# Self-Stabilizing Structure Forming Algorithms for Distributed Multi-Robot Systems

Yansheng Zhang, Farokh Bastani, I-Ling Yen
University of Texas at Dallas
{yxz037000, bastani, ilyen}@utdallas.edu

**Abstact** Object transportation is an important emerging application of multi-robotic swarm systems. It requires a large number of robots to be dynamically coordinated in real-time to form structures around any given rigid body that is to be lifted and transported. This paper systematically investigates the major issues that need to be addressed in methods of dynamically forming robust transportation structures. Two self-stabilizing algorithms are developed to enable a swarm of robots to form a structure to handle object transportation. Even with only a limited localized view of the environment, the algorithm enables individual robots to cooperatively form a safe structure. The stability and fault-tolerance properties of the algorithms are formally proven. The performance of the self-stabilizing algorithms, in terms of their efficiency of convergence, is evaluated via experimental studies and the results show that the system can achieve the goals in real-time.

**Keywords**: Swarm robotic systems, self-organizing, self-stabilizing systems.

## 1. Introduction

A robotic swarm system is a collective multi-agent system composed of multiple autonomous robots, interacting and cooperating with each other to accomplish some specified task. In recent years, a lot of research efforts have been devoted to the development of swarm system models and techniques [4]. Many application systems, such as large-scale unmanned space exploration, underwater missions [2], etc. can potentially take advantage of swarm system techniques. Compared to the traditional individual robot control approach, the swarm robot system has better potential in achieving more complicated tasks and can better adapt to failures or unexpected situations.

One major direction in swarm systems investigates the transportation of large objects, which is frequently required in many robotic applications. For example, in space exploration task, thousands of mobile robots may be launched into asteroid belt to collect data for further analysis. Some interested big object need to be transported by multiple robots back to the spaceship. Another example is pathway clearing. There may be some large trash that can only be removed by group of robots through tight cooperation. To build such a system, we usually divide the whole task into smaller subtasks. The first task is the coalition formation [3] which requires multiple robots form a group to accomplish a common goal. Specifically, the group of robots need to form a suitable structure [5] with which the weight of the load can be distributed evenly on the robots and the balance of the rigid body can be easily maintained in spite of some disturbance during the transportation. Thus the structure should be flexible such that it can be easily adjusted under different situations during the mission.

Self-organizing or self-assembling is the fundamental technique in swarm robotic systems for achieving structure formation. Assume that a group of robots are initially located randomly in an environment. They should automatically assemble to form desired structures to facilitate object movement. The structure should be adjusted dynamically and autonomously during the execution of the transportation tasks. The coordination should be implicit, i.e., the behavior of an individual robot is based only on its local information and interactions with neighboring robots while achieving the desired behavior of the system. Moreover, individual robots should be allowed to dynamically join or leave the group without significantly affecting the emergent system behavior. Thus, self-organizing approach achieves better reliability, extensibility, robustness, and fault tolerance.

Many self-organizing techniques have been developed in the literature. One major approach is to apply artificial evolution techniques to synthesize the individual robot controller so as to achieve a collective behavior of the system. In [5], a group of mobile robots are trained to approach a prey, self-assemble into structures and pull or push the prey towards a target location. The system is capable of coping with different structures and shapes of the prey. But it takes a long time to evolve a good controller. Moreover, the neural network controller may result in multiple structures given the same initial prey shape and robot's position. In some of these structures, all the robots aggregate at one side of the rigid body, which make the robot improperly hold and support the body. Another approach is based on the concept of force field [6]. The attractive potential and repulsive potential are used to manipulate the robots to move toward the target and/or to distribute evenly around a rigid body. Though the system can form flexible structures, every robot is required to get the other robots' position information periodically to calculate the potential. This is not practical since most of the robot being used in the swarm system only have local and limited sensing and communication abilities. Moreover, it takes a long time for these algorithms to converge.

In this paper, we design self-stabilization algorithms [1] for a swarm of robots to form a structure to handle object transportation. We focus on efficient convergence of the algorithm such that the system can fulfill the specified tasks in real-time. Also, the constraint on the number of robots is carefully studied to ensure a safe structure that helps the rigid body keep balance in transportation. Moreover, in the process of transportation, the self-stabilization algorithm supports autonomous structure adaptation to cope with failures or environment changes. Hence, the system is robust and can tolerate failures of individual robots.

The rest of the paper is organized as follows: In Section 2, we present our model for individual robots as well as the problem specification. Section 3 presents two algorithms for the lower level planners to cooperatively form a safe structure. Section 4 describes simulation result and Section 5 summarizes the paper and outlines some future areas of research.

## 2. System Model and Problem Specification

Here we discuss the system model for object transformation by a swarm of robots. A detailed description about the robot capabilities is discussed in Section 2.1. Section

2.2 defines some basic definition about the rigid body and system configuration. Problem specification and the constraint on the formed structure are discussed in Section 2.3.

## 2.1. Mobile Robot Model

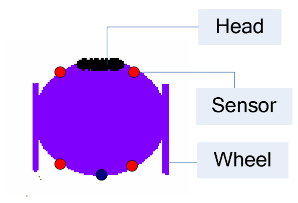


**Fig. 1.** Structure of Each Individual Robot.

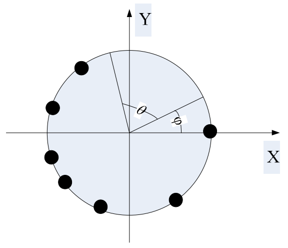


**Fig. 2.** An Example of Unsafe State.

We consider mobile robots with sensors and actuators and wireless communication devices (as shown in Fig. 1). Initially, we assume that *n* robots form a coalition as one group. Let $r_i$ denote the *i*'th robot. To transport a rigid body, the robot should have the following capabilities and properties.

1) The robot has two wheels and their relative positions to the center of the robot are fixed. Each wheel can move independently at the user specified speed. Positive speed yields forward move and negative speed yields backward move. Rotation can be achieved by assigning different speed to two wheels.
2) The robot maximum linear speed is denoted by $\varpi$.
3) Each robot has a set of sonar and color sensors, which can detect obstacles and other robots around it. Furthermore, the robot can determine the distance of an object and can differentiate an obstacle from other robots in the group.
4) Each robot is equipped with a GPS device which allows it to obtain its current position.
5) Each robot is equipped with a gripper that allows the robot to grasp and lift the rigid body at all angles. The gripper can be used to enforce a fine-grained coordination to a certain extent. Slightly uncoordinated behaviors from some individual robots due to physical discrepancies will be forcefully adjusted to the desired behaviors by the force feedbacks from the grippers. Thus, we assume that during motion, the relative positions of the robots are maintained.

## 2.2. Basic Definitions

Let *O* denote the object to be transported. Assume that *O* is a convex polygon and let graph $G = (V, E)$ denote the polygon, where *V* is the set of vertices and *E* is the set of edges. Let *gc* denote the center of gravity and *rad* the radius of *O*, i.e. $rad = \text{Max}\ \{dist\ (gc, v_i) \mid \text{for all } v_i \text{ in } V\}$, where *dist* is the distance between two points.

As discussed earlier, there are *n* mobile robots in the system. For simplicity, each robot is considered as a dot. Let $p_i$ represent robot $r_i$'s angular position relative to the center of gravity. The system state or configuration can be defined by vector $cf = [p_1,$

$p_2$, …, $p_n$]. Configuration vector $cf$ is time variant, so sometimes we use $cf(t)$ to denote the system state at time $t$.

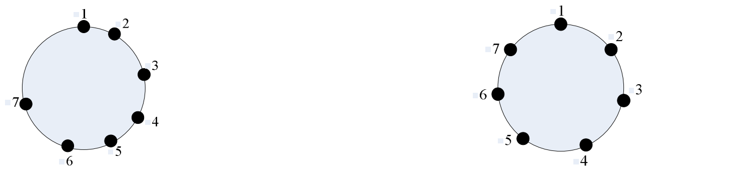
## 2.3. Problem Specification and Analysis

We consider the problem of $n$ robots, $r_1… r_n$, distributing themselves around the rigid body $O$ to support the transportation of $O$ in a balanced manner. The most basic requirement to maintain the balance is: when we arbitrarily draw a line that crosses the center of gravity ($gc$) of the rigid body, there must exist some robots on either side of the rigid body or simply on the line. To allow strengthening the balance requirement and make it flexible, we require that for a given $\theta$, when we arbitrarily draw an angle of size $\theta$ with the tip on $gc$, there always exists at least one robot within the angle. Here $\theta$ can be specified by the system designer to control how evenly the robots should distribute. In the following we give a formal definition for the safe state based on $\theta$.

**Definition 1.** Safe State ***S***: $\forall\varphi$, there always exists a robot that is within the angle range $[\varphi, \varphi+\theta]$ of the rigid body. □

When $\theta = \pi$, the safe state constraint becomes the most basic requirement. Also, $\theta$ should always be less than or equal to $\pi$. In . Fig. **2**, we give a counter example to illustrate an unsafe state. In this configuration, there exists an angular range $[\varphi, \varphi+\theta]$ that does not have any robot.

Generally, the system is expected to not only converge to a safe state, but also remain in the safe state in spite of some individual robot failures so that the group of robots can keep transporting the rigid body without stopping for structure reformation. This is especially important for time-critical rigid body transportation tasks. Thus, it is necessary to consider additional system requirements beyond the safe state requirement. Consider a rigid body that requires at least 3 robots to safely transport it (due to its weight). Thus, the safety requirement have $\theta = 2\pi/3$. Assume that the system actually has 7 robots which form a structure as shown in Fig. 3(a). Note that the system satisfies the safe state constraint. However, if robot 7 fails, the system will no longer be in a safe state since the angle between robot 1 and robot 6 is greater than $2\pi/3$. But if the robots form a structure as shown in Fig. 3(b), then even if one robot fails, the system still stays in the safe state.



(a). Distribution That Cannot Tolerate Failure. (b) Distribution That Can Tolerate Failure.

**Fig. 3.** Two Examples of 7 Robots Distributing around a Circle.

The above analysis leads to a general model of the desired system design. The system is expected to converge to a goal state which always implies the safe state. When failure occurs, the system may no longer be in the goal states, but should remain in the safe states (for example in Fig. 4, the system is driven from state 3 to state 4).

Since the system continuously adjusts its state, it will automatically recover and transfer back to a goal state (state 4 back to state 3). Continuous failures may finally bring the system from a safe state to an unsafe state (for example in Fig. 4, the system is driven from state 1 to state 2), but the desired system design is to let this happen with a very small probability.

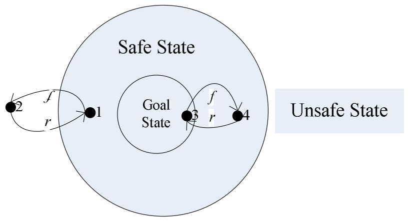


**Fig. 4.** Desired State and Safe State (f=fault, r=recovery).

Mapping the general system design model to our object transportation problem, the goal state is to have robots evenly distributed around the rigid body. When a robot fails, the remaining robots still form a structure that satisfies the safe state constraint. The robots will quickly recover to the goal state and during the recovery process, the system is always in the safe state. In a rare case when the system, after multiple robot failures or do not have a sufficient number of robots, may go into an unsafe state. In this case, if the system can still recover to a safe state, the robot will have to release the rigid body and reform a safe structure and, then, continue the transportation task.

Here we need to define the goal state of the system. The ideal goal state is to have robots evenly distributed around the rigid body. Specifically, the angular distance between any two adjacent robots, denoted by $\alpha$, should have $\alpha = 2\pi/n$ ($n$ is the total number of robots). But this will not be possible in physical systems. Thus, we need to define a tolerance bound without sacrificing safety even under robot failures.

**Definition 2.** **Goal state *G*: T**he angular distance between any two adjacent robots, $\alpha$, should satisfy $\alpha \leq 2\pi/n + \delta$. □

Here $\delta$ is the tolerance bound and it is the control parameter of the goal state. Smaller $\delta$ indicates better robot distribution (more even) structure and potentially better system fault tolerance. In the case of the safe state, we have $\alpha = \theta$.

The problem we consider is that $n$ robots, $r_1 \ldots r_n$, should distribute themselves around the rigid body $O$ such that the goal state is satisfied. When failure occurs, the system should still satisfy the safe state constraint if there is a sufficient number of working robots. When a robot is in motion, in case it cannot coordinate properly with other robots due to partial failures or some obstacles, we assume that the robot fails. The continuous execution of the algorithm should guarantee that the system converges to the goal state after failures.

The system, either starting from an initial state, or being in an intermediate state after failure, should converge to a goal state. This stability requirement is given in the following.

**Definition 3.** **Stability property:** If there exists a time $t$, such that $cf(t) \notin G$, then, there exists a time $t' < t + T$, such that $cf(t') \in G$. □

The stability property requires that, from any state, the system will converge to the goal state *G* in a bounded time *T*.

## 3. Self-Stabilizing Structure Forming Algorithms

In a self-stabilization system, the system may start from an arbitrary state and will converge to a goal state within a bounded time. Self-stabilization concept is very suitable for the design of swarm systems so that the system can converge from an arbitrary state (potentially an unsafe state due to the initial condition or after failures) to a goal state. We apply the self-stabilization principle to the design of the rigid body transportation algorithm for a swarm of robots. In Subsection 3.1, we first try to unify the problem of considering different rigid body shapes by mapping any convex polygons to a circle. Two algorithms for the rigid body transportation problem are presented in Subsections 3.2 and 3.3. For each algorithm, the stability property is analyzed.

### 3.1. Shape mapping

As defined in Section 2.2, we consider a group of robots, $r_1$, …, $r_n$, transporting a rigid body, *O*, that has a convex polygon shape. For structure formation, we first map *O* to a circle, then compute the robot movements on the circle, and finally map the movements to polygon positions.

Let *C*=(*gc*, *rad*) denote the circle that encloses the polygon *O* in which *gc* is the center and *rad* is the radius of *C*. We project a robot $r_i$ next to *O* to the circle *C* (with a light source on *gc*) and call the projection of $r_i$ the shadow robot $sr_i$. Fig. **5** gives an example to illustrate the mapping of a polygon to a circle and mapping of robots next to the polygon to shadow robots (or vise versa).

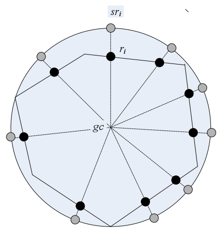


**Fig. 5.** Rigid body and robot mapping.

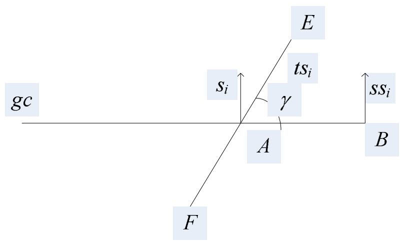


**Fig. 6.** Robot speed mapping.

We use angular coordinates to represent robot positions and movement speeds. Note that the definition of the safe state and goal state are also based on the angular coordinates. As can be seen, $sr_i$ and $r_i$ has the same angular position. Thus, a safe shadow structure implies that the corresponding real structure is safe. So, we only need to consider the movement of the shadow robot for the structure formation problem. Assume that a shadow robot only has two types of simple movements, move around *C* clockwise or anticlockwise. Also assume that clockwise speed is positive and anticlockwise speed is negative. Here we discuss how to map the shadow robot angular speed back to the real robot translational speed.

Without loss of generality, consider that a real robot $r_i$ at position $A$ and $r_i$ is moving along $EF$, the edge of the rigid body (as shown in Figure 6). Let $\gamma$ denote the angle of the intersection between rigid body edge $E$-$F$ with the circle radius $gc$-$B$. Let $s_i$ denote the angular speed of $r_i$ and $ss_i$ denote $sr_i$. Note that the real robot and the shadow should have the same angular speed, i.e. $s_i{=}ss_i$. Also, let $ts_i$ denotes the translational speed of $r_i$ along edge $EF$. We have: $ts_i \times sin(\gamma) = s_i \times dist(gc, A)$. Thus, the speed of the real robot is $ts_i = s_i /(rad\times sin(\gamma))$. For convenience, we only consider $s_i$ in the algorithms and $ts_i$ can be computed accordingly. Similarly, the maximum translational speed of the robots, $\varpi$ defined in Section 2 can be transformed into maximum rotational speed $\tau$, where $\tau = \varpi/rad$.

## 3.2. First algorithm

The simplest strategy for the robots to distribute evenly around the rigid body is to let each robot move to the relative center of its two neighbors. Assume that robot $r_i$ is the current robot. After determining the $LR$ and $RR$ angles, the robot will rotate anticlockwise to reduce the angular difference between $LR$ and $RR$. However, it is well known that this strategy can result in oscillation. Thus, it is necessary to control the speed of the robots in order to assure system convergence. Also, we consider an ideal physical system here and make the following assumptions to ensure the steady convergence to a goal state.

1) We assume that sensors of the robots are accurate and sensitive. It can provide the individual robots with the global information, including all the individual robot positions and the exact shape and position of the rigid body.
2) The planning process of each robot is assumed to be instantaneous.
3) The actuator of each mobile robot is accurate. It can move to any location precisely as the planner specifies. Moreover, speed acceleration and deceleration of each robot is instantaneous.

Assume that the robots are numbered sequentially based on their positions (in clockwise order), i.e., the immediate neighbors to the right and left of robot $r_i$ are $r_{i-1}$ and $r_{i+1}$, respectively. Let $\xi_i$ represent the angular interval between $r_i$ and $r_{i+1}$. $\xi_i$ can also be viewed as the angle formed by $r_i$, $gc$, and $r_{i+1}$ and $\xi_i = p_{i+1}-p_i$. $\xi_n$ is the angular interval between $r_n$ and $r_1$. Robot $r_i$ and $r_{i+1}$ are considered as the boundary robot of interval $\xi_i$. For robot $r_i$, $\xi_i$ and $\xi_{i-1}$ are the left and right angular intervals of $r_i$. Given any two robot $r_i$ and $r_j$, $i > j$, $r_j$ is considered as the $(i-j)$th right neighbor of $r_i$, denoted by $LN_i(i-j)$. Similarly $RN_j(i-j)$ denotes that robot $r_i$ is the $(i-j)$th left neighbor of $r_j$. Specially, $LN_i(0)=RN_i(0)=r_i$. Each robot has a local state. Let $state_i$ denote the state of robot $r_i$. $state_i$ can be either UnEven ($UE$), when the robot's left and right angular intervals are different, or LocalEven ($LE$), when the intervals are equal. If all the robots are in the local even state, then the system is in a global even state. The self-stabilizing structure forming algorithm for the robots is given as follows:

**Algorithm 1**

```
Robot $r_i$:
    $state_i = UE$;
    Periodically execute the following command;
```

```
obtain $\xi_{i-1}$ and $\xi_i$ from the sensors;
if ($\xi_i > \xi_{i-1}$) move the robot clockwise at speed $\tau$;
else if ($\xi_i < \xi_{i-1}$) move the robot anticlockwise at speed $-\tau$;
else //$\xi_i = \xi_{i-1}$
{ $state_i = LE$;
  $minL$ = smallest $k$ such that $LN_i(k)$'state is $UE$;      //if no robot in $UE$, set to −1
  $minR$ = smallest $k$ such that $RN_i(k)$'state is $UE$;     //if no robot in $UE$, set to −1
  $s_L \leftarrow$ speed of $LN_i(minL)$;       // set to 0 if $minL$ equals −1
  $s_R \leftarrow$ speed of $RN_i(minR)$;    // set to 0 if $minL$ equals −1
  $s_i = (minL \times s_R + minR \times s_L) / (minL + minR)$;
}
```

**Theorem 1.** Given any initial state, Algorithm 1 can stabilize within time $\pi/\tau$.
**Theorem 2.** The system can reach a safe state if $\theta \geq \theta_{i,}$, $1 \leq i \leq n$, that is $n \geq 2 \times \pi/\theta$.
**Theorem 3.** The system can reach a goal state.

### 3.3. Second Algorithm

In Algorithm 1, idealistic assumptions were made. In practice, robots generally do not have powerful sensors that can provide global system information. Also, due to physical constraints, it takes some time for the robots to execute the planned command. For example, it takes some time for a robot to accelerate to the computed speed. To address these limitations, we revise some assumptions in Algorithm 1 and design the second algorithm. The assumptions for the second algorithm are as follows.

1) Each robot maintains a clock, which initially is synchronized and accurate throughout the self-stabilization algorithm.
2) The planning task takes a fixed number of clock ticks and the clock tick is an atomic timing unit, denoted by $\lambda$. Planning task a periodical task. Different robots may have different periods and their initial phase may vary.
3) We consider that sensor reading and planning takes almost no time compared with command execution.
4) Each robot is going through a loop: first detecting the environmental changes, planning, and adjusting the robot position. Let $\Delta t$ denotes the maximum time interval between robot sensor detecting and robot actuator actuating. Let $\psi$ denote the angle difference threshold and its value satisfies the following constraint: $\psi > 4 \times \tau \times \Delta t$. This assumption is based on empirical data from analysis of the real system. From our empirical experience with multi-robot systems, sensor reading and planning consume negligible time compared with executing a command since command execution phase involves adjusting the real actuator speed and keep moving for a time period. Thus, we assume that reading sensors and the planning phase start at the beginning of each round and their execution time is zero.

Oscillation may occur due to the local view of the environment. Thus we consider the angle threshold in the process of determining the robot execution command. The robot's state transition is triggered only when the environmental variation has exceeded some threshold.

The self-stabilizing structure forming algorithm for the robots is given as follows:

**Algorithm 2**

```
Robot $r_i$:
    Periodically execute the following command.
    obtain $\xi_{i-1}$ and $\xi_i$ from the sensors;
    if ($\xi_i > \xi_{i-1} + \psi$) move the robot clockwise at speed $\tau$;
    else if ($\xi_i < \xi_{i-1} + \psi$) move the robot anticlockwise at speed $-\tau$;
    else robot stops.
    if (robot is moving) Wait($\Delta t$)
    else  Wait($\lambda$).
```

From Algorithm 2, it's necessary for the planner to wait $\Delta t$ time for the actuator to adjust its speed and move before next round of planning.

**Lemma 1** For any robot $r_i$, if $\xi_i(t) > \xi_{i+1}(t) + \psi$, then for any $t_0$, $t_0 < \Delta t$, $\xi_i(t+t_0) > \xi_{i+1}(t_0)$

**Proof**. To prove this, we only need to show that the difference between a moving robot's left angle and right angle will not change in one round once this robot plans to move. Assume that at the start of round $k$, robot $r_i$ does the planning and moves in the clockwise direction. This means: $\xi_i(k) > \xi_{i+1}(k) + \psi$.

With adversary strategy, we consider robot $r_{i+1}$ and $r_{i-1}$ cooperating to maximally reduce the angular difference between $\xi_i$ and $\xi_{i+1}$, trying to make $\xi_{i+1}$ larger than $\xi_i$ at some time within round $k$ of robot $r_i$. It is easy to see that in the worse case, robot, $r_{i+1}$ and $r_{i-1}$ should move anticlockwise throughout round $k$. Then we have the following fact:

$\xi_i(k+1) \geq \xi_i(k) - 2\times\tau\times\Delta t.$ $\quad \xi_{i+1}(k+1) \leq \xi_{i+1}(k) + 2\times\tau\times\Delta t.$

Thus, we have the following fact:

$\xi_i(k+1) - \xi_{i+1}(k+1) \geq \xi_i(k) - 2\times\tau\times\Delta t - \xi_{i+1}(k) - 2\times\tau\times\Delta t > 0$ □

Similarly, we can proof that if $\xi_i(t) < \xi_{i+1}(t) + \psi$, then for any $t_0$, $t_0 < \Delta t$, $\xi_i(t+t_0) < \xi_{i+1}(t_0)$.

**Lemma 2** : Given any number of robots and initial state, Algorithm2 can successfully terminate.

**Proof.** To analyze the property of the algorithm, we first define $var_i$ as $var_i = (\xi_i - \varepsilon)^2$. Then we can obtain the the summation of all the vars, denoted by var as follows:

$$\mathrm{var} = \sum_{i=1}^{n} \mathrm{var}_i = \sum_{i=1}^{n} (\xi_i - \varepsilon)^2 = \sum_{i=1}^{n} (\xi_i^{\,2} - 2\varepsilon\xi_i + \varepsilon^2) = \sum_{i=1}^{n} \xi_i^{\,2} - n\varepsilon^2$$

Since self-stabilization phase occurs after member recruiting phase, $n$ and $\varepsilon$ are constant when the self-stabilization algorithm starts running. Thus, *var*'s value only depends on the summation of the square of $\xi_i$. Similarly, in the following paragraph, we will show that *var* keep decreasing until the self-stabilization algorithm terminates. We consider the situation in which some robots are moving during time slot $k$ in the process of self-stabilization.

In Fig. 7, at time $k$–1, we assume that robot $r_{m1}$, $r_{m2}$, …, $r_{ml}$ ($m1$, $m2 \ldots ml \in [1,n]$) are moving. Since each robot's period time is an integer of clock ticks, these robots will keep moving throughout the time slot $k$. Let $[p_1 \ldots p_{m1} \ldots p_{m2} \ldots p_{ml} \ldots, p_n]$ denotes the system configuration at time $k$–1 and $[p_1 \ldots p'_{m1} \ldots p'_{m2} \ldots p'_{ml} \ldots, p_n]$ the configuration at time $k$. Then, we can transform the concurrent robots movement at time slot $k$ into sequential movement as follows:

Consider a moving robot $r_{mi}$ in Fig. 8. From the above property, we can see that its movement is consistent with the environment within time slot $k$, independent of its neighbor's movement. Assume that the left angle and right angles of robot $r_{mi}$ are $\xi_{mi}$

and $\xi_{mi+1}$ in the sequential configuration that robot $r_{mi}$ will move. Without loss of generality, we assume that $\xi_{mi}$ is greater than $\xi_{mi+1}$. Then, after robot $r_{mi}$ moves, the angles will be changed as follows:

$\xi'_{mi}=\xi_{mi}-\eta$  $\xi'_{mi+1}=\xi_{mi+1}+\eta$.

where $\eta$ is the minimum angle change of robot $r_{mi}$ in one clock tick, i.e equals $\tau\times\gamma$. Then, the impact of the robot $r_{mi}$'s movement on the *var* is as follows:

$$var' - var = \xi'^2_{mi} + \xi'^2_{mi+1} - \xi^2_{mi} - \xi^2_{mi+1}$$
$$= \xi'^2_{mi} + \xi'^2_{mi+1} - (\xi'_{mi}+\eta)^2 + (\xi'_{mi+1}-\eta)^2 = = 2\eta\,(\xi'_{mi+1}-\xi'_{m}-\eta).$$

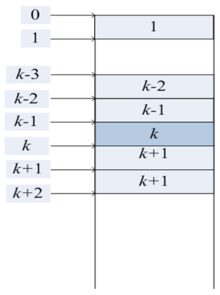


**Fig. 7.** System configuration.

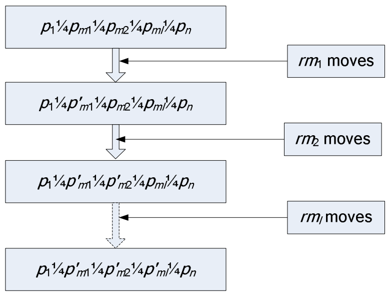


**Fig. 8.** System configuration decomposition.

From Lemma 1, we know that $\xi'_{mi+1}-\xi'_{mi}<0$, so we have $var' - var < -2\eta^2$. Thus, after clock ticks $k$, *var* will decrease at least by $2\times l\times\eta$. Since the largest period of all the robots is $\Delta t$, it is easy to see that the self stabilization algorithm will terminate if the robots stay stationary within time period $2\Delta t$. Thus, the algorithm will terminate. □

Let $\kappa_{max}$ denote the maximum task period of the robot and $\kappa_{min}$ denote the minimum period. Then for any time interval $[t, t+\kappa_{max}\times\gamma]$, every robot planning task is triggered at least one time. Hence, if all the robots stay stationary, the algorithm terminates. Thus we have the following theorem:

**Theorem 4.** Given any initial state, Algorithm II can successfully stabilize within time $2(\kappa_{max}+\kappa_{min})\pi^2/(\kappa_{min}\times\tau^2\times\gamma)$.

**Proof:**

Omitted.

The introduction of the threshold in the process of determining the robot movement may make the stable distribution uneven. The worst case is as follows:

$$\xi_2=\xi_1+4\times\tau\times\Delta t \qquad \xi_3=\xi_2+4\times\tau\times\Delta t \qquad \ldots\ldots$$
$$\xi_{\frac{n}{2}}=\xi_{\frac{n}{2}-1}+4\times\tau\times\Delta t \qquad \xi_{\frac{n}{2}+1}=\xi_{\frac{n}{2}}-4\times\tau\times\Delta t \qquad \ldots..$$

Thus, the system can reach a safe state if $\theta \geq \frac{2\pi}{n}+\frac{n}{4}\times 4\times\tau\times\Delta t$. Consequently, the following theorems hold.

**Theorem 5.** The system can reach a safe state if $\frac{2\theta-\sqrt{\theta^2-8\pi\tau\Delta t}}{2\tau\Delta t} \leq n \leq \frac{2\theta+\sqrt{\theta^2-8\pi\tau\Delta t}}{2\tau\Delta t}$.

**Theorem 6.** The system can reach a goal state if $\delta > n\times\tau\times\Delta t$.

## 4. Experimental study

To illustrate the effectiveness of the proposed approach to the self-organizing behavior of the multi-robot system, we test the algorithm in different cases. The algorithm is performed on a conventional 2G RAM, Intel Core 2 Duo 7600 PC with Windows XP.

The case study considers the even distribution of groups of homogeneous robots around a circle. The radius of the rigid body is 2m (meters). The parameters of the mobile robot are as follows:

| Maximum translational speed ($\varpi$) | Maximum rotational speed ($\tau$) | Clock Tick | Task period | Threshold ($\psi$) |
|---|---|---|---|---|
| 0.2m/s | 0.1 | 0.1s | 5 (0.5s) | 0.2 |

**Table 1.** Parameter of the robot (*m*=meter, *s*=second)

We test the Algorithm II in 100 rounds. In each round, a unique random seed is selected to generate the initial distribution for the robot group of size 0 to 39. Convergence time is illustrated in Fig. 9. The *x* axis denotes the number of rounds; the *y* axis denotes the convergence time for each distribution. Note, for each round, we generate 40 distributions with same random seed and all the distributions have successfully converged.

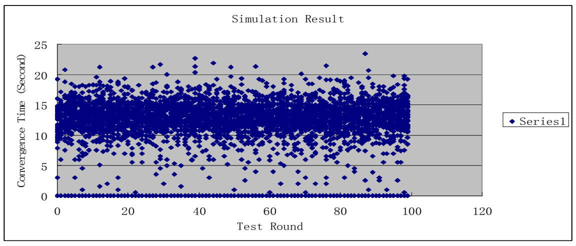


**Fig. 9.** Convergence Time

There are 89.625% initial distributions that converge within the time interval [8*s*-18*s*]. Comparing with the maximum convergence time 25 seconds, the upper bound of convergence time is $\pi/\tau$=31.4s from algorithm I. The convergence time comparison between different numbers of robots is illustrated in Fig.10. The *x* axis denotes the number of robots; the *y* axis denotes the time (seconds). Series 1 is the average convergence time and Series 2 is deviation of the average time.

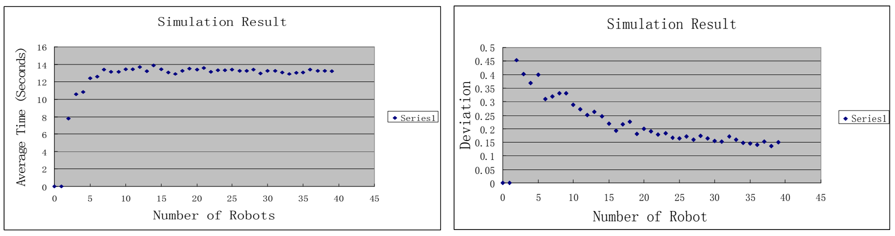


**Fig. 10.** Average Time and Standard Deviation

From Fig. 10, we can conclude that 92.106% distributions converge within time interval [120, 140] in our test. Also the convergence time is irrelevant with the numbers of robot when the latter is above 5. For a general case, assume that the convergence time follows the normal distribution and the average convergence time is less than 14 and standard deviation is less than 0.5. Then 99.8% of the distributions converge within 15.5 seconds.

## 5. Conclusion

We have presented two self-organizing algorithms to adaptively form a safe structure for multiple robots transporting rigid objects. The algorithms are developed based on a safe self-stabilization model, where failures of some robots may bring the system to a non-goal state but the system will always stay in the safe states. The convergence of the algorithms has been formally proven and the convergence speed has been analyzed. The simulation study shows that our algorithm can efficiently converge and the convergence speed is independent of the number of robots. This is especially useful in real-time swarm-robot application since the timing property of the system behavior is predictive.

Our approach is an attempt in forming a structure for swarm-robotic rigid body transportation system. Following this paper, there are several research directions that can be investigated. We will consider more complex structure formation for other real time applications. Also, new self-organizing algorithms which deal with obstacles during rigid body transportation will be developed. Moreover, we plan to test the algorithms using real robots to experimentally validate our approach.

## Reference

[1] Shlomi Dolev: “Self-stabilization”, The MIT Press, Cambridge, Massachusetts, London, England, 2000.

[2] K.P. Valavanis, D. Gracanin, M. Matijasevic, R.Kolluru, and G. A.Demetriou: “Control Architecture for Autonomous Underwater Vehicles”, IEEE Contr. Syst., Dec. 1997, pp. 48-64.

[3] Lovekesh Vig and Julie A. Adams: “Multi-Robot Coalition Formation”, IEEE Transactions on Robotics, Vol. 22, No. 4, AUG 2006.

[4] Lynne E Parker. “Current Research in Multi-Robot Systems”, Journal of Artificial Life and Robotics, 2003.

[5] Marco Dorigo, Vito Trianni, Erol Sahin, Roderich Groß, Thomas H. Labella, Gianluca Baldassarre, Stefano Nolfi, Jean-Louis Deneubourg, Francesco Mondada, Dario Floreano, Luca M. Gambardella: “Evolving Self-Organizing Behaviors for a Swarm-bot”, Auton, Robots, 2004 , pp. 223--245 vol. 17 no. 2—3.

[6] Wang, Y. F., Chirikjian G. S: “A new potential field method for robot path planning”, Proceedings of IEEE International Conference on Robotics and Automation, San Francisco, USA, pp. 977-982.