Cooperative Sensing in Mobile Sensor Networks Based on Distributed Consensus

Hung Manh La and Weihua Sheng

School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA

ABSTRACT

Mobile sensor networks (MSNs) have wide applications such as military target detection and tracking, detection of toxic chemicals in contaminated environments, and search and rescues after disasters. etc. In many applications, a core problem is to conduct cooperative scalar field mapping (or searching) over a large area of interest.

Centralized solutions to the scalar field mapping may not fit for large mobile sensor network due to the single-point-of-failure problem and the limited scalability. In this paper, autonomous mobile sensor networks are deployed to map a scalar field in a cooperative and distributed fashion. We develop a cooperative sensor fusion algorithm based on distributed consensus filters. In this algorithm each agent receives measurements from its neighboring agents within its communication range, and iteratively updates the estimate of the unknown scalar field and an associated confidence map. A motion planning algorithm is used to obtain a path for complete coverage of the field of interest. A distributed flocking control algorithm is adopted to drive the center of the mobile sensor network to track the desired paths. Computer simulations are conducted to validate the proposed algorithms. We evaluate the mapping performance by comparing it with a centralized mapping algorithm. Such a cooperative sensing approach can be used in many military surveillance applications where targets may be small and elusive.

Keywords: Cooperative sensing, Sensor fusion, Flocking control, Mobile sensor networks.

1. INTRODUCTION

Mobile sensor networks (MSNs) have broad applications including military target detection and tracking; cooperative detection of toxic chemicals in contaminated environments; search and rescue operation in disasters; forest fire monitoring, etc. In many applications, a core problem is to conduct cooperative scalar field mapping (or searching) over a large area of interest. Centralized solutions to the scalar field mapping may not fit for large mobile sensor network due to the single-point-of-failure problem and the limited scalability. In recent years, missions that require the mapping of a scalar field become prominent. Measuring and exploring an unknown field of interest have attracted much attention of environmental scientists and control engineers.[1–6] They have numerous applications including military target detection, surveillance, environmental monitoring,[7] and oil spill and toxic-chemical plume tracing.[8,9] 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 the scalar field, one of the important research problems in MSNs is to achieve cooperative sensing among sensors in a distributed fashion. Development of a novel cooperative sensing algorithm based on distributed estimation and control algorithms for MSNs is very challenging. The estimation and control have to be performed in each sensor node using only local information, while as a whole they exhibit collective intelligence and achieve a global goal. In a resource-constrained multi-agent system, the communication range and sensing range of each agent are small compared to the size of a surveillance region. Hence, agents cannot accomplish the mission without an effective flocking control and path planning strategy.

Further author information: (Send correspondence to Weihua Sheng)
Weihua Sheng: E-mail: weihua.sheng@okstate.edu, Telephone: 1 405 744 7590
Hung M. La.: E-mail: hung.la@okstate.edu

Signal and Data Processing of Small Targets 2011, edited by Oliver E. Drummond, Proc. of SPIE Vol. 8137, 81370Y · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.895519

In this paper,the problems of cooperative sensing and cooperative motion control are addressed. Our work has three parts. 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 within its communication range. Each mobile sensor node will then iteratively update the estimate of the scalar field. Second, we use a distributed flocking control algorithm to drive the center of the mobile sensor formation to track the desired paths. Third, we build a path planning strategy to obtain a complete coverage of the field.

The rest of this paper is organized as follows. The next section presents the models of the scalar field and the measurement of each sensor node, as well as the problem formulation. Section 3 describes the distributed consensus filters and the distributed sensor fusion algorithm for building a map of the unknown scalar field. Section 4 presents the flocking control algorithm and the path planning strategy for complete coverage of the scalar field. Section 5 shows the simulation results. Finally, section 6 concludes this paper.

2. 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.

2.1 Model of the Scalar Field

We model the scalar field of interest as

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

here $\Theta = [\theta_1, \theta_2, ..., \theta_K]$, and $\Phi = [\phi_1, \phi_2, ..., \phi_K]$. We can rewrite Equation (1) as

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

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

We can model the function ϕ_j as a multiple variate Gaussian distribution (other distribution functions such as Poisson, Student, Cauchy distributions, ..., can also be used):

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

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

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

where c_j^0 is a correlation factor.

2.2 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 ($k \in \{1, 2, ..., C\}$) at time step t based on the following equation

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

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 uncorrelated noise which satisfies

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

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

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

here $q_i \in R^2$ is the position of sensor node i; $q_c^k \in R^2$ is the location of cell k at its center. This definition tells us that if cell k is inside the sensing range, r_i^s, of sensor node i then cell k can be measured or observed. Otherwise the observability is zero. Note that r_i^s can be the same for all sensors ($r_1^s = r_2^s = ... = r_n^s = r^s$) or different.

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

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

here c_v is the small positive constant between 0 and 1. The reason of introducing c_v is to avoid the variance $V_i^k(t)$ being zero when the distance $\|q_i(t) - q_c^k\|$ equals to zero.

2.3 Problem Formulation

Given the measurements of sensor node i and its neighbors at each cell of the scalar field F as modeled in Equation (3) (see Figure 1), our goal is to build the map for the scalar field F modeled by Equation (1).

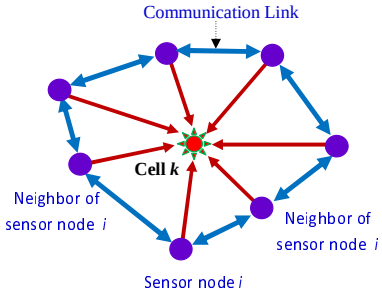

Figure 1. Illustration of the measurement model using multiple mobile sensor nodes.

3. DISTRIBUTED SENSOR FUSION ALGORITHM

3.1 Overall Approach

In this section we present a distributed sensor fusion algorithm to allow each sensor node to find out an estimate of the value at each cell of the scalar field based on its own measurement and its neighbor's measurements. 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 during its movement. To achieve it, we develop two consensus filters. The consensus filter 1 is to find out an estimate of the value of the field F at each cell at time step t. Since each mobile sensor node makes its own measurement at each cell at time step t with its own weight (confidence), the consensus filter 2 is used to find out 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 estimate phase*. Then, during the movement of each sensor node, it will have multiple spatial estimates of each cell associated with their own confidences. Hence, these spatial estimates are fused iteratively through the weighted average protocol, and this process can be called the *temporal estimate phase*. To summarize:

(1) *Spatial estimate phase:*

- Building a weighted average consensus filter (consensus filter 1) to find out an agreement of the estimates among the sensor nodes at each time step t,
- Building an average consensus filter (consensus filter 2) to find out an agreement of the weights (confidences) of the measurements among the sensor nodes at each time step t,

(2) *Temporal estimate phase:*

- Building a weighted average protocol to iteratively update the spatial estimates for sensor node i during its movement,

3.2 Distributed Consensus Filters

Let us consider a dynamic graph G consisting of a vertex set $\vartheta = \{1, 2..., n\}$ 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.

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

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

here r is the communication (active) range, and it can be the same as or less than r^s.

3.2.1 Consensus Filter 1

Distributed consensus[10–15] is an important computational tool to achieve cooperative sensing. We consider distributed linear iterations of the following form

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

here l is iteration index. The initial condition for the state is given as $x_i^k(l=0) = m_i^k(t)$. The weight, $w_{ii}^k(l)$, is the self weight or vertex weight of each sensor to cell k, and $w_{ij}^k(l)$ is the edge weight between sensor i and sensor j. These weights will be discussed more later.

Our problem here is to estimate the value of the field F at each cell k at each time step t. Since each sensor node makes 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 all 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:

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

If Equation (7) converges we have $E_1^k = E_2^k = ... = E_n^k = E^k$. Therefore, our goal is to let

$$lim_{l\to\infty}(x_i^k(l) - E^k(t)) \to 0 \tag{9}$$

We can write Equation (9) in the matrix form

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

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

We can also write Equation (7) in the matrix form

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

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

To make Equation (7) converge to $E^k(t)$ we need

$$lim_{l\to\infty}\mathbf{w}^k(l+1) = \frac{1}{n}\mathbf{1}\mathbf{1}^T \tag{12}$$

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

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

To satisfy this, we have the following designs of weight.

Weight Design 1:

From Equation (13) the vertex weight at node i is obtained as

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

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

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

here, c_1^w is a constant. If both sensor nodes i and j do not observe 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

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

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

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

To satisfy Equation (13) we need

$$0 < \sum_{j\in N_i(t)} w_{ij}^k(t) < 1 \Rightarrow 0 < \sum_{j\in N_i(t)} \frac{c_1^w}{V_i^k(t) + V_j^k(t)} < 1 \Rightarrow 0 < c_1^w < \frac{V_i^k(t) + V_j^k(t)}{|N_i(t)|}, \tag{18}$$

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

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

Weight Design 2:

From Equation (13) by assigning the same value to all edge weights we obtain:

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

here, $w_{ii}^k(l)$ is defined as

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

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

Therefore we have the following weight design

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

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

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

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

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

Finally, the consensus filter 1 (CF1) is summarized as

$$\begin{aligned} CF1 \quad : \quad & x_i^k(l+1) = w_{ii}^k(l) x_i^k(l) + \sum_{j \in N_i(t)} w_{ij}^k(l) x_j^k(l) \\ & w_{ij}^k(l) = \begin{cases} \frac{c_1^w}{V_i^k(t) + V_j^k(t)}, & if \quad i \neq j, j \in N_i(t), \\ 1 - \sum_{j \in N_i(t)} w_{ij}^k(l), & if \quad i = j, \\ 0, & otherwise. \end{cases} \\ & or, \\ & w_{ij}^k(l) = \begin{cases} \frac{c_2^w}{V_i^k(t)}, & if \quad i = j, \\ \frac{1 - w_{ii}^k(l)}{|N_i(t)|}, & if \quad i \neq j, j \in N_i(t), \\ 0, & otherwise. \end{cases} \end{aligned} \tag{25}$$

3.2.2 Consensus Filter 2

Since each sensor node has its own confidence of the measurement of the value of the scalar field at each cell at each time step t we need to find an agreement among the confidences of sensor nodes. The consensus algorithm 2 is introduced to find the overall confidence from each time step t. This overall confidence is the estimated weight, $W_i^k(t)$, of the weighted average protocol as shown in Equation (30).

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 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 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

$$y_i^k(l+1) = w_{ii}^k(l) y_i^k(l) + \sum_{j \in N_i(t)} w_{ij}^k(l) y_j^k(l), \tag{26}$$

In this consensus filter, we use the Metropolis weight[11] as

$$w_{ij}^k(l) = \begin{cases} \frac{1}{1 + max(|N_i(t)|, |N_j(t)|)}, & if \quad i \neq j, j \in N_i(t), \\ 1 - \sum_{j \in N_i(t)} w_{ij}^k(l), & if \quad i = j, \\ 0, & otherwise. \end{cases} \tag{27}$$

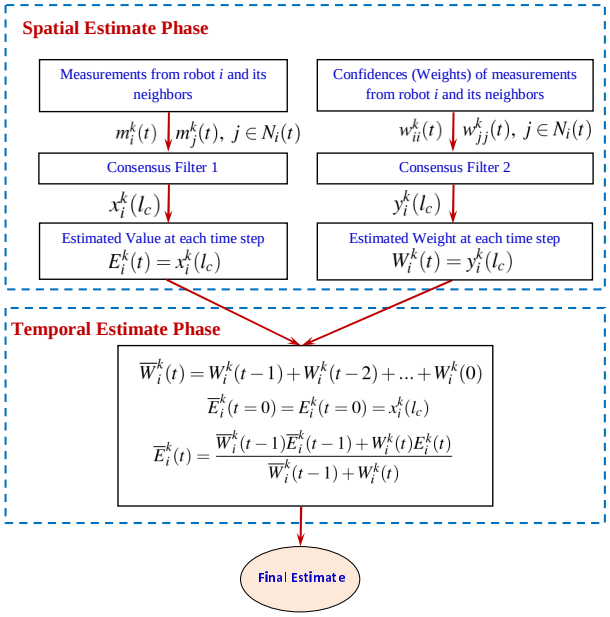

Figure 2. Frame work of distributed sensor fusion algorithm based two different consensus filters.

3.3 Distributed Fusion Algorithm

From the consensus filters 1 and 2 we would like to design a distributed sensor fusion algorithm to allow each sensor node to on-line estimate the value of the scalar field at each cell based on its own measurement and its neighbor's measurements. The overall design of such a distributed sensor fusion algorithm is shown in Figure 2. In this algorithm, we have two phases running at the same time. In the spatial estimate phase, the measurements of each sensor node and its neighbors at cell k at time step t are inputs of the consensus filter 1. Then, the output of this consensus 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 inputs of the consensus filter 2. Then, the output of this consensus is the estimate of the confidence of the measurement of the scalar field at cell k at time step t. During the movement of sensor nodes , each sensor obtain several spatial estimates of the value at cell k associated with its own confidence, hence the final estimate is iteratively updated based on these spatial estimates via the weighted average protocol. The detail to implement this algorithm is shown in Algorithm 1.

4. FLOCKING CONTROL ALGORITHM AND PATH PLANNING STRATEGY

In this section we present the distributed flocking control algorithm that allows the mobile sensor nodes to move together without collision and track the leader (target). Then, we present the path planning strategy to ensure that the MSN can cover the entire scalar field.

4.1 Flocking Control Algorithm

We consider n mobile sensor nodes moving in an m (e.g., $m = 2, 3$) dimensional Euclidean space. The dynamic equations of each sensor node are described as:

$$\begin{cases} \dot{q}_i = p_i \\ \dot{p}_i = u_i, \quad i = 1, 2, ..., n. \end{cases} \tag{31}$$

here $p_i \in R^m$ is the velocity of sensor node i, and u_i is the control input of sensor node i.

The geometry of flocks is modeled by an α-lattice[16] that meets the following condition:

$$\|q_j - q_i\| = d, j \in N_i(t), \tag{32}$$

Algorithm 1: Distributed Sensor Fusion Algorithm

Input: the weight $w_{ii}^k(t)$, and the measurement of sensor node i and its neighbors to cell k, $m_i^k(t)$ and $m_j^k(t)$.

Output: the final estimate of the cell k, $\overline{E}_i^k(1:t_l)$

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 $\|q_i(t) - q_c^k\| \leq r_i^s$

    Sensor node i obtains the measurements of cell k from its neighbors and itself

    for *each iteration* l **do**

      Sensor node i runs the consensus (7) with $w_{ij}^k(t)$ is defined in (16) or (22)

      $x_i^k(l=0) = m_i^k(t)$

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

      Sensor node i runs the consensus (26) with $w_{ij}^k(l)$ is defined in (27)

      $y_i^k(l=0) = w_{ii}^k(t)$

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

    end

    Step2: Obtain the estimate of cell k after running the consensus

    Let l_c be a time step 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 cell k

    - Update weight (confidence):

$$\overline{W}_i^k(t) = W_i^k(t-1) + W_i^k(t-2) + ... + W_i^k(0) \tag{28}$$

    - Update the final estimate based on the weighted average protocol:

$$\overline{E}_i^k(t=0) = E_i^k(t=0) = x_i^k(l_c) \tag{29}$$

$$\overline{E}_i^k(t) = \frac{\overline{W}_i^k(t-1)\overline{E}_i^k(t-1) + W_i^k(t)E_i^k(t)}{\overline{W}_i^k(t-1) + W_i^k(t)} \tag{30}$$

  end

end

here d is a positive constant indicating the distance between sensor node i and its neighbor j. However, at 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 (32) is rewritten in term of σ - norm[16] as follows:

$$\|q_j - q_i\|_\sigma = d^\alpha, j \in N_i(t), \tag{33}$$

here the 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 a map $R^m \Longrightarrow R_+$ defined as $\|z\|_\sigma = \frac{1}{\epsilon}[\sqrt{1+\epsilon\|z\|^2} - 1]$ with $\epsilon > 0$. Unlike the Euclidean norm $\|z\|$, which is not differentiable at $z = 0$, the σ - norm $\|z\|_\sigma$, is differentiable every where. This property allows to construct a smooth collective potential function for sensor nodes.

To achieve three basic flocking rules (flock centering, collision avoidance, and velocity matching). The 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 robots. This function is designed as[16]

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

where c_1^α and c_2^α are positive constants, and each term in (34) is computed as follows:[16]

1. The action function $\phi_\alpha(z)$ that vanishes for all $z \geq r^\alpha$ with $r^\alpha = \|r\|_\sigma$ is defined as follows:

$$\phi_\alpha(z) = \rho_h(z/r_\alpha)\phi(z - d^\alpha) \tag{35}$$

with the sigmoidal function $\phi(z)$ defined as $\phi(z) = 0.5[(a+b)\sigma_1(z+c) + (a-b)]$, here $\sigma_1(z) = z/\sqrt{1+z^2}$ (z is an arbitrary variable), and parameters $0 < a \leq b$, $c = |a-b|/\sqrt{4ab}$ to guarantee $\phi(0) = 0$. The bump function $\rho_h(z)$ with $h \in (0,1)$ is

$$\rho_h(z) = \begin{cases} 1, & z \in [0,h) \\ 0.5[1 + cos(\pi(\frac{z-h}{1-h}))], & z \in [h,1) \\ 0, & otherwise. \end{cases} \tag{36}$$

2. The vector along the line connecting q_i to q_j is

$$n_{ij} = (q_j - q_i)/\sqrt{1+\epsilon\|q_j - q_i\|^2}. \tag{37}$$

3. The elements $a_{ij}(q)$ of the adjacency matrix $[a_{ij}(q)]$ are defined as

$$a_{ij}(q) = \begin{cases} \rho_h(\|q_j - q_i\|_\sigma / r_\alpha), & if \quad j \neq i \\ 0, & if \quad j = i. \end{cases} \tag{38}$$

To allow all sensor nodes to move together and track a leader (static/moving target), the negative feedback function f_i^t is introduced as

$$\begin{aligned} f_i^t &= -c_1^t(q_i - q_t) - c_2^t(p_i - p_t) \\ &\quad -c_1^{mc}(\overline{q}_{(N_i \cup \{i\})} - q_t) - c_2^{mc}(\overline{p}_{(N_i \cup \{i\})} - p_t), \end{aligned} \tag{39}$$

here c_1^t, c_2^t, c_1^{mc} and c_2^{mc} are positive constants.

The pair $(\overline{q}_{(N_i \cup \{i\})},\ \overline{p}_{(N_i \cup \{i\})})$ is defined as $\begin{cases} \overline{q}_{(N_i \cup \{i\})} = \frac{1}{|N_i \cup \{i\}|} \sum_{i=1}^{|N_i \cup \{i\}|} q_i \\ \overline{p}_{(N_i \cup \{i\})} = \frac{1}{|N_i \cup \{i\}|} \sum_{i=1}^{|N_i \cup \{i\}|} p_i \end{cases}$ here $|N_i \cup \{i\}|$ is the number of agents in agent i's local neighborhood including agent i itself.

The final distributed flocking control algorithm is presented as follows:

$$\begin{aligned} u_i &= f_i^\alpha + f_i^t \\ &= c_1^\alpha \sum_{j \in N_i} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_2^\alpha \sum_{j \in N_i} a_{ij}(q)(p_j - p_i) \\ &\quad -c_1^t(q_i - q_t) - c_2^t(p_i - p_t) \\ &\quad -c_1^{mc}(\overline{q}_{(N_i \cup \{i\})} - q_t) - c_2^{mc}(\overline{p}_{(N_i \cup \{i\})} - p_t). \end{aligned} \tag{40}$$

4.2 Path Planning Strategy

Based on the flocking control algorithm (40), all mobile sensor nodes can form a quasi lattice formation, and the center of mass (CoM) of the network as defined in Equation (41) can track the leader (q_t, p_t) successfully.

$$\begin{cases} \overline{q} = \frac{1}{n}\sum_{i=1}^{n} q_i \\ \overline{p} = \frac{1}{n}\sum_{i=1}^{n} p_i \end{cases} \tag{41}$$

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 multi-robot path planning problem becomes a single robot path planning problem. There are some typical types of motion planning for a mobile robot to have complete coverage of the field of interest such as *boustrophedron* motion or wall-following motion.[17] In this paper we plan the leader motion using a typical *boustrophedron* motion.[17] The result of the path planning is shown in Figure 3. and it

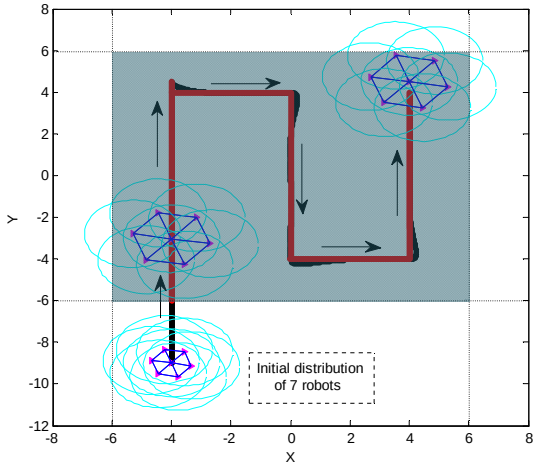

Figure 3. Seven mobile sensor nodes flock together and cover the scalar field (the filled square: 12×12). The motion path (red color) is generated by the leader, and the CoM (black/darker color) of the network tracks the leader with small overshoots at sharp change points of the path.

can generate a path for the network to go so that the field is fully covered.

5. SIMULATION RESULTS

In this section we test our consensus filter 1, then use our distributed sensor fusion algorithm to build the map of the scalar field.

5.1 Simulation for Consensus Filter 1

In this subsection we test our consensus filter 1 for the case of a single cell $k = 1$. We randomly generate a connected network of 10 nodes as shown in Figure 4*a*. The location of the cell measurement is located at the center of the network (the read square in Figure 4*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 to this location.

Each node makes a measurement as

$$m_i^1 = F^1 + n_i^1.$$

here $F^1 = 50$, and n_i^1 is the Gaussian noise, $N(0, V_i^1)$, with $V_i^1 = \frac{\|q_i - \overline{q}\|^2 + c_v}{(r_i^s)^2}$, $c_v = 0.01$, $r_1^s = r_2^s = ... = r_{10}^s = 1.6$, and $\overline{q} = \frac{1}{10}\sum_{i=1}^{10} q_i$.

The results of the convergence of the consensus filter (7) with two different weights, (16) and (22), are presented in Figures 4. We can see that the consensus filter with *Weight Design 2* in Equation (22) converges faster than the one using *Weight Design 1* in Equation (16). We also see that node 4 which has less neighbors than others converges slower.

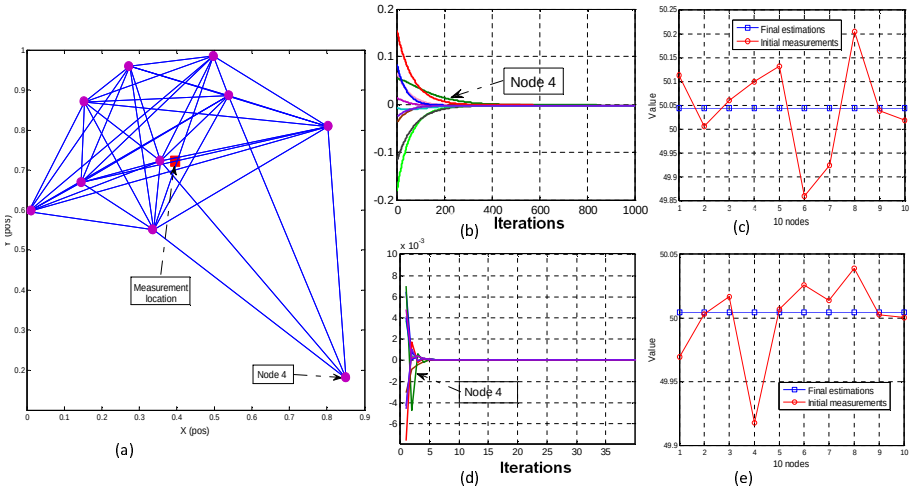

Figure 4. (a). 10 nodes estimate the value at cell k (pink square). (b, c) Result of convergence of 10 nodes, and agreement of 10 nodes when applying *Weight Design 1* in (16). (d, e) Result of convergence of 10 nodes, and agreement of 10 nodes when applying *Weight Design 2* in (22).

5.2 Simulation for Scalar Field Mapping

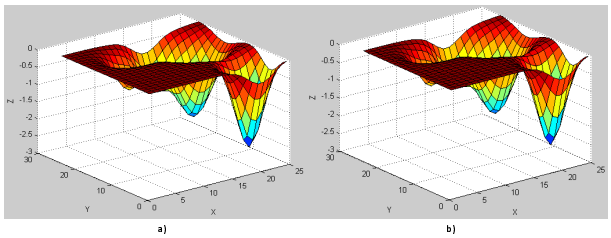

Figure 5. (a) the original map of the scalar field F, (b) the built map of the scalar field F using Algorithm 1.

We model the environment (scalar field F) as multiple variate Gaussian distributions. We set $\Theta = [30\ 10\ 8\ 20]$, and use four multiple variate Gaussian distributions ($K = 4$), and each one is represented as:

$$\phi_1 = \frac{1}{\sqrt{det(C_1)(2\pi)^2}} e^{\frac{1}{2}(x-3)C_1^{-1}(y-2)^T},$$

here $C_1 = \begin{bmatrix} 2.25 & 0.2999 \\ 0.2999 & 2.25 \end{bmatrix}$, with $c_1^0 = 0.1333$.

$$\phi_2 = \frac{1}{\sqrt{det(C_2)(2\pi)^2}} e^{\frac{1}{2}(x-1)C_2^{-1}(y-4.5)^T},$$

here $C_2 = \begin{bmatrix} 1.25 & 0.1666 \\ 0.1666 & 1.25 \end{bmatrix}$, with $c_2^0 = c_1^0$.

$$\phi_3 = \frac{1}{\sqrt{det(C_3)(2\pi)^2}} e^{\frac{1}{2}(x+2)C_3^{-1}(y-3)^T},$$

here $C_3 = C_2$, and $c_3^0 = c_2^0$.

$$\phi_4 = \frac{1}{\sqrt{det(C_4)(2\pi)^2}} e^{\frac{1}{2}(x-4)C_4^{-1}(y+4)^T},$$

here $C_4 = C_3$, and $c_4^0 = c_3^0$.

The field F has a size $x \times y = 12 \times 12$, and it is partitioned into $25 \times 25 = 625$ cells. The result of the built map of the scalar field is shown in Figure 5. The snapshots of multiple sensor nodes forming a flock and building the map of the unknown scalar field are shown in Figure 6. The errors between the built and original maps in one and three dimensions are shown in Figure 7, 8, respectively. Three algorithms, Algorithm 1 with the weighted average update protocol, Algorithm 1 with the normal average update protocol, and the centralized fusion algorithm, are compared. 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. The confidence map which is built based on the summation of the weights at each cell of the field F is shown in Figure 9.

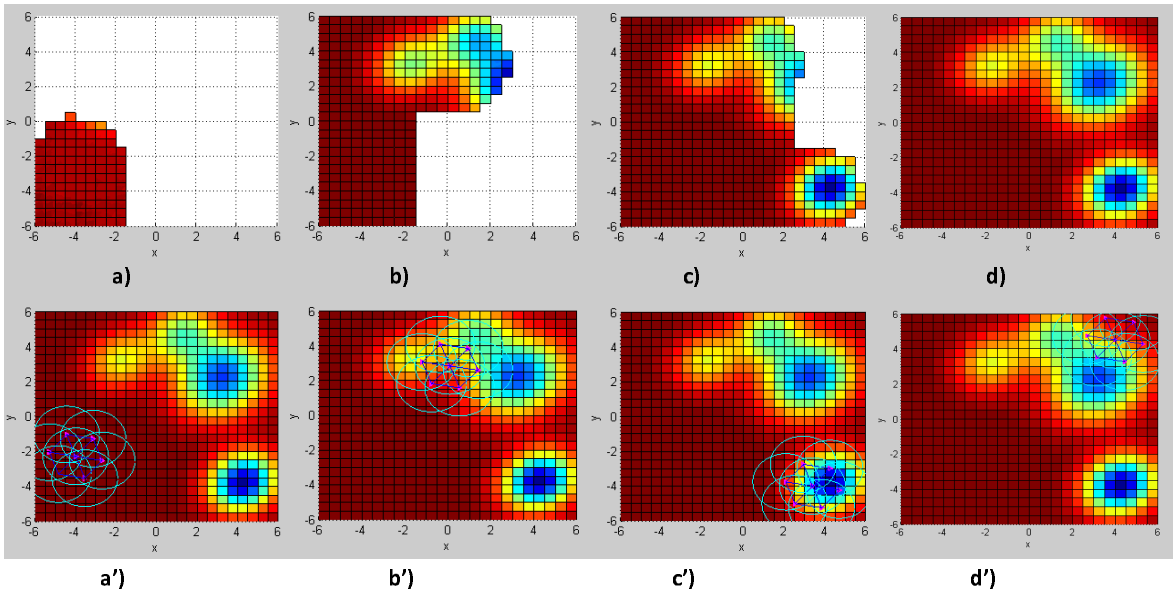

Figure 6. The snapshots of building the map of the scalar field F using Algorithm 1.

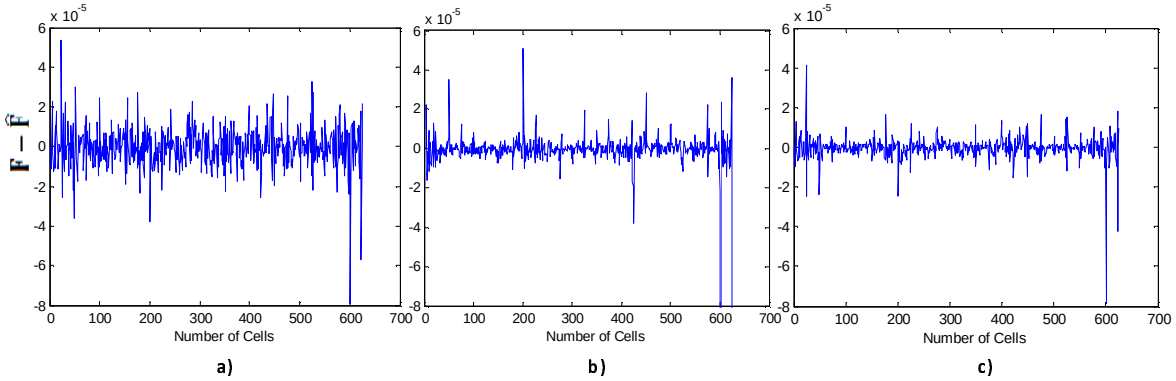

Figure 7. The error between the built and original maps for all cells in one dimension. (a) for Algorithm 1 with the normal average update protocol; (b) for centralized fusion algorithm; (c) for Algorithm 1 with the weighted average update protocol.

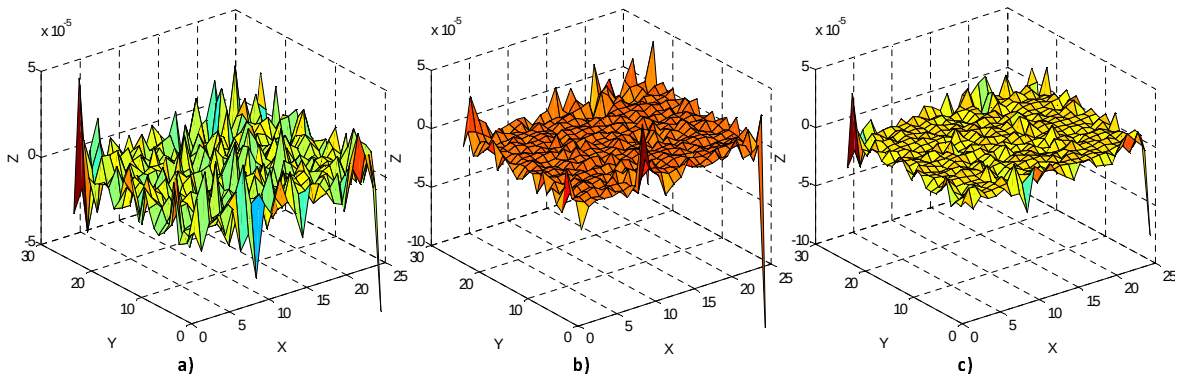

Figure 8. The error between the built and original maps for all cells in three dimensions. (a) for Algorithm 1 with the normal average update protocol; (b) for centralized fusion algorithm; (c) for Algorithm 1 with the weighted average update protocol.

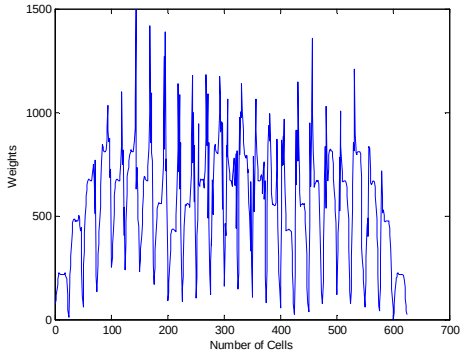

Figure 9. The summation of weights or the confidence of the estimate at each cell of the scalar field F

6. CONCLUSION AND FUTURE WORK

This paper presented a cooperative sensing algorithm for mobile sensor networks that form a flock to build the map of an unknown scalar field. 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 at each cell. The final estimates of the values of the scalar field are updated on-line based on the weighted average protocol. Experimental results are collected to demonstrate the proposed algorithm. To show the effectiveness of our proposed distributed sensor fusion algorithm we compare it with a centralized fusion algorithm in term of mapping error. Such a cooperative sensing approach can be used in many military surveillance applications where targets may be small and elusive.

In the future we plan to implement this algorithm on real sensor nodes, and we would like to investigate how the formation of the network and the motion planning affect to the sensing quality.

ACKNOWLEDGMENTS

This project is supported by the MOET (Ministry of Education and Training) program, Vietnamese Government; the DoD ARO DURIP grant 55628-CS-RIP, USA; the Center for Telecommunications and Network Security, Oklahoma State University; and Oklahoma Transportation Center grant OTCREOS10.1-43.

REFERENCES

[1] Hussein, I. I., "A Kalman filter-based control strategy for dynamic coverage control," *Proceedings of the American Control Conference* , 3271–3276 (2007).

[2] Cortes, J., Martinez, S., Karatas, T., and Bullo, F., "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation* **20**(2), 243–255 (2004).
[3] Zhang, F., Fratantoni, D. M., Paley, D., Lund, J., and Leonard, N. E., "Control of coordinated patterns for ocean sampling," *International Journal on Control* **80**(7), 1186–1199 (2007).
[4] Zhang, F. and Leonard, N. E., "Cooperative filters and control for cooperative exploration," *IEEE Transactions on Automatic Control* **55**(3), 650–663 (2010).
[5] "DOD/ONR MURI: adaptive sampling and prediction project. also available at http://www.princeton.edu/ dcsl/asap/,"
[6] Choi, J., Oh, S., and Horowitz, R., "Distributed learning and cooperative control for multi-agent systems," *Automatica* **45**(12), 2802–2814 (2009).
[7] Dhariwal, A., Sukhatme, G. S., and Requicha, A. A. G., "Bacterium inspired robots for environmental monitoring," *in Proceedings of the IEEE International Conference on Robotics and Automation* , 1436–1443 (2004).
[8] Lilienthal, A., Ulmer, H., Frohlich, H., Stutzle, A., Werner, F., and Zell, A., "Gas source declaration with a mobile robot," *Proceedings of the 2009 IEEE International Conference on Robotics and Automation* , 1430–1435 (2004).
[9] Zarzhitsky, D. V., Spears, D. F., and Thayer, D. R., "Experimental studies of swarm robotic chemical plume tracing using computational fluid dynamics simulations," *International Journal of Intelligent Computing and Cybernetics* , 1–44 (2010).
[10] Xiao, L. and Boy, S., "Fast linear iterations for distributed averaging," *Proceedings of the 42nd IEEE Conference on Decision and Control* (2003).
[11] Xiao, L., Boy, S., and Lall, S., "A scheme for robust distributed sensor fusion based on average consensus," *Proceedings of the International Conference on Information Processing in Sensor Networks* , 63–70 (2005).
[12] Olfati-Saber, R. and Murray, R. M., "Consensus problems in networks of agents with switching topology and time–delays," *IEEE Transactions on Automatic Control* **49**(9), 1520–1533 (2004).
[13] Olfati-Saber, R., Fax, J. A., and Murray, R. M., "Consensus and cooperative in networked multi-agent systems," *Proceedings of the IEEE* **95**(1), 215–233 (2007).
[14] Kar, S. and Moura, J. M. F., "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Transactions on Signal Processing* **57**(1), 355–369 (2009).
[15] Kokiopoulou, E. and Frossard, P., "Distributed classification of multiple observation sets by consensus," *Proceedings of the IEEE* (2010).
[16] Olfati-Saber, R., "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control* **51**(3), 401–420 (2006).
[17] Choset, H., "Coverage of known spaces: the boustrophedon cellular decomposition," *Autonomous Robots* **9**, 247–253 (2000).