

# The Distributed Control and Experiments of Directional Self-assembly for Modular Swarm Robots

Hongxing Wei, *Member, IEEE*, Dezhong Li, Jindong Tan, *Member, IEEE*, Tianmiao Wang

**Abstract**— Self-assembly is a process during which pre-existing components are autonomously organized into some special patterns or structures without human intervention. In this paper, we propose a new control algorithm on distributed self-assembly which is implemented on the Sambot robot platform. A directional self-assembly control model is proposed, in which a configuration connection state table is used to represent the configuration of the robotic structures composed of multiple Sambots. There are three types of Sambots, docking Sambots, SEED Sambot and Connected Sambots. All docking Sambots adopt behavior-based controller that is independent of target configuration. The SEED Sambot and Connected Sambots are used to implement configuration growth. Self-assembly experiments of snake-like and quadruped configurations are conducted on the Sambot platform with five Sambots. The experimental results show the effectiveness and scalability of the distributed self-assembly algorithm.

## I. INTRODUCTION

Self-assembly is a process during which pre-existing components are autonomously organized into some special patterns or structures without human intervention [1]. In the robotic field, self-assembly provides an effective and practical paradigm for the collaboration of multiple robots [2]. At present, researches on self-assembly and autonomous docking of robot mainly focus on modular robots and multiple mobile robots [2, 3].

In the area of autonomous docking of modular robots, Yim *et al.* reported the docking process between PolyBot modular robots, during which a robot arm consisted of six PolyBot could approach and hold up another PolyBot module [4]. Rebenstein *et al.* pointed out that the CONRO robot could implement autonomous docking within a certain distance and angle deviation [5]. Stoy *et al.* used the ATRON self-reconfigurable robot to demonstrate that two three-module robots dock and merge into one large robot [6]. Murata *et al.* proposed a docking method for a self-reconfigurable modular robot M-TRAN, which is based

on simple visual feedback using an additional camera module and LED transmitters equipped on the M-TRAN modules [7].

In the area of docking of mobile robots, Fukuda *et al.* studied a docking system for a cell-structured robot using a hook-type docking mechanism in which the connection mechanism requires a very precise alignment [8]. Hirose *et al.* introduced a distributed and autonomous mobile robot platform as called the Gunryu. Each Gunryu robot was equipped with a common robotic arm that could dock with other Gunryu robots to create multiple Gunryu physical connections [9]. Super Mechano Colony (SMC) was a kind of modular robot that was comprised of the parent module and several sub-unit modules, and each unit was equipped with a robotic arm that could connect the units [10]. Millibot was a modular robotic system that was comprised of several connected track robots, and track robots were connected and locked by the mechanical joints [11]. Delrobaei *et al.* recently found in an experiment that two autonomous mobile robots could successfully dock within certain misalignment [12]. Swarm-bot was composed of a number of autonomous mobile robots known as the s-bot [13]. O'Grady *et al.* proposed a distributed morphology generation mechanism for Swarm-bot, SWARMORPH, which could specify global morphologies using local morphology extension rules that control the self-assembly process [14]. Klavins proposed a graph grammar that could be used to model distributed robotic assembly or formation forming process, and demonstrated the approach using programmable parts testbed (PPT) [15].

There is scarcely any research on self-assemble for multiple robots except Swarm-bot, which does not have the locomotion ability of the chain-typed self-reconfigurable robot system. In our previous researches, we have developed a swarm robot platform called Sambot platform, which has the advantages of both self-reconfigurable robots and self-assembly robots [20]. In this paper, a control algorithm of distributed self-assembly based on the Sambot platform is presented. A directional self-assembly control model is proposed, in which a configuration connection state table is used to represent the configuration of the robotic structures composed of multiple Sambots. All docking Sambots adopt behavior-based controller, which is independent of target configuration. SEED Sambot and Connected Sambots are used to implement configuration growth. For the snake-like and quadruped

This work was supported by the 863 Program of China (2009AA043901, National Natural Science Foundation of China (Grant No. 60525314), the 973 Program of China (2002CB312204-04).

Hongxing Wei, Dezhong Li, Tianmiao Wang are with School of Mechanical Engineering and Automation, Beihang University, Xueyuan Rd. No.37, HaiDian District, Beijing, 100191, China. E-mail: {weihongxing, itm}@buaa.edu.cn.

Jindong Tan, Hongxing Wei is Electrical Engineering Department, Michigan Technological University, Houghton, 49931, USA. E-mail: {jitan, hwei}@mtu.edu.

configuration, the self-assembly experiments have been implemented using five Sambots.

This paper is organized as follows. Section II introduces the Sambot platform. Section III outlines the configuration representation of robotic structure composed of multiple Sambots, and proposes distributed control model of self-assembly for given configuration. Section IV describes control algorithms of SEED, Docking and Connected Sambots in self-assembly process. In section V, experiments of self-assembly for snake-like and quadruped configuration are conducted. Finally, the paper is concluded with prospect of future research in Section VI.

## II. ROBOT PLATFORM

### A. Outline of Sambots

Sambot is an autonomous mobile and self-assembly modular robot consisting of power supply, microprocessor, sensors, actuators, and wireless communication module. As shown in Fig. 1, one Sambot module can connect another module under the guidance of infrared sensors by using hooks. The details design of mechanical structure and control system of the Sambot can be found in the paper [20].

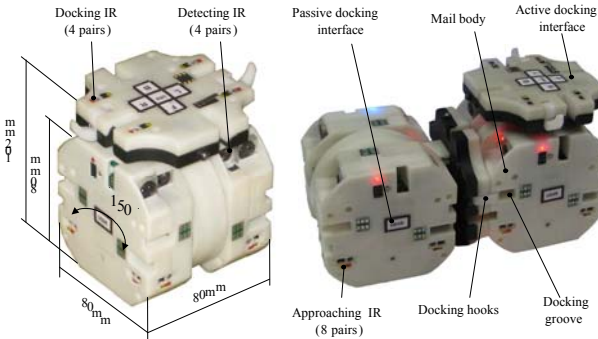


Fig. 1. Outline of the Sambot

### B. Self-assembly and autonomous docking

Self-assembly is one of the major functions and characteristics of Sambot. Autonomous docking between two Sambots is the basis of achieving self-assembly of multiple Sambots.

As shown in Fig. 2, the process of autonomous docking between Sambots is divided into four phases: finding, guiding (navigating), docking and locking. At the beginning of docking, a Sambot (called active Sambot), through its detecting infrared sensors, detects the existence of another Sambot (called passive Sambot). The active Sambot rotates its active docking interface and moves towards the side of the passive Sambot. Meanwhile, the active Sambot switches on the docking infrared sensors at the lower part of the active docking interface. The docking infrared sensor can receive the signals sent by the approaching infrared sensor at the passive docking interface on the passive Sambot. Guided by two pairs of docking infrared sensors, the active Sambot approaches the passive Sambot. When it reaches somewhere

within a certain distance, the mechanical docking touch switch on the active docking interface of the active Sambot is pressed down, which triggers the active docking hooks to lock the docking grooves of the passive Sambot. The docking process is completed.

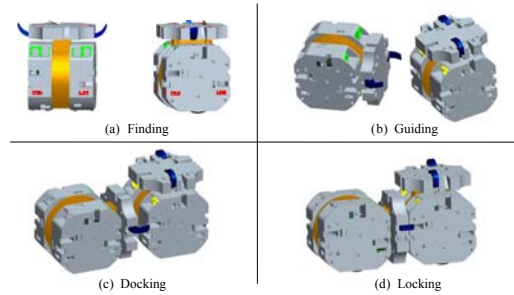


Fig. 2. The autonomous docking of two Sambots

### C. Locomotion of the robotic structure and self-reconfiguration

After multiple Sambots form a robotic structure with certain configuration through self-assembly (as the snake-like robot shown in Fig 3), the robotic structure can move on its own. Additionally, the robotic structure can change its configuration and achieve self-reconfiguration to complete various tasks and adapt to different environments. The mobility of each Sambot enables the robotic structure to realize reconfiguration through disconnection and re-docking between the Sambots. Fig. 3 shows how a robotic structure reconfigures itself from a snake-like robot into a quadruped robot. Since the relative positions and orientations between the Sambots of the snake locomotion are already determined, the Sambot modules in the snake-like configuration can easily determine its position in the quadruped configuration. The flexible locomotion and reconfiguration capability of Sambots enable them to fulfill self-diagnosis and self-repair more easily and more quickly.

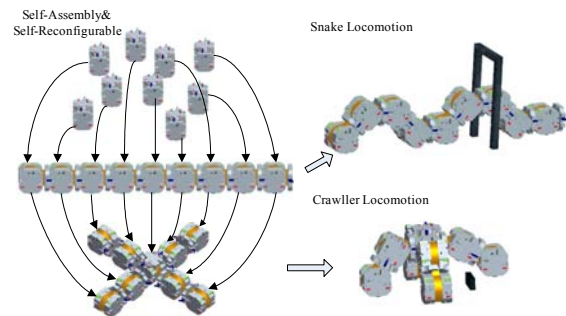


Fig. 3. Self-assembly and locomotion of multiple Sambots

## III. MODEL OF DISTRIBUTED SELF-ASSEMBLY CONTROL

### A. Analysis of configuration

The robotic structure which is comprised of multiple Sambots has outstanding ability of locomotion after assembled. When multiple Sambots are connected, each Sambot is equal to a module of the robotic structure.

As shown in Fig. 4, Sambot can be simplified into a module with one degree of freedom, which is comprised of two links. The central circle stands for the rotating center and it corresponds to the main body of Sambot. The white link stands for the main body of the robot, and the black link stands for the active docking interface.

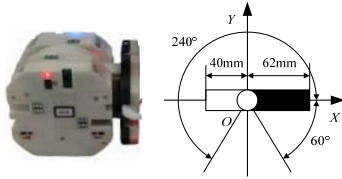


Fig. 4. Simplified configuration of the Sambot

In accordance with distribution relationship between robot joints, we divide the configurations into linear configurations and multipled configurations.

Fig. 5 (a) show a snake-like configuration is composed of six Sambots in line. The robotic structure can make it move like a caterpillar through the rotation of the modules. Fig. 5 (b) shows a four-legged configuration comprised of 15 modules, which resembles a four-legged mammal. The robotic structure can walk through the rotation of leg joints, and foot joints can make sure that the contact between robotic structure and ground is plane, so that the stability can be improved. Fig. 5 (c) ~ (h) show other configurations of different numbers of modules, which can realize different locomotion patterns.

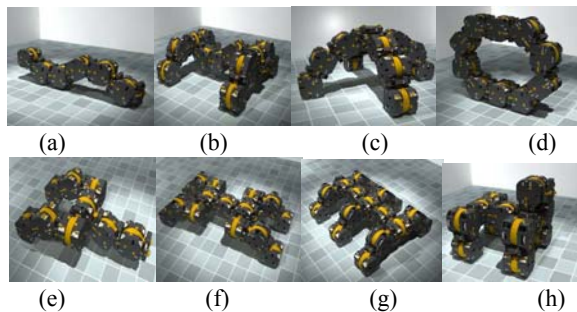


Fig. 5. Basic reconfiguration of the robotic structure. (a) line, (b) track, (c) tripod, (d) parallel quadruped, (e) orthogonal quadruped, (f) six-limbed, (g) extended parallel quadruped, (h) a dog-like configuration.

### B. Configuration representation

Currently, some researches use graph theory to describe the configuration of the self-reconfigurable robot, such as Incidence Matrix [16], adjacency matrix [17], configuration line [18] etc. The Sambot adopts a distributed control strategy with limited computing and storage capacity. We need to reduce the requirements for computing and storage capacity as much as possible. We need to find a unified and effective method to describe the configuration of the robotic structures.

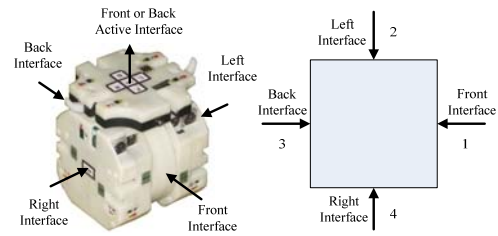


Fig. 6. Connecting relationship of the Sambot

As shown in Fig. 6, each Sambot has an active docking interface and four passive docking interfaces, the Front, the Left, the Back and the Right interface, numbered as 1, 2, 3 and 4 respectively. The connection between each docking interface and the surrounding Sambots can be shown by three kinds of states:

1. Active connection. The interface connects with other Sambot actively, and we set its value to "1". A Sambot can only connect with other Sambot actively through the active docking interface from the Front and the Back.
2. Passive connection. The interface connects with other Sambot passively, and we set its value to "-1". The four interfaces of Sambot, the Front, the Left, the Back and the Right, can be connected as passive docking interface.
3. No connection. The interface does not connect with other Sambot actively or passively, and we set its value to "0".

The Connection states of each Sambot's four interfaces can be shown by an array CS [4], the elements of which respectively express the states of the Front, the Left, the Back and the Right interfaces.

We use graph theory to establish configuration model of the multiple Sambots. We call each Sambot a node, and the connection between each two nodes is called edge. There is only one active docking interface for each Sambot. So, we call the active docking interface directional connection. We can use the tree structures in the graph theory to describe the connections. For example, the tree structure of the quadruped configuration is shown in Fig. 7, where node 1 is the root node, and other nodes are sub-nodes.

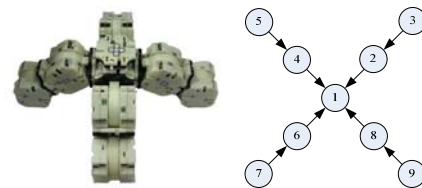


Fig. 7. Tree structure of the quadruped configuration

Because the root node (SEED) of each configuration is unique, we can begin from the root node to traverse the tree structure, and define node sequence of target configuration. The DFS (Depth-first search) algorithm is used to traverse the tree structure in the order of node 1, 2, 3, 4. For a given configuration, we can get combination of its node sequence and the connection state of each node, which is called Configuration Connection State Table (CCST). Table 1 shows the CCST of the quadruped configuration in Fig. 7.

TABLE 1  
Configuration connection state table of the quadruped configuration

Node Order	Node's Connection State	Node Order	Node's Connection State
1	(-1 -1 -1 -1)	6	(1 0 -1 0)
2	(1 0 -1 0)	7	(1 0 0 0)
3	(1 0 0 0)	8	(1 0 -1 0)
4	(1 0 -1 0)	9	(1 0 0 0)
5	(1 0 0 0)		

### C. The task of self-assembly

For a self-assembly task with given configuration, the process start from the Sambot called SEED, which can be selected from any Sambot randomly. After the SEED is selected, we send target configuration's CCST to the SEED. On experimental platform, other robots are known as the Docking Sambot (DSA), which doesn't have any information about the target configuration and any global coordinate and global state information. Considering the differences between current configuration's CCST and final configuration's CCST, SEED decides which docking interface should be docked by the DSA, then switches on approaching infrared sensor (emitter) on this interface, which we call Docking\_Direction, and wait for DSA to dock.

Once DSA docks with SEED, DSA becomes a part of the current configuration called Connected Sambot (CSA). CSA establishes CAN BUS communication with SEED through electrical contact point on the docking interface. At this time, CSA receives final configuration's CCST from SEED, and confirms its label number in the target configuration. Then SEED and CSA update their connection states and current CCST, compares with final configuration's CCST, and decides the next Doking\_Direction.

Fig. 8 shows the self-assembly process of quadruped configuration composed of by nine Sambots. The red arrow points out the next Doking\_Direction. In each self-assembly process, the new DSA randomly selects a Doking\_Direction to join the current configuration to make the structure grow, until it reaches the target configuration.

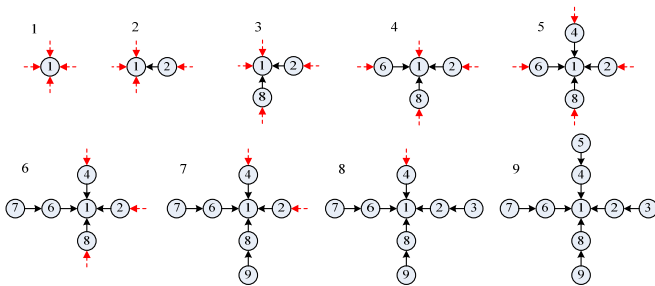


Fig. 8. Self-assembly process of the quadruped configuration

## IV. THE SELF-ASSEMBLY CONTROL ALGORITHM

In this section, we presents the distributed self-assembly control algorithm for a given configuration. We assume that CCST of target configuration is  $CCST_g$ , and it is saved in the SEED Sambot, which is always used as the root node of the new configuration. And the new joined CSA is used as the leaf node of the current configuration. In the process of

self-assembly, SEED, DSA, CSA need to execute different control algorithms and control strategies.

### A. DSA control algorithm

The DSA adopts behavior-based controller [19], which consists of a series of behaviors, and each behavior is used to fulfill a particular task. To perform the function of self-assembly control, Sambot should have four basic behaviors—Wandering, Navigating, Docking and Locking.

1. Wandering. The DSA switches on detecting infrared sensor on the front interface, and wander randomly. The edge height of experimental platform is 40mm (higher than approaching infrared sensor of DSA and lower than detecting infrared sensor of DSA). When DSA wanders to the edge of the experimental platform, approaching infrared sensor gets signals, and then DSA goes backward. When the front detecting infrared sensor receives signals, which means a current configuration is founded, DSA turns to navigating behavior.

2. Navigating. DSA goes anticlockwise around the current configuration to find the Docking\_Direction. This behavior is under the guidance of limited infrared sensors of Sambot. DSA adjusts its orientation to align itself with SEED (the orientation of SEED is predetermined). Then it turns left, until its detecting infrared sensor detects current configuration. Then DSA turns right for a certain angle, and moves forward for a short distance. DSA repeats the process, and it keeps approaching infrared sensor switched on and faced to the current configuration. After receiving a signal, it arrives at the Docking\_Direction. Then the navigating behavior ends, and DSA turns to the docking behavior.

3. Docking. DSA is required to switch on the docking infrared sensors, and adjusts its own position to approach the Docking\_Direction, until the mechanical touch switch on the active docking surface is pressed down. Then the docking hooks can lock the docking groove of SEED. DSA turns to the locking behavior.

4. Locking. This process is simple. The electric touch points of both DSA and SEED contact each other, and send handshaking to affirm completion of CAN BUS communication connection. Afterwards, the DSA sends its basic information such as ID to SEED. DSA becomes CSA. Then, SEED and CSA compare current configuration with target configuration to decide the next Docking\_Direction.

### B. SEED control algorithm

In the self-assembly process, SEED needs to fulfill two tasks as the root node:

1. Configuration comparison: SEED compares current configuration's CCST with target configuration's  $CCST_g$  to decide which interface will become the next Docking\_Direction.

2. Self-assembly process control: SEED collects all the sub-node information connected with SEED as a main control node, and updates current configuration's CCST.

When reaching final configuration, SEED decides whether the self-assembly process has finished.

SEED initializes  $CCST_g$  and the module number ( $Goal$ ) of target configuration, and updates  $CCST$  ( $CCST_j$ ,  $j$  is the module number of the current configuration). In addition, SEED sets the value of its initial connection state as  $Seed\_CS$  to  $[0\ 0\ 0\ 0]$ . The control algorithm of SEED is as follows:

---

```

1. Procedure SEED's control algorithm
2.   Initialization:  $CCST_g$ ,  $goal$ ,  $CCST_j$ ,  $j$ ,  $SEED\_CS=[0\ 0\ 0\ 0]$ 
3.   while( $j \neq Goal$ )
4.     {
5.       while( $SEED\_CS \neq CCST_g.SEED\_CS$ )
6.         {
7.           Decide Docking_Direction of SEED;
8.           DSA dock with SEED; Set new DSA node No.;
9.           Update  $SEED\_CS$ ,  $j$  and  $CCST_j.DSA\_CS=[1\ 0\ 0\ 0]$ ;
10.          Transmit  $CCST_g$  to the DSA;
11.        }
12.      while( not receive information of child node);
13.      Update  $CCST_j$  and  $j$ ;
14.    }
15.  then the Self_Asembly process has been done;
16.  then transmit the result to all nodes
17. End Procedure

```

---

### C. CSA control algorithm

When DSA is connected with SEED or current configuration, DSA become the CSA as a sub-node of the current configuration. A unified control algorithm is executed for all CSA. In the self-assembly process, CSA has two tasks:

1. Configuration comparison: CSA compares its current connection state with final connection state to decide which interface of CSA should be the next Docking\_Direction.

2. Transmitting information to SEED and child-nodes: CSA receives  $CCST_g$  from SEED, and transmits it to its child-nodes. At the same time, CSA collects  $CS$  information sent by its child-nodes, and transmits the  $CS$  information and its  $CS$  information to SEED.

CSA's control algorithm is as follows.  $CSA\_CS_a$  is its current connection state, and 'a' is its own node number in the target configuration.  $CCST_g.CSA\_CS_a$  is the final state.

---

```

1. Procedure CSA's control algorithm
2.   Initialization:  $CSA\_CS_a$ ,  $a$ ,  $CCST_g.CSA\_CS_a$ 
3.   if( received result that the Self_Asembly process has been done )
4.     End Procedure
5.   if( received  $CCST_g$  from the SEED )
6.     {
7.       while( $CSA\_CS_a \neq CCST_g.CSA\_CS_a$ )
8.         {
9.           Decide Docking_Direction of CSA;
10.          DSA dock with CSA; Set new DSA node No.;
11.          Update  $CSA\_CS_a$  and  $j$ , Set  $DSA\_CS=[1\ 0\ 0\ 0]$ ;
12.          Transmit  $CCST_g$  to the DSA;
13.        }
14.      then transmit  $j$ , the No. and  $CSA\_CS_a$  to SEED;
15.      if( received information of the CSA's child node )
16.        {
17.          Transmit the information to SEED;
18.        }
19.    }

```

---

## V. EXPERIMENTS

This section introduces the self-assembly experiments of multiple Sambots. SEED Sambot is located at the center of the bounded experimental platform. DSA could be put anywhere on experimental platform (corner is the most tough position). These experiments show that DSA wanders around to find SEED, navigates to the Docking\_Direction, and autonomously docks with SEED. This assembly process is referred to as directional self-assembly. We have conducted experiments of the self-assembly for snake-like and quadruped configurations.

### A. Self-assembly of snake-like configuration

We conducted experiments of self-assembly of snake like configuration with 5 Sambots on the experimental platform 500 × 800mm. SEED is located at the platform center. The arrow points to the Docking\_Direction. For snake-like configuration, the initial connection state of SEED is  $[0\ 0\ 0\ 0]$ , and its final connection state is  $[-10\ 0\ 0]$ . Therefore, the Docking\_Direction is the Front interface of the SEED.

In order to avoid interference of the infrared sensor of multiple DSA, we introduce one DSA at a time. As shown in Fig. 9, the self-assembly experiment of snake-like configuration with five Sambots is completed within 378 seconds.

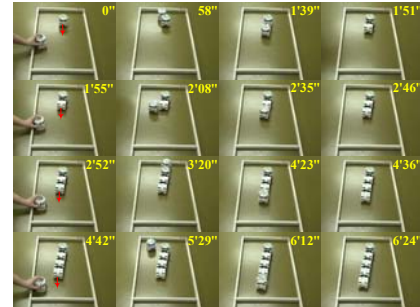


Fig. 9. Self-assembly experiments for snake like configuration

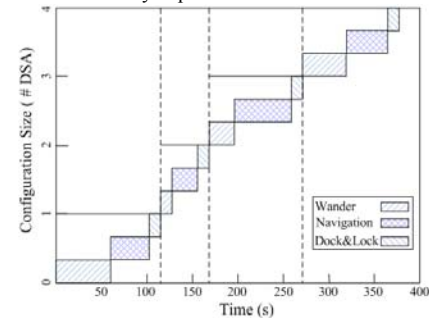


Fig. 10. The directional self-assembly time for snake like configuration with 5 Sambots

Fig. 10 shows the time of self-assembly for the snake like configuration. For each DSA, the Docking and Locking time is relatively even, about 12 seconds. With the growing of the configuration, the wandering time gradually reduces, and navigating time gradually increases.

### B. Self-assembly of quadruped configuration

We conducted experiments of self-assembly of quadruped configuration with 5 Sambot on the experimental platform

600 × 600mm. The initial connection state of SEED is [0 0 0 0], and its final connection state is [-1 -1 -1 -1]. Therefore, the Docking\_Direction of SEED have four interfaces—the Front, the Left, the Back and the Right.

The experiments are shown in Fig. 11 and Fig. 12. They were completed within 290 seconds. This time was shorter than the snake-like configuration. The first reason is that there exist several Docking\_Directions simultaneously to DSA dock. Therefore, the navigating behavior is more efficient. Second, the dimensions of the experimental platform are fewer than the snake-like configuration, so the wandering time is relatively short.

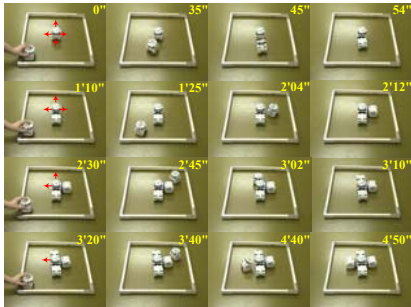


Fig. 11. Self-assembly experiments for quadruped configuration

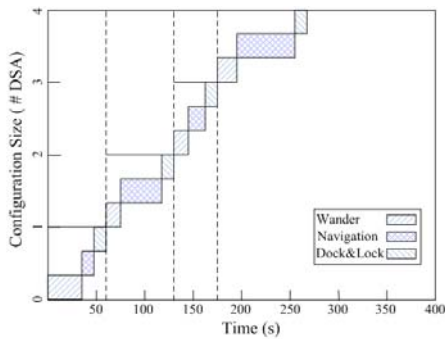


Fig. 12. The directional self-assembly time for quadruped configuration with 5 Sambots

In this directional self-assembly experiment, all the DSA Sambots execute the same control algorithm, which is independent of the target configuration and other Sambots, and the controller can extend to more Sambots. The experimental results show the effectiveness of the algorithm.

## VI. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a distributed self-assembly control algorithm based on Sambot platform. A configuration connection state table is used to represent the configuration composed of multiple Sambots. For a given configuration, the distributed self-assembly control model is proposed. The DSA Sambot adopts a behavior-based controller independent of target configuration. SEED and CSA Sambot are used to control the growth of configuration. We conducted self-assembly experiments for snake-like and quadruped configuration with five Sambots. The experimental results show the effectiveness of the algorithm.

Future research is needed to explore the self-assembly control algorithm simultaneously by multiple Sambots; and the locomotion control methods of the robotic structure composed of multiple Sambots. In addition, the uniform controller from self-assembly to locomotion need to be designed, and a robotic platform whose configuration and function can both evolve need to be built up.

## REFERENCES

- [1] G. M. Whitesides and B. Grzybowski, "Self-Assembly at all scales," *Science*, vol. 295, No. 5564, pp. 2418-2421, Mar. 2002.
- [2] R. Groß and M. Dorigo, "Self-Assembly at the Macroscopic Scale," *Proceedings of the IEEE*, vol. 96(9), pp. 1490 -1508, 2008.
- [3] M. Yim, W. -M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems," *IEEE Robot. Autom. Mag.*, vol. 14, No. 1, pp. 43–52, 2007.
- [4] M. Yim, D. G. Duff, and K. D. Roufas, "PolyBot: A modular reconfigurable robot," in *Proc. Of the 2000 IEEE Int. Conf. Robot. Autom.*, vol. 1, pp. 514–520, Apr. 2000.
- [5] M. Rubenstein, K. Payne, P. Will, and W. Shen, "Docking among independent and autonomous CONRO self-reconfigurable robots," in *Proc. Of the 2004 IEEE Int. Conf. Robot. Autom.*, vol. 3. New York: IEEE, pp. 2877–2882, Apr. 2004.
- [6] K. Stoy, D. J. Christensen, D. Brandt, M. Bordignon, and U. P. Schultz. "Exploit Morphology to Simplify Docking of Self-reconfigurable Robots," *Distributed Autonomous Robotic Systems 8*, Springer Berlin Heidelberg, pp. 441-452, 2009.
- [7] S. Murata, K. Kakomura, and H. Kurokawa. "Toward a scalable modular robotic system: navigation, docking, and integration of M-TRAN," *IEEE Robot. Autom. Mag.*, 14(4): pp. 56-63, 2007.
- [8] T. Fukuda and S. Nakagawa, "Method of autonomous approach, docking and detaching between cells for dynamically reconfigurable robotic system cebot," *JSME Int. J.*, vol. 33, No. 2, pp. 263–268, 1990.
- [9] S. Hirose, T. Shirasu, and E. F. Fukushima, "Proposal for cooperative robot Gunryu composed of autonomous segments," *Robot. Auton. Syst.*, vol. 17, pp. 107–118, 1996.
- [10] R. Damoto, A. Kawakami, and S. Hirose, "Study of super-mechano colony: Concept and basic experimental set-up," *Adv. Robot.*, vol. 15, No. 4, pp. 391–408, 2001.
- [11] H. B. Brown, Jr., J. M. V. Weghe, C. A. Bererton, and P. K. Khosla, "Millibot trains for enhanced mobility," *IEEE/ASME Trans. Mechatron.*, vol. 7, No. 4, pp. 452–461, Jul. 2002.
- [12] M. Delrobaei, K. A. McIsaac, "Connection mechanism for autonomous self-Assembly in mobile robots," *IEEE Trans. Robot.*, pp. 1-7, 2009.
- [13] R. Groß, M. Bonani, F. Mondada and M. Dorigo, "Autonomous self-assembly in swarm-bots," *IEEE Trans. Robot.*, vol. 22, No. 6, pp. 1115-1130, 2006.
- [14] R. O'Grady, A. L. Christensen and M. Dorigo, "SWARMORPH: multi-robot morphogenesis using directional self-assembly," *IEEE Trans. Robot.*, vol. 25, pp. 738-743, 2009.
- [15] E. Klavins, "Programmable Self-Assembly," *IEEE Contr. Syst. Mag.*, Vol. 17, No. 4, pp. 43-56, 2007.
- [16] I.-Ming Chen, Joel W. Burdick, "Enumerating the Non-Isomorphic Assembly Configurations of Modular Robotic Systems," *the Int. J. Robot. Res.*, Vol. 17, No. 7, pp. 702-719, 1998.
- [17] M. Park, S. Chitta, A. Teichman, M. Yim, "Automatic Configuration Recognition Methods in Modular Robots," *the Int. J. Robot. Res.*, Vol. 27, No. 3-4, pp. 403-421, 2008.
- [18] Feili Hou, Wei-Min Shen, "Distributed, dynamic, and autonomous reconfiguration planning for chain-type self-reconfigurable robots," in *proc. Of IEEE Int. Conf. Robot. Autom.*, p2:3135 – 314, 2008.
- [19] R. C. Arkin and T. Balch, "Behavior-based formation control for multi-robot teams," *IEEE Trans. Robot. Autom.*, vol. 14, No. 6, pp. 926-939, 1998.
- [20] Hongxing Wei, Yingpeng Cai, Haiyuan Li, Dezhong Li, Tianmiao Wang, "Sambot: A Self-assembly Modular Robot for Swarm Robot," in *Proc. Of IEEE Int. Conf. on Robot. Autom.*, pp. 66-71, 2010.