Decentralized Model Predictive Control for Constrained Multi-Robot System

Allison J. Seo
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA
ajseo@andrew.cmu.edu

Sha Yi
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA
shayi@andrew.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA
sycara@andrew.cmu.edu

***Abstract*—Multi-robot systems (MRS) have shown collective behaviors and enhanced capabilities in the literature. Real-time control of MRS is challenging due to the exponentially growing state space. This scalability issue becomes more difficult with a highly constrained system. In this paper, we present a Model Predictive Control (MPC) with a decentralized state space and parallelized computations. Our MPC framework effectively models the dense constraints of reconfigurable multi-robot systems and enables scalable real-time control of the system. We show that the proposed MPC enables significantly faster computation compared with an MPC with a centralized state space. We tested our algorithm on up to eight robots in simulation and three robots on the hardware platform. Detailed implementation can be found here: https://github.com/allisonjseo/decentralized-mpc.**

I. Introduction

Multi-robot systems (MRS) have gained significant attention in fields such as search and rescue operations, environmental monitoring, and industrial automation. Through collaboration, these systems have capabilities that surpass those of individual robots. Multiple robots can not only perform efficient exploration [7] but also can form structures that enable them to traverse uneven terrain or cross gaps on the ground [15]. While the wheeled mobile robot has simple kinematics, robot control can be a challenge when there are multiple robots involved. Since constraints can also be different for each robot, it can be difficult to obtain good robot control [4].

To enable scalable control of MRS, researchers have explored ways to implement controllers in a decentralized manner. Xing et al. [12] applied their feedback linear consensus algorithm to a decentralized control law, where robots could access their respective local controller and the states of their neighbors to coordinate a collective behavior. Other researchers have considered various coordination methods, such as Ant Colony Optimization [7] and leader-follower [5]. In the last several decades, Model Predictive Control (MPC), which optimizes an objective function while taking into account the current state, the future predictions, and the constraints of the system, has proven to be an efficient framework for nonlinear systems [3]. As MPC computation time depends on the number of constraints, researchers have explored ways to reduce the number of constraints through decentralized MPC. Tallamraju et al. [9] showed that this can be effective in obstacle avoidance for MRS, and [6, 8] achieved good results in real-time control by combining a form of decentralized MPC with a high-level planner for autonomous robots. A decentralized and prioritized MPC has also been shown to overcome synchronization limitations, leading to faster convergence compared to centralized algorithms, for a system of unmanned aerial vehicles [2].

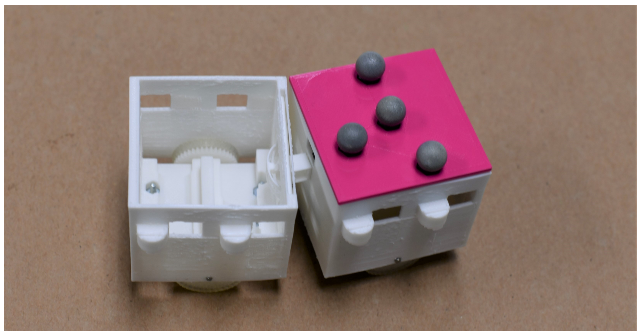
Fig. 1: Two Puzzlebots couple by inserting one Puzzlebot's anchor into the opening of the other robot (circuitry removed for clarity). This coupling mechanism allows for flexible yet stable connections.

MPC remains limited in multi-robot systems due to its computational complexity, making it challenging to solve dynamic optimization problems in real-time [4]. The large state spaces for a centralized system lead to increasingly expensive computations, which inhibits the scalability of MPC. In addition, for robots that physically interact with each other, it is difficult to define and incorporate the desired physical interactions as constraints into the MPC framework. Depending on the robot system and behavior, these constraints can get very dense. For these reasons, it can be anticipated that the MPC computation time will exhibit exponential growth as more agents are added to the system. While decomposing a centralized state space into a decentralized framework may speed up the computation, the optimality may decrease and further influence the performance of the controller. Compared with multi-robot systems that only consider collision avoidance when robots come close to each other, pairwise physical constraints are denser, making it challenging for the controller to run in real-time.

In our previous work [13, 14], we proposed the Puzzlebot system - a multi-robot system where robots physically couple to form functional structures. We modeled the physical inter-

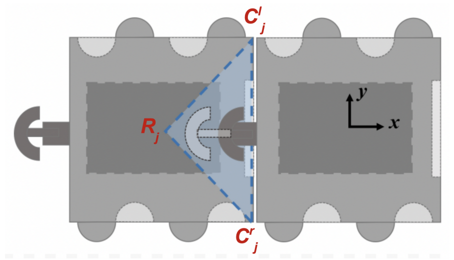

Fig. 2: The blue triangle represents the constraint of the right robot (robot i)'s anchor head while it is coupled with the left robot (robot j). C_j^r and C_j^l denote the front right corner and the front left corner of the robot j, respectively [15].

actions between robots as geometric polygon constraints [15], and incorporated them into a centralized MPC framework. The dense pairwise constraints are essential for maintaining the passive structure of the multi-robot system. While this formulation successfully controls the robot to form a given configuration, it is not scalable to more than four robots in real time. In this paper, We propose a decentralized MPC framework for such a robot system with dense constraints. Our research differs from previous approaches due to the large number of changing physical constraints in our optimization problem between time steps.

In addition to ensuring that constraints, particularly the maintenance of coupling structures among the robots, are met, our MPC framework uses the decentralized state space to define an optimization problem for each robot at every time step. In our simulation results, we test the decentralized MPC on up to eight robots to investigate the scalability of our approach, as well as the optimality and performance compared with the centralized method.

The structure of this paper is as follows. Section II discusses the kinematic model of the robots, the coupling behavior, and the MPC setup. Section III outlines the simulation results. Section IV presents the conclusion and potential future works.

II. Methodology

In this section, we first review the robot model and kinematics. Then, we review the constraints required for coupling the robots and how they are represented in the MPC framework.

A. Robot Model and Dynamics

The robots in this paper are Puzzlebots [15], which are shown in Figure 1. For more details about the robot, our previous paper [15] should be referenced.

Each robot's main structure is 50 mm in depth, 50 mm in width, 45 mm in height, and weighs 68 g. The anchor, which serves as our coupling mechanism, is flexible but robust enough to hold the weight of multiple robots and requires no additional power for the coupling process. Each robot has an anchor on the back and an opening in the front. To couple, one robot's anchor is inserted into the opening of another robot. With this design, the anchor can be inserted easily, requiring much less force than what is provided by the robot. It is more difficult to remove an inserted anchor.

Consider a multi-robot system of N robots. The state and control input of robot $i \in 1, 2, \dots, N$, are $\mathbf{x}_i = [p_{x_i}, p_{y_i}, \theta_i, v_i, w_i]^\intercal$, $\mathbf{u}_i = [\dot{v}_i, \dot{w}_i]^\intercal$ respectively. $p_{x_i}, p_{y_i} \in \mathbb{R}$ are the positions of robot i in the x and y axis. $\theta_i \in (-\pi, \pi]$ is the robot's heading angle (yaw). v_i and w_i are the robot's linear and angular velocities. We model the robot dynamics with a unicycle dynamics model as follows:

$$\dot{\mathbf{x}}_\mathbf{i} = f(\mathbf{x}_\mathbf{i}, \mathbf{u}_\mathbf{i}) = \begin{bmatrix} v_i \cos\theta_i \\ v_i \sin\theta_i \\ w_i \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_\mathbf{i} \tag{1}$$

This equation is incorporated into our MPC framework, which is specified in Section II-C.

B. Control of Coupling Behaviors

Once robots are coupled, they should avoid movement that may cause them to unintentionally decouple. To maintain the coupling status while enabling sufficient flexibility for movement between the robots, we consider the anchor head C_i on robot i to be within a polygon region of robot j, as shown in Figure 2. This polygon formulation can be modeled as linear constraints into the MPC framework [15].

These polygon constraints are a set of linear constraints that we have derived from the Point-In-Polygon problem. Consider the connection pair (C_i, C_j), where C_i is the anchor point on robot i and C_j is the body point on robot j. The linear constraint, where $C_i = [x_i, y_i]^\intercal$ is constrained to be inside a triangle defined by three points on robot j, R_j, C_j^r, and C_j^l (Figure 2), is

$$\begin{bmatrix} y_{k+1} - y_k & -(x_{k+1} - x_k) \end{bmatrix} \left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_k \\ y_k \end{bmatrix} \right) \leq 0 \tag{2}$$

where the point of interest $[x_k, y_k]^\intercal \in \{C_j^r, C_j^l, R_j\}$. We simplify this constraint by writing $pip(C_i | R_j, C_j^r, C_j^l) \leq 0$ in the formulation of the MPC framework in Section II-C.

C. Model Predictive Control Setup and Algorithm

MPC is an optimal control strategy that minimizes an objective function. At each time step, the behavior of the system is predicted over a horizon. Based on these predictions, an objective function is minimized according to a set of constraints. Only the input values for the current time step are used. As the same process is repeated at the next time step, this is known as a moving or receding horizon strategy. This constant recalculation makes MPC adaptable to dynamic environments.

As seen in Equation 1, we use non-linear unicycle dynamics for our robots. Based on these dynamics and other constraints to be introduced later, our Model Predictive Control (MPC) minimizes its cost functions at every time step. For our paper, we consider the behavior of aligning and coupling a set of robots.

The constraints for each robot are as follows. First, there is an initial constraint that will make the optimization start

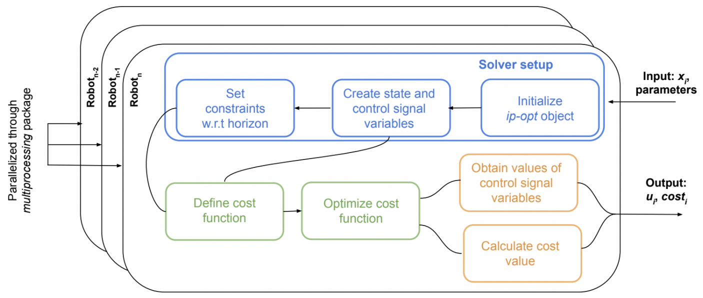
Parallelized through *multiprocessing* package
Robot$_{n-2}$
Robot$_{n-1}$
Robot$_n$
Solver setup
Set constraints w.r.t horizon
Create state and control signal variables
Initialize *ip-opt* object
Input: x_i, parameters
Define cost function
Optimize cost function
Obtain values of control signal variables
Calculate cost value
Output: u_i, cost$_i$

Fig. 3: Overview of the decentralized Model Predictive Controller (MPC) which calculates the optimal control signal of each robot independently and in parallel.

from the robot's current state. Then, there are dynamics update constraints on the robot's state variables between horizons that are based on the unicycle model. Based on the actuation limits, we also set constraints on the linear and angular velocities of the robot. Finally, we check if the robot is part of a connection pair in C_{conn}, which is the set of already coupled robots. If so, we set the polygon constraints to ensure that the robot's anchor point stays in its respective polygon, and thus remains coupled with the other robot, or vice versa.

We define the cost function as the sum of the cost of aligning connection pairs and the stage cost. If the robot is part of an active connection pair in C_{active} - that is, currently in the process of aligning or coupling with another robot - we calculate a weighted norm-2 error of the alignment based on the current status. The same calculations are done if the robot is part of an already-connected pair. Given this framework, the MPC will then optimize one robot's cost function to obtain the control signal based on the current status of the other robots. Compared with the centralized MPC in [15], this formulation decomposed the state space and control signal into a decentralized space. Each robot only optimized its own states and control signals, which significantly reduces the size of state space and constraints.

The MPC is formulated as follows:

$$\min_{x_i,u_i} W_c \sum_{k=0}^{H_c} J(\mathcal{C}_{conn}, t+k|t) + W_m \sum_{k=0}^{H_m-1} J(t+k|t) \tag{3}$$

subject to,

$$x_i(t|t) = x_i(t) \tag{4}$$

$$\begin{aligned} x_i(t+k+1|t) &= x_i(t+k|t) \\ &\quad + f\left(x_i(t+k|t), u_i(t+k|t)\right)\Delta t \end{aligned} \tag{5}$$

$$\text{for } k = 0, \ldots, H_m - 1$$

$$pip(C_i(t+k|t)|R_j(t+k|t), C_j^r(t+k|t), C_j^l(t+k|t)) \tag{6}$$

$$\text{for } k = 1, \ldots, H_c, \text{if } (C_i, C_j) \in \mathcal{C}_{conn} \text{ where } j \neq i$$

$$pip(C_j(t+k|t)|R_i(t+k|t), C_i^r(t+k|t), C_i^l(t+k|t)) \tag{7}$$

$$\text{for } k = 1, \ldots, H_c, \text{if } (C_j, C_i) \in \mathcal{C}_{conn} \text{ where } j \neq i$$

$$x_i(t+k|t) \in \mathcal{X}, u_i(t+k|t) \in \mathcal{U}, \ k = 0, \ldots, H_m \tag{8}$$

In this MPC formulation, we calculate the objective function for robot i at time step t and obtain the control signal u_i. Since the optimization recomputes at every time step, the constraints are not essential for the entire MPC horizon H_m [10]. Thus, the constant horizon $H_c \leq H_m$. The objective function, Equation (3), includes the cost of the coupling behavior $\mathcal{J}$, the weight for maintaining already coupled pairs W_c, and the weight for the stage cost W_m. The set of already coupled pairs is denoted as $\mathcal{C}_{conn}$. There can be up to four sets of constraints for robot i. First, the initial constraint, Equation (4), ensures that the optimization starts from the robot's current state. Then, the constraints for the dynamics update with the unicycle model in Equation (1) is seen in Equation (5). The constraints for coupled pairs are incorporated in Equation (6) and Equation (7), depending on which of the two connection points robot i contains, as outlined in the inequality (2). These constraints are enforced on the pairwise robots with their coupled neighboring robots. It is also necessary to maintain these constraints throughout the execution phase, which makes the constraints dense and challenging. We also incorporate actuation constraints in Equation (8). $\mathcal{U} = \{u_i | u_{\min} \leq u_i \leq u_{\max}\}$ is the constraint set for the acceleration values. $\mathcal{X} = \{x_i | v_i \leq |\frac{w_{\max}}{v_{\max}} w_i|\}$ is the constraint set for the velocities of the robot. Once constraints are set, we use a warm start for the state variables x_i.

Algorithm 1 Decentralized MPC for Coupling Behavior

Input: T: target configuration, x: robot states at time t
Output: u: control input for robots

```
 1: function COUPLEPAIRS($T$, $x$)
 2:     $u$ =[]
 3:     $cost = 0$
 4:     $\mathcal{C}_{conn}$, $\mathcal{C}_{active}$ ← getPairs($x$)
 5:     for $robot_i$ in robots do                    ▷ done in parallel
 6:         $x_i$ = state of $robot_i$
 7:         $u_i$ ← MPC($x_i$, $\mathcal{C}_{conn}$, $\mathcal{C}_{active}$)
 8:         add $u_i$ to $u$
 9:         add $cost_i$ to $cost$
10:     end for
11:     return $u$
12: end function
```

Algorithm 1 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 obtain 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. Next, 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 3. In the MPC function, we initialize the robot's variables and set constraints. The objective function, as defined earlier in this section, is minimized and the control signal u_i is obtained and returned.

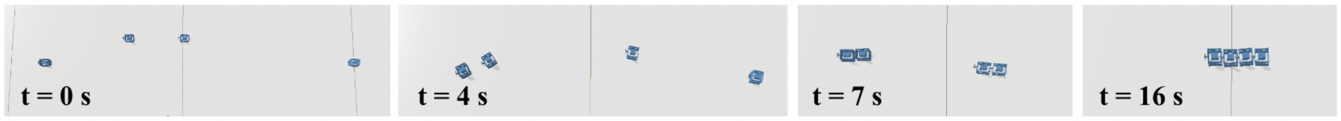

Fig. 4: Screenshots from simulator. Robots first couple into pairs, then the pairs join to create groups of four, and on to eventually form a line of connections. In this example, four robots successfully form a line of connections.

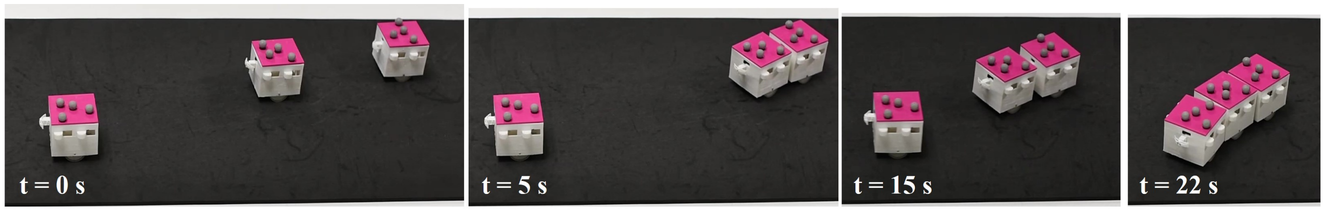

Fig. 5: Screenshots of three robots couple to form a line.

III. Results

We tested our decentralized MPC algorithm on the PuzzleBots [15] system in simulation. We improved the scalability of the highly-constrained MPC. Although the decentralized method provides a sub-optimal result compared with the join-space centralized method, our approach is able to successfully drive the robots to couple and perform the intended tasks.

A. Experiment Setup

We tested our algorithm on a MacBook Pro. The MPC framework is written in Python and uses the CasADi [1] interface's *ip-opt* [11] class, which is a non-linear optimization problem solver. Since each robot's MPC formulation is independent from those of other robots, we use the Python *multiprocessing* package's Pool object to run each robot's computations in parallel. Because Python has the Global Interpreter Lock, users cannot run multiple threads on one process - however, the *multiprocessing* package allows users to spawn multiple Python processes, each with one thread. The Pool object can be reused, so only one is instantiated per simulation run and is reused at every time step.

B. Results

The desired behavior is for all robots to couple as a line. We compare our decentralized MPC algorithm with a centralized one. The simulator, which can be found at https://github.com/allisonjseo/decentralized-mpc, was run 50-60 times for the different numbers of robots for each algorithm. The behavior we tested is coupling into a line formation as shown in Figure 4. For the decentralized MPC, the computation time for each time step equals $max([t_0, \ldots, t_n])$, where t_i is the computation time for robot i. The average computation time and cost of each time step for both algorithms is shown in Figure 6. Compared with the original centralized MPC, the decentralized MPC was significantly faster and more scalable. However, considering the total cost as the optimality of the controller, the decentralized MPC gives sub-optimal results, while still being able to perform the desired behavior. We also tested our algorithm on three PuzzleBots on the hardware system as shown in Figure 5.

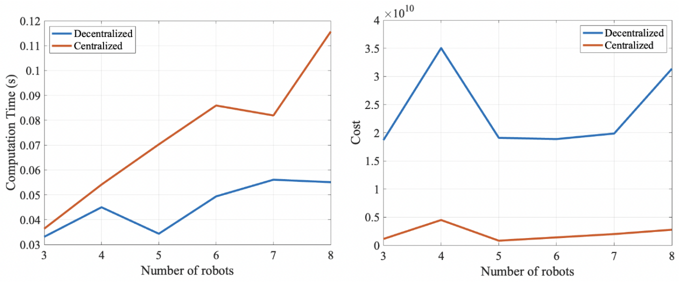

Fig. 6: The total computation time (left) and cost (right) for each time step versus the number of robots.

IV. Conclusion and Future Works

In this paper, we presented an MPC framework with a decentralized state space and parallelized computations for multi-robot systems with dense constraints. The MPC framework included four types of constraints, one of which used connection points to formulate a set of linear constraints derived from the Point-In-Polygon problem. We tested our framework with the behavior of alignment and coupling of Puzzlebots, and we ran the experiments in both simulation and hardware platforms.

While the decentralized MPC framework requires less computation, it has a lower accuracy than the centralized MPC framework. In a semi-decentralized state space, coupled robots can form a local joint space and share the same variables. An MPC framework with a semi-decentralized state space may achieve a lower computation complexity than the centralized MPC and better performance than the decentralized MPC. Furthermore, the experiments of this paper focus on one behavior. By testing on a variety of swarm behaviors, we can understand how the decentralized state space affects the execution of different desired robot interactions.

References

[1] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. volume 11, pages 1–36, 2019. doi: 10.1007/s12532-018-0139-4.

[2] Michal Cáp, Peter Novák, Martin Selecký, Jan Faigl, and Jiff Vokffnek. Asynchronous decentralized prioritized planning for coordination in multi-robot system. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3822–3829, 2013. doi: 10.1109/IROS.2013.6696903.

[3] A.L. Dontchev. Model predictive control. In *Lectures on Variational Analysis, Applied Mathematical Sciences, vol 205*, 2021. doi: doi:10.1007/978-3-030-79911-3_20.

[4] Felipe Kuhne, Walter Fetter Lages, and J Gomes da Silva Jr. Model predictive control of a mobile robot using linearization. In *Proceedings of mechatronics and robotics*, pages 525–530. Citeseer, 2004.

[5] Youhei Kume, Yasuhisa Hirata, Kazuhiro Kosuge, Hajime Asama, Hayato Kaetsu, and Kuniaki Kawabata. Decentralized control of multiple mobile robots transporting a single object in coordination without using force/torque sensors. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 3, pages 3004–3009. IEEE, 2001.

[6] David Hyunchul Shim, H Jin Kim, and Shankar Sastry. Decentralized nonlinear model predictive control of multiple flying robots. In *42nd IEEE International Conference on Decision and Control (CDC)*, volume 4, pages 3621–3626. IEEE, 2003.

[7] Alexandru-Calin Stan. A decentralised control method for unknown environment exploration using turtlebot 3 multi-robot system. In *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6, 2022. doi: 10.1109/ECAI54874.2022.9847497.

[8] 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.

[9] Rahul Tallamraju, Sujit Rajappa, Michael J Black, Kamalakar Karlapalem, and Aamir Ahmad. Decentralized mpc based obstacle avoidance for multi-robot target tracking scenarios. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–8. IEEE, 2018.

[10] 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.

[11] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. volume 106, pages 25–57, 2006. doi: 10.1007/s10107-004-0559-y.

[12] Guansheng Xing, Jianxun Zhang, and Hao Ju. Decentralized control in alignment motion of mobile robots network. In *2008 7th World Congress on Intelligent Control and Automation*, pages 3207–3212. IEEE, 2008.

[13] 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.

[14] Sha Yi, Zeynep Temel, and Katia Sycara. Configuration control for physical coupling of heterogeneous robot swarms. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4268–4274. IEEE, 2022.

[15] Sha Yi, Katia Sycara, and Zeynep Temel. Reconfigurable robot control using flexible coupling mechanisms. *Robotics: Science and Systems (RSS)*, 2023.