

# Multirobot Cooperative Learning for Predator Avoidance

Hung Manh La, *Member, IEEE*, Ronny Lim, and Weihua Sheng, *Senior Member, IEEE*

**Abstract**—Multirobot collaboration has great potentials in tasks, such as reconnaissance and surveillance. In this paper, we propose a multirobot system that integrates reinforcement learning and flocking control to allow robots to learn collaboratively to avoid predator/enemy. Our system can conduct concurrent learning in a distributed fashion as well as generate efficient combination of high-level behaviors (discrete states and actions) and low-level behaviors (continuous states and actions) for multirobot cooperation. In addition, the combination of reinforcement learning and flocking control enables multirobot networks to learn how to avoid predators while maintaining network topology and connectivity. The convergence and scalability of the proposed system are investigated. Simulations and experiments are performed to demonstrate the effectiveness of the proposed system.

**Index Terms**—Flocking control, hybrid system, multirobot systems, reinforcement learning.

## I. INTRODUCTION

COOPERATIVE learning and control in multirobot systems [1], [2] has been attracting growing interest since robot teams have great potentials in tasks, such as reconnaissance, surveillance, and security [3], [4]. When a team of robots is deployed to conduct surveillance, the enemy may react to attack the robot team to disrupt its operation. In this scenario, the robot team should have the ability to deal with the enemy. It is desirable that the robot team can avoid the enemy while maintaining the network topology and connectivity. Biology tells us that there is an effective antipredator function in animal aggregations [5]–[7], where the fish schools or bird flocks move together to create a sensory overload on the predator’s visual channel (Fig. 1). Our paper focuses on the distributed decision making problem, where each individual robot has a number of options (safe places) to move to when predators appear. Often in these decisions, there is a benefit for consensus, where all individuals



Fig. 1. Predator and a school of fish (source: www.eversoblue.blogspot.com).

choose the same safe place. However, the consensus methods [8]–[10] usually require a connected network in which all robots can communicate with each other. This may not be valid in real environments because some robots may not connect to the network during the escape. In this case, the consensus algorithms will fail. Therefore, in this paper, we are interested in the problem of reaching consensus even when some robots cannot connect to the network sometimes, but they could learn from experience to make right decisions. Our method is based on a novel combination of flocking control [11] and reinforcement learning [4], [12]–[14].

Flocking control for multiple mobile agents studied in [11] and [15]–[17] was inspired by the natural phenomena of bird flocks and fish schools [18]. Basically, the flocking control law is designed based on three basic flocking rules proposed in [18]: flock centering (agents try to stay close to nearby flockmates), collision avoidance (agents try to avoid collision with nearby flockmates), and velocity matching (agents try to match their velocity with nearby flockmates). The problems of flocking have also attracted many researchers in physics [19], [20], mathematics [21], biology [22], and especially control science in recent years [11], [23]–[25].

In recent years, machine learning techniques, such as reinforcement learning, have been developed for multirobot systems that allow robots to learn cooperation [14], [26]. However, traditional reinforcement learning assumes discrete and finite state/action spaces; therefore, it is difficult to directly apply reinforcement learning to most real-world applications that inherently involve with continuous space. Furthermore,

Manuscript received April 30, 2013; revised October 25, 2013; accepted February 22, 2014. Date of publication April 4, 2014; date of current version December 15, 2014. Manuscript received in final form March 16, 2014. This work was supported in part by the U.S. Department of Defense through the DoD ARO DURIP under Grant 55628-CS-RIP, in part by the DEPSCoR under Grant W911NF-10-1-0015, in part by the NSF under Grant CISE/IIS 1231671, and in part by the Vietnamese Government through the 322 Project, Ministry of Education and Training. Recommended by Associate Editor D. Liu.

H. M. La and R. Lim are with the Center for Advanced Infrastructure and Transportation, Rutgers University, Piscataway, NJ 08854 USA (e-mail: hung.la11@rutgers.edu; ronny.lim@rutgers.edu).

W. Sheng is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: weihua.sheng@okstate.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCST.2014.2312392

even if the states can be discretized, the learned behaviors are still discrete. In addition, the switching of discrete behaviors usually causes the control of the robots to become nonsmooth that is undesirable in most applications. To tackle these issues, several methods have been proposed to make the reinforcement learning work in continuous environments. The common approach is to use a function approximator to learn a value function, and there are several examples of successful applications [27]–[31]. In this paper, instead of following such a common approach, we try to combine reinforcement learning and flocking control to create a hybrid system. Our new framework allows the proposed system to:

- 1) generate efficient combination of high-level behaviors (discrete states and actions) and low-level behaviors (continuous states and actions) for multirobot cooperation;
- 2) coordinate the concurrent learning process in a distributed fashion.

Part of this paper has been published in [32]. The rest of this paper is organized as follows. Section II presents the flocking control algorithm. Section III describes a general framework to enable cooperative learning. Section IV presents the model of multirobot learning and then proposes a cooperative learning algorithm. Section V analyzes the convergence of the proposed learning algorithm. Section VI shows the simulation and experiment results. Finally, the conclusion and future work of this paper are given in Section VII.

## II. FLOCKING CONTROL ALGORITHM

To describe a dynamic topology of an  $n$ -robot team, we consider an information graph  $G(\mathbf{V}, \mathbf{E})$  consisting of a vertex set  $\mathbf{V} = \{1, 2, \dots, n\}$  and an edge set  $\mathbf{E} = \{(i, j) : i, j \in \mathbf{V}, i \neq j\}$ . In  $G$ , each robot is presented by a vertex and each edge denotes the communication link between two members/robots.

Let  $\mathbf{p}_i, \mathbf{v}_i \in \mathbb{R}^2$  denote the position and velocity vectors of the  $i$ th robot. For simplicity, the kinematic motion of each robot is modeled as a particle

$$\begin{cases} \dot{\mathbf{p}}_i = \mathbf{v}_i \\ \dot{\mathbf{v}}_i = \mathbf{u}_i, \quad i = 1, 2, \dots, n \end{cases} \quad (1)$$

where  $\mathbf{u}_i$  is the control input vector of robot  $i$ th. Equation (1) can be used to model mobile robots that have omni-directional motion capability, such as the Rovio mobile robots [33].

We define an actual neighborhood set of robot  $i$  at time  $t$  as follows:

$$N_i^a(t) = \{j \in \mathbf{V} : \|\mathbf{p}_j - \mathbf{p}_i\| \leq r_a, j \neq i\} \quad (2)$$

where  $r_a$  is an active range and  $\|\cdot\|$  is the Euclidean distance. The superscript  $a$  indicates the actual neighbors of robot  $i$  that is used to distinguish with virtual neighbors in the case of obstacle avoidance. The set of virtual neighbors of robot  $i$  at time  $t$  with  $K$  obstacles is defined as

$$N_i^v(t) = \{k \in \mathbf{V}_o : \|\mathbf{p}_{ik} - \mathbf{p}_i\| \leq r_o, \quad \mathbf{V}_o = \{1, 2, \dots, K\}\} \quad (3)$$

where  $r_o$  is the obstacle detection range.  $\mathbf{V}_o$  is a set of obstacles.  $\mathbf{p}_{ik}$  is the position of robot  $i$  projecting on the  $k$ th obstacle. For convenience from now on we just use  $N_i^a$  and  $N_i^v$  by dropping  $t$ . The virtual neighbors are used to generate the repulsive force to push the robots away from the obstacles. The moving obstacles are considered as the predators.

Robots in the team will work together with neighbors to form a certain geometry formation (e.g., quasi lattice formation) while avoiding collision with each other and static/moving obstacles. The distributed flocking control law  $\mathbf{u}_i$  for each robot is introduced [11], [15] consisting of three terms: formation control  $\mathbf{f}_i^f$ , obstacle avoidance  $\mathbf{f}_i^o$ , and navigation  $\mathbf{f}_i^n$ , namely

$$\mathbf{u}_i = \mathbf{f}_i^f + \mathbf{f}_i^o + \mathbf{f}_i^n \quad (4)$$

where  $\mathbf{f}_i^f = c_1^a \sum_{j \in N_i^a} \phi_a(\|\mathbf{p}_j - \mathbf{p}_i\|_\sigma) n_{ij} + c_2^a \sum_{j \in N_i^a} a_{ij}(p)(\mathbf{v}_j - \mathbf{v}_i)$ , here  $\phi_a(\cdot)$  is a pair-wise attractive/repulsive action function to allow robots to maintain a certain distance with its neighbors. The  $\sigma$ -norm,  $\|\cdot\|_\sigma$ , defined as  $\|\mathbf{x}\|_\sigma = 1/\epsilon[\sqrt{1 + \epsilon\|\mathbf{x}\|^2} - 1]$  with  $\epsilon > 0$ , unlike the Euclidean norm  $\|\cdot\|$ , is differentiable everywhere.  $\mathbf{f}_i^o = c_1^o \sum_{j \in N_i^v} \phi_o(\|\mathbf{p}_{ik} - \mathbf{p}_i\|_\sigma) n_{ik} + c_2^o \sum_{j \in N_i^v} b_{ik}(p)(\mathbf{v}_{ik} - \mathbf{v}_i)$ , here  $\phi_o(\cdot)$  is a repulsive action function to allow robots to avoid obstacles.  $n_{ij}$  and  $n_{ik}$  are vectors along the line connecting  $\mathbf{p}_j$  to  $\mathbf{p}_i$  and  $\mathbf{p}_{ik}$  to  $\mathbf{p}_i$ , respectively. Matrices  $A = [a_{ij}]$  and  $B = [b_{ik}]$  are the adjacency matrices defined by graph  $G$  [34].

To allow all robots to move together and track a virtual leader (static/moving target or safe place), the negative feedback function  $\mathbf{f}_i^n$  is defined as

$$\begin{aligned} \mathbf{f}_i^n = & -c_1^t(\mathbf{p}_i - \mathbf{p}_t) - c_2^t(\mathbf{v}_i - \mathbf{v}_t) \\ & -c_1^l(\bar{\mathbf{p}}_{(N_i^a \cup \{i\})} - \mathbf{p}_t) - c_2^l(\bar{\mathbf{v}}_{(N_i^a \cup \{i\})} - \mathbf{v}_t) \end{aligned}$$

where  $c_1^t, c_2^t, c_1^l$ , and  $c_2^l$  are positive constants. The pair  $(\bar{\mathbf{p}}_{(N_i^a \cup \{i\})}, \bar{\mathbf{v}}_{(N_i^a \cup \{i\})})$  is defined as

$$\begin{cases} \bar{\mathbf{p}}_{(N_i^a \cup \{i\})} = \frac{1}{|N_i^a \cup \{i\}|} \sum_{i=1}^{|N_i^a \cup \{i\}|} \mathbf{p}_i \\ \bar{\mathbf{v}}_{(N_i^a \cup \{i\})} = \frac{1}{|N_i^a \cup \{i\}|} \sum_{i=1}^{|N_i^a \cup \{i\}|} \mathbf{v}_i \end{cases}$$

where  $|N_i^a \cup \{i\}|$  is the number of agents in agent  $i$ 's local neighborhood, including agent  $i$  itself.  $\mathbf{p}_t$  and  $\mathbf{v}_t$  are position, and velocity of the moving target.

The result of the flocking control of multirobot based on algorithm (4) is shown in Fig. 2.

## III. SYSTEM INTEGRATION OF FLOCKING CONTROL AND REINFORCEMENT LEARNING

In this section, we build a framework of cooperative learning for multirobot cooperation based on flocking control and reinforcement learning. In this framework our goals are to:

- 1) allow the robots to learn with continuous states and actions;
- 2) coordinate the concurrent learning process to generate an efficient control policy in a distributed fashion.

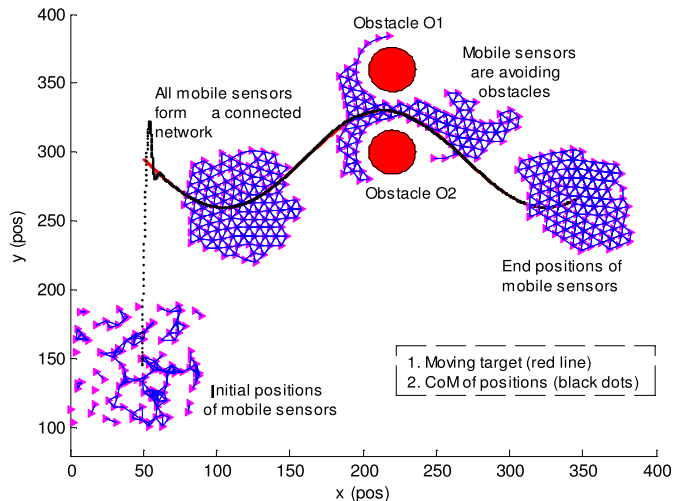


Fig. 2. Multiple robots flock together to track a target moving in a sine wave trajectory based on the flocking control algorithm (4) [15].

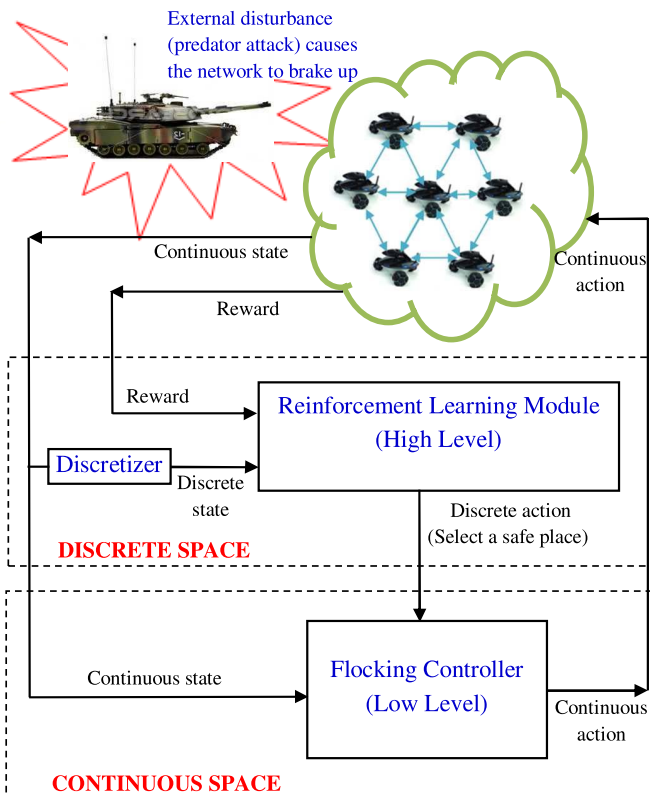


Fig. 3. Hybrid system for reinforcement learning and flocking control in a multirobot domain.

With regard to the limitation of discrete and finite space, we propose a hybrid system of reinforcement learning in a discrete space and flocking controller in a continuous space as shown in Fig. 3. This control architecture has two main parts, the reinforcement learning module (high level) and the flocking controller module (low level).

The flocking controller (4), which works in a continuous space, is the network controller that controls all robots to move together without collision and track a stationary or moving target. In general, the target  $(\mathbf{p}_t, \mathbf{v}_t)$  is defined

as follows:

$$\begin{cases} \dot{\mathbf{p}}_t = \mathbf{v}_t \\ \dot{\mathbf{v}}_t = \mathbf{f}_t(\mathbf{p}_t, \mathbf{v}_t). \end{cases} \quad (5)$$

In this paper, we only consider a stationary target (a fixed point or safe place). Then,  $\mathbf{p}_t$  and  $\mathbf{v}_t$  are considered to be constant vectors. When the predator is detected, several safe places  $(\mathbf{p}_{t_1}, \mathbf{p}_{t_2}, \dots, \mathbf{p}_{t_N}, N \in \mathbb{Z})$  are generated by the prey. These safe places are generated based on the moving direction of the predator to maximize the escaping probability.

The flocking controller also allows the robots to avoid the predators based on a repulsive force generated from an artificial potential field induced by the predators. However, this repulsive force usually breaks up the network. Therefore, we need to combine both flocking control and reinforcement learning so that they can avoid the predators while maintaining network formation (topology) and connectivity.

The reinforcement learning module that works in a discrete space, is the key to the controller. The goal is to agree on one of the safe places for the flocking controller. By retrieving the states (after they are discretized) and the rewards, the reinforcement learning module finds the appropriate safe place so that the network topology and connectivity can be maintained.

Our framework is valid in real situations when the predators continuously attack the prey network. Since all robots in a cooperative multirobot system can influence each other, it is important to ensure that the actions are selected by the individual robots resulting in effective decisions for the whole group.

#### IV. MODELING AND COOPERATIVE LEARNING ALGORITHM

In this section, we build a model of multirobot learning and then develop the cooperative learning algorithm.

##### A. Model of Multirobot Learning

The multirobot learning problem can be shown in Fig. 4. In this figure, the robots learn to make the same decision, i.e., selecting the same safe place to escape, so that the network will not break up, and the network topology and connectivity can be maintained. Based on the moving direction of the predator, the safe places are realtimely generated by the prey network. The prey will keep escaping as long as the predator is keeping attacking.

When the predators attack the prey network, they try to strike the center of the network. This behavior of the predators is usually adopted in existing works [35] and [36]. Usually such attacks will cause the prey network to break up. As a result, the prey will not flock or move together. This is one of the reasons that the prey have to learn in a cooperative fashion so that they can agree on the same safe place to escape the predators [5]. Therefore, we model the prey or robots as follows.

- 1) Initially, all robots flock together in free space and form an  $\alpha$ -lattice formation (Fig. 2) based on the distributed flocking control algorithm (4).

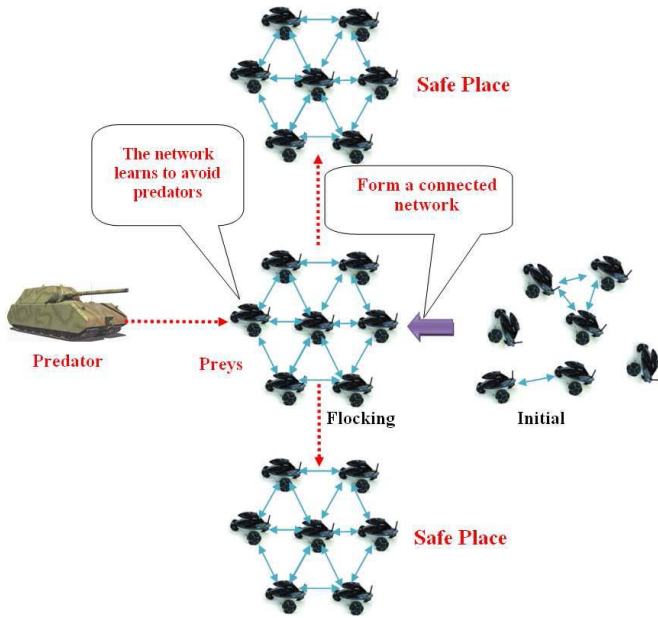
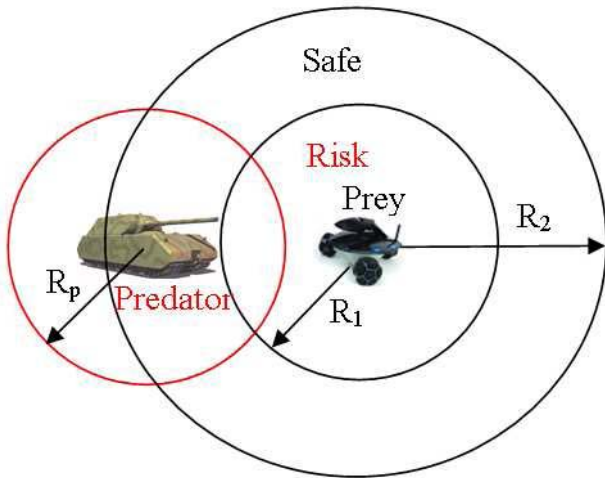


Fig. 4. Illustration of the safe places to choose.


 Fig. 5. Illustration of the predator and prey detection ranges.  $R_1$  is the active range of the robot (prey),  $R_2$  is the predator detection range, and  $R_p$  is the prey detection range.

- 2) If the predators come into the detection range ( $R_2$ ), the robot can sense the location of the predators. The robot will learn and select one of the safe places to move to (Fig. 5).
- 3) If the predators come into the risk area ( $R_1$ ), the robot will move away based on the repulsive force via the function  $f_i^o$  as mentioned in (4). Here, we can set  $R_1$  equal to  $r_o$ , as defined in Section II.

### B. Cooperative Learning Algorithm

In this section, we present a cooperative reinforcement learning algorithm based on  $Q$ -learning, which is based on an action-value function that gives the expected utility of taking a given action in a given state. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state [13], [37].

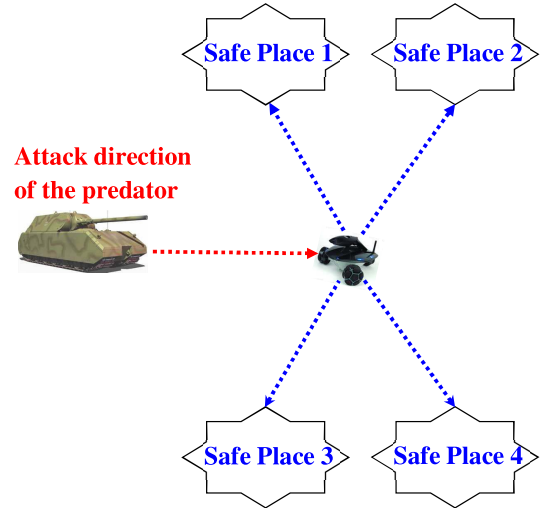


Fig. 6. Four safe places are generated based on the moving direction of the predator.

1) *State, Action, and Reward*: Let the current state, action, and reward of robot  $i$  be  $s_i, a_i, r_i$ , respectively, and the next state and action of robot  $i$  be  $\hat{s}_i, \hat{a}_i$ , respectively. At each moment, we have a partially observable environment. This means that not all robots are able to see the predators, and each robot only communicates with its neighbors to exchange local information. We have the following models for the state, action, and reward.

a) *State*: we assume that when the learning starts the state is initialized. For each robot  $i$ , the state is defined as the number of the predators  $n_p$  in the detection range  $R_2$ , the attacking direction  $d_p^k, k = 1, \dots, n_p$ , of the predators (east, west, south, and north represented by numbers 1, 2, 3, and 4, respectively), and the number of neighboring robots  $|N_i^a|$  in its active range  $r_a$ . As a result, we have  $s_i = [n_p, d_p, |N_i^a|]^T$ ,  $d_p = [d_p^k], k = 1, \dots, n_p$ . For example, if two predators are in the detection range of robot  $i$ , one attacks the robot from north, and the other one attacks from south, and six neighboring robots are in the active range of robot  $i$ , then the state for robot  $i$  is  $[2, 4, 3, 6]^T$ . If only six neighboring robots are in the active range, and no predator is in the detection range of robot  $i$  then the state for robot  $i$  is  $[0, 0, 6]^T$ . If the robot  $i$  performs the action, i.e., selecting one safe place, it will keep moving until the state changes to a different state,  $\hat{s}_i \neq s_i$ . The maximum number of states for each robot or the state list ( $S_i$ ) depends on the number of the neighboring robots, predators, and their attacking direction.

b) *Action*: we assume that the predators can come from any direction with different paths. However, when they detect the prey they try to move into the center of the prey network. Therefore, the desired action of the prey is to move to one of four safe places to escape. If we encode the four safe places as numbers 1, 2, 3, and 4, we have the action list for each robot  $A_i = [1, 2, 3, 4]$ . When the predators enter the risk area, the robot will generate the repulsive force to move away from them. Additional actions can be introduced if needed. The illustration of this scenario is shown in Fig. 6. The action, selecting one of the safe places, is generated

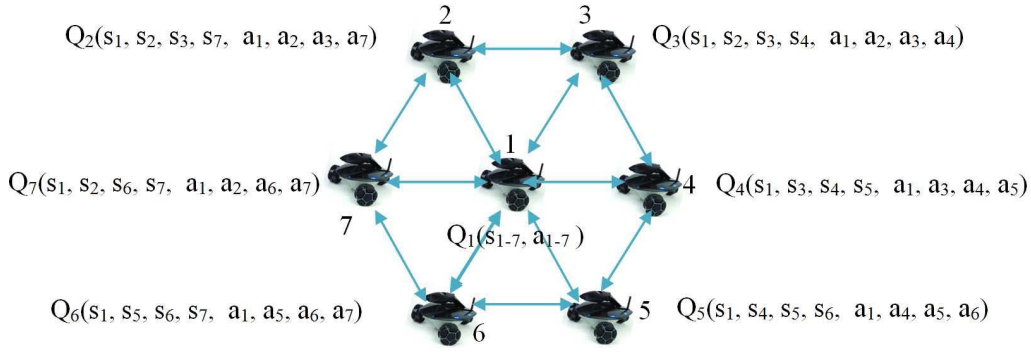


Fig. 7.  $Q$ -value collection based on the robot's action/state and its neighboring actions/states.

in the reinforcement learning module. Then, this action is implemented in the flocking controller (Fig. 3).

*c) Reinforcement reward:* the reinforcement reward signal changes in the experiments, depending on the input data that is received. In  $\alpha$ -lattice configuration (hexagonal lattice configuration), a robot inside the network has six neighbors, and the robot on the border of the network has one to five neighbors. Our purpose is to maintain this network configuration, hence we define the reward as: if  $|N_i^a| < 6$  then  $r_i = |N_i^a|$ , else  $r_i = 6$ . This reward definition basically implies that the maximum reward for each agent is six that corresponds to the hexagonal lattice configuration of the network.

*2) Independent Learning:* For comparison purpose, we implement an independent learning algorithm [13], in which the robots ignore the actions and rewards of other robots, and learn their strategies independently. Each robot stores and updates an individual table,  $Q_i$ , as follows:

$$Q_i^{k+1}(s_i, a_i) \leftarrow Q_i^k(s_i, a_i) + \alpha[r_i^k + \gamma \max_{\hat{a}_i \in A_i} Q_i^k(\hat{s}_i, \hat{a}_i) - Q_i^k(s_i, a_i)] \quad (6)$$

where  $\alpha$  is a learning rate,  $\gamma$  is a discounting factor,  $\hat{a}_i$  is a next action in the action list  $A_i$ , and  $\hat{s}_i$  is a next state in the state list  $S_i$ .

*3) Cooperative Learning:* In a cooperative fashion, we have two phases. The first phase is  $Q$ -value update, and the second one is action selection. In the first phase, we let each robot collect its own  $Q$ -value based on its action/state and its neighbor's action/state. The idea of this collection is shown in Fig. 7. In the second phase, the action selection is based on the maximum  $Q$ -value approach [14], [38].

*a) Q-value update:* our purpose is to allow each robot to aggregate the information of its neighbors via the  $Q$ -value. Therefore each robot updates its  $Q$ -value based on the following:

$$Q_i^{k+1}(s_i, a_i) \leftarrow w Q_i^k(s_i, a_i) + (1 - w) \sum_{j=1}^{|N_i^a|} Q_j^k(s_j, a_j) \quad (8)$$

where  $Q_i^k(s_i, a_i)$  is computed based on (6), and  $|N_i^a|$  is the number of neighbors of robot  $i$ .  $w$  is the weight, and it is designed as  $0 \leq w \leq 1$ . We can clearly see that if  $w = 1$ , the robot only updates its  $Q$ -value by itself (trusts itself only), and

this is exactly the independent learning algorithm as discussed above (6). If  $w = 0$  the robot updates its  $Q$ -value based on its neighborhood  $Q$ -values only (trusts neighbors only).

*b) Action selection strategy:* usually the next action selection in reinforcement learning is based on the Boltzmann action selection strategy [39] or the maximum  $Q$ -value [14], [38]. Therefore, we can select the next action for the learning algorithm as

$$a_i \equiv \max_{a_i \in A_i} Q_i(s_i, a_i) \quad (9)$$

where  $Q_i(s_i, a_i)$  is computed via (8).

*c) Cooperative learning algorithm summary:* From (8), we can see that the experiences of neighboring robots can be incorporated into the  $Q$ -value of robot  $i$  through a weighted linear combination of their state values, and this can be observed in Fig. 7. In this manner, each robot can learn from the previous experiences of its neighbors. For example, if robot A detects the predator, it selects the safe place 1 to escape. This action may cause the robot to break out of the network and consequently gets the reward of 0 (no neighbor,  $|N_i^a| = 0$ ), it saves this experience into its state value. When the same scenario occurs to robot B, and robot A is within its neighborhood, robot B will have low tendency to choose the same action to escape the predator because the experience of robot A at the current state has been used to update the current state of robot B. Therefore, robot B could learn faster than the robots under the independent  $Q$ -learning approach. Overall, the cooperative learning algorithm is summarized in Algorithm 1.

### C. Stop Criterion

The stop criterion of the cooperative learning Algorithm 1 is determined based on the network connectivity and topology, that is, the training will be terminated if the network connectivity and topology do not change for certain time or iterations.

We know that the link (connectivity) between robot  $i$  and robot  $j$  is maintained if the distance  $0 < \|p_i - p_j\| \leq r_a$ . Otherwise this link is considered broken. A dynamic graph  $G(V, E)$  is connected at iteration  $k$  if there exists a path between any two vertices. Hence, to analyze the connectivity of the network we define a connectivity matrix  $[c_{ij}(k)]$  as

**Algorithm 1:** Distributed Cooperative Learning Algorithm

---

Set parameters  $\alpha$  and  $\gamma$ .  
 Build the state list ( $S_i$ ) and action list ( $A_i$ ) for each robot.  
 Initialize the training environments:  
 - Set position of obstacles;  
 - Initialize attacking direction and trajectory of predators.  
 - Initializes the matrix  $Q_i$ .

**for each episode do**  
   **for each iteration  $k$  do**  
     **for each robot  $i$  do**  
       **Initialization phase:**  
       - Update the matrix  $Q_i$ .  
       - Find initial state ( $s_i$ ) that corresponds to the one in the state list ( $S_i$ ).  
       - Randomly select one action ( $a_i$ ) in the action list ( $A_i$ ).  
       **Update phase** (after robot  $i$  does the selected action):  
       - Update the next state ( $\acute{s}_i$ ).  
       - Select the next action ( $\acute{a}_i$ ) based on the maximum  $Q_i$  value.  
       - Compute the reward  $r_i$ .  
       - Compute  $Q_i$  value:  
       
$$\zeta_i^k(s_i, a_i) \Leftarrow Q_i^k(s_i, a_i) + \alpha[r_i^k + \gamma \max_{\acute{a}_i \in A_i} Q_i(\acute{s}_i, \acute{a}_i) - Q_i(s_i, a_i)]$$
  
       - Update  $Q_i$  based on its neighbors:  
       
$$Q_i^{k+1}(s_i, a_i) \Leftarrow w \zeta_i^k(s_i, a_i) + (1-w) \sum_{j=1}^{|N_i^a|} \zeta_j^k(s_j, a_j) \quad (7)$$
  
       where,  $|N_i^a|$  is the number of neighbors of robot  $i$ .  
       - Set the next state as the current state.  
       **Action implementation phase:**  
       - Final action is selected based on  
       
$$a_i \equiv \max_{a_i \in A_i} Q_i(s_i, a_i).$$
  
     **end**  
   **end**  
 - Update the matrix  $Q_i$ .  
 - Reinitialize the training environments:  
 - Change position and number of obstacles;  
 - Change attacking direction and trajectory of predators.  
 - Training is terminated if both network connectivity  $C$  as defined in (11) and network topology  $T$  as defined in Algorithm 2 do not change for a certain period of time.

---

**end**

follows:

$$[c_{ij}(k)] = \begin{cases} 1, & \text{if } j \in N_i^a(k), \quad i \neq j \\ 0, & \text{if } j \notin N_i^a(k), \quad i \neq j \end{cases} \quad (10)$$

**Algorithm 2:** Topology Evaluation Algorithm

---

**for each robot  $i$  do**  
   **if**  $|N_i^a(k)|$  *changes* **then**  
     Topology:  $T_i(k) = \text{abs}(|N_i^a(k)| - |N_i^a(k-1)|)$   
     ( $k$  is time step or iteration)  
   **else if**  $|N_i^a(k)|$  *does not change, but indices of*  $|N_i^a(k)|$  *change* **then**  
     Topology:  $T_i(k) = \text{number of index changes}$   
   **else**  
     Topology:  $T_i(k) = 0$   
   **end**  
**endfor**  
 For the whole network :Topology:  $T(k) = \sum_{i=1}^n T_i(k)$

---

and  $c_{ii} = 0$ . Since the rank of Laplacian of a connected graph  $[c_{ij}(k)]$  of order  $n$  is at most  $(n-1)$  or  $\text{rank}([c_{ij}(k)]) \leq (n-1)$ , the relative connectivity of a network at time  $t$  is defined as [11]

$$C(k) = \frac{1}{n-1} \text{rank}([c_{ij}(k)]). \quad (11)$$

If  $0 \leq C(k) < 1$  the network is broken, and if  $C(k) = 1$  the network is connected.

To evaluate the topology maintenance, we define a measure  $T$  to reflect the change of the number of neighbors for each robot. The topology of the network is evaluated based on Algorithm 2.

From Algorithm 2, we see that if  $T(k) = 0$ , the topology of the network does not change at this iteration, and if  $T(k) > 0$ , the topology of the network changes.

## V. CONVERGENCE ANALYSIS

The convergence of the flocking control algorithm (4) is already discussed in [15]. Basically, by applying the distributed control protocol (4), the center of mass of positions, and velocities of all agents in the network will exponentially converge to the target. In addition, the formation of all mobile agents will maintain in the process of the target tracking. This statement was validated and shown in Fig. 2. Therefore, as stated we just need to show the convergence of our proposed cooperative learning Algorithm 1.

First, we evaluate the convergence of the proposed system based on the average values of the  $\Delta \mathbf{Q}(s, \mathbf{a})$  of all  $\Delta Q_i = Q_i^{k+1}(s_i, a_i) - Q_i^k(s_i, a_i)$ . The idea is to show the  $Q$ -value of reinforcement learning converging to the optimal  $Q^*$ -value defined by Bellman equation in the stochastic case

$$Q^*(s, a) = R(s, a) + \gamma \max_a \sum_s P(\acute{s}|s, a) Q^*(\acute{s}, a) \quad (12)$$

where  $R(s, a)$  is the reward for taking the action  $a$  giving the highest expected return, and  $P(\acute{s}|s, a)$  is the state transition probability.

We should show that the expectation  $E(Q_i^{k+1}(s_i, a_i))$  converges to the optimal value  $Q^*(s, a)$  as defined in (12). However, for simplicity here we just show the deterministic

case in which  $Q_i^{k+1}(s_i, a_i)$  converges to  $Q^*(s, a)$  defined as

$$Q^*(s, a) = R(s, a) + \gamma \max_a Q^*(\acute{s}, a). \quad (13)$$

Hence, we can state that if the average of  $\Delta \mathbf{Q}(\mathbf{s}, \mathbf{a})$  goes to zero the proposed system is stable. Recall that  $|N_i^a|$  is the number of robots in robot  $i$ 's local neighborhood, and  $Q_i^k(\acute{s}_i, \acute{a}_i) = \max_{\acute{a}_i \in \acute{A}_i} Q_i^k(\acute{s}_i, \acute{a}_i)$ . Without losing the generality, from Algorithm 1 by selecting  $w = 1/2$  we have  $Q_i^{k+1} = 1/2 \sum_{j=1}^{|N_i^a \cup \{i\}|} \zeta_j^k(s_i, a_i)$  and  $Q_i^k = 1/2 \sum_{j=1}^{|N_i^a \cup \{i\}|} \zeta_j^{k-1}(s_i, a_i)$ . Then we have

$$\begin{aligned} \Delta Q_i^k(s_i, a_i) &= Q_i^{k+1}(s_i, a_i) - Q_i^k(s_i, a_i) \\ &= \frac{1}{2} \sum_{j=1}^{|N_i^a \cup \{i\}|} [\zeta_j^k(s_j, a_j) - \zeta_j^{k-1}(s_j, a_j)] - \frac{1}{2} \sum_{j=1}^{|N_i^a \cup \{i\}|} \\ &\quad \times [\Delta Q_j^k(s_i, a_j) + \alpha [r_j^k + \gamma Q_j^k(s_j^*, a_j^*) \\ &\quad - Q_j^k(s_j, a_j)] \\ &\quad - \alpha [r_j^{k-1} + \gamma Q_j^{k-1}(s_j^*, a_j^*) - Q_j^{k-1}(s_j, a_j)]] \\ &= \frac{1}{2} \sum_{j=1}^{|N_i^a \cup \{i\}|} \Delta r_j^{k-1} + \frac{1-\alpha}{2} \sum_{j=1}^{|N_i^a \cup \{i\}|} \Delta Q_j^{k-1}(s_j, a_j) \\ &\quad - \frac{\alpha\gamma}{2} \sum_{j=1}^{|N_i^a \cup \{i\}|} \Delta Q_j^{k-1}(s_j^*, a_j^*). \end{aligned} \quad (14)$$

We can expand (14) to  $n$  robots into space representation as follows:

$$\Delta \mathbf{Q}(\mathbf{s}, \mathbf{a}) = \frac{1}{\alpha} \mathbf{R}(\mathbf{s}, \mathbf{a}) - \frac{\alpha\gamma}{2} \mathbf{Q}(\acute{\mathbf{s}}, \acute{\mathbf{a}}) + \frac{1-\alpha}{2} \mathbf{Q}(\mathbf{s}, \mathbf{a}) \quad (15)$$

where

$$\Delta \mathbf{Q}(\mathbf{s}, \mathbf{a}) = [\Delta Q_1(s_1, a_1), \Delta Q_2(s_2, a_2), \dots, \Delta Q_n(s_n, a_n)]^T$$

with  $\mathbf{s} = [s_1, s_2, \dots, s_n]^T$  and  $\mathbf{a} = [a_1, a_2, \dots, a_n]^T$

$$\begin{aligned} \mathbf{Q}(\mathbf{s}, \mathbf{a}) &= \begin{bmatrix} \sum_{j=1}^{|N_1^a \cup \{i\}|} \Delta Q_j(s_j, a_j), & \sum_{j=1}^{|N_2^a \cup \{i\}|} \Delta Q_j(s_j, a_j), & \dots, \\ \sum_{j=1}^{|N_n^a \cup \{i\}|} \Delta Q_j(s_j, a_j) \end{bmatrix}^T \\ \mathbf{Q}(\acute{\mathbf{s}}, \acute{\mathbf{a}}) &= \begin{bmatrix} \sum_{j=1}^{|N_1^a \cup \{i\}|} \Delta Q_j(\acute{s}_j, \acute{a}_j), & \sum_{j=1}^{|N_2^a \cup \{i\}|} \Delta Q_j(\acute{s}_j, \acute{a}_j), & \dots, \\ \sum_{j=1}^{|N_n^a \cup \{i\}|} \Delta Q_j(\acute{s}_j, \acute{a}_j) \end{bmatrix}^T \end{aligned}$$

with  $\acute{\mathbf{s}} = [\acute{s}_1, \acute{s}_2, \dots, \acute{s}_n]^T$  and  $\acute{\mathbf{a}} = [\acute{a}_1, \acute{a}_2, \dots, \acute{a}_n]^T$

$$\mathbf{R} = \begin{bmatrix} \sum_{j=1}^{|N_1^a \cup \{i\}|} \Delta r_j(s_j, a_j), & \sum_{j=1}^{|N_2^a \cup \{i\}|} \Delta r_j(s_j, a_j), & \dots, & \sum_{j=1}^{|N_n^a \cup \{i\}|} \Delta r_j(s_j, a_j) \end{bmatrix}^T$$

*Proposition 1:* Consider a system of  $n$  robots, that have dynamics (1) and are controlled by the control law (4). Based

on Algorithm 1 and a state differential equation (15), the vector  $\Delta \mathbf{Q}(\mathbf{s}, \mathbf{a})$  will converge to a zero vector.

*Proof:* Since each learning robot will select the action that holds the maximum  $Q$ -value we can write (15) into a continuous fashion as

$$\dot{\mathbf{Q}}(\mathbf{s}, \mathbf{a}) = \mathbf{B}\mathbf{Q}(\mathbf{s}, \mathbf{a}) + \alpha \mathbf{R}(\mathbf{s}, \mathbf{a}) \quad (16)$$

where

$$\mathbf{B} = \begin{bmatrix} \frac{1-\alpha-\alpha\gamma}{2} & 0 & \dots & 0 \\ 0 & \frac{1-\alpha-\alpha\gamma}{2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \frac{1-\alpha-\alpha\gamma}{2} \end{bmatrix}_{n \times n}.$$

Now, we can consider (16) as a standard feedback control system ( $\dot{X} = AX + Bu$ ), that is, the model of the system is  $\mathbf{Q}(\mathbf{s}, \mathbf{a})$ , and the control input is  $\alpha$ , and  $\mathbf{R}$  is the output signal of the controller. Therefore, we can easily see that if all of the roots of the characteristic equation of (16) have negative real parts (16) is stable, or the vector  $\Delta \mathbf{Q}(\mathbf{s}, \mathbf{a})$  will converge to a zero vector.

We have the characteristic equation

$$\det(\lambda \mathbf{I} - \mathbf{B}) = 0 \quad (17)$$

here

$$\lambda \mathbf{I} - \mathbf{B} = \begin{bmatrix} \lambda - \frac{1-\alpha-\alpha\gamma}{2} & 0 & \dots & 0 \\ 0 & \lambda - \frac{1-\alpha-\alpha\gamma}{2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda - \frac{1-\alpha-\alpha\gamma}{2} \end{bmatrix}_{n \times n}.$$

From (17) we can obtain

$$\left( \lambda - \frac{1-\alpha-\alpha\gamma}{2} \right)^n = 0. \quad (18)$$

Since  $0 < \alpha < 1$  and if we select  $\gamma > (1 - \alpha/\alpha)$ , all roots of (18):  $\lambda_1 = \lambda_2 = \dots = \lambda_n < 0$ . Hence, we can conclude that the proposed system (16) is stable, or the vector  $\Delta \mathbf{Q}(\mathbf{s}, \mathbf{a})$  will converge to a zero vector. ■

## VI. SIMULATION AND EXPERIMENT RESULT

In this section, we test our distributed cooperative learning Algorithm (Algorithm 1) and the distributed flocking control algorithm (4) in both simulation and experiment. We compare the proposed cooperative learning algorithm with the independent learning algorithm (6) in term of connectivity, topology, convergence, action, and reward.

### A. Simulation Results

In simulation, we test the proposed learning algorithm in two cases: single predator and two predators.

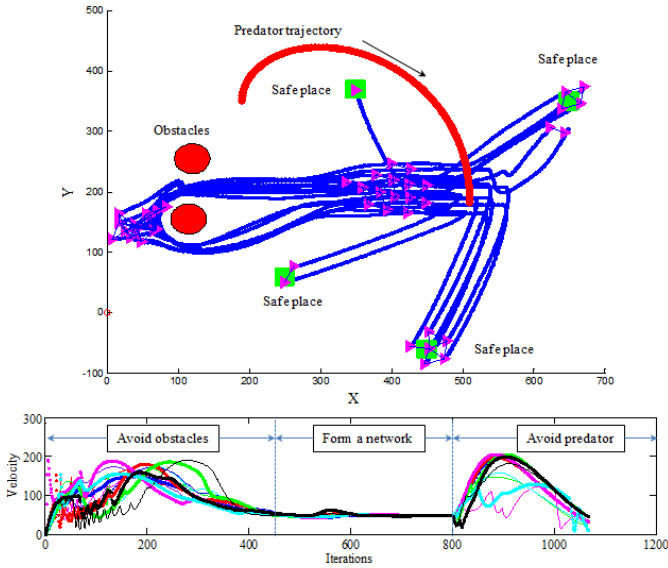


Fig. 8. Top: trajectory/position profile of 15 agents in the first learning episode. The four green squares are the four safe places. The filled red circles are the obstacles. The red trajectory is the position profile of the predator. Bottom: velocity profile of 15 agents during the first learning episode.

1) *Case 1 (Single Predator)*: In this simulation, we use 15 robots (prey), and four actions (four safe places to escape predator). This results in  $4^{15} = 1073741824$  ( $\approx 1$  billion) possible joint actions.

In each learning episode, we randomly setup initial deployments of the prey, location of obstacles, as well as trajectory and initial location of predators. The learning task is to find out one of four optimal joint actions. The parameters of flocking control are as follows: the desired distance among the prey  $d = 16$ , the scaling factor  $k_c = 1.2$ , the active range  $r_a = k_c \times d = 19.2$ , and the constant  $\epsilon = 0.1$  for the  $\sigma$ -norm. The parameters of the learning Algorithm 1 are as follows: updated weight  $w = 0.5$ , learning rate  $\alpha = 0.2$ , and discount factor  $\gamma = 0.9$ .

Fig. 8 shows the result of trajectory and velocity profiles of the prey network in the first training episode. It can be seen that at the beginning (iteration 1–400) the prey have to avoid obstacles, hence their velocity did not agree on the same value as shown in Fig. 8 (bottom). After avoiding obstacles the prey form a network of  $\alpha$  lattice configuration, and their velocities agree on the same value (iteration 400–800). When the predator appears and attacks the prey network (iteration 800 to the end), each prey tries to escape by selecting its safe place. At this moment, the prey increase velocity to escape the predator, and since the prey's behavior changes over time the velocity does not match. This happens because the prey do not have much learning experience, they fail to agree on the same action. Hence, the network is broken.

In the second learning episode, which is not shown in this paper, we observed that the learning result is better since more than 50% of the prey agree on the same safe place. In the third episode (Fig. 9), the learning converges and all the prey select the same action and agree on the same velocity. Therefore, the connectivity is maintained. We can see that even when the trajectories of the predator change as well as the locations of the obstacles change, the learning results still hold.

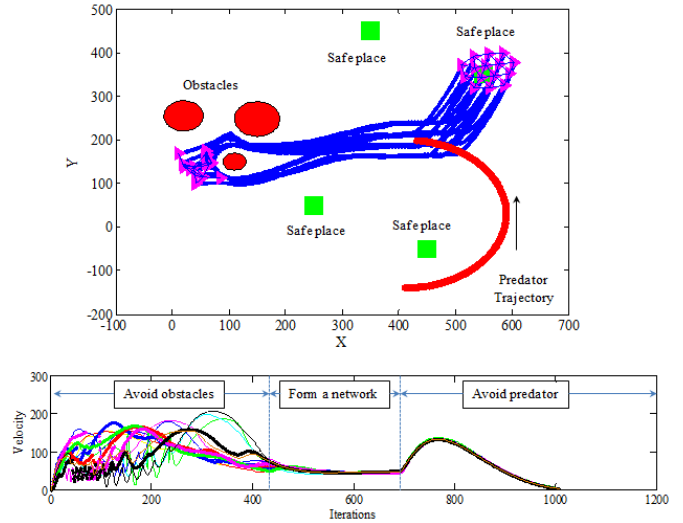


Fig. 9. Top: trajectory/position profile of 15 agents in the third learning episode. The four green squares are the four safe places. The filled red circles are the obstacles. The red trajectory is the position profile of the predator. Bottom: velocity profile of 15 agents during the third learning episode.

2) *Case 2 (Two Predators)*: The results of this case are shown in Figs. 10 and 11. Similar to Case 1, at the first learning episode the robots do not have any experience, they fail to agree on the same action (Fig. 10). After five learning episodes, the learning converges and all the robots choose the same action (Fig. 11). We can also see at the case that the robot (prey) network reaches the safe place [Fig. 11(b)], and the predator continues to attack the network, but the prey can escape to the other safe place [Fig. 11(c)].

## B. Experiment Results

In real experiments, we use eight Rovio robots [33] that have omni-directional motion capability as shown in Fig. 12. In this figure, seven Rovio robots are used as preys, and one Rovio robot is used as a predator. To distinguish with other prey robots the predator robot is marked by a yellow cup mounted on it. Basically, these robots can freely move in eight directions. The dynamic model of the Rovio robot can be approximated by (1). However, the accuracy of the localization of the Rovio robot is low, and the robot does not have any sensing device to sense the pose (position and velocity) of its neighbors, predator, and obstacles. Hence, we use a VICON motion capture system [40] in our lab (Fig. 13) that has 12 infrared cameras to track moving objects. This tracking system can give the location and velocity of each moving object with high accuracy.

Fig. 14 shows the experimental result of the first training episode. Similar to the simulation results, since in the first episode the robots do not have any experience of the behavior of the predator, they failed to agree on the same action. Hence, the network is broken. In the third episode as shown in Fig. 15, the learning converges and all the robots choose the same action (same safe place). Therefore, the topology and the connectivity are maintained.

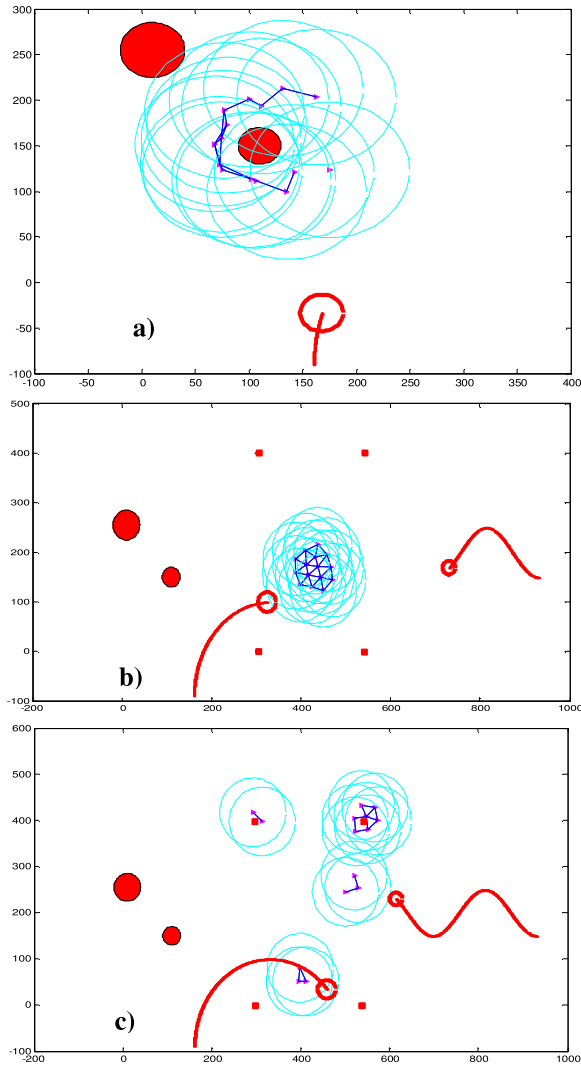


Fig. 10. Snapshots of the proposed cooperative learning algorithm in the first episode. (c) and (d) The four red/darker dots are the four safe places. The empty red circles are the predators. The filled red circles are the obstacles. (a) Robots are avoiding obstacles. (b) Robots are forming a network. (c) Robots are avoiding predators, and did not agree to go to the same place.

### C. Performance Evaluation

In this section, we evaluate the connectivity, topology, convergence, and reward performance of our proposed distributed cooperative learning algorithm (Algorithm 1), then compare with those of the independent learning algorithm (6).

1) *Connectivity Evaluation*: According to the network connectivity evaluation as defined in (11), from the result in Fig. 16, we can see that for the cooperative learning algorithm, the connectivity of the network is maintained after three training episodes, while for the independent learning algorithm, the connectivity is not maintained even after 100 training episodes. This means that the robots do not agree on the same action. Note that in Fig. 16 (right), the connectivity is only lost from iteration 1–400 because the prey have to avoid the obstacles. After that the predator appears, and the preys can avoid it and maintain the connectivity based on the proposed cooperative learning algorithm. In contrast, the connectivity is lost using the independent learning algorithm as shown in Fig. 16 (left).

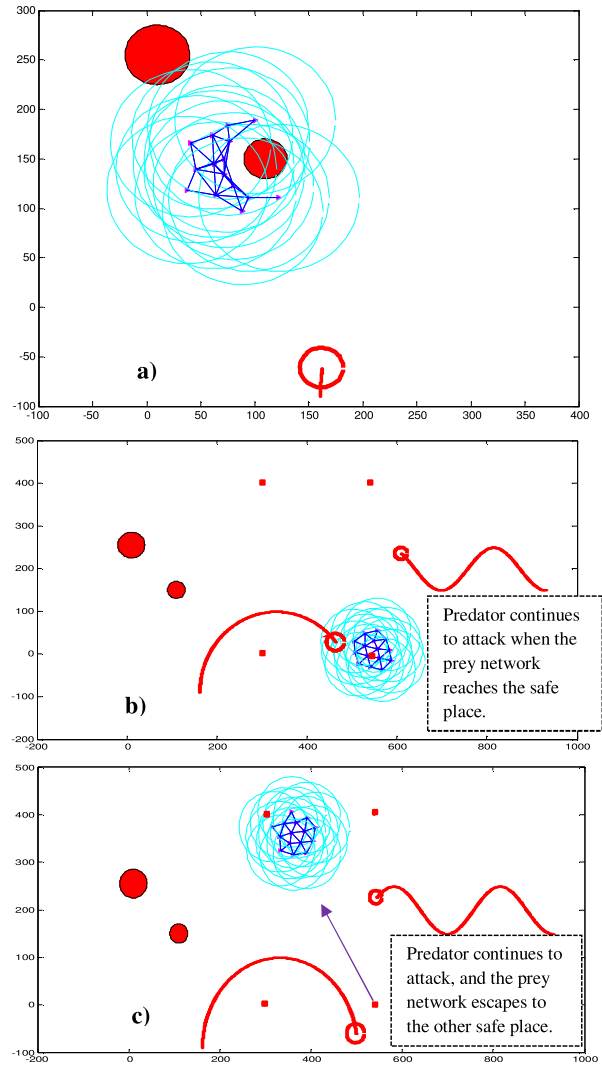


Fig. 11. Snapshots of the proposed cooperative learning algorithm in the fifth episode. (c) and (d) The four red/darker dots are the four safe places. The empty red circles are the predator. The filled red circles are the obstacles. (a) Robots are avoiding obstacles. (b) Robots are avoiding predators. (c) All robots agreed on the same action to go to the same safe place to escape the predator.



Fig. 12. Seven Rovio prey robots and one Rovio predator robot (marked with a yellow cup) are used in the experiment.

2) *Topology Evaluation*: According to the network topology evaluation as presented in Algorithm 2, from the result in Fig. 17, we can see that for our proposed cooperative learning

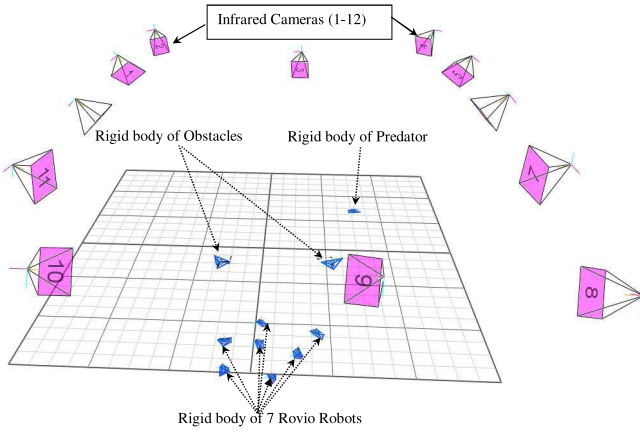


Fig. 13. Infrared cameras tracking system for experimental setup.

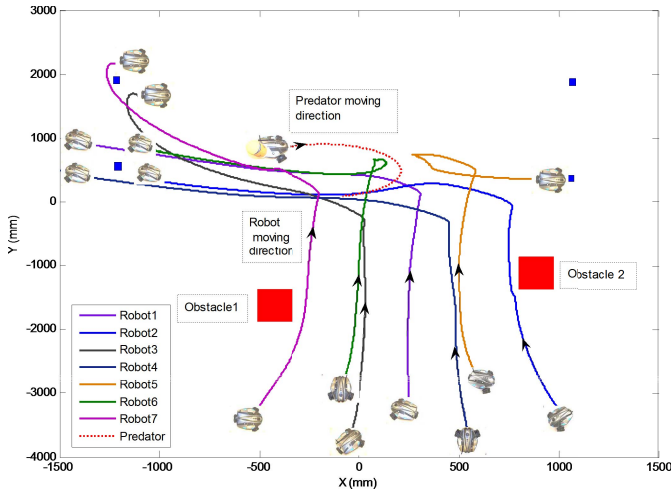


Fig. 14. Trajectories of seven Rovio robots and one predator in the first learning episode. The green small squares are the safe places, and the filled red squares are the obstacles.

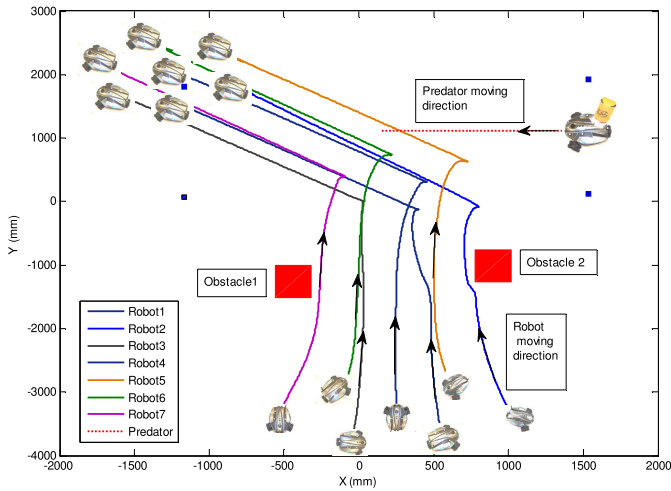


Fig. 15. Trajectories of seven Rovio robots and one predator in the third learning episode. The green small squares are the safe places, and the filled red squares are the obstacles.

algorithm, the topology of the network does not change after three training episodes, while for the independent learning algorithm, the topology changes in all training episodes. Note that in Fig. 17 (right), the topology only changes when the

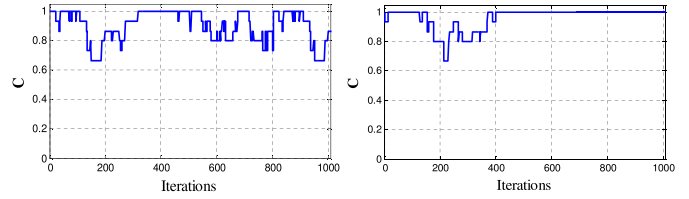


Fig. 16. Connectivity evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right) at the third learning episode. The capital  $C$  on the vertical axis is computed by (11).

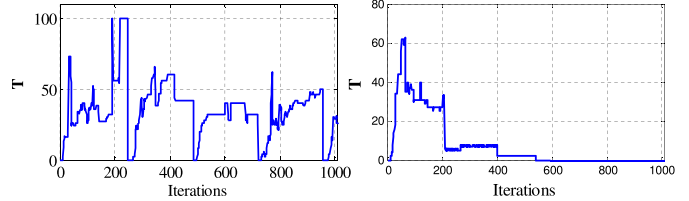


Fig. 17. Topology evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right). The capital  $T$  on the vertical axis is computed by algorithm (2).

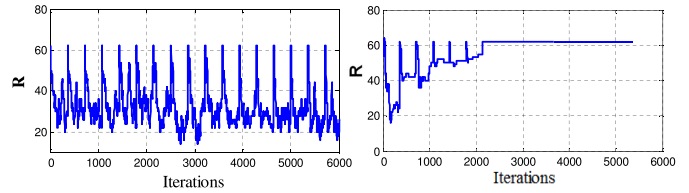


Fig. 18. Global reward evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).

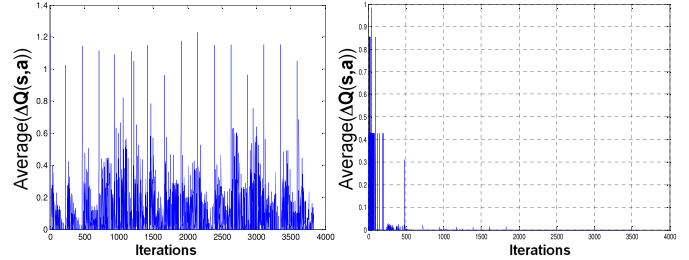


Fig. 19. Convergence of  $Q$ -values for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).

prey have to avoid the obstacles, and it is maintained when they are avoiding the predator.

3) *Reward Evaluation*: The global reward is computed as  $R = \sum_{i=1}^n r_i$ . From the result in Fig. 18 with our proposed cooperative learning algorithm, we can obtain a maximum global reward with a value of 62 after about 2000 iterations, but with the independent learning algorithm the global reward does not converge to a stable value.

4) *Q-value and Action Evaluation*: As stated in Proposition 1, we can see that if the average of  $\Delta Q(s, a)$  goes to zero the proposed system is stable. From the result of average of  $\Delta Q(s, a)$  for 15 agents, as shown in Fig. 19, we can see that for our proposed cooperative learning algorithm the average of  $\Delta Q(s, a)$  goes to zero after 2000 iterations, while for the independent learning algorithm, it does not converge to zero.

Second, we show the action performance of all agents, which is evaluated based on the encoded values for the agent

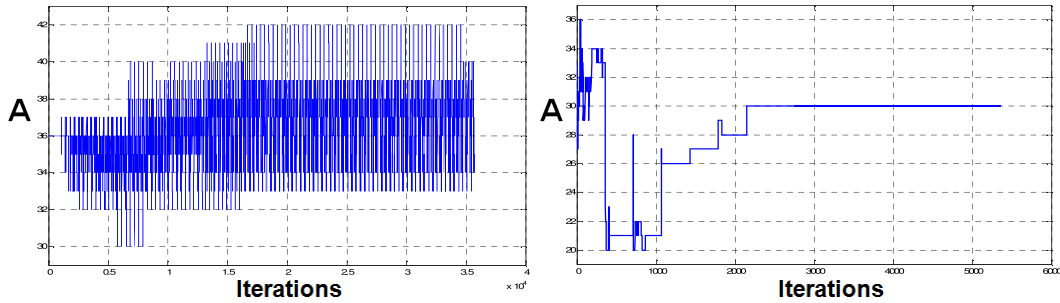


Fig. 20. Action selection evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).  $A$  is computed in (19).

action selection. The total value of action is computed as

$$A = \sum_{i=1}^n E_v(a_i) \quad (19)$$

where  $E_v(a_i)$  is the encoded value corresponding to the selected action  $a_i$ . For example, if we encode four safe places (four actions) by numbers 1–4, and assume that all agents select the same action 3 or  $E_v = 3$ , then total value of action  $A$  of 15 agents is 45. As another example, if six prey choose action 1, and others choose action 2, then  $A = 6 \times 1 + 9 \times 2 = 24$ .

From the result in Fig. 20, we can see that with our proposed cooperative learning algorithm all robots agree on the same action which is encoded as a value of two after 2000 iterations (two learning episodes). Therefore, the summation of valued actions of 15 individuals equals to 30. This means that all prey agreed on the same action to satisfy the tasks, such as maintaining the same topology and connectivity. However, with the independent learning algorithm the summation of action values of individuals does not converge to a stable value. The prey change their action/decision over time, therefore the network is broken.

## VII. CONCLUSION

In this paper, we proposed a hybrid system that integrates flocking control and reinforcement learning to allow robots to behave intelligently in a continuous space. Reinforcement learning is developed based on the cooperative  $Q$ -learning algorithm. We evaluated the proposed hybrid system in the context of multirobot learning to avoid predators. We showed that the proposed cooperative  $Q$ -learning allows the network to find out the effective joint action more quickly than the independent  $Q$ -learning. This also allows the network to maintain its topology and connectivity while avoiding the predator. Both simulation and experiment results are collected to demonstrate the effectiveness of our proposed system.

In the future work, we intend to extend our learning framework to a cooperative and active sensing problem in mobile sensor networks (MSNs) to allow the robots to learn the network configuration to adapt to the complex sensing environments. The current learning framework can be an useful tool for developing the cooperative and active sensing to improve the sensing performance in MSNs, where our initial cooperative sensing was reported in [41] and [42].

## REFERENCES

- [1] J. Choi, S. Oh, and R. Horowitz, "Distributed learning and cooperative control for multi-agent systems," *Automatica*, vol. 45, no. 12, pp. 2802–2814, 2009.
- [2] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, 2012.
- [3] M. A. Fields, E. Haas, S. Hill, C. Stachowiak, and L. Barnes, "Effective robot team control methodologies for battlefield applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2009, pp. 5862–5867.
- [4] J. Ding, J. Gillula, H. Huang, M. P. Vitus, W. Zhang, and C. J. Tomlin, "Hybrid systems in robotics: Toward reachability-based controller design," *IEEE Robot. Autom. Mag.*, vol. 18, no. 3, pp. 33–43, Jan. 2011.
- [5] H. Milinski and R. Heller, "Influence of a predator on the optimal foraging behavior of sticklebacks," *Nature*, vol. 275, pp. 642–644, Oct. 1978.
- [6] G. Roberts, "Why individual vigilance declines as group size increases," *Animal Behavior*, vol. 51, no. 5, pp. 1077–1806, 1996.
- [7] J. Krause, G. Ruxton, and D. Rubenstein, "Is there always an influence of shoal size on predator hunting success?" *J. Fish Biol.*, vol. 52, no. 3, pp. 494–501, 1998.
- [8] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proc. Amer. Control Conf.*, 2005, pp. 1859–1864.
- [9] R. Olfati-Saber, J. Alex Fax, and R. M. Murray, "Consensus and cooperative in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.
- [10] E. Kokiopoulou and P. Frossard, "Distributed classification of multiple observation sets by consensus," *IEEE Trans. Signal Process.*, vol. 59, no. 1, pp. 104–114, Jan. 2011.
- [11] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.
- [12] B. Luo, H. Wu, and T. Huang, "Off-policy reinforcement learning for  $H_\infty$  control design," *Preprint: arXiv:1311.6107*, 2014.
- [13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
- [14] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.
- [15] H. M. La and W. Sheng, "Flocking control of a mobile sensor network to track and observe a moving target," in *Proc. IEEE Inter. Conf. Robot. Autom.*, May 2009, pp. 3129–3134.
- [16] H. M. La and W. Sheng, "Flocking control of multiple agents in noisy environments," in *Proc. IEEE Int. Conf. Robotics Autom.*, May 2010, pp. 4964–4969.
- [17] H. M. La and W. Sheng, "Dynamic targets tracking and observing in a mobile sensor network," *Robot. Auto. Syst.*, vol. 23, no. 7, pp. 996–1009, 2012.
- [18] C. Reynolds, "Flocks, birds, and schools: A distributed behavioral model," *ACM SIGGRAPH, Comput. Graph.*, vol. 21, no. 4, pp. 25–34, 1987.
- [19] T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen, and O. Schochet, "Novel type of phase transitions in a system of self-driven particles," *Phys. Rev. Lett.*, vol. 75, no. 6, pp. 1226–1229, 1995.

- [20] H. Levine, W. J. Rappel, and I. Cohen, "Self-organization in systems of self-propelled particles," *Phys. Review. E*, vol. 63, pp. 017101–017104, Dec. 2000.
- [21] A. Mogilner, L. Edelstein-Keshet, L. Bent, and A. Spiros, "Mutual interactions, potentials, and individual distance in a social aggregation," *J. Math. Biol.*, vol. 47, no. 4, pp. 353–389, 2003.
- [22] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups," *J. Theory Biol.*, vol. 218, no. 1, pp. 1–11, 2002.
- [23] H. G. Tanner, A. Jadbabai, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 863–868, May 2007.
- [24] H. Su, X. Wang, and Z. Lin, "Flocking of multi-agents with a virtual leader," *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 293–307, Feb. 2009.
- [25] H. M. La, R. S. Lim, H. Chen, and W. Sheng, "Decentralized flocking control with minority of informed agents," in *Proc. 6th IEEE Conf. Ind. Electron. Appl.*, Jun. 2011, pp. 1851–1856.
- [26] H. Guo and Y. Meng, "Distributed reinforcement learning for coordinate multi-robot foraging," *J. Intell. Robot. Sys.*, vol. 60, nos. 3–4, pp. 531–551, 2010.
- [27] J. C. Santamaria, R. S. Sutton, and A. Ram, "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adapt. Behaviour*, vol. 6, no. 2, pp. 163–218, 1998.
- [28] G. J. Gordon, "Stable function approximation in dynamic programming," in *Proc. 12th Inter. Conf. Mach. Learn.*, 1995, pp. 261–268.
- [29] C. Gaskett, D. Wettergreen, and A. Zelinsky, *Q-Learning in Continuous State and Action Spaces*. Berlin, Germany: Springer-Verlag, 1999, pp. 417–428.
- [30] W. D. Smart and L. P. Kaelbling, "Practical reinforcement learning in continuous spaces," in *Proc. Int. Conf. Mach. Learn.*, 2000, pp. 903–910.
- [31] B. Baddeley, "Reinforcement learning in continuous time and space: Interference and not ill conditioning is the main problem when using distributed function approximators," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 950–956, Aug. 2008.
- [32] H. M. La, R. S. Lim, and W. Sheng, "Hybrid system of reinforcement learning and flocking control in multi-robot domain," in *Proc. Conf. Theoretical Appl. Comput. Sci.*, 2010, pp. 7–13.
- [33] (2011). *Rovio Robot* [Online]. Available: <http://www.wowwee.com/en/support/rovio>
- [34] F. Bullo, J. Cortes, and S. Martinez, "Distributed control of robotic networks," *Applied Mathematics Series*. Princeton, NJ, USA: Princeton Univ., 2009.
- [35] S. H. Lee, "Predator's attack-induced phase-like transition in prey flock," *Phys. Lett. A*, vol. 357, nos. 4–5, pp. 270–274, 2006.
- [36] K. Morihiro, H. Nishimura, T. Isokawa, and N. Matsui, "Learning grouping and anti-predator behaviors for multi-agent systems," in *Proc. Int. Conf. Knowl. Based Intell. Inf. Eng. Syst.*, 2008, pp. 426–433.
- [37] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dyn. Syst.*, vol. 13, nos. 1–2, pp. 41–77, 2003.
- [38] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, 1996.
- [39] M. Coggan, "Exploration and exploitation in reinforcement learning," in *Proc. 4th Int. Conf. Comput. Intell. Multimedia Appl.*, 2001, pp. 1–44.
- [40] (2011). *VICON Motion Capture System* [Online]. Available: <http://www.vicon.com/>
- [41] H. M. La and W. Sheng, "Cooperative sensing in mobile sensor networks based on distributed consensus," in *Proc. Signal Data Process. Small Targets Conf.*, 2011, pp. 81370Y1–81370Y14.
- [42] H. M. La and W. Sheng, "Distributed sensor fusion for scalar field mapping using mobile sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 766–778, Apr. 2012.



**Hung Manh La** (M'09) received the Ph.D. degree in electrical engineering from Oklahoma State University, Stillwater, OK, USA, and the B.S. and M.S. degrees in electrical engineering from the Thai Nguyen University of Technology, Thai Nguyen, Vietnam, in 2011, 2001, and 2003, respectively.

He was a Lecturer with the Department of Electrical Engineering, Thai Nguyen University of Technology, from 2001 to 2007. He is currently a Research Associate Scientist with the Center for Advanced Infrastructure and Transportation, Rutgers University, Piscataway, NJ, USA. He has been involved in research projects with the Federal Highway Administration, National Institute of Standards and Technology, U.S. Department of Transportation, Department of Defense, and National Science Foundation. He has authored more than 30 papers published in major journals, book chapters, and international conference proceedings. His current research interests include mobile robotic systems, mobile sensor networks, cooperative control-learning and sensing, and intelligent transportation systems.

Dr. La received two Best Paper Awards at the 2009 and 2010 Conferences on Theoretical and Applied Computer Science, and one Best Paper Presentation at the Network Control Session of the 2009 American Control Conference.



**Ronny Lim** received the M.S. degree in electrical and computer engineering from Oklahoma State University, Stillwater, OK, USA, and the B.S. degree from Pelita Harapan University, Tangerang, Indonesia, in 2011 and 2008, respectively.

He is currently an Application Developer with the Center for Advanced Infrastructure and Transportation, Rutgers University, Piscataway, NJ, USA. His current research interests include robotics and automation for civil infrastructure, intelligent transportation systems, and multiagent robotic control.



**Weihua Sheng** (SM'08) received the M.S. and B.S. degrees in electrical engineering from Zhejiang University, Hangzhou, China, in 1997, and the Ph.D. degree in electrical and computer engineering from Michigan State University, East Lansing, MI, USA, in 1994 and 2002, respectively.

He was a Research Engineer with the Research and Development Center, Huawei Technologies Company, Ltd., Shenzhen, China, from 1997 to 1998. From 2002 to 2006, he taught with the Electrical and Computer Engineering Department, Kettering University (formerly General Motor Institute), Flint Township, MI, USA. He is an Associate Professor with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. He has participated in organizing various IEEE international conferences and workshops in intelligent robots and systems. He holds one U.S. patent and has authored more than 130 papers in major journals and international conferences. His current research interests include wearable computing, mobile robotics, human-robot interaction, and intelligent transportation systems. His research is supported by the National Science Foundation, U.S. Department of Defense, Department of Defense Experimental Program to Stimulate Competitive Research, and Department of Transportation.

Dr. Sheng is currently an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He was a recipient of six Best Paper Awards from many international conferences.