Parallel Self-Assembly with SMORES-EP, a Modular Robot

Chao Liu[1], Qian Lin[2], Hyun Kim[1], and Mark Yim[1]

***Abstract*— Self-assembly of modular robotic systems enables the construction of complex robotic configurations to adapt to different tasks. This paper presents a framework for SMORES types of modular robots to efficiently self-reconfigure into tree topologies. These modular robots form kinematic chains that have been shown to be capable of a large variety of manipulation and locomotion tasks, yet they can reconfigure using a mobile reconfiguration. A desired kinematic topology can be mapped onto a planar pattern with optimal module assignment based on the modules' location, then the mobile reconfiguration assembly process can be executed in parallel. The framework is demonstrated on the SMORES-EP platform.**

I. INTRODUCTION

It is common in nature that groups of individuals can form a variety of structures in order to overcome the limited capability of each individual, especially for insects who often need to collaborate in large groups to finish tasks. This collective intelligence has inspired researchers in robotics to develop similar robotic systems. Modular robots are composed of numerous simple building blocks, or modules, which can be formed into various morphologies. While each individual usually has limited capability, multiple modules are able to execute complicated tasks by assembling into suitable configurations. Separated modules that have independent mobility can autonomously come together to form arbitrary configurations. Determining the sequence of motions to form a goal configuration of modules is called *self-assembly planning*.

Self-assembly is related to self-reconfiguration for modular robots. There are in general three styles of self-reconfiguration among the variety of self-reconfigurable robot architectures: lattice, chain and mobile style [1]. Lattice style reconfiguration occurs with modules rearranging themselves to positions on a virtual lattice while maintaining a single connected component. Chain style occurs between modules forming kinematic chains again in a single connected component. The mobile style occurs where modules can separate from each other and move on the environment. Thus self-assembly is mostly related to the latter style.

The self-assembly ability is able to change the interaction between the robot and the environment dramatically. For example, a SMORES [2] module has four active rotational degrees-of-freedom (DOF), pan, tilt and left/right wheels. It has differential wheeled drive using its left and right wheels. An example collaborative behaviour using this mobility is obstacle crossing. Whereas a single module cannot cross a gap that is larger than the width of one module, multiple modules can form a snake configuration to overcome this difficulty. Other applications include reaching tall spaces or rapid simultaneous exploration. These behaviors are similar to the swarming behaviors of bees and ants.

Modular robots are capable of assembling themselves into different kinematic structures with locomotion and manipulation capabilities, such as a walker with four legs and a mobile vehicle mounted with a manipulator. There are several challenges needed to be addressed:

1) Efficiency is important, especially for modular robotic systems which are supposed to have a large number of modules involved;
2) Many physical constraints have to be considered for real hardware applications;
3) Accurate docking is required which is usually a hard problem for modular robots.

The hardware platform SMORES (**S**elf-assembly **MO**dular **R**obot for **E**xtreme **S**hape-shifting) is first presented in [2] and SMORES-EP is the current version where EP refers to the Electro-Permanent magnets as its connector [3]. Each module is a four degree-of-freedom (*LEFT DOF*, *RIGHT DOF*, *PAN DOF* and *TILT DOF*) system with four connectors (*LEFT Face* or **L**, *RIGHT Face* or **R**, *TOP Face* or **T** and *BOTTOM Face* or **B**) which are equipped with an array of electro-permanent magnets as illustrated in Fig. 1. In particular, LEFT DOF, RIGHT DOF and PAN DOF can continuously rotate to produce a twist motion of docking ports relative to the module body, and TILT DOF is limited to $\pm 90°$ to produce a bending joint. LEFT DOF and RIGHT DOF can also be used as driving wheels when doing differential drive locomotion.

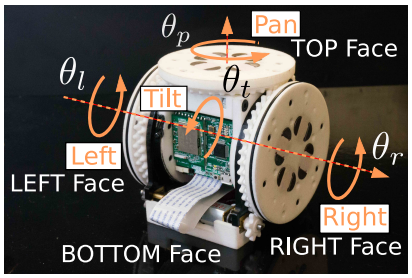


Fig. 1. A SMORES-EP module with four active rotation degrees-of-freedom and four connectors using an array of electro-permanent (EP) magnets.

[1]Chao Liu, Hyun Kim and Mark Yim with GRASP Lab and the Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104, USA {chaoliu, hkim8, yim}@seas.upenn.edu

[2]Qian Lin with the Department of Mechanical Engineering, Tsinghua University, Beijing, China lqa16@mails.tsinghua.edu.cn

The SMORES-EP modules are unique among self-reconfigurable systems in that they can reconfigure in any of the three styles noted earlier. The system can also form chains which has been shown to be one of the more useful configurations for doing things like manipulation of objects or different styles of locomotion. While lattice style reconfiguration mechanisms have been studied in depth [4], the mobile form of reconfiguration similar to self-assembly has not been.

In this paper, a parallel self-assembly algorithm for kinematic topology as well as controllers for hardware execution are presented using SMORES-EP modules. A target kinematic structure can be simply defined as a graph where each vertex is a module and each edge is the connection between adjacent modules. Given the current locations of all modules, every individual is mapped to a module in the target configuration in an optimal way by solving a task assignment problem. Then the assembly actions can be executed in a parallel manner satisfying some physical constraints. A motion controller is also developed to guarantee the success of docking process. The effectiveness and robustness of the framework is demonstrated by the physical hardware.

The paper is organized as follows. Sec. II reviews relevant and previous work. Sec. III introduces the modular robot configuration and how to determine the goal location of each module in an optimal way. The parallel assembly algorithm is presented in Sec. IV and some hardware experiments are shown in Sec. V. Finally, Sec. VI talks about the conclusion and future work.

II. RELATED WORK

Self-assembly problems have been studied for different modular robotic systems. Swarm-bot [5] modules are small mobile platforms and each module has an arm that can be used to link them together. The assembly is limited to simple 2-dimensional structures for planar applications and have been shown with dozens of modules cooperating together. Larger scale planar systems with nearly on thousand modules have been shown with kilobots [6]. Modular boats have formed shapes in water [7] and modular quadrotors have assembled planar shapes in midair [8]. These systems are not applicable to complex locomotion and manipulation tasks. Some self-assembly algorithms are presented in [9], [10], [11], [12], [13] and [14] but limited to modules which do not have non-holomic constraints or planar structures. An approach to solve configuration formation is presented in [15] but in sequential manner which makes the formation process slow. Assembling structures in 3-dimensional space is shown in [16] and [17] but do not deal with the physical constraints of land-mobile platforms. All chain-type modular robot reconfiguration algorithms implicitly solve kinematic models. For example, reconfiguration for Polybot in kinematic topology was presented in [18] but without demonstration on physical systems. Also for chain-type modular robots, modules remain connected during the motion process, thus these systems cannot do self-assembly. Similarly, for lattice-type modular robots, reconfiguration usually happens in 3-dimensional space and some useful techniques have been introduced [19], but these techniques are not applicable to the self-assembly of mobile class of modular robot problem since initially modules are not in one connected component. A distributed reconfiguration algorithm for SMORES-EP is introduced in [20] which focuses on topology reconfiguration by manipulating a part of modules in the initial configuration and it has a different goal that is to minimize the number of reconfiguration actions.

Our work differs from the structural self-assembly work in that our assembly goal is to build a kinematic topology, such as a multi-limbed form or a snake, which has similar locomotion and manipulation ability with other type of robots to interact with environments. Secondly, the modules are not limited to holonomic vehicles with passive connectors. In addition, the hardware in this work, SMORES-EP, has more connectors resulting in more possible configurations since there are more ways to connect two modules, which also complicates the self-assembly process and hardware control. With the locomotion ability of each module, the self-assembly problem can be solved in a parallel manner while taking hardware constraints into consideration. Finally, the motion planner is validated on physical systems with multiple experiments to show its effectiveness and robustness.

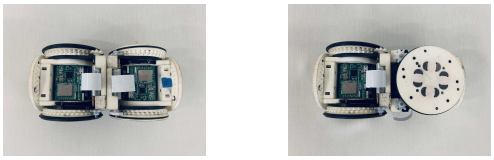
III. ROBOT CONFIGURATION AND ASSEMBLY

A. Modular Robot Topology Configuration

A graph model of modular robot topology was presented in [21]. A modular robot topology can be represented as a graph $G = (V, E)$ where V is the set of vertices of G representing all modules and E is the set of edges of G representing all the connections among modules. Graphs with only one path between each pair of vertices are *trees*. It is convenient to start with tree topology and, if a configuration has loops, it can be converted into an acyclic configuration by running a spanning tree algorithm. Therefore, this work only focuses on configurations in tree topology.

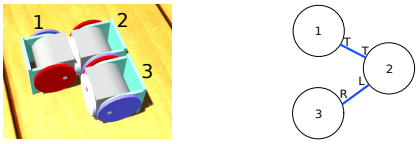
A tree $G = (V, E)$ can be rooted with respect to a vertex $\tau \in V$. Given a modular robot configuration in tree topology, the root is selected as the center of the graph defined in [22]. A linear-time algorithm to compute the root of a modular robot configuration is shown in [21]. Then the *degree* of a vertex (module) v in $G = (V, E)$ denoted as $d(v)$ can be defined that is the number of edges from v to the root τ. There are multiple ways to connect module u and module v denoted as $\mathrm{connect}(u, v)$ from module u's point of view and $\mathrm{connect}(v, u)$ from module v's point of view. Each connection has three attributes: *Face*, *Face2Con* and *Orientation* [21]. Some seemingly different connections are actually equivalent in topology. In a SMORES-EP configuration, attribute *Orientation* is trivial for the connections among LEFT Face, RIGHT Face and TOP Face. For connections between two BOTTOM Faces which cannot rotate, *Orientation* needs to be considered (*Orientation*$\in [0, 1]$) shown in Fig. 2.

With all these information, a SMORES-EP configuration can be fully defined. For example, a three-module config-



(a) (b)

Fig. 2. Connection between two BOTTOM Faces: (a) *Orientation* is 0 and (b) *Orientation* is 1.



(a) (b)

Fig. 3. (a) A three-module configuration and (b) its graph.

uration is shown in Fig. 3a and its graph representation is shown in Fig. 3b. In addition, the root module is Module 2.

B. Self-Assembly Problem

Assume there is a team of modules $\mathcal{M} = \{m_1, m_2, \cdots, m_n\}$ in the Eculidean space $\mathbb{R}^2$. The state of a module $m_i \in \mathcal{M}$ is defined as $p_i = [x_i, y_i, \theta_i]^\intercal$ where $o_i = [x_i, y_i]^\intercal$ is the location of the center of m_i and θ_i is the orientation of m_i. Then the distance between module m_i and m_j can be derived as $\|o_i - o_j\|$. Every SMORES-EP module is a cube with a side length of w. The assembly goal is a SMORES-EP tree topology Configuration $G = (V, E)$ where $|V| = n$. Not all kinematic topology can be built by self-assembly process. Only the kinematic topology that can be unfolded onto a plane can be achieved by a bunch of separated modules on the ground.

***Definition** 1:* The target kinematic topology that can be self-assembled by separated modules is a modular robot configuration $G = (V, E)$ that can be fully unfolded to a plane satisfying:

1) G is a connected graph;
2) The distance between two adjacent modules is w;
3) The center of every module occupies a unique location.

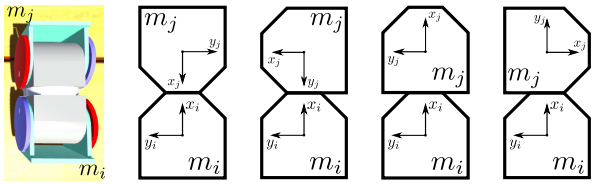
The modules are located at arbitrary locations with constraint that the distance between any pair of modules m_i and m_j denoted as d_{ij} is greater than w. The kinematic topology self-assembly problem is stated. Given a target kinematic topology $G = (V, E)$ and a team of n modules $\mathcal{M} = \{m_1, m_2, \cdots, m_n\}$ where $n = |V|$, find a sequence of collision-free assembly actions to form the target kinematic topology.

IV. PARALLEL SELF-ASSEMBLY ALGORITHM

We propose a parallel self-assembly algorithm for a set of modules to form a desired kinematic topology.

A. Task Assignment

Given a target kinematic topology $G = (V, E)$, first check if it can be derived by self-assembly and, if yes, fully unfold



(a) (b) (c) (d) (e)

Fig. 4. (a) m_i TOP Face is connected with m_j TOP Face and (b) its kinematic diagram. (c) (d) (e) are the kinematic diagrams of three other cases when m_i TOP Face is involved in the connection.

it to the ground. The root module τ of G can be computed in linear time [21]. Then the state of each module $v_i \in V$ with respect to this root module τ denoted as $\bar{p}_i$ after fully unfolding G can be computed in breadth-first search order starting from τ. For example, module m_i is the parent of module m_j, and TOP Face of m_j is attached to TOP Face of m_i shown in Fig. 4a as well as its kinematic diagram shown in Fig. 4b. There are three more cases with TOP Face of m_i involved shown in Fig. 4c — Fig. 4e. In breadth-first search order, when visiting m_j, $\bar{p}_i = [\bar{x}_i, \bar{y}_i, \bar{\theta}_i]$ is already known. The state of m_j with respect to m_i denoted as $\bar{p}_{ji}$ is determined by the involved connectors, e.g. $\bar{p}_{ji} = [w, 0, \pi]^\intercal$ for Fig. 4b. Then

$$\bar{p}_j = \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \bar{p}_{ji} + \bar{p}_i \tag{1}$$

in which $R = \begin{bmatrix} \cos\bar{\theta}_i & -\sin\bar{\theta}_i \\ \sin\bar{\theta}_i & \cos\bar{\theta}_i \end{bmatrix}$.

Given a set of n modules $\mathcal{M}$, each module $m \in \mathcal{M}$ needs to be mapped to a module $v \in V$ in an optimal way that is called the task assignment problem. We want to minimize the total distance that all modules have to travel in order to assemble G. First, the center location of all modules can be defined as $o_c = [x_c, y_c]^\intercal$ where $x_c = \sum_{i=1}^{n} x_i/n$ and $y_c = \sum_{i=1}^{n} y_i/n$. Then the root module is selected as

$$m_\tau = \arg\min_{m_i \in \mathcal{M}} \|o_i - o_c\| \tag{2}$$

The state of every module $m_i \in \mathcal{M}$ with respect to m_τ denoted as $\tilde{p}_i$ can be computed simply by rigid body transformation. Obviously $\tilde{p}_\tau = [0, 0, 0]^\intercal$. Recall that $\bar{o}_i$ is the location of the center of v_i with respect to $\tau \in V$. Given m_τ is mapped to τ, namely $\|\tilde{o}_\tau - \bar{o}_\tau\| = 0$, the distance between every pair of $m_i \in \mathcal{M} \setminus \{m_\tau\}$ and $v_j \in V \setminus \{\tau\}$ is simply $\|\tilde{o}_i - \bar{o}_j\|$ which is the cost of task — moving to location of v_j — for module m_i. Some other factors can also be included in the cost rather than just the distance, such as the orientation. The optimal task assignment problem can be solved by Kuhn-Munkres algorithm or other concurrent assignment and planning of trajectories algorithms in polynomial time [23]. The output is a one-to-one and onto mapping $f : V \to \mathcal{M}$ which is used by a motion planner later to generate the assembly sequence.

B. Parallel Assembly Actions

With mapping $f : V \to \mathcal{M}$, the assembly sequence is computed from root to leaves of G. In each step, all the

Algorithm 1: Parallel Assembly

Input: Target Kinematic Topology $G = (V, E)$ with root τ and depth $d(G)$, $f : V \to \mathcal{M}$
Output: Parallel Assembly Sequence $\mathcal{A}$

$d \leftarrow 1$;
while $d \leq d(G)$ **do**
  Create empty action queue A;
  $\overline{V} \leftarrow \{v \in V | d(v) = d\}$;
  for $v \in \overline{V}$ **do**
    $m \leftarrow f(v)$;
    $m^c \leftarrow f(v^c)$;
    A.enqueue$((m, c, m^c, c'))$;
  $\mathcal{A}$.enqueue(A);
  $d \leftarrow d + 1$;

modules in $\mathcal{M}$ mapped to the modules in the target kinematic topology G with the same depth can be executed in parallel manner. $d(G)$ and $d(v)$ are the depth of rooted graph G and vertex $v \in V$ respectively, then for any vertex with $d(v) > 0$, we denote its parent connected via its connector c as v^c and the mating connector of v^c as c'. An assembly action is simply a tuple in the form of (m_i, c_i, m_j, c_j) meaning connect m_i's connector c_i with m_j's connector c_j. The parallel assembly algorithm is shown in Algorithm 1.

For each group of assembly actions $A \in \mathcal{A}$, all the actions can be executed in parallel except when multiple modules are docking with different connectors of the root module. The group of assembly actions A with the root module involved is separated into two subgroups: one group contains the actions for LEFT Face and RIGHT Face of the root module which is executed first and the other group contains the actions for TOP Face and BOTTOM Face of the root module which is executed later. This is because the root module is not fixed to the ground and it is hard to ensure all attached modules can approach the root module simultaneously. For each assembly action $a = (m_i, c_i, m_j, c_j) \in \mathcal{A}$, a motion controller is running to first navigate m_i to a location close to m_j and the distance is determined by the grid size of the environment, then adjust the pose of m_j to align the connector c_i with c_j, and finally approach c_j to finish docking process.

C. Docking Control

Docking is a difficult process when doing self-assembly. We separate the docking process into three steps to ensure its success: *navigation*, *pose adjustment* and *approach*.

1) Navigation: When doing self-assembly, each SMORES-EP module $m \in \mathcal{M}$ can behave as a differential-drive vehicle with the following kinematics model:

$$\dot{\tilde{p}} = \begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{y}} \\ \dot{\tilde{\theta}} \end{bmatrix} = \begin{bmatrix} \cos\tilde{\theta} & 0 \\ \sin\tilde{\theta} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3}$$

in which v is the linear velocity along x-axis of body frame m_i and ω is the angular velocity around z-axis of body frame m_i which are determined by the velocity of LEFT DOF and RIGHT DOF. Given a set of assembly actions A, the collision-free paths to navigate all involved modules can be generated by some multi-vehicle planner, e.g. [24]. Then, $\forall a = (m_i, c_i, m_j, c_j) \in A$, a simple path following controller is running to control m_i to a location close to m_j.

2) Pose Adjustment: Once the navigation procedure is done, m_i starts to adjust its pose to align the involved connector. If c_i is either LEFT Face or RIGHT Face, then adjust x_i' and θ_i' to zeros where x_i' is the x location of m_i with respect to body frame of $f^{-1}(m_i)$ (the goal pose of m_i) and similar to θ_i'. Otherwise, adjust y_i' and θ_i' to zeros. A kinematics model for the second case can be derived as

$$\begin{bmatrix} \dot{y}_i' \\ \dot{\theta}_i' \end{bmatrix} = \begin{bmatrix} \sin\theta_i' & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4}$$

A control law to make y_i' and θ_i' to converge to zeros is

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \sin\theta_i' & 0 \\ 0 & 1 \end{bmatrix}^{-1} K_{2\times 2} \begin{bmatrix} -y_i' \\ -\theta_i' \end{bmatrix} \tag{5}$$

where K is positive definite and $K = \mathrm{diag}(2, 1)$ in our experiments. Similar controller can be derived for the first case.

3) Approach: The last step is to approach c_j by moving in a straight line which is similar to the controller used in navigation step to follow a given trajectory. If c_i is either TOP Face or BOTTOM Face, then module m_i will first adjust c_i to the right position and then keep moving and pushing m_j until c_i and c_j are fully connected. Otherwise, a helping module shown in Fig. 5 is needed. A new assembly action $(m_H, \mathbf{T}, m_i, \bar{c}_i)$ where m_H is a helping module and $\bar{c}_i$ is LEFT Face if c_i is RIGHT Face, or the reverse. After m_H is connected with m_i, m_i will be lifted up so that it can adjust c_i to the right position, and then lifted down. At last, m_H will keep moving to push m_i to approach m_j for docking. In our setup, there is only one helping module. It is possible to have more to execute assembly actions requiring helping modules in parallel.

V. EXPERIMENTS

Our algorithm was demonstrated with SMORES-EP modules on three tasks. We use a VICON motion capture system to localize the pose of every module and all paths for modules are generated on an empty grid map.

A. Task 1: Mobile Manipulator

The first task is to form a mobile vehicle with an arm that can reach higher locations as is shown in Fig. 6g. There are seven modules involved and the initial locations of all

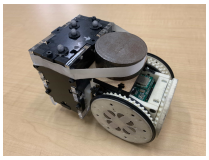
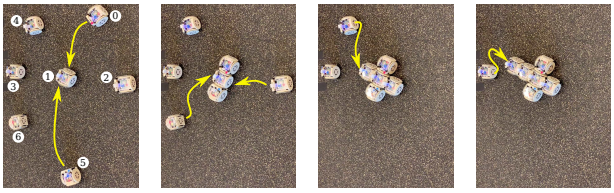


Fig. 5. Helping Module with Some Payload

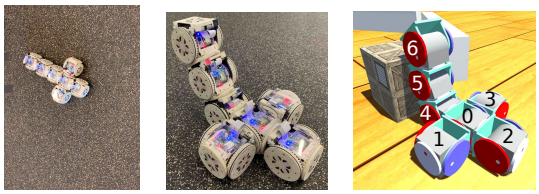
TABLE I

INITIAL LOCATIONS OF ALL MODULES IN TASK 1

Module	x (m)	y (m)	θ (rad)
Module 0	0.017	0.357	1.142
Module 1	0.0	0.0	0.0
Module 2	0.305	0.129	0.641
Module 3	-0.318	-0.132	0.454
Module 4	-0.318	0.158	0.823
Module 5	0.264	-0.448	-0.763
Module 6	-0.172	-0.380	-2.431



(a) (b) (c) (d)



(e) (f) (g)

Fig. 6. SMORES-EP hardware mobile manipulator self-assembly: (a) Execute actions $(0, \mathbf{B}, 1, \mathbf{L})$ and $(5, \mathbf{B}, 1, \mathbf{R})$; (b) Execute actions $(2, \mathbf{B}, 1, \mathbf{T})$ and $(6, \mathbf{T}, 1, \mathbf{B})$; (c) Execute action $(4, \mathbf{T}, 6, \mathbf{B})$; (d) Execute action $(3, \mathbf{T}, 4, \mathbf{B})$. (e) — (f) are the final assembly. (g) is the target kinematic topology.

modules are shown in Table I. Without self-assembly, this task is not feasible. Module 1 is the root module which is closest to the center of all module locations. Then the mapping $f : V \to \mathcal{M}$ is $0 \to 1$, $1 \to 5$, $2 \to 2$, $3 \to 0$, $4 \to 6$, $5 \to 4$ and $6 \to 3$ by Kuhn-Munkres algorithm. The assembly sequence starts from all vertices $v \in V$ with depth of 1 in the target topology $G = (V, E)$. Module 0 and Module 5 start moving first to dock with LEFT Face and RIGHT Face of Module 1 respectively (Fig. 6a), then Module 2 and Module 6 begin the docking process with TOP Face and BOTTOM Face of the root module (Fig. 6b). At last, Module 4 and Module 3 execute the assembly actions successively (Fig. 6c and Fig. 6d). The path of each module in this experiment was recorded shown in Fig. 7. The final assembled configuration is shown in Fig. 6f.

In the docking process, controller for pose adjustment and approach ensures the success of docking. For example, for the last assembly action $(3, \mathbf{T}, 4, \mathbf{B})$, Module 3 first adjusts its body frame (Fig. 8a) so that its TOP Face is aligned with BOTTOM Face of Module 4 (Fig. 8b) shown in Fig. 9a. Then Module 3 moves forward to Module 4 for final docking (Fig. 8c) shown in Fig. 9b.

B. Task 2: Holonomic Vehicle

The second task is to assemble nine modules into a holonomic vehicle in order to do transportation activities

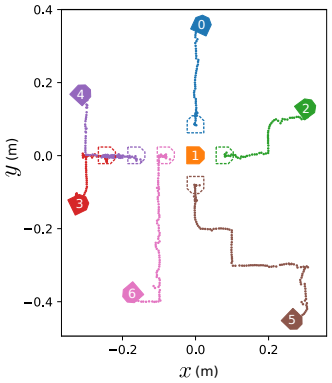


Fig. 7. Actual path of each module for Task 1.

TABLE II

INITIAL LOCATIONS OF ALL MODULES IN TASK 2

Module	x(m)	y(m)	θ(rad)
Module 0	-0.421	0.388	-0.760
Module 1	0.0	0.0	0.0
Module 2	-0.110	-0.469	1.445
Module 3	0.386	-0.143	0.509
Module 4	-0.343	0.082	-2.961
Module 5	0.118	-0.472	1.700
Module 6	0.215	0.428	-2.058
Module 7	-0.342	-0.044	-1.249
Module 8	0.270	-0.317	2.416

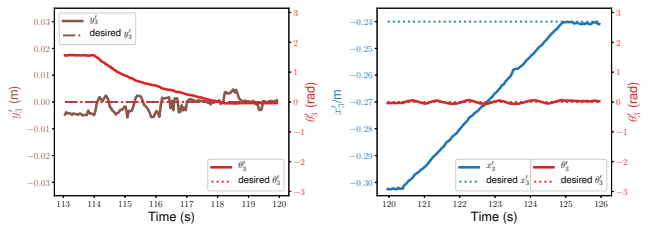
shown in Fig. 10f. The initial location of all modules are shown in Table II and the root module is then selected as Module 1. The mapping $f : V \to \mathcal{M}$ is $0 \to 1$, $1 \to 0$, $2 \to 8$, $3 \to 5$, $4 \to 4$, $5 \to 6$, $6 \to 3$, $7 \to 2$ and $8 \to 7$. The assembly process is shown in Fig. 10a — Fig. 10c and the final assembly is shown in Fig. 10d and Fig. 10e. With our planner and controllers, the recorded actual path of every module in the experiment is illustrated in Fig. 11.

C. Task 3: RC Car

The last task is to assemble seven modules as a vehicle in order to push heavy items shown in Fig. 12f. The initial location of all modules are shown in Table III and the root module is selected as Module 2. The mapping $f : V \to \mathcal{M}$ is $0 \to 2$, $1 \to 1$, $2 \to 3$, $3 \to 5$, $4 \to 7$, $5 \to 6$ and $6 \to 4$. The assembly actions are shown in Fig. 12a — Fig. 12c. For the first step, we need to dock Module 1 RIGHT Face with Module 2 LEFT Face and dock Module 3 LEFT Face with Module 2 RIGHT Face. These two assembly actions

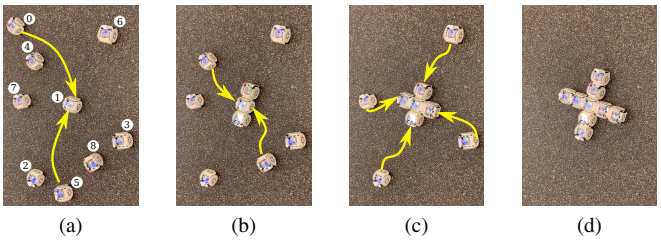
(a) (b) (c)

Fig. 8. Adjustment of position and orientation before execute assembly action $(3, \mathbf{T}, 4, \mathbf{B})$. (a) Module 3 finished navigation process and started to adjust its pose. (b) y_3' and θ_3' have been adjusted and it started to approach the goal for docking. (c) The docking process of Module 3 completed.

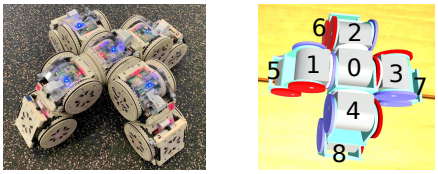


(a) (b)

Fig. 9. Pose adjustment of Module 3 before docking in Task 1: (a) Adjusting of y_3' and θ_3' and (b) adjusting x_3' while maintaining the correct orientation.



(a) (b) (c) (d)



(e) (f)

Fig. 10. SMORES-EP hardware holonomic vehicle self-assembly: (a) Execute assembly actions $(0, \mathbf{T}, 1, \mathbf{L})$ and $(5, \mathbf{T}, 1, \mathbf{R})$; (b) Execute assembly actions $(4, \mathbf{T}, 1, \mathbf{B})$ and $(8, \mathbf{T}, 1, \mathbf{T})$; (c) Execute assembly actions $(3, \mathbf{T}, 8, \mathbf{B})$, $(2, \mathbf{T}, 5, \mathbf{B})$, $(7, \mathbf{T}, 4, \mathbf{B})$ and $(6, \mathbf{T}, 0, \mathbf{B})$. (d) — (e) are the final assembly. (f) is the target kinematic topology.

require the help of a helping module shown in Fig. 12b. The final assembly result is shown in Fig. 12d and Fig. 12e. The recorded path in the experiment is shown in Fig. 13.

VI. CONCLUSIONS

In this paper, we present a parallel modular robot self-assembly algorithm for kinematic topology which can significantly improve the capability of modular robots to interact with the environment. Modules are mapped to those in the target configuration in an optimal way and then the assembly actions can be computed and executed in parallel. Motion controllers are developed to ensure the success of docking among modules. Hardware demonstrations show the effectiveness and robustness of the algorithm.

Future work includes creating demonstrations of arbitrary 3D structures with the SMORES-EP system. This is a capability which, in principle, should be possible with small changes to the algorithm, but is much harder to demonstrate using hardware with the concomitant complications of constraints from actuator limitations for lifting modules. Simulations with much larger structures would also demonstrate the scalability of this algorithm.

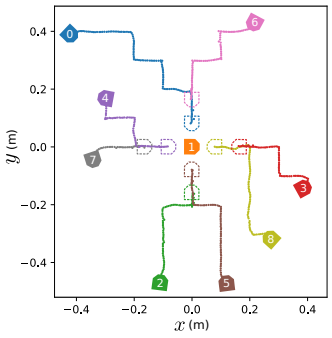
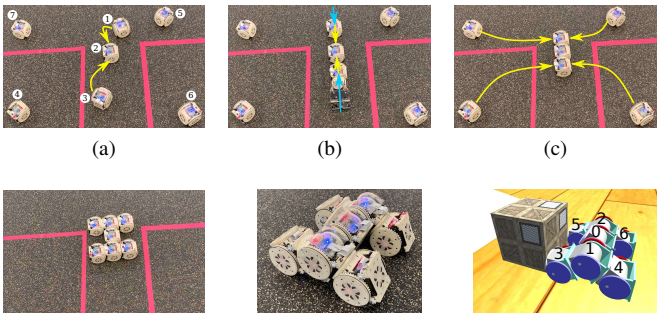


Fig. 11. Actual path of each module for Task 2.

TABLE III

INITIAL LOCATIONS OF ALL MODULES IN TASK 3

Module	x(m)	y(m)	θ(rad)
Module 1	0.070	0.155	-1.352
Module 2	0.0	0.0	0.0
Module 3	-0.066	-0.250	0.997
Module 4	-0.299	0.170	0.811
Module 5	0.311	0.197	-2.539
Module 6	0.330	-0.373	-2.728
Module 7	-0.487	0.218	-0.436



(a) (b) (c)

(d) (e) (f)

Fig. 12. SMORES-EP hardware RC car self-assembly: (a) Execute actions $(1, \mathbf{R}, 2, \mathbf{L})$ and $(3, \mathbf{L}, 2, \mathbf{R})$; (b) Helping module is used to execute the last docking actions; (c) Execute actions $(4, \mathbf{T}, 3, \mathbf{B})$, $(7, \mathbf{T}, 1, \mathbf{B})$, $(6, \mathbf{B}, 3, \mathbf{T})$ and $(5, \mathbf{B}, 1, \mathbf{T})$. (d) — (e) are the final assembly. (f) is the target kinematic topology.

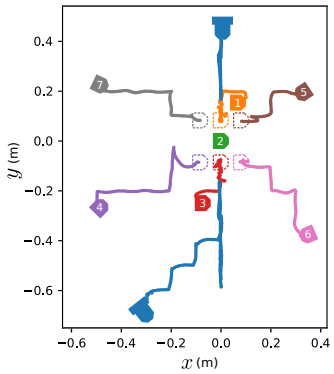


Fig. 13. Actual path of each module in Task 3. The blue blocks without number labeled represent the helping modules.

REFERENCES

[1] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems: Grand challenges of robotics," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[2] J. Davey, N. Kwok, and M. Yim, "Emulating self-reconfigurable robots — design of the smores system," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4464–4469.

[3] T. Tosun, J. Davey, C. Liu, and M. Yim, "Design and characterization of the ep-face connector," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 45–51.

[4] K. Stoy, "How to construct dense objects with self-recondfigurable robots," in *European Robotics Symposium 2006*, H. I. Christensen, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 27–37.

[5] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in a swarm-bot," in *Proc. of the 3rd Int. Symp. on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, K. Murase, K. Sekiyama, N. Kubota, T. Naniwa, and J. Sitte, Eds. Springer, Berlin, Germany, 2006, pp. 314–322.

[6] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[7] I. O'Hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, J. Seo, M. Turpin, V. Kumar, and M. Yim, "Self-assembly of a swarm of autonomous boats into floating structures," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1234–1240.

[8] D. Saldaña, B. Gabrich, G. Li, M. Yim, and V. Kumar, "Modquad: The flying modular structure that self-assembles in midair," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 691–698.

[9] E. Klavins, R. Ghrist, and D. Lipsky, "A grammatical approach to self-organizing robotic systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 949–962, June 2006.

[10] J. Werfel, D. Ingber, and R. Nagpal, "Collective construction of environmentally-adaptive structures," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 2345–2352.

[11] L. Murray, J. Timmis, and A. Tyrrell, "Modular self-assembling and self-reconfiguring e-pucks," *Swarm Intelligence*, vol. 7, no. 2, pp. 83–113, Sep 2013.

[12] J. Seo, M. Yim, and V. Kumar, "Assembly sequence planning for constructing planar structures with rectangular modules," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5477–5482.

[13] H. Li, T. Wang, and G. S. Chirikjian, "Self-assembly planning of a shape by regular modular robots," in *Advances in Reconfigurable Mechanisms and Robots II*, X. Ding, X. Kong, and J. S. Dai, Eds. Cham: Springer International Publishing, 2016, pp. 867–877.

[14] D. Saldaña, B. Gabrich, M. Whitzer, A. Prorok, M. F. M. Campos, M. Yim, and V. Kumar, "A decentralized algorithm for assembling structures with modular robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 2736–2743.

[15] A. Dutta, P. Dasgupta, and C. Nelson, "Distributed configuration formation with modular robots using (sub)graph isomorphism-based approach," *Autonomous Robots*, vol. 43, pp. 837–857, April 2019.

[16] J. Werfel and R. Nagpal, "Three-dimensional construction with mobile robots and modular blocks," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 463–479, 2008.

[17] M. T. Tolley and H. Lipson, "On-line assembly planning for stochastically reconfigurable systems," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1566–1584, 2011.

[18] A. C. Mark H. Yim, David Goldberg, "Connectivity planning for closed-chain reconfiguration," in *SPIE 4196, Sensor Fusion and Decentralized Control in Robotic Systems III*, vol. 4196, Oct 2000.

[19] D. Brandt, "Comparison of a and rrt-connect motion planning techniques for self-reconfiguration planning," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 892–897.

[20] C. Liu, M. Whitzer, and M. Yim, "A distributed reconfiguration planning algorithm for modular robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4231–4238, Oct 2019.

[21] C. Liu and M. Yim, "Configuration recognition with distributed information for modular robots," in *IFRR International Symposium on Robotics Research*, 2017.

[22] G. McColm, "On the structure of random unlabelled acyclic graphs," *Discrete Mathematics*, vol. 277, no. 1, pp. 147–170, 2004.

[23] M. Turpin, N. Michael, and V. Kumar, "Concurrent assignment and planning of trajectories for large teams of interchangeable robots," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 842–848.

[24] B. Binder. tuw_multi_robot. TU Wien. [Online]. Available: http://wiki.ros.org/tuw_multi_robot