Distributed Sensor Fusion for Scalar Field Mapping Using Mobile Sensor Networks

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

***Abstract*—In this paper, autonomous mobile sensor networks are deployed to measure a scalar field and build its map. We develop a novel method for multiple mobile sensor nodes to build this map using noisy sensor measurements. Our method consists of two parts. First, we develop a distributed sensor fusion algorithm by integrating two different distributed consensus filters to achieve cooperative sensing among sensor nodes. This fusion algorithm has two phases. In the first phase, the weighted average consensus filter is developed, which allows each sensor node to find an estimate of the value of the scalar field at each time step. In the second phase, the average consensus filter is used to allow each sensor node to find a confidence of the estimate at each time step. The final estimate of the value of the scalar field is iteratively updated during the movement of the mobile sensors via weighted average. Second, we develop the distributed flocking-control algorithm to drive the mobile sensors to form a network and track the virtual leader moving along the field when only a small subset of the mobile sensors know the information of the leader. Experimental results are provided to demonstrate our proposed algorithms.**

***Index Terms*—Cooperative sensing, flocking control, mobile sensor networks (MSNs), sensor fusion.**

I. Introduction

A. Motivation

MOBILE sensor networks (MSNs) [1] have broad applications, including target tracking, cooperative detection of toxic chemicals in contaminated environments, search and rescue operations after disasters, forest fire monitoring, etc. In recent years, missions that require the mapping of a scalar field have become prominent. Measuring and exploring an unknown field of interest have attracted much attention of environmental scientists and control engineers [2]–[8]. There are numerous applications, including environmental monitoring [9] and oil spill and toxic-chemical plume tracing [10], [11]. Because the scalar field is often distributed across a large area, we need many sensors to cover the field if the sensors are mounted at fixed locations. MSNs in which sensors can move together and take measurements along their motion trajectories are ideal candidate for such missions.

In order to create the map of a scalar field, the MSN should be able to achieve cooperative sensing among sensors in a distributed fashion. The development of a novel cooperative sensing algorithm based on distributed estimation and control algorithms for MSNs is very challenging. The estimation and the control have to be performed in each sensor node using only local information, while as a whole, the network should exhibit collective intelligence and achieve a global goal. In a resource-constrained multiagent system, the communication range and sensing range of each agent are small compared with the size of the surveillance region. Hence, agents cannot accomplish the mission without an effective flocking control and path planning strategy. Therefore, this paper aims to develop a cooperative sensing scheme to address both estimation and motion control problems to build the map of the scalar field. Our idea is to combine distributed estimation, distributed motion control, and path planning strategy to allow mobile sensors to obtain the estimates of the field during their movements.

B. Literature Review

Cooperative sensing in MSNs has been recently studied by researchers in control engineering [2], [6], [8], [12]–[20], and it can be utilized in target tracking, environmental mapping, monitoring, exploration, and coverage.

The early work on this technique can be found in [15] and [16], where the cooperative sensing algorithm for MSNs to estimate the state of dynamic targets is proposed. The localization and the tracking of dynamic targets are addressed. To achieve active sensing, the mobility of sensing agents is utilized to improve the sensing performance. However, the gradient controller for cooperative sensing is designed in a centralized way. To relax this limitation, the distributed gradient controller is proposed in [18], and it is designed by constructing a dynamic average consensus estimator and using a one-hop neighbor for communication, so that both formation control and cooperative sensing are integrated in order improve the sensing performance.

In addition, the developed cooperative sensing algorithms for target estimation and the cooperative sensing algorithms for source seeking and radiation mapping have been also developed [19]–[24]. The problem of source seeking is first addressed

Manuscript received October 7, 2011; revised May 4, 2012; accepted August 11, 2012. Date of publication October 9, 2012; date of current version April 16, 2013. This work was supported in part by the Ministry of Education and Training, Vietnamese Government, under Program MoET 322; by the Department of Defense, Army Research Office Defense University Research Instrumentation Program, USA, under Grant 55628-CS-RIP; and by the National Science Foundation, USA, under Grant NSF CISE/CNS MRI 0923238. This paper was recommended by Editor E. Santos, Jr.

H. M. La is with the Center for Advanced Infrastructure and Transportation, Rutgers University, Piscataway, NJ 08854 USA, and also with the Department of Electronics Engineering, Thai Nguyen University of Technology, Thai Nguyen, Vietnam (e-mail: hung.la11@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/TSMCB.2012.2215919

in [21], and then, it is thoroughly studied in [22]–[24] for the case when direct gradient information of the measured quantity is unavailable. Specifically, Pang and Farrell [22] address chemical plume source localization by constructing a source likelihood map based on Bayesian inference methods. Mesquita *et al.* [23] induces source-seeking behavior without direct gradient information by mimicking E. coli bacteria. Mayhew *et al.* [24] propose a hybrid control strategy to locate a radiation source utilizing only radiation intensity measurements. Moreover, cooperative sensing for radiation mapping is developed in [19] and [20]. The control algorithm takes into account sensing performance, as well as dynamics of the observed process; therefore, it can steer mobile sensors to locations at which they maximize the information content of the measurement data.

In addition, cooperative sensing algorithms have been developed for environmental estimation [4], sampling, and exploring [5], [6], [25]. The extensive survey of the current work in this area can be seen in [26]. In [4], an MSN is deployed in an environment of interest and takes measurements of a dynamic physical process modeled as a spatiotemporal random field. A distributed Kriged Kalman filter is developed to infer the random field and its gradient. In [25], underwater vehicles are deployed to measure temperature and currents. The vehicles communicate via an acoustic local area network and coordinate their motion in response to local sensing information and the evolving environments. This MSN aims to sample the environment adaptively in space and time. In [5], a class of underwater vehicles are used to obtain a sampling coverage over a large area. A cooperative control method is proposed to control vehicles to generate patterns on closed smooth curves. To further improve the cooperative sensing performance, both cooperative motion control and cooperative sensing are integrated based on cooperative Kalman filters [6] to control the shape of the sensor node formation in order to minimize the estimation error.

Other significant works in cooperative sensing for environmental modeling and coverage can be found in [2], [3], [8], and [27]. Cooperative sensing based on the gradient descent algorithms to obtain the optimal coverage is developed in [3]. For dynamic environment coverage, a control strategy based on the discrete Kalman filter is developed [2]. The approach relies on the Kalman filter to estimate the field and on the filter's prediction step to plan the vehicles' next move to maximize the estimation quality. In [27], an optimal filtering approach toward fusing local sensor data into a global model of the environment is developed. Their approach is based on the use of average consensus filters to distributively fuse the sensory data through the communication network. Along with the consensus filters, the control laws are developed for mobile sensors to maximize their sensory information relative to current uncertainties in the model.

Overall, to our best knowledge the existing works in the area of cooperative sensing using MSNs mostly focus on target(s) tracking, environmental estimation, sampling, modeling and coverage, sensor placement, source seeking, and radiation mapping. The problem of the scalar field estimation and mapping based on multiagent cooperative sensing has not been fully addressed.

In this paper, the problems of cooperative sensing and cooperative motion control are addressed. First, we develop a distributed sensor fusion algorithm by integrating two different distributed consensus filters to achieve cooperative sensing among sensor nodes. In this algorithm, each sensor node obtains measurements from itself and its neighboring sensor nodes. Each mobile sensor node will then iteratively update the estimate of the scalar field. Second, to allow the mobile sensors to move together and take measurements, we develop a distributed flocking-control algorithm to drive them to form a network and track a virtual leader. The trajectories of the virtual leader are planned, so that the MSN can cover the entire field.

In summary, the contributions of our paper are:

1) Development of a distributed sensor fusion algorithm to achieve cooperative sensing among sensor nodes to build the map of a scalar field.
2) Development of a distributed flocking-control algorithm for an MSN to navigate the scalar field for the mapping process.

The rest of this paper is organized as follows: The succeeding section presents the models of the scalar field and the measurement of each sensor node, as well as the problem formulation. Section III describes the distributed consensus filters and the distributed sensor fusion algorithm for building a map of the unknown scalar field. Section IV presents the flocking-control algorithm and the path planning strategy for complete coverage of the scalar field. Section V shows the simulation results. Finally, Section VI concludes this paper.

II. Scalar Field and Measurement Modeling and Problem Statement

In this section, we present the model of the scalar field, the model of the measurement of each sensor node, and the problem statement.

A. Model of the Scalar Field

We model the scalar field of interest as

$$F = \Theta\Phi^T. \tag{1}$$

Here, $\Theta = [\theta_1, \theta_2, \ldots, \theta_K]$, and $\Phi = [\phi_1, \phi_2, \ldots, \phi_K]$, where j is the index, and K is the total number of function distributions. We can rewrite (1) as

$$F = \sum_{j=1}^{K} \theta_j \phi_j. \tag{2}$$

Here, ϕ_j is a function representing the density distribution, and θ_j is the weight of the density distribution of function ϕ_j.

We can model function ϕ_j as a bivariate Gaussian distribution, i.e.,

$$\phi_j = \frac{1}{\sqrt{\det(C_j)(2\pi)^2}} \times e^{\frac{-1}{2}(x-\mu_x^j)C_j^{-1}(y-\mu_y^j)^T}, \quad j \in [1, 2, \ldots, K].$$

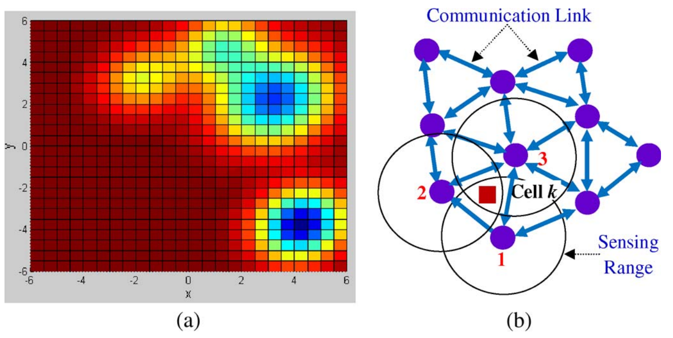


Fig. 1. (a) One example of the scalar field and it is partitioned into multiple cells. (b) Illustration of the measurement model using multiple mobile sensor nodes.

Here, $[\mu_x^j \ \mu_y^j]$ is the mean of the distribution of function ϕ_j, and C_j is the covariance matrix (positive definite) and is represented by

$$C_j = \begin{bmatrix} \left(\sigma_x^j\right)^2 & c_j^o \sigma_{xy}^j \\ c_j^0 \sigma_{xy}^j & \left(\sigma_y^j\right)^2 \end{bmatrix}$$

where c_j^0 is a correlation factor.

One example of the scalar field is shown in Fig. 1(a). In this field, we have four Gaussian distributions associated with different weights of Θ. Each Gaussian distribution can represent an oil leak or chemical leak, etc.

B. Measurement Model

We partition the scalar field F into a grid of C cells. Each sensor i makes an observation (measurement) of the scalar field at cell k ($k \in \{1, 2, \ldots, C\}$) at time step t, i.e.,

$$m_i^k(t) = O_i^k(t)\left[F^k(t) + n_i^k(t)\right]. \tag{3}$$

Here, $n_i^k(t)$ is the Gaussian noise with zero mean and variance $V_i^k(t)$ at time step t. We assume that n_i^k is the uncorrelated noise which satisfies

$$\text{Cov}\left(n_i^k(s), n_i^k(t)\right) = \begin{cases} V_i^k, & \text{if } s = t \\ 0, & \text{otherwise.} \end{cases}$$

Here, Cov is the covariance. $O_i^k(t)$ is the observability of the sensor node i at cell k at time step t, and it is defined as

$$O_i^k(t) = \begin{cases} 1, & \text{if } \left\|q_i(t) - q_c^k\right\| \leq r_i^s \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Here, $q_i \in R^2$ is the position of the sensor node i; $q_c^k \in R^2$ is the location of cell k. This definition implies that, if cell k is inside the sensing range r_i^s, of the sensor node i, then cell k can be measured or observed. Otherwise, the observability is zero.

Each mobile sensor node makes a measurement at cell k corresponding to its position. We assume that variance $V_i^k(t)$ is related to the distance between the sensor node i and cell k according to

$$V_i^k(t) = \begin{cases} \frac{\left\|q_i(t) - q_c^k\right\|^2 + c_v}{\left(r_i^s\right)^2}, & \text{if } \left\|q_i(t) - q_c^k\right\| \leq r_i^s \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Here, c_v is the small positive constant between 0 and 1. The reason of introducing c_v is to avoid variance $V_i^k(t)$ being zero when distance $\|q_i(t) - q_c^k\|$ is equal to zero. This means that, if the position of agent i is exactly at the location of cell k, then agent i makes the measurement with the smallest variance of noise but still greater than zero.

C. Problem Formulation

Given the measurements of the sensor node i and its neighbors at each cell of the scalar field F as modeled in (3) [see Fig. 1(b)], our goal is to build the map for the scalar field F modeled by (1) with a certain level of confidence.

Fig. 1(b) shows one example of multiple mobile sensor nodes measuring the value at cell k of the scalar field F. Since cell k is in the sensing range of sensor nodes 1, 2, and 3, it is measured by these sensors only. Each of these sensor nodes measures the value at cell k with its own confidence and exchanges the measurement to its neighbors. In general, cell k may be measured multiple times as long as it is inside the sensing range of the sensor node. Through the distributed consensus filter, the measurements of cell k are propagated over the network; hence, all sensor nodes can obtain the estimate of cell k. Our task is to build a cooperative sensing scheme to allow mobile sensors to build the map of the scalar field during their movements.

III. Distributed Sensor Fusion Algorithm

A. Overall Approach

In this section, we present a distributed sensor fusion algorithm to allow each sensor node to obtain an estimate of the value at each cell of the scalar field. Our algorithm has two phases. First, each sensor node finds an estimate of the value of the scalar field F at each cell at time step t. Second, each sensor node finds a final estimate of the value of the scalar field F at each cell over the time. To achieve these two phases, we develop two consensus filters. Consensus filter 1 is to obtain an estimate of the value of field F at each cell at time step t. Since each mobile sensor node may have its own measurement at each cell at time step t with its own weight (confidence), consensus filter 2 is used to obtain an agreement among these confidences.

At each time step t, each mobile sensor node needs to find an estimate of the value of each cell based on consensus filter 1 and find an overall confidence of this estimate based on consensus filter 2. This process can be called the *spatial estimation phase*. Then, during the movement of each sensor node, it will have multiple spatial estimates of each cell associated with the corresponding confidences. Hence, these spatial estimates are iteratively fused through a weighted average protocol, and this process can be called the *temporal estimation phase*.

B. Distributed Consensus Filters

Let us consider a dynamic graph G consisting of a vertex set $\vartheta = \{1, 2 \ldots, n\}$ and an edge set $E \subseteq \{(i, j) : i, j \in \vartheta, j \neq i\}$. In this graph, each vertex denotes a mobile sensor node, and each edge denotes the communication link between sensor nodes.

During the movement of the sensor nodes, the relative distance between them may change; hence, the neighbors of each sensor node also change. Therefore, we define a neighborhood set of the sensor node i at time step t as follows:

$$N_i(t)=\{j\in\vartheta : \|q_j-q_i\|\leq r, \vartheta=\{1,2,\ldots,n\}, j\neq i\}. \quad (6)$$

Here, $q_i \in R^2$ is the position of the sensor node i, and r is the communication (active) range and can be the same as or less than r^s.

1) Consensus Filter 1: Distributed consensus [28]–[33] is an important computational tool to achieve cooperative sensing. We consider distributed linear iterations of the following form:

$$x_i^k(l+1) = w_{ii}^k(t)x_i^k(l) + \sum_{j\in N_i(t)} w_{ij}^k(t)x_j^k(l). \quad (7)$$

Here, l is the iteration index in the running process of the consensus filter, and t is the time step when sensor i makes an observation/measurement at cell k. Specifically, at each time step t, sensor i runs the consensus filter (7) to find an agreement among all measurements of sensors by exchanging information with its neighbors only.

The initial condition for the state is given as $x_i^k(l=0) = m_i^k(t)$. Weight $w_{ii}^k(t)$ is the self-weight or the vertex weight of each sensor at cell k, and $w_{ij}^k(t)$ is the edge weight between sensor i and sensor j.

The problem here is to estimate the value of field F at each cell k at each time step t. Since the network may have more than one sensor node making the observation at cell k at time step t based on its own confidence (weight), the consensus should converge to the weighted average of all observations (measurements) at cell k from these sensor nodes in the network. This weighted average is the estimate of the value at cell k at time step t, and it is computed as

$$E^k(t) = \frac{\sum_{i=1}^n w_{ii}(t)m_i(t)}{\sum_{i=1}^n w_{ii}(t)}. \quad (8)$$

If (7) converges, we have $E_1^k(t) = E_2^k(t) = \ldots = E_n^k(t) = E^k(t)$; here, $E_i^k(t)$, with $i = 1,\ldots,n$, is the estimate of the field at cell k at time step t of the sensor node i. Therefore, our goal is to let

$$\lim_{l\to\infty}\left(x_i^k(l) - E^k(t)\right) \to 0. \quad (9)$$

We can write (9) in the following matrix form:

$$\lim_{l\to\infty} \mathbf{x}^k(l) = E^k(t)\mathbf{1}. \quad (10)$$

Here, $\mathbf{x}^k(l) = [x_1^k(l), x_2^k(l),\ldots,x_n^k(l)]_{n\times 1}^T$, and $\mathbf{1} = [1, 1, \ldots, 1]_{n\times 1}^T$.

We can also write (7) in the following matrix form:

$$\mathbf{x}^k(l+1) = \mathbf{w}^k(t)\mathbf{x}^k(l) \quad (11)$$

with the initial condition $\mathbf{x}^k(0) = \mathbf{m}^k(t)$; here, $\mathbf{m}^k(t) = [m_1^k(t), m_2^k(t),\ldots,m_n^k(t)]_{n\times 1}^T$.

To make (11) converge to $E^k(t)$, we need

$$\mathbf{w}^k(t) = \frac{1}{n}\mathbf{1}\mathbf{1}^T. \quad (12)$$

In order to achieve this, we need to ensure that the sum of all weights including the vertex and edge weights at each node is equal to 1 or

$$w_{ii}^k(t) + \sum_{j\in N_i(t)} w_{ij}^k(t) = 1. \quad (13)$$

To satisfy this, we can design the weights of the consensus filters as follows:

Weight design 1: From (13), the vertex weight at node i is obtained as

$$w_{ii}^k(t) = 1 - \sum_{j\in N_i(t)} w_{ij}^k(t). \quad (14)$$

Here, $w_{ij}^k(l)$ is defined as

$$w_{ij}^k(t) = \frac{c_1^w}{V_i^k(t) + V_j^k(t)}, \quad i\neq j, \quad j\in N_i(t). \quad (15)$$

Here, c_1^w is a designed factor. If none of the sensor nodes i and j observes cell k $(O_i^k(t) = O_j^k(t) = 0)$, then to avoid dividing by zero, the edge weight $w_{ij}^k(l)$ is set to zero.

Therefore, we have the following form of weight design:

$$w_{ij}^k(t) = \begin{cases} \frac{c_1^w}{V_i^k(t)+V_j^k(t)}, & \text{if } i\neq j, j\in N_i(t), \\ 1-\sum_{j\in N_i(t)} w_{ij}^k(t), & \text{if } i=j, \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

Now, we need to find c_1^w to satisfy (13). We know that $\min(V_i^k(t)) = \min(\|q_i(t) - q_c^k\|^2 + c_v/(r_i^s)^2) = c_v/(r_i^s)^2$ if $\|q_i(t) - q_c^k\| = 0$. Hence, we have

$$min\left(V_i^k(t)\right) + min\left(V_j^k(t)\right) = \begin{cases} \frac{2c_v}{(r^s)^2}, & \text{if } \left(r_i^s = r_j^s = r^s\right) \\ \frac{c_v}{\left(r_i^s\right)^2} + \frac{c_v}{\left(r_j^s\right)^2}, & \text{otherwise.} \end{cases} \quad (17)$$

To satisfy (13), we need

$$0 < \sum_{j\in N_i(t)} w_{ij}^k(t) < 1 \Rightarrow 0 < \sum_{j\in N_i(t)} \frac{c_1^w}{V_i^k(t) + V_j^k(t)} < 1$$

or

$$0 < c_1^w < \frac{V_i^k(t) + V_j^k(t)}{|N_i(t)|}. \quad (18)$$

Here, $|N_i(t)|$ is the number of neighbors of the sensor node i at time t, and from (17) and (18), we can select c_1^w as

$$\begin{cases} 0 < c_1^w < \frac{2c_v}{(r_i^s)^2 |N_i(t)|}, & \text{if } r_i^s = r_j^s = r^s \\ 0 < c_1^w < \frac{1}{|N_i(t)|}\left(\frac{c_v}{\left(r_i^s\right)^2} + \frac{c_v}{\left(r_j^s\right)^2}\right), & \text{otherwise.} \end{cases} \tag{19}$$

Weight design 2: From (13), by assigning the same value to all edge weights, we obtain

$$w_{ij}^k(t) = \frac{1 - w_{ii}^k(t)}{|N_i(t)|}. \tag{20}$$

Here, $w_{ii}^k(t)$ is defined as

$$w_{ii}^k(t) = \frac{c_2^w}{V_i^k(t)} \tag{21}$$

where c_2^w is a designed factor. If the sensor node i does not observe cell k ($O_i^k(t) = 0$), then the vertex weight $w_{ii}^k(t)$ is set to zero.

Therefore, we have the following weight design:

$$w_{ij}^k(t) = \begin{cases} \frac{c_2^w}{V_i^k(t)}, & \text{if } i = j \\ \frac{1 - w_{ii}^k(t)}{|N_i(t)|}, & \text{if } i \neq j, j \in N_i(t) \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

Now, we discuss how to select constant c_2^w. In order to satisfy (13), we need the following condition:

$$0 < \frac{c_2^w}{V_i^k(t)} < 1. \tag{23}$$

Since $\min(V_i^k(t)) = c_v/(r_i^s)^2$ when $\|q_i(t) - q_c^k\| = 0$, we have

$$0 < \frac{c_2^w}{\frac{c_v}{\left(r_i^s\right)^2}} < 1 \Rightarrow 0 < c_2^w < \frac{c_v}{\left(r_i^s\right)^2}. \tag{24}$$

2) Consensus Filter 2: Since each sensor node has its own confidence (weight) of the measurement at each cell at each time step t, we need to find an agreement among the confidences of sensor nodes. Consensus algorithm 2 is introduced to find the overall confidence from each time step t.

In this consensus algorithm, the input is the confidence of the measurement of each sensor node, and it is defined as the weight of the measurement in (14) or (21). The output of the consensus is the overall confidence, which is the estimated weight $W_i^k(t)$ of the weighted average protocol, as shown in (27) (28) (29).

Let $y_i^k(l = 0)$ be the confidence of the measurement of the value of the scalar field at cell k at each time step t for the sensor node i, or $y_i^k(l = 0) = w_{ii}^k(t)$. Let $y_j^k(l = 0)$ be the confidence of the measurement of the value of the scalar field at cell k at each time step t for the sensor node j with $j \in N_i(t)$, or $y_j^k(l = 0) = w_{jj}^k(t)$. Then, we have the following consensus filter:

$$y_i^k(l+1) = M_{ii}^k(t) y_i^k(l) + \sum_{j \in N_i(t)} M_{ij}^k(t) y_j^k(l). \tag{25}$$

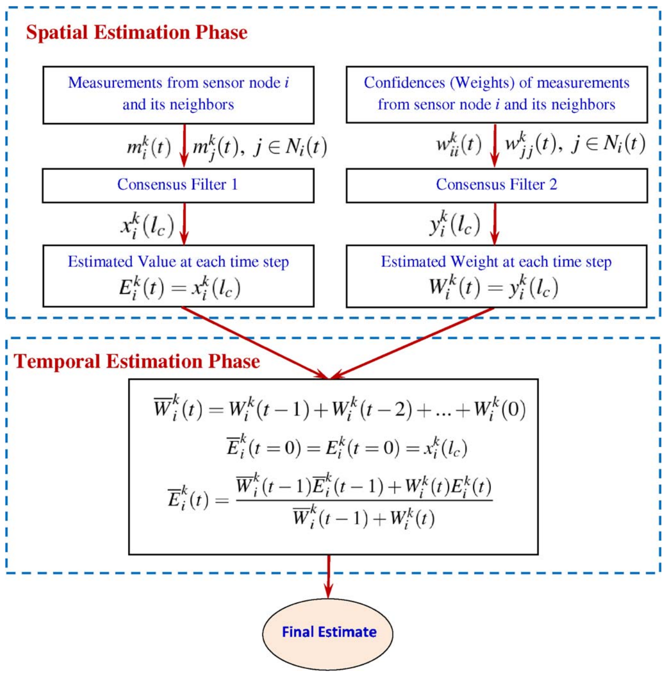

Spatial Estimation Phase
Measurements from sensor node i and its neighbors
Confidences (Weights) of measurements from sensor node i and its neighbors
$m_i^k(t)$, $m_j^k(t)$, $j \in N_i(t)$
$w_{ii}^k(t)$, $w_{jj}^k(t)$, $j \in N_i(t)$
Consensus Filter 1
Consensus Filter 2
$x_i^k(l_c)$
$y_i^k(l_c)$
Estimated Value at each time step $E_i^k(t) = x_i^k(l_c)$
Estimated Weight at each time step $W_i^k(t) = y_i^k(l_c)$
Temporal Estimation Phase
$\overline{W}_i^k(t) = W_i^k(t-1) + W_i^k(t-2) + \ldots + W_i^k(0)$
$\overline{E}_i^k(t=0) = E_i^k(t=0) = x_i^k(l_c)$
$\overline{E}_i^k(t) = \frac{\overline{W}_i^k(t-1)\overline{E}_i^k(t-1) + W_i^k(t)E_i^k(t)}{\overline{W}_i^k(t-1) + W_i^k(t)}$
Final Estimate

Fig. 2. Framework of distributed sensor fusion algorithm.

Here, $M_{ij}^k(t)$ is the Metropolis weight [29] as

$$M_{ij}^k(t) = \begin{cases} \frac{1}{1 + \max(|N_i(t)|, |N_j(t)|)}, & \text{if } i \neq j, j \in N_i(t) \\ 1 - \sum_{j \in N_i(t)} M_{ij}^k(t), & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

C. Distributed Fusion Algorithm

Based on consensus filters 1 and 2, we can design a distributed sensor fusion algorithm to allow each sensor node to estimate the value of the scalar field at each cell using its own measurement and its neighbor's measurements. The overall design of such a distributed sensor fusion algorithm is shown in Fig. 2. In this algorithm, we have two phases running at the same time. In the spatial estimation phase, the measurements of the sensor node i and its neighbors at cell k at time step t are the inputs of consensus filter 1. Then, the output of this consensus filter is the estimate of the value of the scalar field F at cell k at time step t. In the temporal estimate phase, the confidences (weights) of the measurements of each sensor node and its neighbors at cell k at time step t are the inputs of consensus filter 2. Then, the output of this consensus filter is the estimate of the confidence of the measurement of the scalar field at cell k at time step t. During the movement of the sensor, each sensor node obtains a sequence of estimates of the value at cell k with associated confidence. Hence, the final estimate is iteratively updated based on these spatial estimates via the weighted average protocol. The details of the algorithm are shown in Algorithm 1. Note that the motion of each sensor node is controlled based on the flocking-control algorithm discussed in the succeeding section.

```
Algorithm 1: Distributed Sensor Fusion Algorithm
Input: weight $w_{ii}^k(t)$ and the measurement of the sensor node $i$ and its neighbors to cell $k$, $m_i^k(t)$, and $m_j^k(t)$.
Output: the final estimate at each cell $k$, $\overline{E}_i^k(t)$.
for each time step $t$ do
    for each sensor node $i$ do
        Step1: Make a measurement (observation) $m_i^k(t)$ to cell $k$ if $\|q_i(t) - q_c^k\| \leq r_i^s$.
        The sensor node $i$ obtains the measurements of cell $k$ from its neighbors and itself.
        for each iteration $l$ do
            The sensor node $i$ runs consensus (7) with the weight defined in (16) or (22):
            $x_i^k(l=0) = m_i^k(t)$,
            $x_i^k(l+1) = w_{ii}^k(t)x_i^k(l) + \sum_{j \in N_i(t)} w_{ij}^k(t)x_j^k(l)$.
            The sensor node $i$ runs consensus (25) with the weight defined in (26):
            $y_i^k(l=0) = w_{ii}^k(t)$,
            $y_i^k(l+1) = M_{ii}^k(t)y_i^k(l) + \sum_{j \in N_i(t)} M_{ij}^k(t)y_j^k(l)$.
        end
        Step2: Obtain the estimate of cell $k$ after running the consensus.
        Let $l_c$ be an iteration that both consensus filters converge, then we have:
        $E_i^k(t) = x_i^k(l_c)$; $W_i^k(t) = y_i^k(l_c)$.
        Step3: Update process to find the final estimate of the value of the scalar field at each cell $k$.
        -Update weight (confidence), i.e.,
        $\overline{W}_i^k(t) = W_i^k(t-1) + W_i^k(t-2) + \cdots + W_i^k(0)$.    (27)
        -Update the final estimate based on the weighted average protocol, i.e.,
        $\overline{E}_i^k(t=0) = E_i^k(t=0) = x_i^k(l_c)$    (28)
        $\overline{E}_i^k(t) = \frac{\overline{W}_i^k(t-1)\overline{E}_i^k(t-1) + W_i^k(t)E_i^k(t)}{\overline{W}_i^k(t-1) + W_i^k(t)}$.    (29)
    end
end
```

IV. Flocking-Control Algorithm and Path Planning Strategy

In this section, we first present the distributed flocking-control algorithm, which can steer the MSN to track a virtual leader that moves in the desired paths. These desired paths are generated based on the path planning strategy, which ensures that the MSN can cover the entire scalar field.

A main issue for multiple mobile sensors to move together is that these sensors need to avoid colliding with each other, which requires the use of cooperative control methods [34]–[36]. One of these methods is the flocking control [35], which is inspired by the flocking or schooling phenomenon in which a number of mobile agents move together and interact with each other locally while ensuring no collision, velocity matching, and flock centering [37]. In the nature, fish, birds, ants, and bees, etc. demonstrate the phenomenon of flocking. In natural flocking, usually, only few agents in a group have the information of the target, such as the knowledge about the location of a food source or the migration route [38], [39]. However, they can still flock together in a group based on local information. Inspired by this natural phenomenon, a flocking-control algorithm is designed to coordinate the motion of multiple mobile sensors. Based on our algorithm, all mobile sensors can form a network and track the target (virtual leader) even only very few of them know the information of the leader. In this algorithm, we adopt a target navigation term in order to reduce the large tracking force at the initial tracking time, so that the mobile sensors can form a network and track the leader. We use a damping force term to reduce the tracking overshoot.

A. Flocking-Control Algorithm

We consider n mobile sensor nodes moving in a 2-D Euclidean space. The dynamic equations of each sensor node are described as

$$\begin{cases} \dot{q}_i = p_i \\ \dot{p}_i = u_i, \quad i = 1, 2, \ldots, n. \end{cases} \tag{30}$$

Here, $q_i \in R^2$ and $p_i \in R^2$ are the position and the velocity of the sensor node i, respectively, and u_i is the control input of the sensor node i.

The geometry of flocks is modeled by an α-lattice [35] that meets the following condition:

$$\|q_j - q_i\| = d, \quad j \in N_i(t). \tag{31}$$

Here, d is a positive constant indicating the distance between the sensor node i and its neighbor j. However, at the singular configuration $(q_i = q_j)$, the collective potential used to construct the geometry of flocks is not differentiable. Therefore, the set of algebraic constraints in (31) is rewritten in terms of σ-norm [35] as follows:

$$\|q_j - q_i\|_\sigma = d^\alpha, \quad j \in N_i(t). \tag{32}$$

Here, constraint $d^\alpha = \|d\|_\sigma$ with $d = r/k_c$, where k_c is the scaling factor. The σ-norm $\|.\|_\sigma$ of a vector is map $R^m \Longrightarrow R_+$ defined as $\|z\|_\sigma = 1/\epsilon[\sqrt{1+\epsilon\|z\|^2} - 1]$ with $\epsilon > 0$. Unlike the Euclidean norm $\|z\|$, which is not differentiable at $z = 0$, the σ-norm $\|z\|_\sigma$ is differentiable everywhere. This property allows constructing a smooth collective potential function for sensor nodes.

To allow mobile sensors to move together without collision and track a moving target (virtual leader) while only some of the sensors know the location and the velocity of the leader, we propose the distributed flocking-control algorithm as follows:

$$u_i = f_i^\alpha + f_i^t + f_i^{\mathrm{dam}}. \tag{33}$$

Here, f_i^α is the formation control term, f_i^t is the navigation term, and f_i^{dam} is the damping term. The detail of each term is provided below.

1) Formation Control Term: To achieve three basic flocking rules (flock centering, collision avoidance, and velocity matching), function f_i^α, which consists of a gradient-based component and a consensus component, is used to regulate the artificial potential forces (repulsive or attractive forces) and the velocity among agents. This function is designed as [35]

$$f_i^\alpha = c_1^\alpha \sum_{j\in N_i} \phi_\alpha\left(\|q_j - q_i\|_\sigma\right) n_{ij} + c_2^\alpha \sum_{j\in N_i} a_{ij}(q)(p_j - p_i) \quad (34)$$

where c_1^α and c_2^α are positive constants, $\phi_\alpha(z)$ is the action function, n_{ij} is the vector along the line connecting q_i to q_j, and $[a_{ij}(q)]$ is the adjacency matrix. For more details on these terms, please refer to [35].

Algorithm 2: Design of the Navigation Term

```
for each informed agent $j$, $j \in N_I$ do
    if $\|q_i^{\inf}(t) - q_t(t)\| > K_1\|q_i^{\inf}(0) - q_t(0)\|$ then
        $$f_j^t = -\frac{K_2}{\|q_i^{\inf}(t) - q_t(t)\|}\left(q_j^{\inf} - q_t\right) - \frac{K_3}{\|q_i^{\inf}(t) - q_t(t)\|}\left(p_j^{\inf} - p_t\right).$$
        Here, $0.9 < K_1 < 1$, $K_2 > 0$, and $K_3 > 0$,
    else
        $$f_i^t = -c_1^t\left(q_j^{\inf} - q_t\right) - c_2^t\left(p_j^{\inf} - p_t\right). \quad (35)$$
    end
end
```

Algorithm 3: Design of the Damping Force Term

```
for each informed agent $j$, $j \in N_I$ do
    if $\|q_i^{\inf}(t) - q_t(t)\| < K_4 r$ then
        $$f_j^{\mathrm{dam}} = -K_{\mathrm{dam}} p_j^{\inf}. \quad (36)$$
        Here, $0 < K_4 < 1$ and $K_{\mathrm{dam}} > 0$,
    else
        $$f_j^{\mathrm{dam}} = 0. \quad (37)$$
    end
end
```

2) Navigation Term: First, let us define N_I as a subset of informed agents and N_{UI} as a subset of uninformed agents with $|N_I| \ll |N_{UI}|$. Hence, we have $N_I \cup N_{UI} = N$; here, N is the set of all agents (uninformed and informed agents), or $N = \{1, 2, \ldots, n\}$.

The navigation term allows the agents to stay together. The main idea behind this term is that, if we let the informed agents keep strong cohesion to uninformed agents at the initial time of the target-tracking process, the connectivity can be maintained. In order to do this, we have to reduce the initial momentum of the attractive force to the target for the informed agents. This means that we should have small attractive force at the initial time when the distance between the informed agent and the target is large. Based on this analysis, we design the navigation term as shown in Algorithm 2. In this algorithm, constant K_1 chosen between 0.9 and 1 is used to ensure that a small attractive force is applied at the initial time of the target-tracking process. Weights $K_2/\|q_i^{\inf}(t) - q_t(t)\|$ and $K_3/\|q_i^{\inf}(t) - q_t(t)\|$ are designed, so that the attractive force is small enough at the initial time, and then, it becomes bigger when distance $\|q_i^{\inf}(t) - q_t(t)\|$ decreases. Here, K_2 and K_3 are positive constants. If these constants are large, the sensor node i can reach the target faster.

3) Damping Force Term: Since only the informed agents N_I have the information of the target, the damping force can be only applied to these agents. The idea behind this damping force is to reduce the tracking overshoot when the informed agents are close to the target. That is, the damping force for the informed agents is only effective if the distance between the informed agent and the target is less than a certain threshold. This threshold is designed based on the active range r.

Algorithm 4: Decentralized Flocking-Control Algorithm with a Minority of Informed Agents

```
Input: Position and velocity of each agent $(q_i, p_i)$; position and velocity of the target $(q_t, p_t)$ for the informed agents $(N_I)$.
Output: Control law for each agent $u_i$
for each agent $i$ do
    Compute
    $$f_i^\alpha = \sum_{j\in N_i} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + \sum_{j\in N_i} a_{ij}(q)(p_j - p_i) \quad (38)$$
end
for each informed agent $j$, $j \in N_I$ do
    if $\|q_i^{\inf}(t) - q_t(t)\| > K_1\|q_i^{\inf}(0) - q_t(0)\|$, then
        $$f_j^t = -\frac{K_2}{\|q_i^{\inf}(t) - q_t(t)\|}\left(q_j^{\inf} - q_t\right) - \frac{K_3}{\|q_i^{\inf}(t) - q_t(t)\|}\left(p_j^{\inf} - p_t\right)$$
        $(0.9 < K_1 < 1, K_2 > 0$ and $K_3 > 0)$
    else
        $$f_i^t = -c_1^t\left(q_j^{\inf} - q_t\right) - c_2^t\left(p_j^{\inf} - p_t\right) \quad (39)$$
    end
    if $\|q_i^{\inf}(t) - q_t(t)\| < K_4 r$ then
        $$f_j^{\mathrm{dam}} = -K_{\mathrm{dam}} p_j^{\inf} \quad (40)$$
        $(0 < K_4 < 1$ and $K_{\mathrm{dam}} > 0)$
    else
        $$f_j^{\mathrm{dam}} = 0 \quad (41)$$
    end
end
for each uninformed agent $k$, $k \in N_{UI}$ do
    $f_k^{\mathrm{dam}} = 0$ and $f_k^t = 0$
end
Update the control law for each agent $i$
    $$u_i = f_i^\alpha + f_i^{\mathrm{dam}} + f_i^t. \quad (42)$$
```

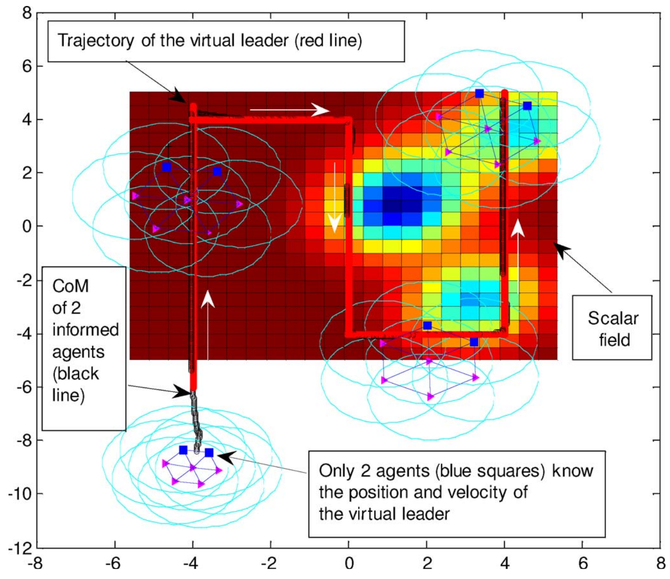


Fig. 3. Seven mobile sensor nodes flock together and cover the scalar field. (Red) The motion path is generated by the leader, and (black/darker color) the CoM of the network tracks the leader with small overshoots at sharp turning points of the path.

This means that, when the target is inside the active range of the informed agent j, the damping force f_j^{dam} is applied; otherwise, $f_j^{\text{dam}} = 0$. In order to do that, constant K_4 is used $(0 < K_4 < 1)$. When the damping force f_j^{dam} is applied, the informed agent j will reduce its speed gradually to approach the target. Hence, the tracking overshoot is reduced. Overall, the damping force is designed in Algorithm 3. Here, K_{dam} is the gain of the damping force. If K_{dam} is large, the network moves to the target slowly.

Finally, the whole decentralized flocking-control algorithm is proposed in Algorithm 4.

B. Path Planning Strategy

Based on the flocking-control Algorithm 4, all mobile sensor nodes can form a lattice formation, and the center of mass (CoM) of the informed agents (sensors) as defined in (43) can track leader (q_t, p_t), i.e.,

$$\begin{cases} \overline{q} = \frac{1}{|N_I|} \sum_{i=1}^{|N_I|} q_i^{\text{inf}} \\ \overline{p} = \frac{1}{|N_I|} \sum_{i=1}^{|N_I|} p_i^{\text{inf}}. \end{cases} \tag{43}$$

Here, $|N_I|$ is the number of informed agents.

The results of the flocking control can be shown in Figs. 3 and 4. Fig. 3 shows the snapshots of seven mobile sensors forming a network and tracking the leader moving in the red line, while only two of them know the location and velocity of the leader. Fig. 4 shows the tracking error between the position of the virtual leader (q_t) and the average of the position of the two informed agents $(\text{mean}(q^{\text{inf}}))$. We can see that the tracking performance has a small offset distance between the virtual leader and the informed agents. At the sharp turning points of the path of the virtual leader, the tracking performance has bigger errors.

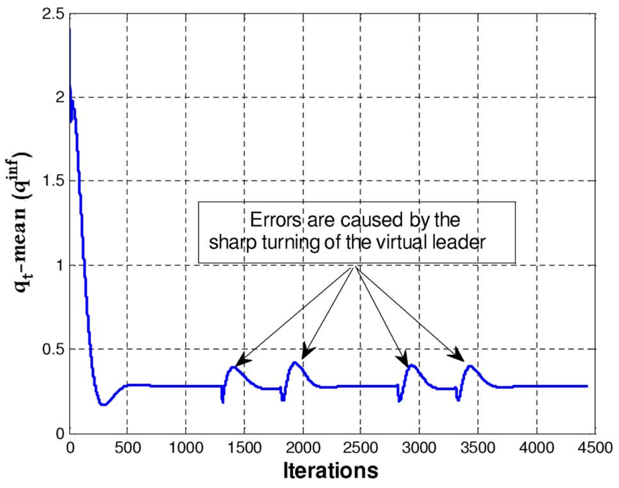


Fig. 4. Tracking error between the position of the virtual leader (q_t) and the average of the position of the two informed agents $(\text{mean}(q^{\text{inf}}))$.

Since the network can track the leader, to allow the network to cover the entire scalar field, we only need to design the path of the leader, so that the field is fully covered. We assume that the leader knows the total number of sensor nodes in the network. Then, based on the distance between sensor nodes (d^α), the leader can compute the size of the network. Then, our multirobot path planning problem becomes a single robot path planning problem. There are some typical algorithms of motion planning for a mobile robot to have complete coverage of the field of interest, such as *boustrophedron* motion or *wall-following* motion [40]. In this paper, we plan the leader motion using a typical *boustrophedron* motion.

V. Simulation and Experimental Results

In this section, we test consensus filters 1 and 2, and flocking-control Algorithm 4. Then, we use the proposed distributed sensor fusion Algorithm 1, along with flocking-control Algorithm 4, to build the map of a scalar field.

A. Test of Consensus Filters 1 and 2

Consensus filter 1 is tested for the case of a single cell $k = 1$. We randomly generate a connected network of 10 nodes, as shown in Fig. 5(a). The cell is located at the center of the network [see the red square in Fig. 5(a)]. The ground truth of the measurement at this location is 50. In this case, the location of the measurement is inside the sensing range of all nodes; hence, all nodes can make its own measurement at this location.

Each node makes a measurement as

$$m_i^1 = F^1 + n_i^1. \tag{44}$$

Here, $F^1 = 50$, and n_i^1 is the Gaussian noise, $N(0, V_i^1)$, with $V_i^1 = \|q_i - \overline{q}\|^2 + c_v/(r_i^s)^2$, $c_v = 0.01$, $r_1^s = r_2^s = \ldots = r_{10}^s = 1.6$, and $\overline{q} = 1/10 \sum_{i=1}^{10} q_i$. The initial condition for running consensus filter 1 is $x_i^1(l = 0) = m_i^1$.

The results of the convergence of consensus filter 1 associated with two different weights, i.e., *weight design 1* defined in (16) and *weight design 2* defined in (22), respectively, are presented in Fig. 5. In Fig. 5(a), to compare the speed of

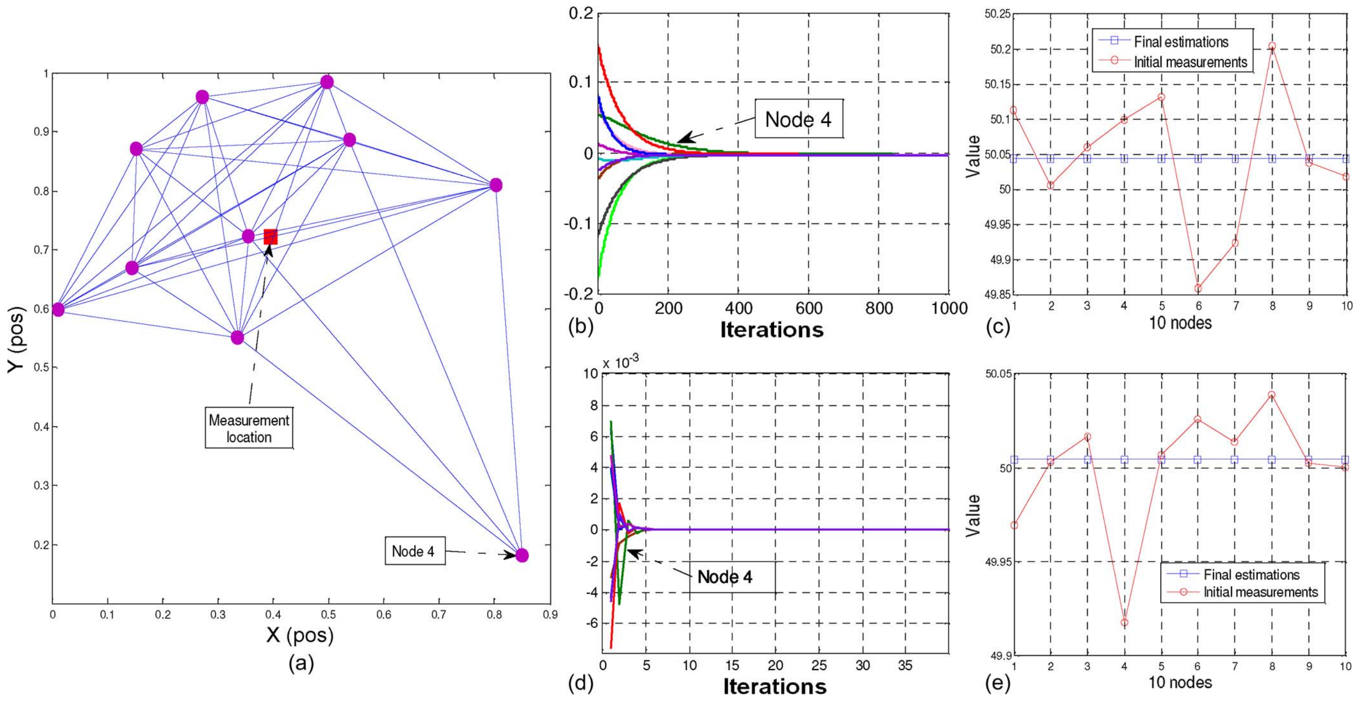


Fig. 5. (a) Ten nodes estimate the value at (red/black square) cell k. (b) and (c) Result of convergence of ten nodes and agreement of the ten nodes when applying the consensus filter 1 with *weight design 1* in (16). (d) and (e) Result of convergence of ten nodes and agreement of the ten nodes when applying consensus filter 1 with *weight design 2* in (22).

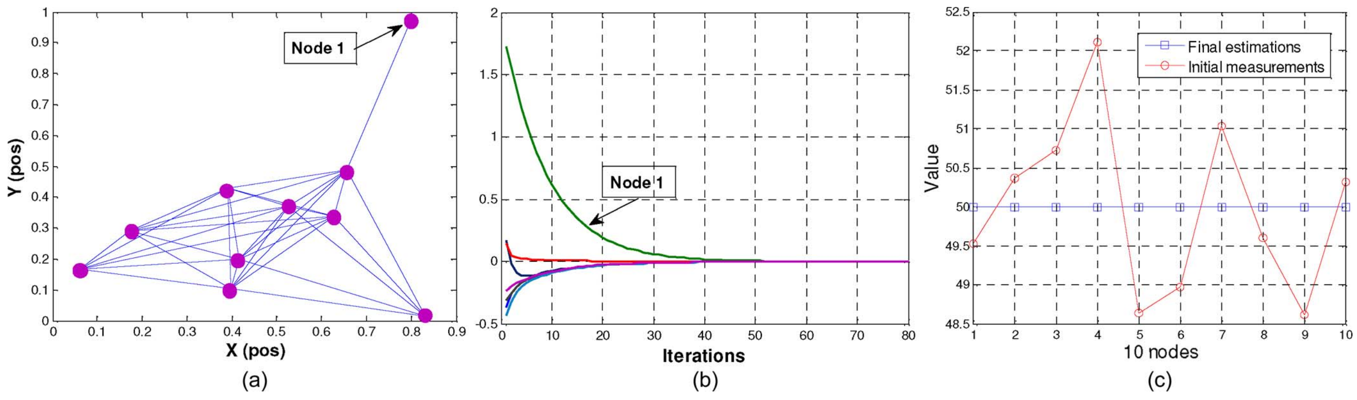


Fig. 6. (a) Distribution of ten nodes. (b) and (c) Result of convergence of ten nodes and agreement of the ten nodes when applying consensus filter 2 in (25) with the Metropolis weight (26).

the convergence of $(x_i^1(l) - E^1)$ among nodes, we generate a connected network with ten nodes in which we let the node 4 have only four neighbors, while other nodes have more than or equal to seven neighbors. From Fig. 5(b) and (d) we can see that $(x_i^1(l) - E^1)$, converge to zero after 300 iterations for *Weight Design 1* and 5 iterations for *Weight Design 2*. Therefore, it is better to use *Weight Design 2* since it can converge faster. We can also see that node 4 converges slower than the other nodes; this is because it has less neighbors. Additionally, to clearly see the convergence, we show the result of the agreement among nodes in Fig. 5(c) and (e).

For testing consensus filter 2, we let each sensor make its own measurement as with $F^1 = 50$, and n_i^1 is the Gaussian noise, $N(0,1)$. The initial condition for running consensus filter 2 is $y_i^1(l=0) = m_i^1$.

The results of the convergence of consensus filter 2 are presented in Fig. 6. Fig. 6(b) shows the convergence of $(y_i^1(l) - 1/10\sum_{I=1}^{10} y_i^1(0))$, and we can see that they converge to zero after 40 iterations. Fig. 6(c) shows the agreement among ten nodes, and all nodes in the network can agree on the same average value $(1/10\sum_{i=1}^{10} y_i^1(0))$. We also see that node 1, which has less neighbors than others, converges slower.

B. Experimental and Simulation Results of the Flocking-Control Algorithm 4

In this section, we test our proposed flocking-control Algorithm 4. First, we test our algorithm with seven physical robots. Then, to show the effectiveness and the scalability of our algorithm, we test it with 50 robots in simulation.

In the experiment, we use seven Rovio robots [41] that have omnidirectional motion capability. Basically, these robots can freely move in six directions. The dynamic model of the Rovio robot can be approximated by (30). 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

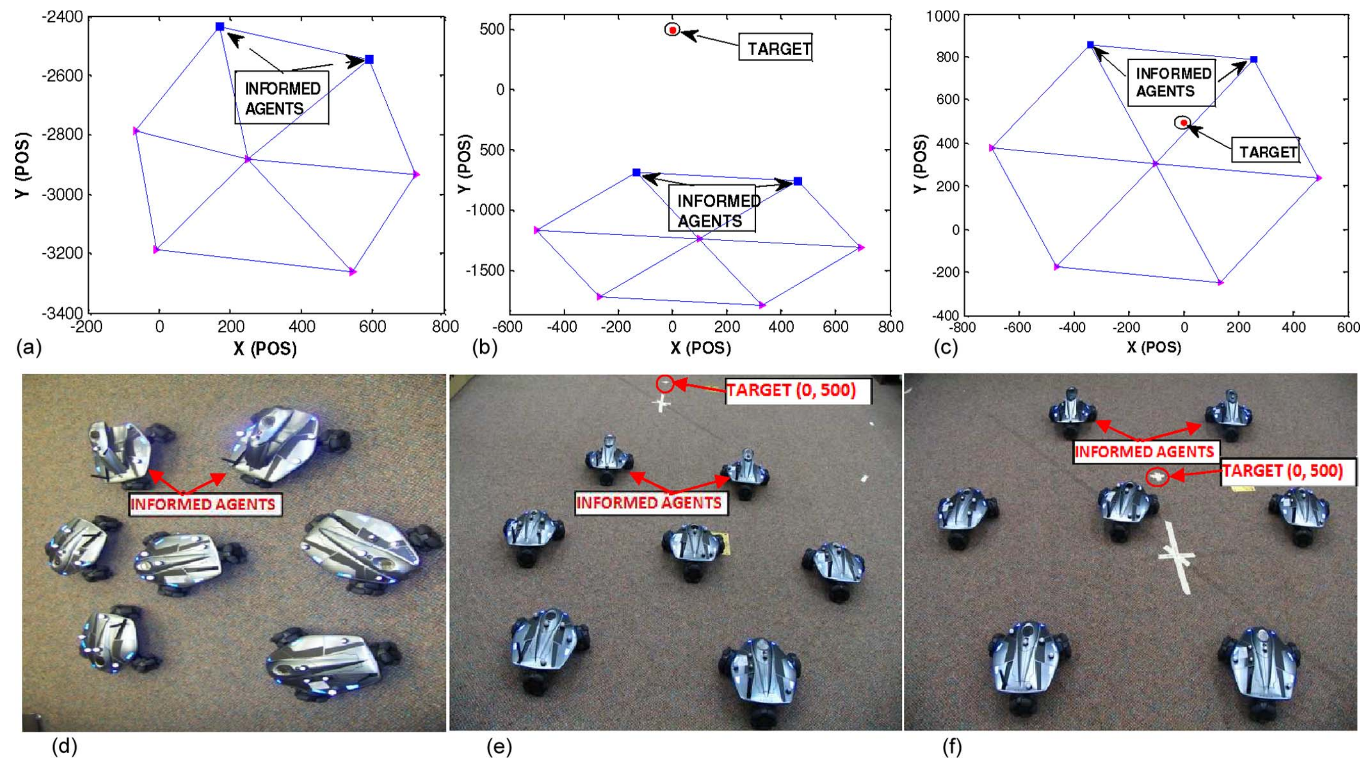


Fig. 7. Snapshots of seven Rovio robots flocking together when applying our proposed flocking-control Algorithm 4.

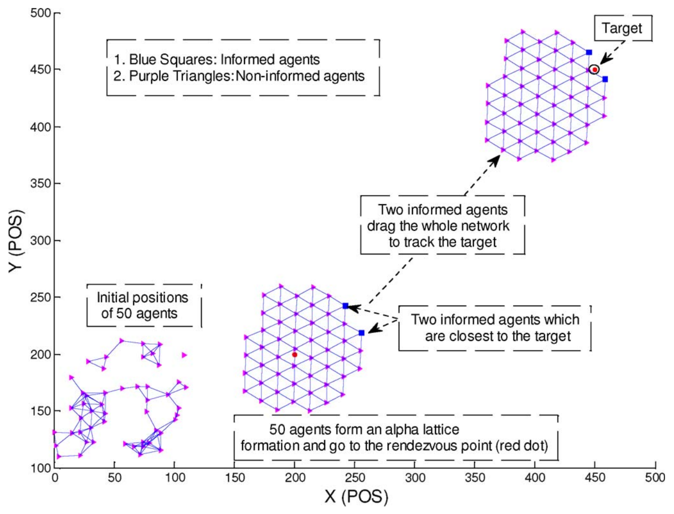


Fig. 8. Snapshots of 50 robots flocking together (simulation) with two of them knowing the information of the target.

velocity) of its neighbors. Hence, we use a VICON motion capture system [42] to track objects. Initially, the seven Rovio robots are randomly deployed. Then, two robots, which are closest to the target, are selected to be the informed agents [two robots have cameras facing up as shown in snapshot in Fig. 7(d)]. Based on the flocking-control Algorithm 4, all robots can flock together and track the target.

In simulation, we test our proposed Algorithm 4 with 50 robots. First, these robots form a network and go to the rendezvous point (red square), as shown in Fig. 8. After that, we let two robots (blue squares), which are closest to the target, know the position and velocity of the target. By observing Fig. 8, we can see that the two informed robots can drag all 48 other robots (purple triangles) to flock together. More simulation and experiment results, and parameters used to run the algorithm are referred to [43].

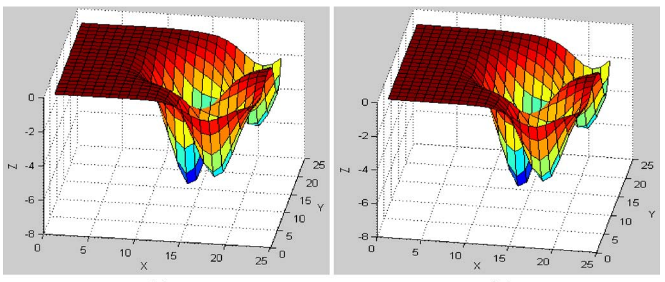


Fig. 9. (a) Original map of the scalar field F. (b) Built map of the scalar field F using Algorithm 1.

C. Simulation of Scalar Field Mapping

In this section, we use the proposed distributed sensor fusion Algorithm 1 and the proposed flocking-control algorithm with a minority of informed agents, i.e., Algorithm 4, to build the map of a scalar field. In the flocking-control algorithm, only a few sensor nodes closest to the virtual leader know its position and velocity. Based on our algorithm, all mobile sensor nodes can flock together and form a network of lattice formation. Our flocking-control algorithm allows the MSN to maintain the connectivity and reduce the tracking overshoot (see Figs. 3 and 4).

To model the environment, four multiple variate Gaussian distributions $(K = 4)$ with $\Theta = [20\ 50\ 35\ 40]$ are used, and each one is represented as

$$\phi_1 = \frac{1}{\sqrt{\det(C_1)(2\pi)^2}} e^{\frac{-1}{2}(x-2)C_1^{-1}(y-2)^T}$$

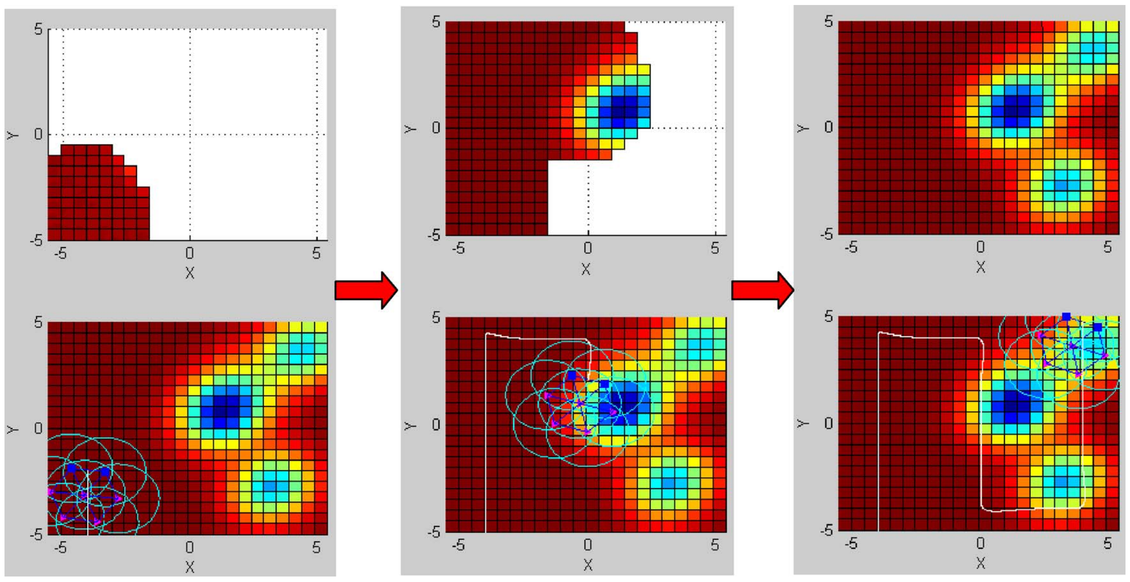


Fig. 10. Snapshots of multiple mobile sensors flocking together and building the map of the scalar field. In these snapshots, (blue squares) only two mobile sensors have information of the virtual leader. (White line) Trajectory of the center of position of two informed mobile sensors.

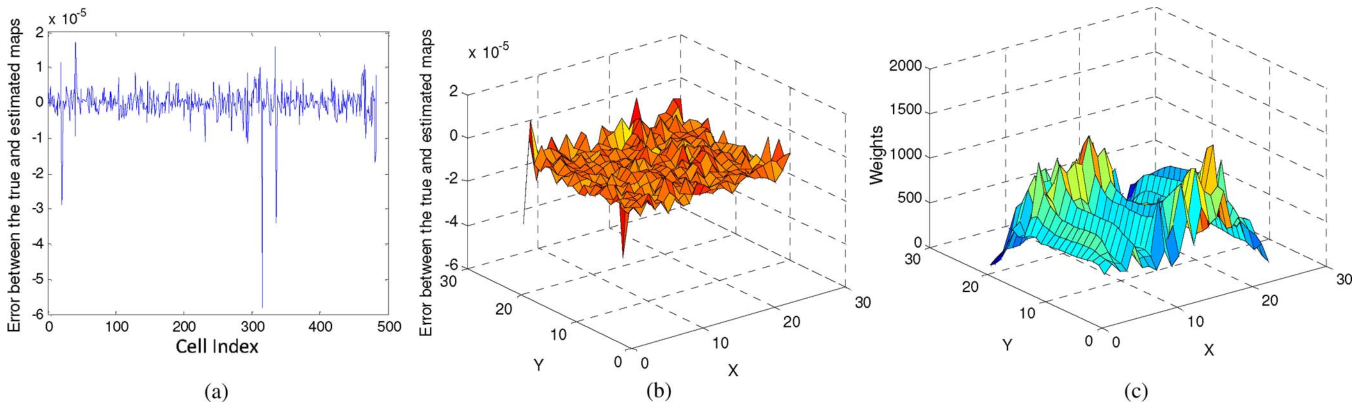


Fig. 11. (a) Error between the built and true maps for all cells in one dimension. (b) Error between the built and true maps for all cells in three dimensions. (c) 3-D confidence map.

where $C_1 = \begin{bmatrix} 2.25 & 0.2999 \\ 0.2999 & 2.25 \end{bmatrix}$, with the correlation factor $c_1^0 = 0.1333$

$$\phi_2 = \frac{1}{\sqrt{\det(C_2)(2\pi)^2}} e^{\frac{-1}{2}(x-1)C_2^{-1}(y-.5)^T}$$

where $C_2 = \begin{bmatrix} 1.25 & 0.1666 \\ 0.1666 & 1.25 \end{bmatrix}$, with the correlation factor $c_2^0 = c_1^0$

$$\phi_3 = \frac{1}{\sqrt{\det(C_3)(2\pi)^2}} e^{\frac{-1}{2}(x-4.3)C_3^{-1}(y-3.5)^T}$$

where $C_3 = C_2$, with the correlation factor $c_3^0 = c_2^0$, and

$$\phi_4 = \frac{1}{\sqrt{\det(C_4)(2\pi)^2}} e^{\frac{-1}{2}(x-3)C_4^{-1}(y+3)^T}$$

where $C_4 = C_3$, with the correlation factor $c_4^0 = c_3^0$.

Field F is partitioned into a grid of cells. The result of the built map of the scalar field is shown in Fig. 9. The snapshots of multiple sensor nodes forming a flock and building the map of the unknown scalar field are shown in Fig. 10. In this figure, we can see that only two mobile sensors (blue squares) have information of the virtual leader (q_t, p_t), but they can drag the whole network to track the virtual leader while maintaining the network connectivity. The errors between the built and original maps in one and three dimensions are shown in Fig. 11(a), (b), respectively. The final confidence of the estimate at each cell of field F is shown in Fig. 11(c). The confidence map represents the accuracy of the estimate of the field. The higher confidence, the better accuracy of the estimate. The cells near the border of the field have less measurements compared with the ones inside the field. Therefore, these border cells have lower accuracy [see Fig. 11(c)].

To see the effectiveness of the proposed method, three algorithms, i.e., Algorithm 1 with the weighted average update protocol, Algorithm 1 with the normal average update protocol, and the centralized fusion algorithm, are compared in terms of map error. We see that the map error in Algorithm 1 with the weighted average update protocol is similar to the one using the centralized fusion algorithm but slightly smaller than the one using Algorithm 1 with the normal average update protocol.

VI. Conclusion and Future Work

This paper has presented a cooperative sensing algorithm for MSNs that form a flock to build the map of an unknown scalar field. The developed flocking-control algorithm can drive mobile sensors to cover the whole field and take measurements. The proposed distributed sensor fusion algorithm consists of two different distributed consensus filters, which can find an agreement on the estimates and an agreement on the confidences among sensor nodes. Each sensor node cooperates with neighboring sensors to estimate the value of the field. The final estimates of the values of the scalar field are updated online based on the weighted average protocol. Simulation results validate for both distributed consensus filters and the distributed sensor fusion algorithm. In the future, we plan to implement this algorithm on real sensor nodes, and we will investigate how the formation of the network and the motion planning affect to the sensing quality.

Acknowledgment

The authors would like to thank R. Lim for helping in part to implement the flocking-control algorithm on Rovio robots.

References

[1] S. He, J. Chen, Y. Sun, D. K. Y. Yau, and N. K. Yip, "On optimal information capture by energy-constrained mobile sensors," *IEEE Trans. Veh. Technol.*, vol. 59, no. 5, pp. 2472–2484, Jun. 2010.

[2] I. I. Hussein, "A Kalman filter-based control strategy for dynamic coverage control," in *Proc. Amer. Control Conf.*, 2007, pp. 3271–3276.

[3] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[4] J. Cortes, "Distributed Kriged Kalman filter for spatial estimation," *IEEE Trans. Autom. control*, vol. 54, no. 12, pp. 2816–2827, Dec. 2009.

[5] F. Zhang, D. M. Fratantoni, D. Paley, J. Lund, and N. E. Leonard, "Control of coordinated patterns for ocean sampling," *Int. J. Control*, vol. 80, no. 7, pp. 1186–1199, 2007.

[6] F. Zhang and N. E. Leonard, "Cooperative filters and control for cooperative exploration," *IEEE Trans. Autom. Control*, vol. 55, no. 3, pp. 650–663, Mar. 2010.

[7] DOD/ONR MURI: Adaptive sampling and prediction project, 2011. [Online]. Available: http://www.princeton.edu/~dcsl/asap/

[8] J. Choi, S. Oh, and R. Horowitz, "Distributed learning and cooperative control for multi-agent systems," *Automatica*, vol. 45, no. 12, pp. 2802–2814, Dec. 2009.

[9] A. Dhariwal, G. S. Sukhatme, and A. A. G. Requicha, "Bacterium inspired robots for environmental monitoring," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 1436–1443.

[10] A. Lilienthal, H. Ulmer, H. Frohlich, A. Stutzle, F. Werner, and A. Zell, "Gas source declaration with a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 1430–1435.

[11] D. V. Zarzhitsky, D. F. Spears, and D. R. Thayer, "Experimental studies of swarm robotic chemical plume tracing using computational fluid dynamics simulations," *Int. J. Intell. Comput. Cybern.*, vol. 3, no. 4, pp. 1–44, Nov. 2010.

[12] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for target tracking," *IEEE Signal Process. Mag.*, vol. 19, no. 2, pp. 61–72, Mar. 2002.

[13] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. De Schutter, "Active sensing for robotics—A survey," Katholieke Univ. Leuven, Dept. Mech. Eng., Heverlee, Belgium, 2003.

[14] J. Spletzer and C. Taylor, "Dynamic sensor planning and control for optimally tracking targets," *Int. J. Robot. Res.*, vol. 22, no. 1, pp. 7–20, Jan. 2003.

[15] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray, "On a decentralized active sensing strategy using mobile sensor platforms in a network," in *Proc. 43rd IEEE Conf. Decision Control*, 2004, pp. 1914–1919.

[16] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray, "On a decentralized active sensing strategy using mobile sensor platforms in a network," in *Proc. IEEE International Conf. on Decision and Control*, 2004, pp. 1914–1919.

[17] S. Martinez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661–668, Apr. 2006.

[18] P. Yang, R. A. Freeman, and K. M. Lynch, "Distributed cooperative active sensing using consensus filters," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 405–410.

[19] R. A. Cortez and H. G. Tanner, "Radiation mapping using multiple robots," in *Proc. 2nd Int. Joint Top. Meeting Emergency Prep. Response Robot. Remote Syst.*, 2008, pp. 1–6.

[20] H. G. Tanner, R. A. Cortez, and R. Lumia, "Distributed robotic radiation mapping," in *Experimental Robotics—The Eleventh International Symposium*, vol. 54, *Springer Tracts in Advanced Robotics*, P. O. Khatib and V. Kumar, Eds., 2009, pp. 147–156, Springer-Verlag, Berlin, Germany.

[21] C. Zhang, D. Arnold, N. Ghods, A. Siranosian, and M. Krstic, "Source seeking with nonholonomic unicycle without position measurement—Part 1: Tuning of forward velocity," in *Proc. IEEE Conf. Decision Control*, 2006, pp. 3040–3045.

[22] S. Pang and J. A. Farrell, "Chemical plume source localization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 5, pp. 1068–1080, Oct. 2006.

[23] A. R. Mesquita, J. P. Hespanha, and K. Astrom, "Optimotaxis: A stochastic multi-agent on site optimization procedure," in *Hybrid Systems: Computation and Control*. Berlin, Germany: Springer-Verlag, 2008, pp. 358–371.

[24] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel, "Robust source-seeking hybrid controllers for autonomous vehicles," in *Proc. Amer. Control Conf.*, 2007, pp. 1185–1190.

[25] T. B. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb, "Autonomous oceanographic sampling networks," *Oceanography*, vol. 6, no. 3, pp. 86–94, 1993.

[26] R. Graham and J. Cortes, "Spatial statistics and distributed estimation by robotic sensor networks," in *Proc. Amer. Control Conf.*, 2010, pp. 2422–2427.

[27] K. M. Lynch, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Trans. Robot.*, vol. 24, no. 3, pp. 710–724, Jun. 2008.

[28] L. Xiao and S. Boy, "Fast linear iterations for distributed averaging," in *Proc. 42nd IEEE Conf. Decision Control*, 2003, pp. 4997–5002.

[29] L. Xiao, S. Boy, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. Int. Conf. Inf. Process. Sens. Netw.*, 2005, pp. 63–70.

[30] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time–delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[31] 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.

[32] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 355–369, Jan. 2009.

[33] 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.

[34] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[35] 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.

[36] R. M. Murray, "Recent research in cooperative control of multi–vehicle systems," *J. Dyn. Syst., Meas., Control*, vol. 129, no. 5, pp. 571–583, 2007.

[37] C. Reynolds, "Flocks, birds, and schools: A distributed behavioral model," in *Proc. ACM SIGGRAPH*, Anaheim, CA, 1987, vol. 21, no. 4, pp. 25–34.

[38] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision making in animal groups on the move," *Lett. Nat.*, vol. 433, no. 7025, pp. 513–516, Feb. 2005.

[39] S. G. Reebs, "Can a minority of informed leaders determine the foraging movements of a fish shoal?" *Animal Behav.*, vol. 59, no. 2, pp. 403–409, Feb. 2000.

[40] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Auton. Robots*, vol. 9, no. 3, pp. 247–253, Dec. 2000.

[41] Rovio Robot, 2011. [Online]. Available: http://www.wowwee.com/en/support/rovio

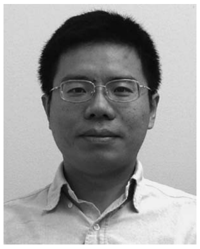
[42] VICON Motion Capture System, 2011. [Online]. Available: http://www.vicon.com/

[43] H. M. La, R. S. Lim, H. Chen, and W. Sheng, "Decentralized flocking control with minority of informed agents," in *Proc. IEEE ICIEA*, 2011, pp. 1851–1856.

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

He is a Research Associate with the Center for Advanced Infrastructure and Transportation (CAIT), Rutgers University, Piscataway, NJ, and a Lecturer with the Department of Electronics Engineering, Thai Nguyen University of Technology, Thai Nguyen. From 2011 to 2012, he was a Postdoctoral Associate with the CAIT. He has been actively involved in research projects with the Federal Highway Administration, Department of Transportation, the Department of Defense, and the National Science Foundation. His research includes mobile robots; mobile sensor networks; cooperative control, learning, and sensing; and intelligent transportation systems.

Dr. La was a recipient of two best paper awards in the 2009 and 2010 Conferences on Theoretical and Applied Computer Science, and one best presentation of session in the 2009 American Control Conference.



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

From 1997 to 1998, he was a Research Engineer with the Research and Development Center, Huawei Technologies Company, China. He is an Associate Professor with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater. From 2002 to 2006, he was with the Department of Electrical and Computer Engineering, Kettering University (formerly General Motor Institute), Flint, MI. He has participated in organizing several IEEE international conferences and workshops in the area of intelligent robots and systems. He is the author of more than 120 papers in major journals and international conferences. He is the holder of one U.S. patent. His current research interests include wearable computing, mobile robotics, human-robot interaction, distributed sensing and control, and intelligent transportation systems. His research is supported by the National Science Foundation, the Department of Defense, the Defense Experimental Program to Stimulate Competitive Research, the Department of Transportation, etc.

Dr. Sheng is currently an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.