Supplementary Materials

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 S1(A) with a switch for the power. On the bottom S1(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 $3.7\,\mathrm{V}$ $1000\,\mathrm{mA\,h}$ lithium polymer battery (ASR00012, TinyCircuits) to power the robot. The robot consumes $0.1\,\mathrm{A}$ at $3.7\,\mathrm{V}$ on average, hence can run for approximately 10 hours with a fully charged battery.

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, 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 S2A and B. We tested the force-displacement

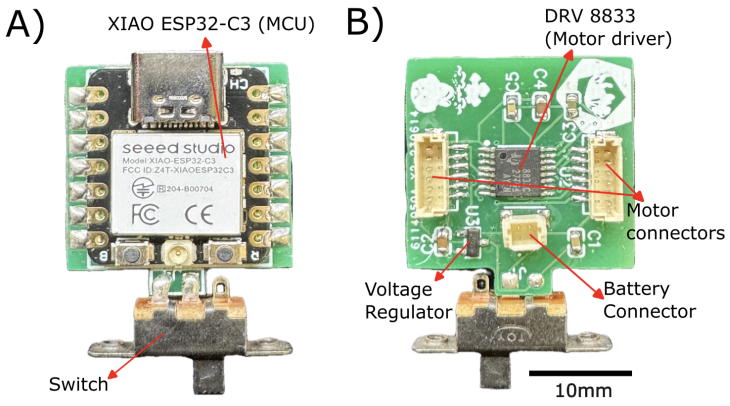

Figure S1: Pictures of the top view(A) and bottom view(B) of the populated PCB.

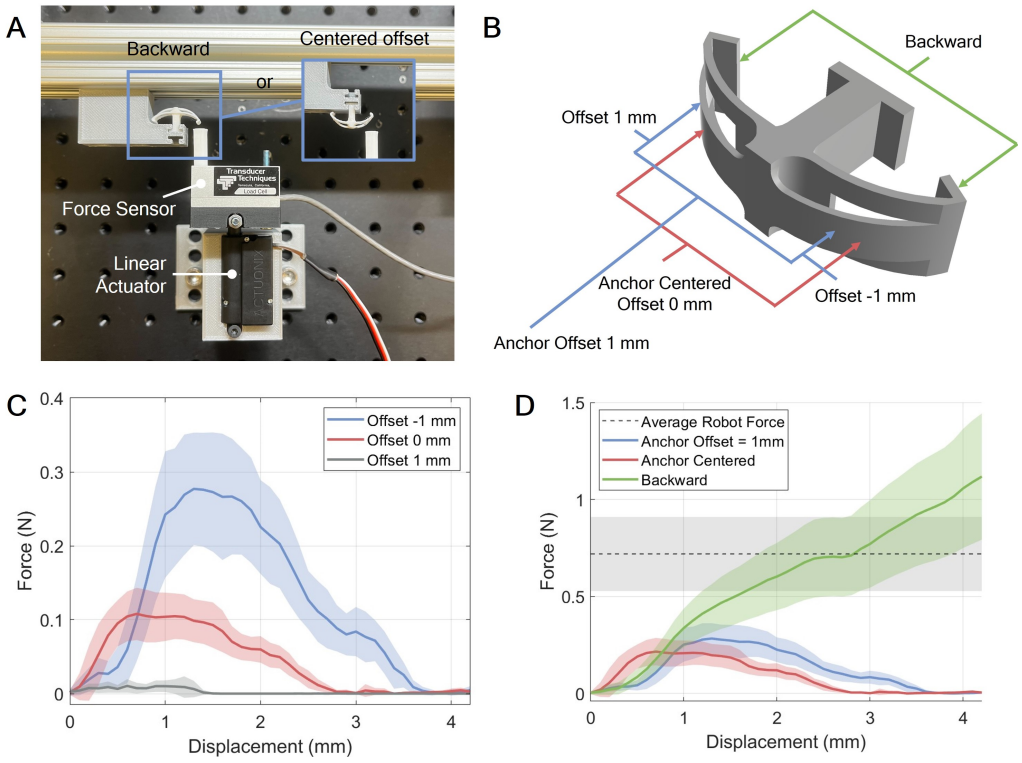

Figure S2: (A) Force experiment setup of pushing from the front or back. (B) The different pushing positions for the force profile. (C) Forces with different offset distances from the centered position versus displacement. (D) Forces of pushing from the anchor in the centered position, offset position, and backward positions versus displacement.

profiles of the anchor from different positions. Figure S2A 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 S2C 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 S2D 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.

2 Additional force experiments

As illustrated in Figure S3, 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 (*?*). 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 intended robot formations.

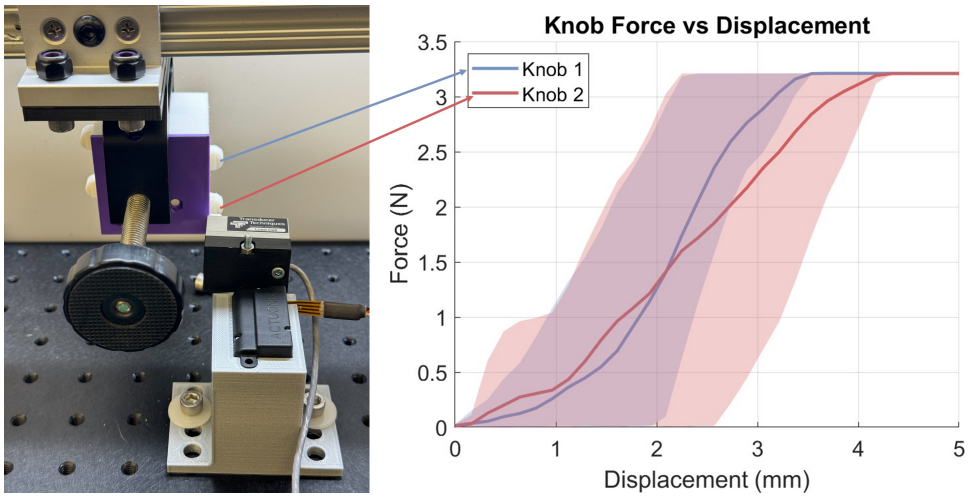

Figure S3: Force plot of the two side knobs.

3 Simulation setup

In this research, we evaluated our robots using the Bullet physics engine (*?*). 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 S2.

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 S2.

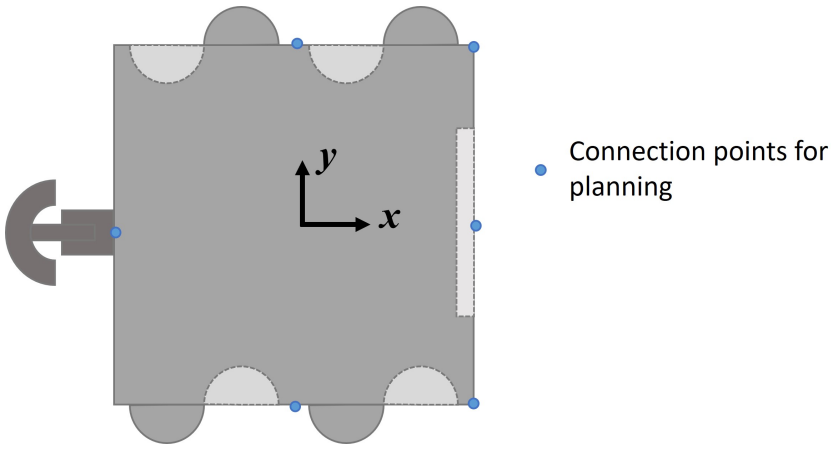

Figure S4: Connection points on a robot for computing the connection pair task queue in Figure **??**A.

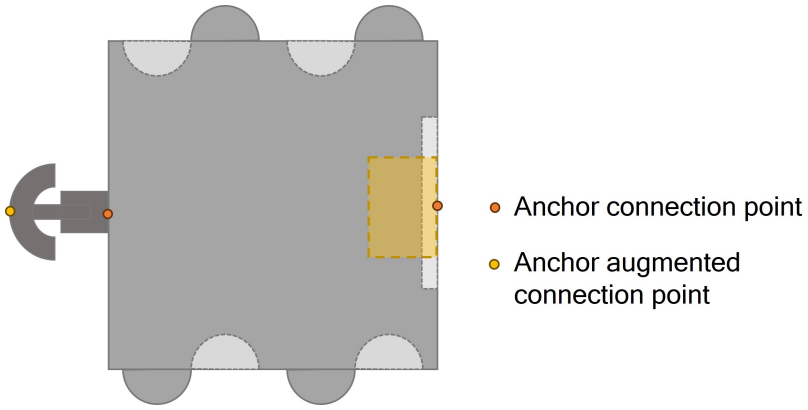

Figure S5: Anchor connection points for planning, the augmented anchor connection point, and the polygon constraints for coupling and connection maintenance.

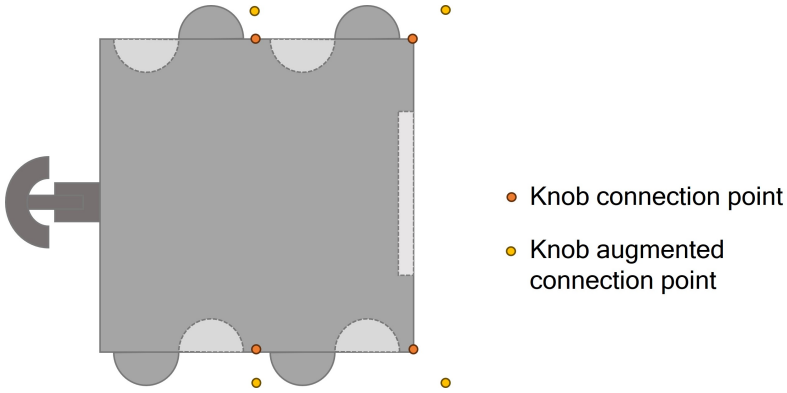

Figure S6: Knob connection points for planning, and the augmented connection points for coupling and connection maintenance.

4 Connection pair augmentation

As shown in Figure S4, we define six connection points on the robot body for initial task assignment. To obtain the task queue as shown in Figure **??**, the planner iterate 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 (***?***).

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.

Algorithm 1 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($\mathcal{C}$)
  for (C_i, C_j) in $\mathcal{C}$ **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
    $\tilde{\mathcal{C}}$.append((C_i, C_j))
  return $\tilde{\mathcal{C}}$

After obtaining the goal connection pairs based on a distance-induced graph, we augment the pairs as in Algorithm 1. In Algorithm 1, 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 2, we 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.

5 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

Algorithm 2 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($\mathcal{C}_{conn}$, $\mathcal{C}_{active}$, $\tilde{\mathcal{C}}$, ϵ)
  for (C_i, C_j) in $\tilde{\mathcal{C}}$ **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 $\leftarrow$ *head_aligned*
    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 $\leftarrow$ *head_inserted*
      if R_a.head not in polygon(R_b, ϵ) **then**
        (C_i, C_j).status $\leftarrow$ *decoupled*
    if (C_i, C_j).status is *head_inserted* **then**
      $\mathcal{C}_{active}$.remove((C_i, C_j))
      $\mathcal{C}_{conn}$.append((C_i, C_j))
  return $\mathcal{C}_{conn}$, $\mathcal{C}_{active}$

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 S7, 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 (***?***), 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

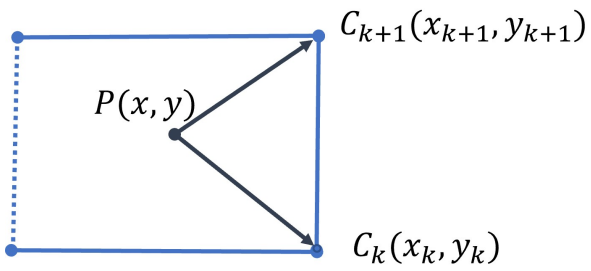

Figure S7: Point in polygon.

vector $\overrightarrow{C_kC_{k+1}}$. According to the right-hand rule of cross-product, we have

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

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

$$\begin{aligned} \begin{bmatrix} x_k - x \\ y_k - y \end{bmatrix} \times \begin{bmatrix} x_{k+1} - x \\ y_{k+1} - y \end{bmatrix} &\geq 0 \\ (x_k - x)(y_{k+1} - y) - (y_k - y)(x_{k+1} - x) &\geq 0 \\ \begin{bmatrix} y_{k+1} - y_k & -(x_{k+1} - x_k) \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_k \\ y_k \end{bmatrix} \right) &\leq 0 \end{aligned} \tag{2}$$

Denote the augmented connection point frame in Figure S5 to be $P_a \in SE(2)$. The homogeneous transformation from the robot i body frame R_i to the connection point frame is

$$g_{R_iP_a} = \begin{bmatrix} 1 & 0 & a_x \\ 0 & 1 & a_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

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.

$$g_{R_jP_a} = g_{R_jW}\ g_{WR_i}\ g_{R_iP_a} \tag{4}$$

$$= g_{WR_j}^{-1}\ g_{WR_i}\ g_{R_iP_a} \tag{5}$$

$$= \begin{bmatrix} \mathcal{R}(-\theta_j) & -\mathcal{R}(-\theta_j)\begin{bmatrix} x_j \\ y_j \end{bmatrix} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{R}(\theta_i) & \begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & a_x \\ 0 & 1 & a_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} \mathcal{R}(-\theta_j) & -\mathcal{R}(-\theta_j)\begin{bmatrix} x_j \\ y_j \end{bmatrix} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{R}(\theta_i) & \mathcal{R}(\theta_i)\begin{bmatrix} a_x \\ a_y \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ 0 & 1 \end{bmatrix} \tag{7}$$

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

$$\mathcal{R}(-\theta_j)\left(\mathcal{R}(\theta_i)\begin{bmatrix} a_x \\ a_y \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \end{bmatrix}\right) - \mathcal{R}(-\theta_j)\begin{bmatrix} x_j \\ y_j \end{bmatrix} \tag{8}$$

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 (8) into the coordinates of the point $P = [x, y]^\intercal$ in Equation (2), and denote this as:

$$pip(P_a, \overrightarrow{C_kC_{k+1}}) \leq 0 \tag{9}$$

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 (***?***), 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.

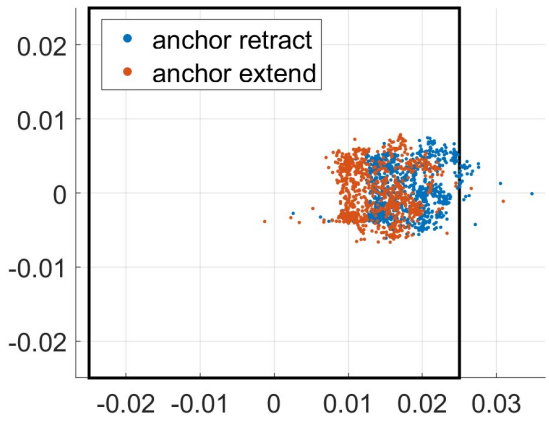

Figure S8: Projected anchor head positions when two robots are coupled. The plot shows the projected positions when the anchor is fully retracted and fully extended.

6 Modeling constraint geometry

To create the bounding polygon as a constraint to the passive soft anchor as shown in Figure S7, 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 S8. 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 S9. 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 robot body. For Figure S9A and Figure S9B constraints respectively, we have

$$g^x_{R_iC_j} \geq L/2 \tag{10}$$

$$g^x_{R_jC_i} \leq -L/2 \tag{11}$$

where $g^x_{R_iC_j}$ represents the translation of corner point C_j in x-axis of the robot body frame R_i,

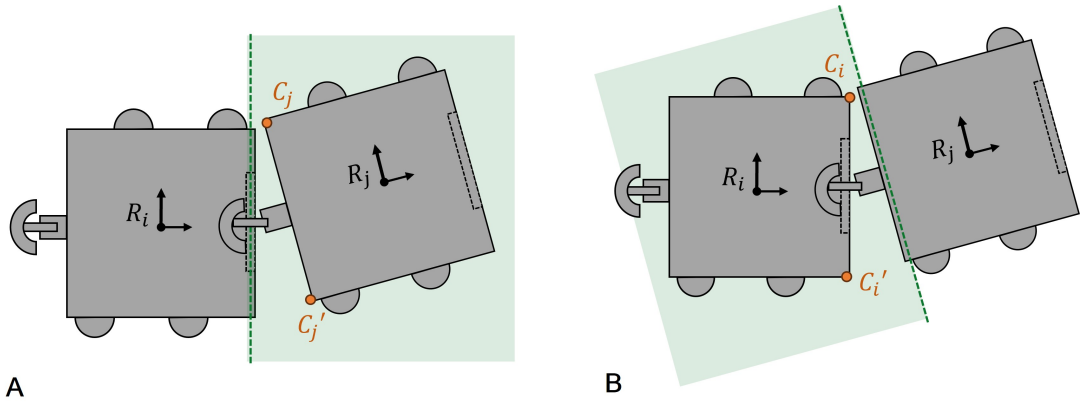

Figure S9: The cutting plane constraints for inter-robot connections.

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.

7 Distributed MPC framework details

Algorithm 3 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 **??**. 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.

$$\min_{x_i,u_i} w_f\|\mathbf{c}_i(H)-\mathbf{c}_j\|_2^2 + w_c\sum_{k=0}^{H-1}\|\mathbf{c}_i(k)-\mathbf{c}_j\|_2^2$$
$$+ w_s\sum_{k=0}^{H-1}\|\mathbf{u}_i(k)-\mathbf{u}_i(k+1)\|_2^2 \tag{12}$$
$$\text{s.t. } x_i(k=0) = x_i(t) \tag{13}$$
$$x_i(k+1) = f(x_i(k), u_i(k)) \tag{14}$$
$$x_i(k)\in\mathcal{X}, u_i(k)\in\mathcal{U},\ k=0,\dots,H \tag{15}$$
$$pip(P_a(k), \overrightarrow{C_m(k)C_{m+1}(k)}) \le 0,\ k=0,\dots,H_c \tag{16}$$
$$line(x_i(k), x_j(k)) \le 0,\ k=0,\dots,H_c \tag{17}$$

The MPC formulation for a couping behavior is shown above. For robot i, the objective in Equation 12 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 $H \in \mathbb{Z}^+$.

Algorithm 3 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, $\mathcal{C}_{conn}$, $\mathcal{C}_{active}$)
  for $robot_i$ in robots **do** ▷ done in parallel
    x_i = state of $robot_i$
    $u_i \leftarrow$ MPC(x_i, $\mathcal{C}_{conn}$, $\mathcal{C}_{active}$)
    add u_i to u
    add $cost_i$ to $cost$
  return u

8 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 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 (12). For a none leader robot within a segment, the objective of the MPC becomes

$$w_f\|\mathbf{x}_i(H) - \mathbf{x}_{leader}\|_2^2 + w_c \sum_{k=0}^{H-1} \|\mathbf{x}_i(k) - \mathbf{x}_{leader}\|_2^2 + w_s \sum_{k=0}^{H-1} \|\mathbf{u}_i(k) - \mathbf{u}_i(k+1)\|_2^2 \qquad (18)$$

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 (12). 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

$$w_f\|\mathbf{x}_i(H) - \mathbf{x}^{ref}(H)\|_2^2 + w_c \sum_{k=0}^{H-1} \|\mathbf{x}_i(k) - \mathbf{x}^{ref}(k)\|_2^2 \qquad (19)$$

The constraints remain the same as in the previous section. The non-leader robots follow the MPC formulation as in Equation (18).

9 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 $0\,\mathrm{mm}$, and then artificial, 3D printed rough terrains with surface variances of $1\,\mathrm{mm}$, $2\,\mathrm{mm}$, $3\,\mathrm{mm}$, and $4\,\mathrm{mm}$.

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 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 $0\,\mathrm{mm}$, 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 $0\,\mathrm{mm}$ (flat terrain) and $4\,\mathrm{mm}$, 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 $5\,\mathrm{mm}$, individual and paired PuzzleBots are statistically significantly different in all three metrics for the pairwise t-tests. Figure **??** shows that the paired PuzzleBots outperformed individual PuzzleBots for the 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.

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 1: Experiment analysis for terrain traversal.

10 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.