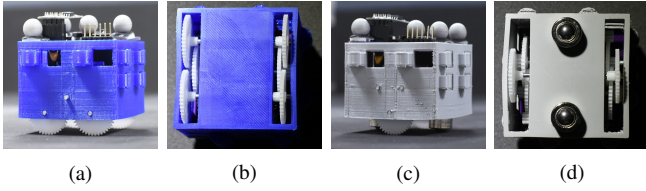
(a)   (b)   (c)   (d)

Fig. 4: Pilot robot: (a) side view, (b) bottom view. Non-pilot robot: (c) side view, (d) bottom view.

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

$$v = k_v \frac{M_r + M_l}{2}, \omega = k_\omega (M_r - M_l) . \tag{10}$$

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

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

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 $A\mathbf{u} \le b$ in Equation (7). We will present the calibration of the feasibility region in Section V-A as well.

V. 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 [13] with the Vortex Studio[1] as the physics engine for concave objects. We conducted three sets of experiments: Section V-A presents the hardware calibration of the pilot and non-pilot robots. Section V-B shows a set of screenshots from a 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 V-C presents quantitative results of the gap-crossing performance based on the line and mesh configuration.

A. 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, and the result is shown in Figure 5. The parameters obtained from the experiments are $k_{v,\text{non-pilot}} = 0.0006$, $k_{\omega,\text{non-pilot}} = 0.0142$, $k_{v,\text{pilot}} = 0.0024$, $k_{\omega,\text{pilot}} = 0.0033$. Figure 5a and 5c

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

(a) Non-pilot robot linear velocity

(b) Non-pilot robot angular velocity

(c) Pilot robot linear velocity

(d) Pilot robot angular velocity

(e) Non-pilot robot $v - \omega$ constraints
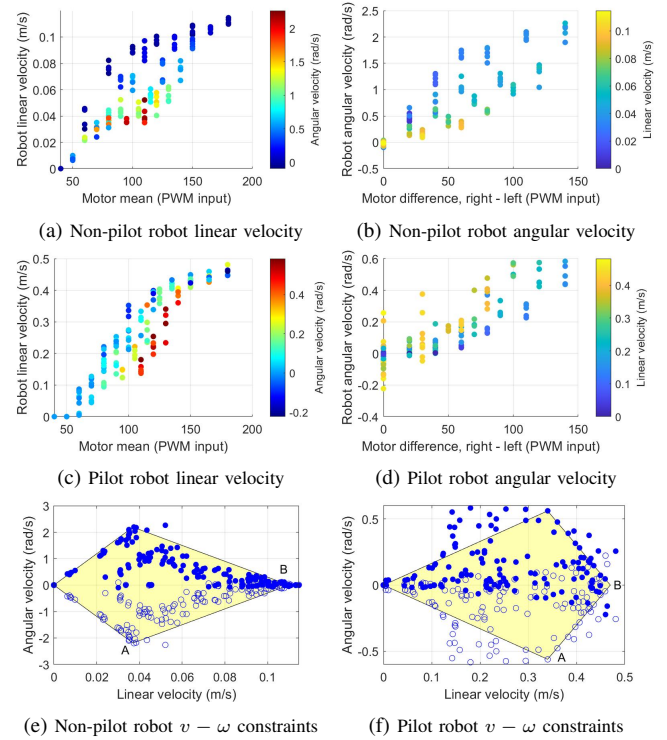
(f) Pilot robot $v - \omega$ constraints

Fig. 5: Calibration results of non-pilot and pilot robot. (a) - (d): linear and angular velocities with respect to PWM input (0 to 255). (e) (f): feasibility region in terms of $v - \omega$ of the non-pilot and pilot robots.

show that with the same motor mean, the robot has higher linear velocity when the angular velocity is small. As shown in Figure 5e and 5f, 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.

B. Combined Behavior Sequences

As shown in Figure 6, 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 this case, we see that the robots are able to first form

Fig. 6: Screenshots of four robots coupling to form a mesh configuration, crossing the gap, and decouple.

(a) Line configuration

(b) Mesh configuration

(c) Line configuration

(d) Mesh configuration
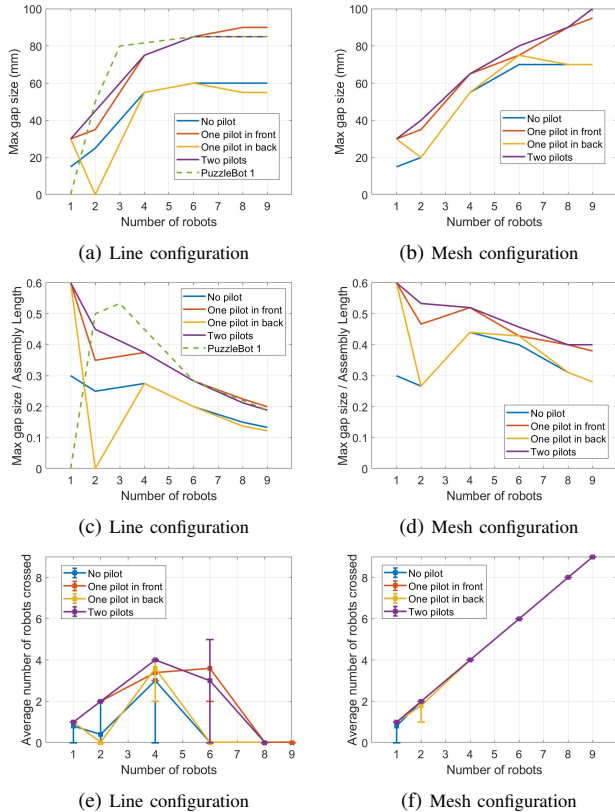
(e) Line configuration

(f) Mesh configuration

Fig. 7: Result of various number of robots crossing a gaps with 6 mm height difference, and 20° heading angle. (a) (b) Maximum gap size the line/mesh configuration assembly can cross against the number of robots; (c) (d) Maximum gap size compared with the entire line/mesh assembly length, against the number of robots; (e) (f) Average number of robots that crossed a gap length = 25% length of the line/mesh assembly, against the number of robots.

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.

C. Gap-crossing Performances

We performed experiments of various number 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 [20] 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 gaps lengths ranging from 10 mm to 105 mm, with 5 mm increment. Each experiment is recorded five times.

We record the number of robots successfully reach the other platform. The result of the experiments is shown in Figure 7.

Figure 7a and 7b show that the robots in either line or mesh formation maintain the gap-crossing ability as in [20], and can cross larger gaps with more than eight robots compared with [20]. 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 7c and 7d, 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 7e and 7f 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.

VI. 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.

References

[1] 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.

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

[3] 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.

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

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

[6] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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, November 2013.

[15] 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.

[16] 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.

[17] 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.

[18] 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, October 2016.

[19] 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.

[20] 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.