Decentralized Algorithm for Repeating Pattern Formation by Multiple Robots

Shan Jiang*, Junbin Liang†, Jiannong Cao*, Jia Wang*, Jinlin Chen*, Zhixuan Liang*
*Department of Computing, The Hong Kong Polytechnic University
†School of Computer and Electronic Information, Guangxi University
{cssjiang,csjcao,csjiawang,csjlchen}@comp.polyu.edu.hk, liangjb@gxu.edu.cn, zhixuan.liang@connect.polyu.hk

***Abstract*—Recently, much attention is paid to multi-robot systems due to their widespread applications such as warehouse robotics, persistent surveillance, and exploration of unknown environments. Although urgently required by the applications, coordination among multiple robots remains to be challenging. Among the problems of multi-robot coordination, *pattern formation* serves a fundamental one. It aims to control a group of robots to form a desired shape with some certain goals such as best formation quality, minimum makespan or minimum total distance. Existing works mainly focus on the formation of certain patterns, such as repeating squares or a circle. those approaches cannot be generalized to arbitrary pattern formation. In this paper, we propose a decentralized algorithm for a multi-robot system to generate a given formation with an arbitrary repeating pattern. We introduce *basic pattern graph* and *assembling graph* to define a repeating pattern and *formation quality* for measurement. Towards solving the *repeating pattern formation problem*, our approach is divided into two phases. The robots are grouped into multiple basic patterns in the first phase, and the patterns are assembled level by level in the second phase. Simulations and real-world experiments indicate the effectiveness and practicability of our approach.**

***Index Terms*—multi-robot system, pattern formation, distributed algorithm**

I. Introduction

Recent advances in robotics have enabled the coordination of a vast number of capacity-limited robots to perform complex tasks such as search, patrol, and escort. A system containing large numbers of simple physical robots is called a multi-robot system (MRS) [1]. MRS has been attracting more and more attentions because of its great potential to carry more complex tasks with lower cost than a single robot.

Informally speaking, a pattern is an overall appearance of a system. For instance, a group of students is aligning into a line, a troop of soldiers is assembled into a square, and a flock of gooses is flying in a "V" shape. Here, the "line", "square", and "V" are so-called patterns. As for a system, forming a pattern has several advantages such as enhancing the coordination efficiency and reducing the external impacts. Thus, pattern formation, which aims to arrange a group of robots into structured positions relative to each other, is an important and fundamental problem of MRS [2].

Pattern formation has been a hot research topic in the field of MRS for a long time [3]. However, existing works are dedicated to some specific patterns, *e.g.*, circle [4], or repeating square. Those approaches are hard to be generalized for arbitrary repeating patterns. Also, few of them are deployed in real-world testbed. Thus, the practicability and effectiveness can not be guaranteed.

To solve pattern formation problem in centralized MRS is relatively easy. In a centralized MRS, there exists a central computing station in the system. The central station can be a powerful workstation or a specific robot in the MRS. The station gathers the information from, compute the actions for, and deliver the commands to all the robots. In this case, the problem of pattern formation can be abstracted as a matching and optimally solved in polynomial time [5].

When it comes to distributed MRS, the pattern formation problem becomes very challenging. In a distributed MRS, there is no central computing station, and the robots have to make decisions by themselves. Therefore, coordination and co-operation between multiple robots are required, which serves as the key issue. Compared to centralized MRSs, distributed MRSs are more flexible and robust. Thus, to solve pattern formation problem in distributed MRS is significant and the researchers have been paying many efforts. However, existing works mainly focus on the formation of certain patterns, such as repeating squares or a circle. Those approaches can hardly be generalized to an arbitrary pattern specified by the users.

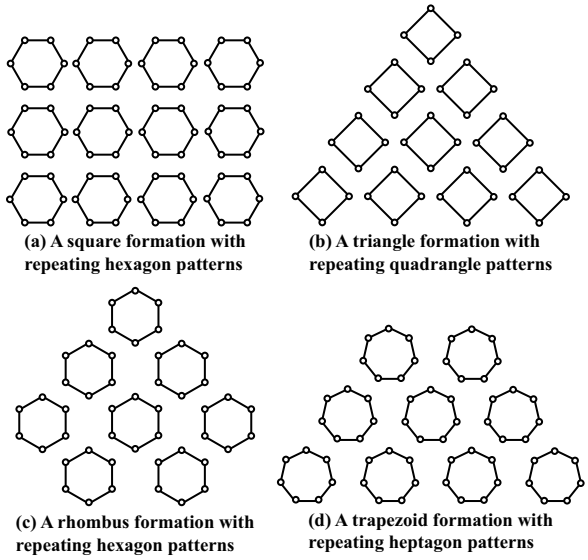
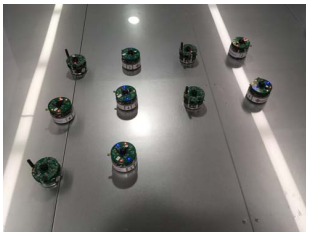


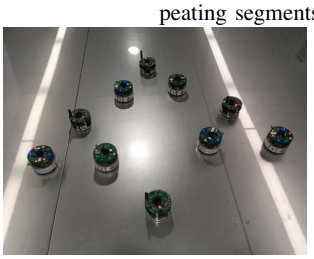
Fig. 1: Four Examples of repeating patterns

978-1-7281-2583-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICPADS47876.2019.00090



(a) A square formation of repeating dots

(b) A triangle formation of repeating segments



(c) A rhombus formation of repeating triangles

Fig. 2: Real-world experiments

In this paper, we tackle the problem of arbitrary repeating pattern formation in distributed MRSs. Firstly, we introduce two concepts *basic pattern graph* and *assembling graph* to facilitate defining a repeating pattern. For example, the four repeating patterns in Fig. 1 are defined by basic patterns graphs in Fig. 3 and assembling graphs in Fig. 4. Secondly, we introduce the concept *formation quality* to evaluate how good an MRS forms a repeating pattern. We use the concept of formation quality to formulate the *repeating pattern formation problem*. Thirdly, we propose a decentralized algorithm towards solving the problem. Our algorithm runs in two phases. In the first phase, the robots forms dispersed basic patterns according to the input of basic pattern graph. In the second phase, the dispersed basic patterns are assembled level by level according to the input of assembling graph. Finally, we conduct simulations in randomly generated MRSs and deploy our algorithm in a real-world testbed. As shown in Fig. 2, three repeating patterns are successfully formed by 10 robots. The main contributions of the paper are:

- We formally define the problem of repeating pattern formation in MRS. The problem formulation includes two important concepts, *i.e.*, basic pattern graph and assembling graph, and a critical metric, *i.e.*, formation quality. To the best of our knowledge, we are the first to define the problem and the performance evaluation criteria formally.
- We propose a decentralized algorithm for repeating pattern formation. The simulation results show that our approach outperforms existing ones significantly.
- We deploy our algorithm in a real-world testbed. The successful deployment demonstrates the practicability of our algorithm. To the best of our knowledge, real-world experiments are not included in most existing works.

II. Related Work

Pattern formation is a typical coordination problem in MRS. Traditional algorithms can be divided into three categories: behavior-based algorithms [6], leader-follower algorithms [7] [8], and virtual structure algorithms [9] [10].

In behavior-based algorithms, simple behaviors are designed for each robot, such as collision avoidance, trajectory tracking, obstacle avoidance, goal seeking and formation keeping. The robots can perform more complex behaviors by interacting with other robots. Behavior-based algorithms are often used by combining with the potential field algorithms [11], which control the movement of the robots through gradients of potential fields. The gradients are defined as the sum of attractive virtual forces and repulsive forces in the potential fields overlaid over the working area. Behavior-based algorithms are easy to deploy and implement. However, they are hard to converge and terminate within a short time, which would cause excessive energy consumption due to continuous robotic movements.

Leader-follower algorithms select one or several robots as leaders and take the others as followers. The leaders compute their final positions in the formation and deciding geometric relationship with the followers. The followers just follow the movement of the leaders and maintain their relationship. This kind of algorithms is also easy to implement. However, they are easy to be affected by the failure of the leaders and lacks of feedbacks among the leaders and the followers.

Virtual structure algorithms organize all the robots into a rigid formation. Expected dynamics of the virtual structure are defined firstly, and then related control laws for the movement of the structure are translated into desired motion of each robot. Coordination of the robots is easy to be implemented by the algorithms, and the generated formation is easy to be maintained. However, applications of the algorithms are limited since the generated formation is hard to be reconfigured.

Above traditional algorithms mainly focus on generating simple formations with single patterns in small-scale MRS. They are not suitable for complex formations with repeating patterns in large-scale MRS. In recent years, some works were conducted to solve the repeating pattern formation problem in large-scale MRS. In particular, they map the problem as a *distributed task assignment problem* by considering different optimization objectives such as minimum total movement distance, or minimum task deadline miss rate, *etc.* Then, some auction and consensus based approaches [12] [13] are designed to solve the problem.

However, these works require that the whole information of the formation and the costs (*e.g.*, time duration or energy consumption) that each robot reaches its target position are known by each robot beforehand, which is impractical. That is because each robot just has limited storage space and it cannot conserve all the position information of the formation. Moreover, due to the uncertainty of the working area, it is hard for each robot to estimate or predict the costs of its movement.

Recently, some works were proposed to conquer the above drawbacks. Shakeel Ahmad et al. [14] proposed a decentralized solution by using a task division method. The global goal

of forming the desired formation is achieved by dividing it into multiple small tasks of forming local patterns. Firstly, a centralized neighbor selection method is used to construct the local patterns. Secondly, a distributed singularity-robust task-priority inverse kinematics method is proposed to control the robots to move and be located at expected positions in the patterns. Since a robot may be located at an overlapping point of several patterns, it would receive multiple moving orders from different neighbors in various patterns. Then, a distributed null space behavioral (NSB) approach is proposed to select appropriate movement based on priorities of the moving orders. This solution has high reliability and low memory requirement, but it is designed for specific formations with losing the generality.

Yang Song et al. [15] [16] proposed a fully distributed algorithm to generate a formation with the arbitrary repeating pattern. A kind of multigraph, called *lattice graph*, is introduced to represent the pattern. The algorithm contains 2 phases. Firstly, all the robots form a tree. Secondly, the root of the tree arranges its children to form a pattern according to the lattice graph; Then, all the component robots in the pattern continue to arrange their children in the tree to form new patterns according to the lattice graph. Above operations continue until all the robots in the system are arranged. The algorithm is general and has low memory requirement. However, it cannot guarantee the shape of the generated formation, *i.e.*, the generated formation may not be the desired formation. That is because it just considers the generation of patterns and does not consider how the patterns are connected to achieve the desired formation.

III. Preliminaries

In this section, we present the system model in Sec. III-A, and formally define the repeating pattern formation problem in Sec. III-B.

A. System Model

Consider a group of n computational entities $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$, namely *robots*, located on an Euclidean plane $\mathbb{R}^2$. The robots are identical and have the capabilities of localization, communication, and actuation. In detail,

- Each robot r_i has a unique identifier, which is used for comparison only.
- At an arbitrary given time t, each robot r_i is aware of its position and orientation $l_i^t = (x_i^t, y_i^t, \theta_i^t)$, where $p_i^t = (x_i^t, y_i^t)$ is the coordinate in the Cartesian space and θ_i^t is the clockwise angle to the direction of the y-axis. All the robots share a common Cartesian coordinate system.
- Each robot r_i can communicate with, *i.e.*, send and receive messages to and from, all the robots in $\mathcal{R}$.
- Each robot can move and rotate precisely in $\mathbb{R}^2$ with a maximum linear velocity v and a maximum angular velocity ω. The robots can dynamically choose a collision-free path during movement.

In addition to the capabilities of each robot, the whole system $\mathcal{R}$ forms a traditional message-passing system. We assume that $\mathcal{R}$ is asynchronous with no failure. That is to say, the procedure execution of each robot is not only completely arbitrary but independent with the ones of the other robots as well. In particular, there is no fixed upper bound on the time it takes for a message to be delivered. Also, the robots are always performing correct operations, and the communication between the robots will not fail.

Definition 1. *(Configuration) A configuration $C^t = \{p_1^t, \cdots, p_n^t\}$ is the collection of positions of $\mathcal{R}$ at time t. Here, we limit $p_i^t \neq p_j^t$ for any $i \neq j$.*

In the beginning, $\mathcal{R}$ is in an arbitrary initial configuration, and the robots in $\mathcal{R}$ orientate arbitrarily.

B. Problem Definition

Definition 2. *(Sub-configuration) A sub-configuration $S^t = \{p_{s_1}^t, \cdots, p_{s_k}^t\}$ is the collection of positions of a subset of $\mathcal{R}$, i.e., $\{r_{s_1}, \cdots, r_{s_k}\}$, at time t. Without loss of generality, we assume that $1 \leq s_1 < s_2 < \cdots < s_k \leq n$. Here, we limit $k \geq 1$ and $p_{s_i}^t \neq p_{s_j}^t$ for any $i \neq j$.*

Definition 3. *(Basic Pattern Graph, BPG) A BPG $B = \{(x_1, y_1), \cdots, (x_k, y_k)\}$ is a collection of non-overlapping points in which $x_1 = 0$, $y_1 = 0$, and $(x_i, y_i) \in \mathbb{R}^2$ for all $1 \leq i \leq k$. Here, we limit $k \geq 1$. The **center** of B, notated as $O(B)$ is defined to be center of gravity of all the points in B, i.e., $O(B) = \frac{1}{k} \sum_{i=1}^{k} (x_i, y_i)$.*

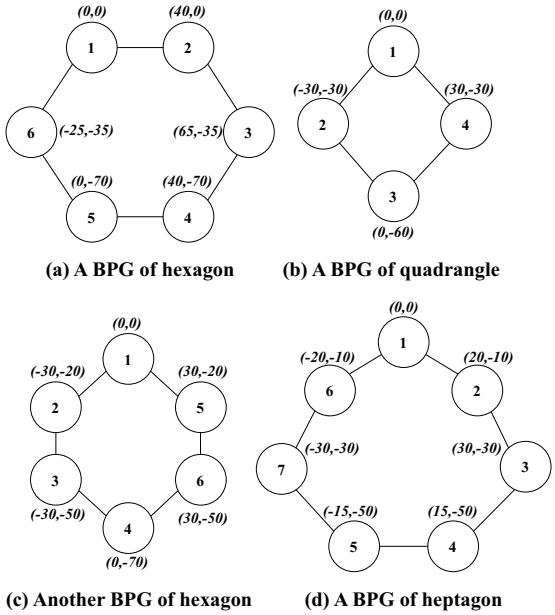


Fig. 3: Four examples of BPG

As shown in Fig. 3, BPGs can be used to represent basic geometric patterns, *e.g.*, hexagon, quadrangle, and heptagon.

Definition 4. *(Relationship between sub-configuration and BPG) Consider a sub-configuration $S^t = \{p_{s_1}^t, \cdots, p_{s_{k_1}}^t\}$*

*and a BPG $B = \{(x_1, y_1), \cdots, (x_{k_2}, y_{k_2})\}$. S^t is said to be **compatible** with B at time t if and only if there exists a vector $v \in \mathbb{R}^2$ such that $\forall 1 \leq i \leq k_1, \exists 1 \leq j \leq k_2, p^t_{s_i} + v = (x_j, y_j)$. The **center** of S^t w.r.t. B is defined to be $O(S^t, B) = O(B) - v$. If S^t is compatible with B, the **completion rate** of S^t to B, notated as $\alpha(S^t, B)$ for short, is defined to be k_1/k_2. Otherwise, we define $\alpha(S^t, B)$ to be 0.*

It is evident that the completion rate must be a real number between 0 (inclusive) and 1 (inclusive). The completion rate is used to evaluate how good a subset of robots forms a specific basic pattern. The larger the completion rate is, the better the subset of robots forms the basic pattern.

Definition 5. *(Relationship between sub-configuration and configuration) A set of sub-configurations $\{S^t_1, S^t_2, \cdots, S^t_k\}$ is said to be a **partition** of a configuration C^t at time t if and only if $S^t_1 \cup S^t_2 \cup \cdots \cup S^t_k = C^t$ and $S^t_i \cap S^t_j =$ for any $i \neq j$. We notate all the partitions of C^t as $\mathcal{T}(C^t)$.*

Definition 6. *(Basic Pattern Fulfillment Ratio) The fulfillment ratio, notated as β, of a partition $\{S^t_1, S^t_2, \cdots, S^t_k\}$ of a configuration C^t to a BPG B is defined to be the average of all the completion rates of S^t_i to B. That is:*

$$\beta(\{S^t_1, S^t_2, \cdots, S^t_k\}, B) = \frac{1}{k} \sum_{i=1}^{n} \alpha(S^t_i, B) \tag{1}$$

The basic pattern fulfillment ratio of a configuration C^t to a BPG B, notated as $\gamma(C^t, B)$ for short, is defined to be the maximum value of the fulfillment ratios of all the partitions of C^t. That is:

$$\gamma(C^t, B) = \max_{\tau \in \mathcal{T}(C^t)} \beta(\tau, B) \tag{2}$$

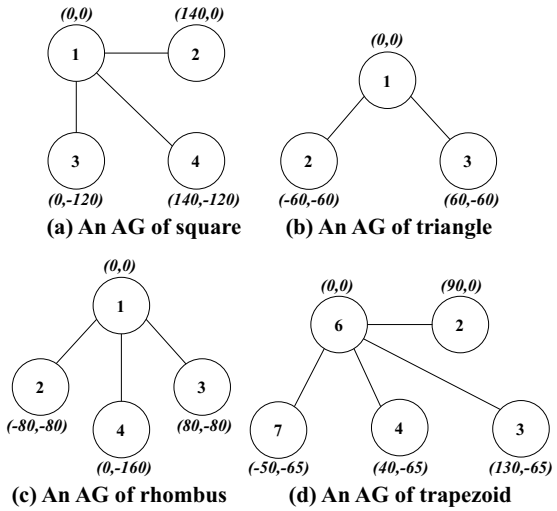
*Here, the partition τ is called the **optimal partition** of C^t w.r.t. B.*

It is evident that the basic pattern fulfillment ratio must be a real number between 0 (inclusive) and 1 (inclusive). Using Eq. 2, we can evaluate how good the whole MRS forms a specific kind of basic pattern. The larger the basic pattern fulfillment ratio is, the better the MRS forms the basic pattern. Using Def. 6, we can partition a MRS $\mathcal{R}$ into a collection of basic patterns optimally. Now, we consider the centers of the basic patterns and assemble them using assembling graph.

Definition 7. *(Assembling Graph, AG) An AG $A = \{(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)\}$ is a collection of distinct vectors, where $(x_i, y_i) \in \mathbb{R}^2$ for all $1 \leq i \leq k$.*

As shown in Fig. 4, AGs can be used to represent the assembled patterns, *e.g.*, square, triangle, rhombus and trapezoid.

Definition 8. *(Assembling Fulfillment Ratio) Consider an configuration C^t, a BPG B, and an optimal partition τ. We notate the centers of the sub-configurations in τ as $O = \{o_1, o_2, \cdots, o_k\}$. Given an AG $A = \{(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)\}$, we link o_i with o_j if and only if $o_i + (x_l, y_l) = o_j$ for some $1 \leq l \leq k$. In this way, we get a graph of O, whose number of connected components is notated as $\mathcal{C}$. The assembling fulfillment ratio, notated as δ is defined to be 1 by $\mathcal{C}$. That is:*



(a) An AG of square (b) An AG of triangle (c) An AG of rhombus (d) An AG of trapezoid

Fig. 4: Four examples of AG

$$\delta(\tau, A) = \frac{1}{\mathcal{C}} \tag{3}$$

Using Eq. 3, we can calculate the assembling fulfillment ratio, which implies the quality of the assembled formation. It is evident that the assembling fulfillment ratio is a real number between 0 (exclusive) and 1 (inclusive). Ther higher the assembling fulfillment ratio is, the better the MRS is assembled as desired. Combining the basic pattern fulfillment ratio and the assembling fulfillment ratio, we introduce the metric **formation quality** to judge how good a MRS form repeating patterns.

Definition 9. *(Formation Quality) The formation quality, notated as ϵ, is defined as the product of the basic pattern fulfillment ratio and the assembling fulfillment ratio. That is, for a configuration C^t, a BPG B, an optimal partition τ, and a AG A,*

$$\epsilon(C^t, B, \tau, A) = \gamma(C^t, B) \cdot \delta(\tau, A) \tag{4}$$

In Fig. 1, The four patterns are perfect (formation qualities are 1) *w.r.t.* BPGs in Fig. 3 and AGs in Fig. 4.

Definition 10. *(Repeating Pattern Formation Problem) Given a MRS $\mathcal{R}$, a BPG B, and an AG A, move the robots such that the formation quality of $\mathcal{R}$ is maximized.*

IV. A Decentralized Algorithm

Our algorithm is a decentralized algorithm executed on each node independently. The whole process of the algorithm is divided into two phases, namely basic pattern generation, and pattern assembling respectively. In the phase of basic pattern generation, the robots coordinate with each other to generate a set of basic patterns according to the BPG. In the pattern

assembling phase, the basic patterns are gradually assembled with each other to form the final pattern according to the AG. Next, we will describe the two phases in detail.

A. Basic pattern generation

In the phase of basic pattern generation, a group-based method is used to divide the robots into groups with a constrained number of members. The number of robots in a group equals to the number of nodes in the BGP. When the system starts to run, a robot is assigned to a new group if it has not joined a group and owns the largest ID in the set of unassigned robots. Then, the assigned robot adds several neighbors to its group. The number of selected neighbors is $|BPG| - 1$, that is the size of the BPG minus 1. Inside the group, the robots are organized into the basic pattern according to the input BGP. The robot with the largest ID is the head of this pattern. The detailed operations for each robot to generate basic patterns are shown in Alg. 1.

Algorithm 1 The basic pattern generation algorithm for each robot r_i

Input: $\mathcal{R} = \{r_1, \cdots, r_n\}$: the MRS; B: the BPG to form; $\mathcal{P} = \{p_1, \cdots, p_n\}$: the positions of the robots
Output: several basic patterns are formed according to B

$r_i.head \leftarrow False$
Upon r_i **finds that it has the largest ID in** $\mathcal{R}$
  $r_i.head \leftarrow True$
  $B' \leftarrow B \setminus \{(0,0)\}$
  $m \leftarrow |B'|$
  $B' \leftarrow B' + p_i$ ▷ each element in B' is added to a vector p_i
  Select m nearest neighbors $\mathcal{N} = \{r_{c_1}, \cdots, r_{c_m}\}$ from $\mathcal{R}$
  $\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{N} \setminus \{r_i\}$
  Compute an assignment of robots in C to positions in B' with minimum sum of distances
  Notate the assignment result as $\{p'_{c_1}, \cdots, p'_{c_m}\}$
  Send $Group(\mathcal{R}, (r_i, p_i), (r_{c_1}, p'_{c_1}), \cdots, (r_{c_m}, p'_{c_m}))$ to all its neighbors
Upon r_i **receives** $Group(\mathcal{R}', (r_{c_1}, p'_{c_1}), \cdots, (r_{c_m}, p'_{c_m}))$
  $\mathcal{N} = \{r_{c_1}, \cdots, r_{c_m}\}$
  If $r_i \in \mathcal{N}$
    Find the record (r_i, p'_i) in the $Group$ message
    Move to p'_i
    Regard r_{c_1} as its head and maintains relative position to r_{c_1}
  EndIf
  $\mathcal{R} \leftarrow \mathcal{R}'$

In Alg. 1, let $\mathcal{R}$ represents the set of available robots that are not assigned to form the basic pattern yet. When the system starts, each node has the information of the set $\mathcal{R}$. However, each node does not know whether it can be a head or not at this time, so the variable $head$ is set as $False$ in the beginning.

For each robot r_i, if it finds that it has the largest ID in $\mathcal{R}$, it sets up a new group with itself to be the head. The head is responsible for forming and maintaining the pattern. Then, it selects $|BPG| - 1$ nearest neighbors from $\mathcal{R}$ and computes a matching of these neighbors to the target positions in the pattern with the minimum sum of moving distances.

The matching process is mapped to a problem of maximum matching in a weighted bipartite graph. Inside the maximum matching problem, the positions of the selected neighbors are in one independent set of vertices, the target positions are in the other set of vertices, and the weights are distances between the selected neighbors and the target positions. The problem can be optimally solved by using Hungarian algorithm [5].

When the matching process is finished in r_i, it gets matching result $\{p'_{c_1}, \cdots, p'_{c_m}\}$, where m is group size. Then, r_i sends a message $Group(\mathcal{R}, (r_i, p_i), (r_{c_1}, p'_{c_1}), \cdots, (r_{c_m}, p'_{c_m}))$ to inform all the robots, where $r_i, r_{c_1}, \cdots, r_{c_m}$ are the assigned robots, and $p_i, p'_{c_1}, \cdots, p'_{c_m}$ are the target positions where the assigned robots should go.

When a robot r_i receives a message $Group(\mathcal{R}', (r_{c_1}, p'_{c_1}), \cdots, (r_{c_m}, p'_{c_m}))$, it checks whether its ID is contained in the message. If true, *i.e.*, a record (r_i, p'_i) is found in the message, it means that r_i has been assigned a target position by robot r_{c_1}. Therefore, r_i moves to the assigned position p'_i. After reaching the position, r_i regards r_{c_1} as its head and maintains the relative position *w.r.t.* r_{c_1}. If false, it does not perform the above operations. Whatever r_i's ID is in the message or not, it will update the set $\mathcal{R}$ as $\mathcal{R}'$ contained in the message.

When Alg. 1 terminates, *i.e.*, $A = \varnothing$ and all the robots have reached the assigned positions, several basic patterns are generated. Inside each group of a basic pattern, the non-head robots will follow the head's action to maintain the pattern. Therefore, we only need to consider the head's movement in the second phase.

B. Pattern assembling

After the first phase, there are multiple basic patterns in the working area. In the second phase, *i.e.*, the phase of pattern assembling, a layer-based mechanism is proposed to assemble the basic patterns. The basic patterns form a tree structure layer by layer according to the input AG.

Firstly, a head with the largest ID in the system is selected as the root in the formation. The root is the first layer. Secondly, the root selects a given number of nearest heads, which are not in the formation, as its children. The number is decided by the input AG. Thirdly, it computes a maximum matching for the selected heads to target positions according to the AG. Fourthly, the selected heads move to the assigned positions; then the second layer is formed. After the selected heads arrive their positions, they continue to select other heads that are not in the formation to join the tree and form a new layer of the formation. Above process continues until the final pattern is generated. A detailed description of the above operations is shown in Alg. 2.

In Alg. 2, let $\mathcal{H}$ be the set of unassigned heads in the system, and let $\mathcal{Q}$ be a queue to control the assigned heads to perform their assignment operations one by one to avoid collision, *i.e.*, more than one heads are assigned to the same

Algorithm 2 The assembling algorithm for each robot r_i

Input: $\mathcal{H} = \{h_1, \cdots, h_m\}$: the head of the basic patterns; $R = \max_{i=1}^{m} \mathcal{H}$: the maximum robot ID in $\mathcal{H}$; $\mathcal{P} = \{p_1, \cdots, p_m\}$: the positions of the heads; A: the input AG

Output: The repeating pattern is formed

$\mathcal{Q} \leftarrow \{R\}$ ▷ $\mathcal{Q}$ is as a queue of assigned leaf robots

$\mathcal{O} \leftarrow \varnothing$ ▷ $\mathcal{O}$ is a set of assigned heads

Upon r_i **finds that it is the first element in** $\mathcal{Q}$

$A' = A \backslash \{(0,0)\} \backslash \mathcal{O}$

$l \leftarrow |A'|$

$A' \leftarrow A' + p_i$ ▷ each element in A' is added to a vector p'

Select l nearest neighbors $E = \{r_{c_1}, \cdots, r_{c_l}\}$ from $\mathcal{H}$

Compute an optimal matching between E and A' with minimum sum of distances

Notate the matching result as $\{p'_{c_1}, \cdots, p'_{c_l}\}$

Sort the robots in E according to the node IDs in ascending order

$\mathcal{H} = \mathcal{H} \backslash E \backslash \{r_i\}$

$\mathcal{Q} = \mathcal{Q} \cup E \cup \{r_i\}$

Send $Layer(\mathcal{H}, \mathcal{Q}, (r_i, p_i), (r_{c_1}, p'_{c_1}), \cdots, p'_{c_l})$ to all its neighbors

Upon r_i **receives a message** $Layer(\mathcal{H}', \mathcal{Q}', (r_{c_1}, p'_{c_1}), \cdots, (r_{c_l}, p'_{c_l}))$

$\mathcal{N} \leftarrow \{r_{c_1}, \cdots, r_{c_l}\}$

If $r_i \in \mathcal{N}$

Find the record (r_i, p'_i) in the message

Move to p'_i

EndIf

$\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{N}$

$\mathcal{H} \leftarrow \mathcal{H}'$

$\mathcal{Q} \leftarrow \mathcal{Q}'$

position. Here, the unassigned heads are the heads that have not assigned target positions in the desired formation and vice versa. At first, $\mathcal{H}$ contains all the heads in the system, and $\mathcal{Q}$ just contains the robot with the largest ID in $\mathcal{H}$. Each robot has the initial information of $\mathcal{H}$ and $\mathcal{Q}$.

If a robot r_i finds that it is the first element in $\mathcal{Q}$, it takes its role as the first node in the input AG. Then, it checks which target positions in the AG are not occupied by other robots. In our algorithm, an array $\mathcal{O}$ is used to record the positions have been occupied. If a target position is not occupied, it needs to be assigned a robot. Next, r_i computes an assignment (maximum matching) of neighboring unassigned heads to the unoccupied target positions.

After the assignment is finished, these neighboring heads become assigned heads, so they are removed from H. Moreover, they are added to $\mathcal{Q}$ to wait for generating the next layer of the formation. Then, r_i sends a $Layer$ message contains the assignment result and the current values of $\mathcal{H}$ and $\mathcal{Q}$ to other robots in the system.

When a robot r_j receives the $Layer$ message, it firstly checks if its ID is contained in the message. Then, it moves to the target position according to the new coordinates. If the result of the checking is false, it does not perform above operations. Whatever a robot r_j's ID is contained in the $Layer$ message or not, the $\mathcal{H}$ and $\mathcal{Q}$ values are updated by using the new values of $\mathcal{H}$ and $\mathcal{Q}$ contained in the $Layer$ message.

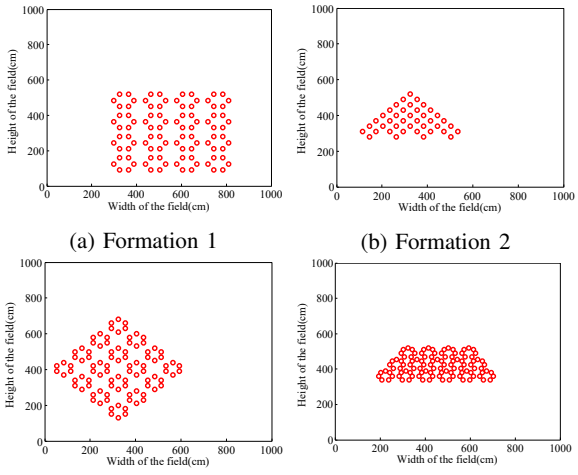
V. Simulation and Real-world Deployment

To evaluate the performance of our algorithm, we conduct sufficient experiments in this section. In Sec.V-A, we compare our algorithm with the existing works in a Matlab simulator. In Sec.V-B, we deploy our algorithm in a real-world testbed in our laboratory. The experimental results have indicated the effectiveness and practicability of our algorithm.

A. Simulation

We use a simulator implemented by Matlab to evaluate our algorithm and existing works. We compare our algorithm with Song's algorithm [16], since it is the only approach has the same feature with our algorithm (*i.e.*, it is also a decentralized and general approach for generating formation with arbitrary repeating pattern). The four types of formations shown in Fig. 1 are selected as desired formations. For each desired formation F_d, $|F_d|$ robots are placed randomly and uniformly in the working plane. The robots can move without colliding with other robots in a constant speed of 10 cm/s. Two metrics, namely formation quality and total movement distance are selected to evaluate the algorithms.

To provide a visualized comparison, we firstly execute our algorithm and Song's algorithm to generate the four formations in Fig. 1 respectively. The working area of the system is set as a square of $10m \times 10m$. The formations generated by our algorithm are shown in Fig. 5, and the formations produced by Song's algorithm are shown in Fig. 6.



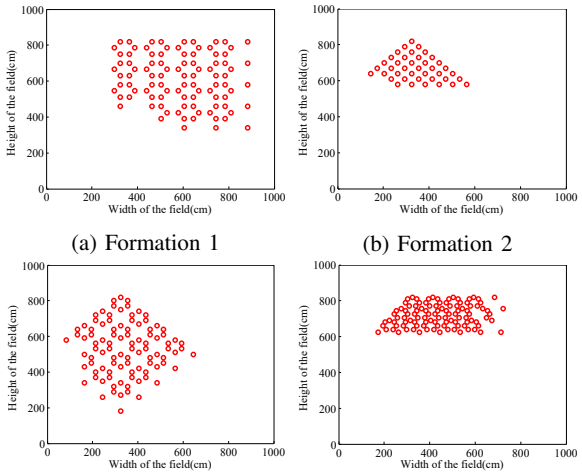
(a) Formation 1 (b) Formation 2

(c) Formation 3 (d) Formation 4

Fig. 5: Formations generated by our algorithm

We can see that our algorithm can generate the four formations accurately, but Song's algorithm cannot achieve the desired formations. That is because Song's algorithm uses a

lattice graph to represent robot's relative positions in each formation, but the lattice graph cannot reflect the shape of the formation. In our algorithm, we use not only a basic pattern graph to represent robots' relative positions in a pattern, but also an assembling graph to describe how can the patterns be assembled to generate the desired formation. Based on the two graphs, our algorithm firstly organizes the robots into basic patterns and then organizes the patterns to assemble with each other to generate the desired formation. Therefore, our algorithm can guarantee the shape of the generated formations.



(a) Formation 1 (b) Formation 2

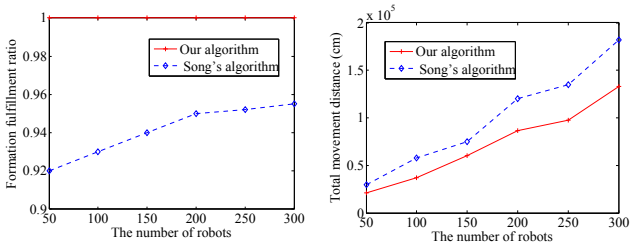
(c) Formation 3 (d) Formation 4

Fig. 6: Formations generated by Song's algorithm

Then, we formally evaluate the average performances of the algorithms. We perform the two algorithms to generate the four types of formations shown in Fig. 1 in different scales of MRS. To accommodate more robots, we extend the working area of the system to a square of $15m \times 15m$. We set the number of robots ranging from 50 to 300 with steps of 50. Then, the two algorithms are executed 20 time respectively, and the average values of the performances are calculated.

We first test the average formation qualities of the two algorithms. To make the generated formation and the desired formation comparable, we select the leftmost point in the first line of each formation as the origin of a local coordinate system. All the other points' coordinates can be computed by using the local coordinate system. By this way, the coordinate systems of the two formations become the same. The formation quality can be obtained by calculating the percentage of points that locate in both the generated formation and the desired formation. The comparison result is shown in Fig. 7a. We can see that our algorithm can achieve the formation quality of 1, no matter how many the number of robots in the systems. However, the Song's algorithm can only achieve the formation quality of 0.92 to 0.96 in the systems. Therefore, our algorithm has a better formation quality than Song's algorithm.

The total movement distances of the two algorithms are shown in Fig. 7b. We can see that our algorithm has shorter total movement distances than that of the Song's algorithm. In



(a) Comparison of formation quality (b) Comparison of total movement distance

Fig. 7: Comparison between our algorithm and existing work

a system with 50 robots, our algorithm can achieve an average total movement distance of around $212cm$, but the total movement distance of Song's algorithm is around $299m$. In this situation, our algorithm's total movement distance is 28.8% lower than that of the Song's algorithm. In a system with 300 robots, our algorithm can achieve an average total movement distance of around $1334m$, but the total movement distance of Song's algorithm is around $1820m$. In this situation, our algorithm's total movement distance is 26.7% lower than that of the Song's algorithm. Therefore, in different scale systems, our algorithm achieves shorter total movement distance than Song's algorithm.

B. Real-world Deployment

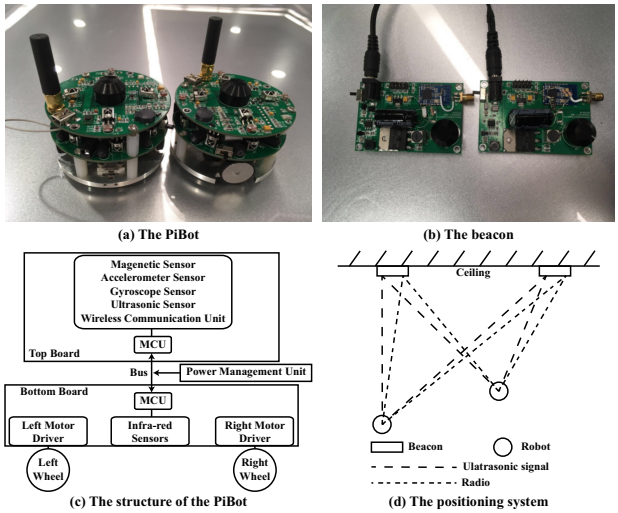


Fig. 8: Experiment Setup

To demonstrate the practicability of the proposed algorithm in Sec. IV, we deploy a realistic $2m \times 3m$ testbed with ten intelligent robots. The robots as shown in Fig. 8(a), namely PiBots (The Hong Kong Polytechnic University Intelligent Robot) [17], are designed in our laboratory. All the robots are in the same shape of a cylinder with radius $7cm$ and height $18cm$. The structure of a PiBot is illustrated in Fig. 8(c). The whole robot is composed of four parts. From the bottom to

the top, the four components are the wheels, the bottom board, the power management unit, and the top board respectively.

The wheels are linked with the motor drivers on the bottom board. By linkage, the motors can get and set the rotating speeds of the wheels. There are also eight infra-red sensors around the bottom board, which are responsible for detecting the surrounding objects. The data from the motors and the infra-red sensors are stored and processed in the MCU (micro-controller). The bottom board are connected with the top board using an SPI (Serial Peripheral Interface) bus to achieve data transmission. On the top board, there are plentiful sensors such as an ultrasonic receiver, a 3-axis accelerator sensor, a 3-axis gyroscope, and a 3-axis geomagnetic sensor. Besides the sensors, there is a 2.4G wireless communication unit. The wireless communication unit uses IEEE 802.15.4 as its protocol to enable inter-robot communication. There is also a MCU to store and process data on the top board. The MCUs on the top and bottom board are of STM32 family with 8M static random access memory (SRAM) or 512K flash memory.

Equipped with the sensors, each robot can localize itself on the testbed with assist of two beacons as shown in Fig. 8(b). The localization mechanism is illustrated in Fig. 8(d). Each beacon is composed of a 2.4 G wireless communication module, an STM32 MCU, an ultrasonic transmitter, a temperature sensor, and a battery. Every one tenth second, the primary beacon will send out a packet of radio signal via the wireless communication module and a packet of ultrasonic signal via the ultrasonic transmitter. When the secondary beacon receives a radio signal from the primary beacon, it will send out a radio signal and a ultrasonic signal with the same packet number. Due to the different propagation speeds of the radio signal and the ultrasonic signal, the robot will receive them at different time. Specifically, each robot can calculate the distance between itself and the two beacons using the principle of TDoA (Time Difference of Arrival). After calculating the two distances, the robots can localize themselves on a half plane. Since the two beacons are on the same side of the testbed, each robot can acquire the only possible location on the testbed. Moreover, we consider the different ultrasonic velocities under different temperatures with a temperature sensor to achieve a more precise positioning system.

The programming environment is based on FreeRTOS, a popular real-time operating system kernel for embedded devices. In FreeRTOS, a set of tasks (similar to threads in operating systems) are defined with priorities to be executed concurrently. This kind of task-based operating system fits in distributed system naturally [18]. For each *Upon* statement in the algorithm, we create a corresponding task. We testify three repeating patterns on our test-bed, namely a square formation of repeating dots, a triangle formation of repeating segments, and a rhombus formation of repeating triangles. A set of running result is demonstrated in Fig. 2.

VI. Conclusion

In this paper, we studied the problem of repeating pattern formation in MRS. Existing works have some drawbacks such as lacking generalization or cannot guarantee the shapes of the generated formations. We proposed a decentralized algorithm for generating arbitrary repeating patterns. Some key concepts are introduced to define repeating patterns and the formation quality for measurement. We implement our algorithm in a Matlab simulator and a real-world testbed. The simulation result shows that our algorithm outperforms the existing works. The real-world deployment indicates the practicability of our approach.

Acknowledgments

This research is supported by Shenzhen STIC Basic Research Funding Scheme with Ref. No. JCYJ20170818104222072 and RGC GRF with Ref. No. PolyU 152133/18.

References

[1] L. Bayındır, "A review of swarm robotics tasks," *Elsevier Neurocomputing*, vol. 172, pp. 292–321, 2016.

[2] P. Flocchini, G. Prencipe, and N. Santoro, "Distributed computing by oblivious mobile robots," *Morgan & Claypool Synthesis lectures on distributed computing theory*, vol. 3, no. 2, pp. 1–185, 2012.

[3] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM J. Comput.*, vol. 28, no. 4, pp. 1347–1363, 1999.

[4] S. Jiang, J. Cao, J. Wang, M. Stojmenovic, and J. Bourgeois, "Uniform circle formation by asynchronous robots: A fully-distributed approach," in *IEEE ICCCN*, 2017, pp. 1–9.

[5] H. W. Kuhn, "The hungarian method for the assignment problem," *Wiley Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[6] F. Arrichiello, S. Chiaverini, and T. I. Fossen, "Formation control of underactuated surface vessels using the null-space-based behavioral control," in *IEEE/RSJ IROS*, 2006, pp. 5942–5947.

[7] D. Panagou and V. Kumar, "Cooperative visibility maintenance for leader–follower formations in obstacle environments," *IEEE Trans. Robotics*, vol. 30, no. 4, pp. 831–844, 2014.

[8] A. Loria, J. Dasdemir, and N. Jarquin-Alvarez, "Decentralized formation-tracking control of autonomous vehicles on straight paths," in *IEEE CDC*, 2014, pp. 5399–5404.

[9] A. Benzerrouk, L. Adouane, L. Lequievre, and P. Martinet, "Navigation of multi-robot formation in unstructured environment using dynamical virtual structures," in *IEEE/RSJ IROS*, 2010, pp. 5589–5594.

[10] A. Ailon and I. Zohar, "Control strategies for driving a group of nonholonomic kinematic mobile robots in formation along a time-parameterized path," *IEEE/ASME Trans. Mechatronics*, vol. 17, no. 2, pp. 326–336, 2012.

[11] L. Wachter, J. Murphy, and L. Ray, "Potential function control for multiple high-speed nonholonomic robots," in *IEEE ICRA*, 2008, pp. 1781–1782.

[12] C. Nam and D. A. Shell, "Assignment algorithms for modeling resource contention and interference in multi-robot task-allocation," in *IEEE ICRA*, 2014, pp. 2158–2163.

[13] T. Mercker, D. W. Casbeer, P. T. Millet, and M. R. Akella, "An extension of consensus-based auction algorithms for decentralized, time-constrained task assignment," in *IEEE ACC*, 2010, pp. 6324–6329.

[14] S. Ahmad, Z. Feng, and G. Hu, "Multi-robot formation control using distributed null space behavioral approach," in *IEEE ICRA*, 2014, pp. 3607–3612.

[15] Y. Song and J. M. O'Kane, "Decentralized formation of arbitrary multi-robot lattices," in *IEEE ICRA*, 2014, pp. 1118–1125.

[16] ——, "Forming repeating patterns of mobile robots: A provably correct decentralized algorithm," in *IEEE/RSJ IROS*, 2016, pp. 5737–5744.

[17] S. Jiang, J. Liang, J. Cao, and R. Liu, "An ensemble-level programming model with real-time support for multi-robot systems," in *IEEE Percom Workshops*, 2016, pp. 1–3.

[18] S. Jiang, J. Cao, Y. Liu, J. Chen, and X. Liu, "Programming large-scale multi-robot system with timing constraints," in *IEEE ICCCN*, 2016, pp. 1–9.