# Self-assembly Planning of a Shape by Regular Modular Robots

**Haiyuan Li, Tianmiao Wang and Gregory S. Chirikjian**

**Abstract** We present a self-assembly planning algorithm to allow regular modular robots to assemble into some specified shapes. By tiling a shape using connected lattices in accordance with the robotic geometry, a parallel sequence is obtained to plan self-assembly in successive layers. Robots are identical and autonomous. They use local interactions to seek for a grown shape and update the connection information to determine the corresponding shape. We calculate self-assembly steps of different sizes of shapes to obtain the optimal initial node for square shapes and simulate the self-assembly process.

**Keywords** Self-assembly · Swarm robots · Collective construction

## 1 Introduction

Self-assembly has been viewed as an intelligent approach to enable shaping what are desired by autonomously assembling components. Especially, it has been attracting the robotic community over the last decades [1, 2], because it seems to improve automating construction to some extent and adaptation in setting where human activity is difficult. Self-assembly in robotics can make a shape specified by a user or constrained by environments, which resembles morphogenesis that is a

H. Li (✉) · T. Wang
Beihang University, 37 Xueyuan Road, Beijing 100191, China
e-mail: lihaiyuan@buaa.edu.cn

T. Wang
e-mail: itm@buaa.edu.cn

G.S. Chirikjian
Johns Hopkins University, 3400 N. Charles Street, Baltimore, MD 21218, USA
e-mail: gregc@jhu.edu

© Springer International Publishing Switzerland 2016

biological process where physical processes and constraints affect biological growth to develop an organism's shape. To build a desired two-dimensional shape, unlike using static blocks, we use a group of regular modular robots which have the simple autonomous capability to assemble themselves.

There are three types of robots creating self-assembly ability: robot swarms, collective robotic construction, self-reconfigurable robots. In robot swarms, each individual component forming a shape is an autonomous robot. Inspired by multicellular organisms at micro-scale level or social insects at the colony level, robot swarms cooperate through local interactions in a large group for shape formation. Although an individual robot is simple, swarm intelligence can make the group more powerful. Based on interactions in local area, a swarm of simple and identical agents can construct a polygon in the plane [3]. In addition to the polygon formation, self-assembly of a continuous-space shape was addressed in algorithms and physical system [4]. Nagpal's group demonstrates a thousand-robot swarm form different shapes only using local message and sensing. There robots can aggregate into a formation but they don't have inter-robot physical connection for whole structure. Equipped with the appropriate grasping or connecting mechanisms, distributed robot swarms can connect with the beacon to build a shape [5, 6]. This shape can move collectively or be capable to stand up in 3D space [7].

Collective robotic construction combines the passive blocks and active mobile robots. Inspired by social insects, the mobile ground robots [8] or aerial vehicles [9] or external propulsion/fabrication devices [10] can collect the modular passive blocks to construct complex shapes. A great challenge for collective construction is how to find individual rules for robots, given a shape. Two problems are *forward prediction* and *inverse solution*. Based on individual rules of a robot, *forward prediction* is used to define what they can use passive blocks to build while inversely given a shape, a construction path must be solved in *inverse solution*.

Self-reconfigurable robots can be viewed as a large swarm of connected robots to reconfigure the shape of the single entity without disassembly [2]. Chain-like robot can been used to reconfigure into multi-branch structures [11, 12], however the potential shapes are limited. Chirikjian presents a hexagonal metamorphic robot which can reconfigurable into multi-branch or solid shapes [13]. Reconfiguration planning from an initial configuration into a goal is a key issue [14].

Our work is combination of robot swarms and collective construction. We present a self-assembly planning algorithm for regular modular robots to form some possible shapes according to a certain sequence. A continuous space of shape can be tiled by a large group of tiny regular lattices. These lattices with specified inter-lattice connection can be replaced by robots with same geometry and inter-robot connection. We begin with an overview of the self-assembly problem in Sect. 2. Then we present a complete self-assembly planning algorithm in Sect. 3. In Sect. 4, we analyze the algorithm using some examples and simulate self-assembly process. Finally, we draw conclusions.

## 2 Self-assembly Problem for a Pre-designed Shape

We consider that a large-scale robot swarm assemble into a two-dimensional shape by tiling a user-specified plane with regular polygons. A polygon is called *lattice* which can be occupied by a robot as a part of structure. When these lattices are connected in certain way corresponding to connection mechanism between two robots, it can generate a stable and complete structure.

Considering that a lattice geometry is a regular polygon, the interior angle and edge number of each polygon is $\alpha$ and $n$, respectively. There exits $k \in \mathrm{N}$ such that $k\alpha = 2\pi$ when polygons titling a plane meet at each vertex. Meanwhile, sum of exterior angles of a regular polygon of $n$ edges satisfies $n(\pi - \alpha) = 2\pi$. We can obtain $n = 2 + \frac{4}{k-2}$. The possible value of $n$ is 3, 4 and 6 respectively corresponding to $k$ = 6, 4, and 3. Therefore, the regular polygons to tile the space of a shape are triangles, squares and hexagons. In this paper, we focus on self-assembly of a swarm of robots with *square* geometry. As shown in Figs. 1 and 2, a shape is described in black pixels in a binary image. A group of robots are capable of moving with self-contained controllers and sensors to interact with others and position themselves. Meanwhile, by connection mechanisms, these robots can attach themselves to the grown structure. The self-assembly algorithm presented in this paper work on the swarm robot characterized by the following properties. (1) Modularity. The robots are homogeneous with regular geometry (2) Autonomy, including Mobility, Sensing, and Computations. Robots can move autonomously and interact with others. (3) Inter-robot Communication. Robots can communicate with neighboring robots via interfaces when connected. (4) Connection. The inter-robot connection is created using a pair of mating active/passive connectors.

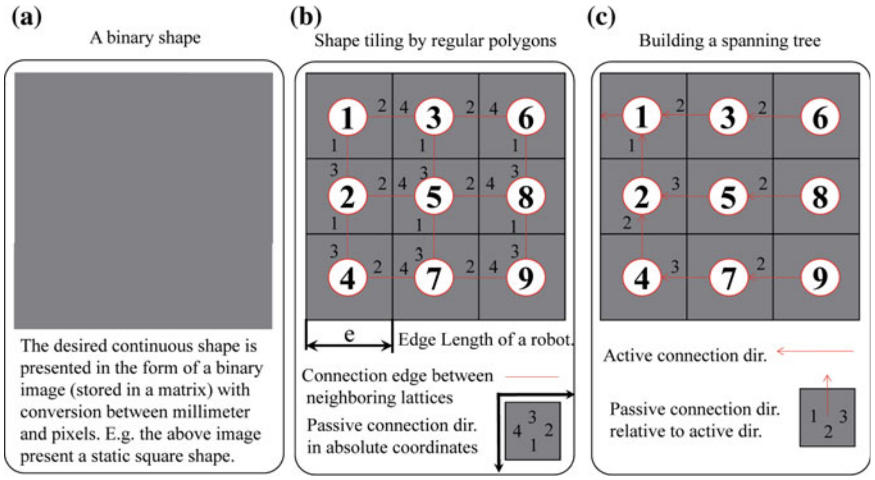
(a)
A binary shape
The desired continuous shape is presented in the form of a binary image (stored in a matrix) with conversion between millimeter and pixels. E.g. the above image present a static square shape.
(b)
Shape tiling by regular polygons
e
Edge Length of a robot
Connection edge between neighboring lattices
Passive connection dir. in absolute coordinates
(c)
Building a spanning tree
Active connection dir.
Passive connection dir. relative to active dir.

**Fig. 1** Self-assembly planning for a spanning tree from a binary shape

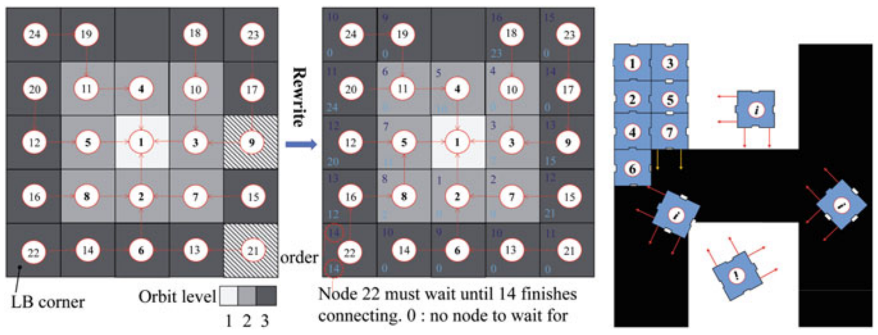
Rewrite
order
LB corner
Orbit level
1 2 3
Node 22 must wait until 14 finishes connecting. 0 : no node to wait for

**Fig. 2** *Left* Rewriting of a self-assembly spanning tree to avoid constrained space. *Right* A group of mobile robots connect to form a shape

These robot capacities have been demonstrated in a variety of modular mobile robots [7]. Connection way between robots determines the self-assembly sequence of a desired shape.

## 3 Self-assembly Planning

A desired structure is presented by a binary image. Each robot occupying a lattice is denoted by a node $v_i$ and the connection between neighboring robots is denoted by an edge $e_{ij}$. A graph $G$ is used to describe self-assembly planning of robot swarms forming the shape.

### *3.1 Graph Construction by Tiling a Shape*

We use a breadth-first algorithm to tile a shape using lattices. Firstly, we build a coordinate system whose originates at the center of the root node to obtain shape matrix $M_{shape}$ from an image. In the counterclockwise direction, starting from the bottom side, four areas are visited one by one. As for each area, a matrix $M_{mask}$ representing the geometry of a robotic lattice is used to cover the pixels. If the visited area is a part of shape (black pixels) and has never been visited, it will be assigned a number and an edge as a newly added node. Algorithm 1 gives the pseudocode procedure of transferring a shape into a group of lattices with edges. We uses ray casting algorithm to find the points inside a robotic lattice and build $M_{mask}$ representing the pixel distribution of a robot shape.

Algorithm 1: Graph construction by using robot swarms' geometry to tile a shape

```
1:  $M_{shape}$ :={$m_{ij}$},  $S$ :=$StackInit$, $G$ := {{$v$},{$e_{ij}$},{$l_e$}}
2:  Define position and orientation of the root node;
3:  visit root node and insert root node into G;
4:  Push($v_0$,$S$)
5:  **while** $S$ != Null, **do**
6:    $V_j$ := Pop($S$)
7:    **for** each neighboring area $i$ around $V_j$, **do**
8:      State = Threshold($M_{shape}$,$M_{mask\_i}$)
9:      **if** "State: unvisited in $i$"
10:       insert new {$v$,$e_{ij}$,$l_e$} in $G$
11:       Push ($v_i$,$S$)
12:     **else if** State:visited area occupied by $j$ in $i$
13:       update {$v$,$e_{ij}$,$l_e$} in $G$
14:     **end if**
15:   **end for**
16: **end while**
17: **return** $G$
```

## 3.2 A Spanning Tree Defining Connection

A robot has only one active connector. Therefore, a robot can only use one active connector to dock with another. Although the shape may have a loop branch, its topology graph is connected and have no cycles. That is, the graph presenting the shape can be transferred into a directed rooted tree.

As for a node, there exits at most one directed edge starting from it and there are at most three directed edges ending up with it. The edge is assigned a weight to denote which side the neighboring node is connected with it on. Problem is to find out a spanning tree $T_{tree} = (V', E', l')$, $V' = \{v_i\}$, $E' \subseteq V' \times V'$, $l' : E' \mapsto R'$, $R' = \{1, 2, \ldots, n\}$, rooted at a node $v_0$. In Fig. 1c, we can obtain a directed rooted tree from a graph and the pseudo code is presented in Algorithm 2. We choose a lattice as a root node to start building a spanning tree, based on breadth-first searching algorithm (BFS). BFS can make more robots participate in self-assembly in parallel because robots can approach and connect with the grown structure from all directions. Because Algorithms 2 also use BFS, it can be inserted in Algorithms 1. However, here we still present it alone in case of a graph given directly rather than a binary image.

Algorithm 2: Self-assembly sequence using breadth-first searching algorithm to obtain a spanning tree

Input $G=(V,E,l_E)$

```
1:  $S_{level}:=\{v_0\}$, $V_{tree}:=\{v_0\}$, $T_{tree}:=\{V_{tree},E_{tree},W_{tree}\}$
2:  for each $x$ in $S_{level}$, do
3:    for each neighbor $y$ of $x$ in $V-V_{tree}$, do
4:      if $x$ != $v_0$
5:        find the parent $z$ of $x$ in
6:      else
7:        $l_E(v_0,v_0)$:= active dir. of the root node
8:      end if
9:      update $w_E(x,y)$ in $W_{tree}$ from $l_E(x,y)$, $l_E(z,x)$
10:     $V_{tree}:=\{V_{tree},y\}$, $E_{tree}:=\{E_{tree},(x,y)\}$
11:   end for
12:   if no neighbor $y$ of $x$ in $V-V_{tree}$
13:     return $T_{tree}:=\{V_{tree},E_{tree},W_{tree}\}$
14:   end  if
15:     $S_{level}$:= unvisited neighbors of $S_{level}$ in $V$
16: end for
```

## 3.3 Rewriting

There are some criteria concerning the topological order for robots in self-assembly planning. A directed path rooted at the root is considered as a critical path. Lattices with circled number are parts of a shape in Fig. 2 while empty lattices don't belong to this shape. A quad-tree is obtained according to Algorithm 2. There are eight critical paths, {1, 2, 6, 13, 21}, {1, 2, 6, 14, 22}, {1, 2, 7, 15}, {1, 2, 8, 16}, {1, 3, 9, 17, 23}, {1, 3, 10, 18}, {1, 4, 11, 19, 24}, and {1, 5, 12, 20}. That is the robot $i + 1$ cannot start docking until the robot $i$ in the same critical path has finished. In the coordinate originated at the root node numbered 1, we think all the nodes along the four edges of a square centered at the root are in the same orbit which has the same background grayscale in Fig. 2. Robots in different critical paths can dock with the grown shape simultaneously but there must be sufficient room in the potential docking side for a robot to maneuver and align its active connector with the passive connector installed on the parent robot. The left or right adjacent square lattice must be open. If two adjacent square lattices like node 9 and 21 are occupied, the robot can't move inside into the position of node 15 to attach to node 7. Therefore, we improve the topological order of the tree and rewrite the rules in Algorithm 3. In this algorithm, we assume except the root node, the parent node of a node $i$ on orbit $O_j$ has to lie on the same orbit $O_j$ or the inner orbit $O_{j-1}$.

```
Algorithm 3 Rewriting the self-assembly sequence
1:  for each $O_j$ in $T_{tree}$, do
2:    for each $P_i$ in $O_j$, do
3:      if $P_i$ is $O_1$ or $P_i$ is not adjacent to $P_{i-1}$
4:        $f(C_{p1})=f(O_{j-1})+1$
5:      else if $P_i \cap P_{i-1}=C_{Pi}$
6:        $f(C_{pi})=f(C_{pi-1})$
7:      else if $P_i$ is adjacent to $P_{i-1}$ and $P_i$ is CW
8:        Transfer $P_i$-$C_{pi}$ into a CW path $'P_{i-1}$,
9:        Attach $'P_{i-1}$ to $P_{i-1}$, Update $f(c_x)$ in $'P_{i-1}$
10:     else if $P_i$ is adjacent to $P_{i-1}$ and $P_i$ is CCW
11:       $f(C_{pi})=f(P_{i-1})+1$
12:       WaitForNode($C_{pi}$) = neighbor node in $P_{i-1}$ to $C_{pi}$
13:     end if
14:     update all $f(c_{k+1})=f(c_k)+1$, $c_k(k=2\ldots m)$ in $P_1$
15:   end for
16:     If $P_n$ is adjacent to $P_1$
17:       Divide $P_n \cup P_1$ into two new paths $P_n'$ and $P_1'$
18:    {$P_n'$ ends up with LB corner, $P_1'$ starts after it}
19:       WaitForNode($C_{LB\ Corner}$) = the first node in $P_1$
20:       Update $f(c_x)$ in $P_1$, $f(c_y)$ in $P_n$
21:     end if
22: end for
23: return ($T_{tree}$, $f(T_{tree})$, WaitForNode($T_{tree}$))
```

The rules consist of two principal metrics. (1) The shape is assembled from inner orbits to outer orbits; (2) any lattice to be docked has at least one empty neighboring lattice. An orbit is denoted by a set $O_j$, e.g. $O_j$ = {2, 7, 3, 10, 4, 11, 5, 8}. The connected nodes on an orbit are in the same critical path. All the paths on the same orbit are denoted $P_k$. On $O_3$, there are eight critical paths $P_1 = \{6, 14, 22\}$, $P_2 = \{6, 13, 21\}$, $P_3 = \{15\}$, $P_4 = \{9, 17, 23\}$, $P_5 = \{18\}$, $P_6 = \{19, 24\}$, $P_7 = \{12, 20\}$, and $P_8 = \{16\}$. Each $P_i$ on $O_{j+1}$ have a child node connected to a parent node on $O_j$ and we call the child node $C_{pi}$. If $P_i$ extends counterclockwise along an orbit, it is a CCW path, e.g. {6, 13, 21}, otherwise it is a CW path, e.g. {6, 14, 22}. Starting visiting all nodes counterclockwise from the left bottom corner, $P_i$ including the first $C_{Pi}$ is $P_1$ and the next path is $P_2 \ldots P_n$. $f(x)$ is the order of a node $x$; $f(O_i)$ presents the maximum of orders of all the nodes on an orbit $O_i$; $f(P_i)$ is the order of the last node when visiting a path $P_i$ on $O_i$ counterclockwise. These results can be seen $P_i \le P_{i+1} \Leftrightarrow f(P_i) \le f(P_{i+1})$, $O_j \le O_{j+1} \Leftrightarrow f(O_j) \le f(O_{j+1})$, $P_i \cap P_{i+1} \neq NULL \Rightarrow P_i \cap P_{i+1} = C_{Pi} = C_{Pi+1}$. If two paths on an orbit are not adjacent, they can begin self-assembly with their own $C_{Pi}$ at the same time, otherwise the latter must wait until the neighboring node in former path has finished connecting. It needs to be mentioned that if the last path is adjacent to the first path, they are rewritten and separated by the node in the left bottom corner. In Fig. 2, we get $T_{tree}$, $f(T_{tree})$ and WaitForNode ($T_{tree}$), where WaitForNode ($x$) means a node for $x \in \mathrm{V}_{\mathrm{tree}}$ to wait for.

### 3.4 Self-assembly Control for Robots

Once the self-assembly order for a desired shape is planned above, the root robot can start the self-assembly process. There are two pairs of emitter-detector infrared sensors installed on each side of a robot. If this robot needs another robot to dock with on a side, it can open the corresponding infrared sensors on this side, and then when a free mobile robot receives this signal, it will connect with it. As for each robot attached to the assembled shape, they determine whether and when to open their sensors according to $T'_{\mathrm{tree}}$. For example, firstly, root node 1 checks there are no empty nodes whose order are smaller than it and starts self-assembly on the next orbit 2. There are 2, 3, 4, 5 child nodes on orbit 2 and however, node 3 needs to wait for node 7, node 4 does wait for node 10 and node 5 does wait for node 11. Node 1 opens the emitter sensors on the side reserved for node 2. Once a free mobile robot maneuvers and detects this signal, it connects with node 1 and becomes node 2. Furthermore, node 1 will send node 2 a sub tree rooted at node 2 belonging to $T'_{\mathrm{tree}}$. After that, node 2 determines whether and when to open its sensors like node 1. And so on, all the nodes on orbit 2 finish connecting to the grown shape, and continue on the next orbits.

If a large number of free mobile robots are recruited, they move toward the grown shape (phototaxis). When they encounter the grown shape, they use edge-following algorithms presented in previous work [15] around the stationary shape to search for the signals. After that, the free mobile robots will dock with the parent nodes. These procedures including phototaxis, edge-following and docking are identical for all the robots maneuvering in parallel. Using these free robots, the shape builds up in successive orbits in the designed self-assembly order.

## 4 Results and Discussion

We use the algorithms to plan self-assembly order for a group of shapes of interest. We assume the size of a robotic lattice is $100 \times 100$ in pixels. Figure 3 shows a few examples. A square shape with size $700 \times 700$ is tiled by 49 lattices and a cross shape is assembled by 45 lattices. Note that number of steps for a group of robots to assemble into a shape depends on the initial position of the root node, partially because we start to plan all the lattices from the left bottom corner on an orbit. Meanwhile, the gaps between the lattices can obviously reduce number of steps on an orbit, although it increases number of orbits.

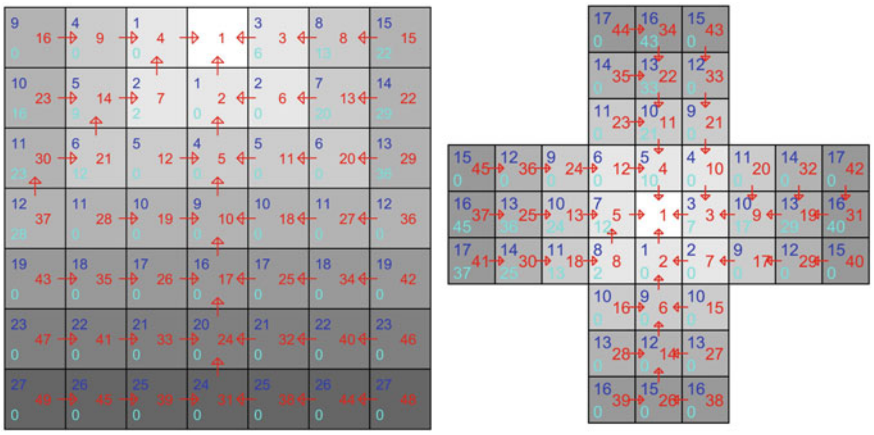
Fig. 3 Examples: self-assembly planning of a square shape and a cross shape

Only considering square shapes, Fig. 4 shows number of steps when 1600 robotic lattices form a 800 × 800 square, starting at different root nodes. Minimum number of steps occurs when root nodes are in the right parts of the first row (Row = 1, Column = 20–40). We implement the algorithms on a variety of sizes of square shapes whose edges are 100$k$ in length. It means these shapes will be assembled by $k^2$ robots. Maximum number of steps for $n = k^2$ robots is $n$, that is, these robots will connect to the grown shape one by one. As shown in Fig. 4 Right, statistically, we found minimum of self-assembly steps of $n$ robotic lattices is $\frac{1}{2}(n + \sqrt{n}), n = k^2$. Here, we prove this equation. When the root note lies on the right parts of the first row, if the length of a square shape increases from $k - 1$ to $k$, minimum of self-assembly steps increases by $k$ steps. Using this recursive formula,

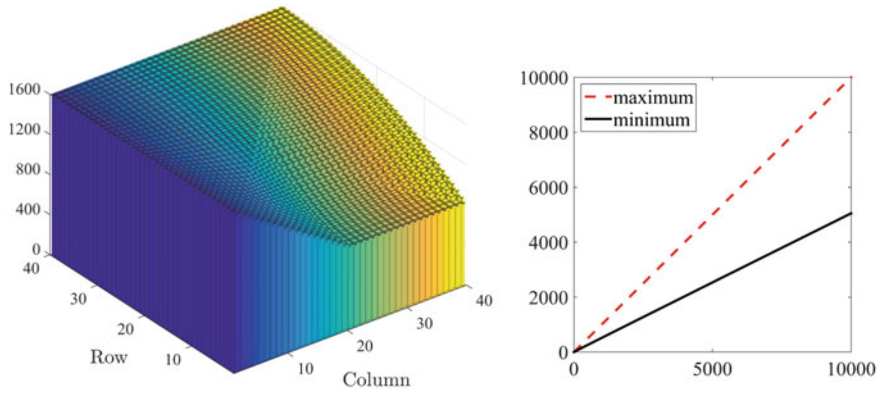

Fig. 4 *Left* number of self-assembly steps for 1600 robots starting at different initial nodes. *Right* Number of self-assembly steps versus number of robots

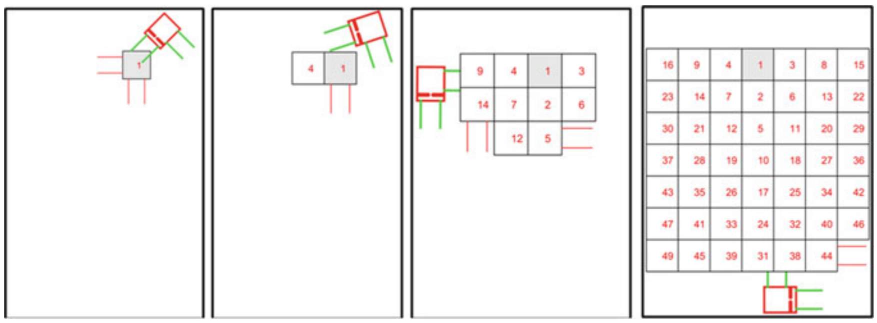

**Fig. 5** Self-assembly demonstration of a square shape

we can obtain $1 + \cdots + k = \frac{1}{2}(k^2 + k)$. The shapes considered to be formed are limited and the constraint is that as for each node in the tree except the root one, its parent node has to lie on the same orbit or the neighboring inner orbit. Therefore, a convex shape can start from center. For a concave shape or ones including holes, a root node should be chosen to meet the above constrains if exists.

We use the self-assembly control to simulate a self-assembly process of a shape. Successive snapshots during this process are shown in Fig. 5. Here, we let only one robot enter the edge-following stage once. The robot in red is following the edge of the grown shape to find an infrared beam in red. Planned self-assembly order gives the robot at least two lattices of space to enter. We believe adding multi-robots simultaneously will greatly improve the efficiency, especially for shapes including more gaps.

## 5 Conclusion

This paper presented a method by which regular robots can use local interactions to build a shape without knowing any information about the desired shape. They only implement their motion until receiving a connection signal from the grown shape. The new node is in self-assembly sequence which is designed by tiling a shape and planning a specified spanning tree. Through analysis of square shapes, we obtained self-assembly starting at the nodes at the second half of the first row can get the least steps. The constraints defining possible shapes were given.

Gaps in a desired shape can make the self-assembly fast, because they can open more nodes at a time. Although in simulation we only put a robot once, the robot may have more than one signal to choose and if so, it will choose the first one, which makes the self-assembly fast to some extent. When multi-robots work simultaneously, self-assembly can be implemented in parallel.

**Acknowledgments** The authors would like to thank China Scholarship Council, “863” Program of China (2012AA041402) and Fundamental Research Funds for the Central Universities (YWF-14-YHXY-015) that supported Mr. Haiyuan Li’s visit at JHU and partial work. Chirikjian’s contribution to this material is based upon work supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

1. Groß, R., Dorigo, M.: Self-assembly at the macroscopic scale. P. IEEE **96**, 1490–1508 (2008)
2. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems (Grand Challenges of Robotics). IEEE Robot. Autom. Mag. **14**(1), 43–52 (2007)
3. Arbuckle, D.J., Requicha, A.A.G.: Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations. Auton. Robot. **28**(2), 197–211 (2010)
4. Michael, R., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science **345**(6198), 795–799 (2014)
5. Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarm-bots. IEEE Trans. Robot. **22**(6), 1115–1130 (2006)
6. Wolfe, K.C., Moses, M.S., Kutzer, M.D.M., Chirikjian, G.S.: M3Express: a low-cost independently-mobile reconfigurable modular robot. In: 2012 IEEE International Conference on Robotics and Automation, pp. 2704–2710. IEEE Press, New York (2012)
7. Wei, H.X., Li, H.Y., Tan, J.D., Wang, T.M.: Self-assembly control and experiments in swarm modular robots. Sci. China Ser. E. **55**(4), 1118–1131 (2012)
8. Werfel, J., Petersen, K., Nagpal, R.: Designing collective behavior in a termite-inspired robot construction team. Science **343**(6172), 754–758 (2014)
9. Lindsey, Q., Mellinger, D., Kumar, V.: Construction with quadrotor teams. Auton. Robot. **33** (3), 323–336 (2012)
10. Moses, M.S., Ma, H., Wolfe, K.C., Chirikjian, G.S.: An architecture for universal construction via modular robotic components. Robot. Auton. Syst. **62**(7), 945–965 (2014)
11. Hou, F., Shen, W.M.: Graph-based optimal reconfiguration planning for self-reconfigurable robots. Robot. Auton. Syst. **62**(7), 1047–1059 (2014)
12. Spröwitz, A., Möckel, R., Vespignani, M., Bonardi, S., Ijspeert, A.J.: Roombots: A hardware perspective on 3D self-reconfiguration and locomotion with a homogeneous modular robot. Robot. Auton. Syst. **62**(7), 1016–1033 (2014)
13. Chirikjian, G., Pamecha, A., Ebert-Uphoff, I.: Evaluating efficiency of self-reconfiguration in a class of modular robots. J. Robotic Syst. **13**(5), 317–338 (1996)
14. Larkworthy, T., Ramamoorthy, S.: A characterization of the reconfiguration space of self-reconfiguring robotic systems. Robotica **29**(01), 73–85 (2011)
15. Wang, T.M., Li, H.Y., Meng, C.: Self-assembling for swarm modular robots using MIMO fuzzy control. Adv. Mech. Eng. 2013, Article ID 598647 (2013)