Autonomic Mobile Sensor Network with Self-Coordinated Task Allocation and Execution

Kian Hsiang Low, Wee Kheng Leow, *Member, IEEE*, and Marcelo H. Ang, Jr., *Member, IEEE*

***Abstract*— The work in this paper describes a distributed layered architecture for resource-constrained multi-robot cooperation, which is utilized in autonomic mobile sensor network coverage. In the upper layer, a dynamic task allocation scheme self-organizes the robot coalitions to track efficiently across regions. It uses concepts of ant behavior to self-regulate the regional distributions of robots in proportion to that of the moving targets to be tracked in a non-stationary environment. As a result, the adverse effects of task interference between robots are minimized and network coverage is improved. In the lower task execution layer, the robots use self-organizing neural networks to coordinate their target tracking within a region. Both layers employ self-organization techniques, which exhibit autonomic properties such as self-configuring, self-optimizing, self-healing, and self-protecting. Quantitative comparisons with other tracking strategies such as static sensor placements, potential fields, and auction-based negotiation show that our layered approach can provide better coverage, greater robustness to sensor failures, and greater flexibility to respond to environmental changes.**

***Index Terms*— task allocation, motion control, multi-robot architecture, swarm intelligence, self-organizing neural networks**

I. Introduction

SENSOR networks have recently received significant attention in the areas of networking, embedded systems, pervasive computing, and multi-agent systems [1] due to its wide array of real-world applications (e.g., disaster relief, environment monitoring). In these applications, the distributed sensing task is achieved by the collaboration of a large number of static sensors, each of which has limited sensing, computational, and communication capabilities.

One of the fundamental issues that arises in a sensor network is coverage. Traditionally, network coverage is maximized by determining the optimal placement of static sensors in a centralized manner, which can be related to the class of art gallery problems [2]. However, recent investigations in sensor network mobility reveal that mobile sensors can self-organize to provide better coverage than static sensors ([3], [4]). Existing applications have only utilized uninformed mobility (i.e., random motion or patrol) [1]. In contrast, our work focuses on informed, intelligent mobility to further improve coverage.

K. H. Low is with Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA (email:bryanlow@cs.cmu.edu).

W. K. Leow is with Department of Computer Science, National University of Singapore, 3 Science Drive 2, S(117543), Singapore (email: leowwk@comp.nus.edu.sg).

M. H. Ang, Jr. is with Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, S(119260), Singapore (email: mpeangh@nus.edu.sg).

Our network coverage problem is motivated by the following constraints that discourage static sensor placement or uninformed mobility: a) unknown target distributions or motion patterns, b) limited sensing range, and c) large area to be observed. All these conditions may cause the sensors to be unable to cover the entire region of interest. Hence, fixed sensor locations or uninformed mobility will not be adequate in general. Rather, the sensors have to move dynamically in response to the motion and distribution of targets and other sensors to maximize coverage.

Inspired by robotics, the above problem may be regarded as that of low-level motion control to coordinate the sensors' target tracking movements in continuous workspace. Alternatively, it can be cast as a high-level task allocation problem by segmenting the workspace into discrete regions (Fig. 1a) such that each region is assigned a group or *coalition* of sensors to track the targets within. This paper presents a distributed architecture that integrates low-level motion control with high-level task allocation for autonomic mobile sensor network coverage in complex, dynamic environments (Section III). We will now refer to mobile sensors as robots since they are the same in this paper's context.

II. Related Work on Coverage

Existing sensor network coverage applications can be classified under the following characteristics: a) network mobility (static vs. mobile), b) network density (dense vs. sparse), c) target distributions (known vs. unknown), and d) target motion patterns (e.g., static, random, evasive). Static sensor networks [5] are often densely deployed for complete coverage of the area to be observed. Such networks typically require manual positioning of the sensors and cannot be easily deployed in contaminated or hostile regions. Mobile sensors, on the other hand, can be used for this purpose. Current implementations of mobile sensor networks have focused on evenly dispersing the sensors from a source point throughout the observed region [6] without considering the target distributions. Recent efforts have attempted to self-organize the mobile sensors to that of the target distributions, which can potentially decrease the number of deployed sensors (Section VI-B.1). However, the target distributions are either static [3] or known beforehand [7]. Our work in this paper differs from all these by deploying a sparse network of mobile sensors to track unknown, time-varying target distributions.

III. Overview of Mobile Sensor Architecture

Our mobile sensor architecture consists of two layers of coordination (Fig. 1b): (1) lower task execution layer, and (2)

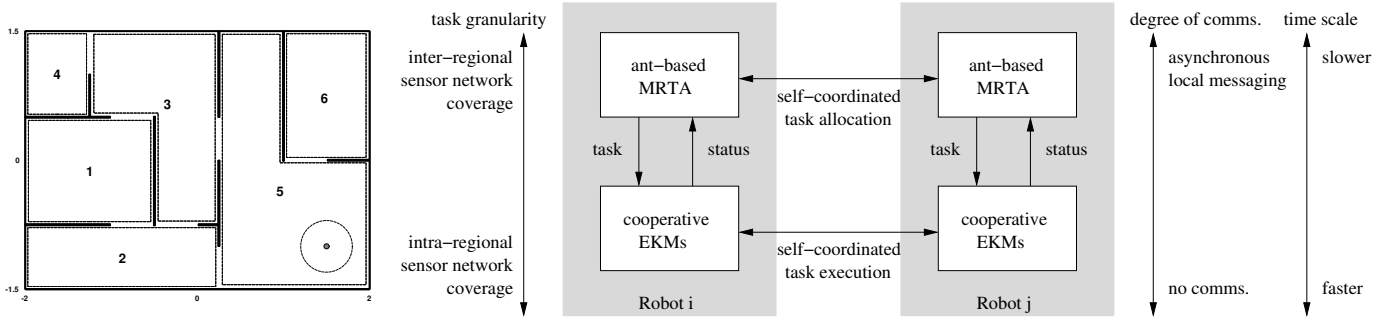


Fig. 1. (a) A 4 m × 3 m environment that is divided into 6 regions. The circle at the bottom right represents the robot's sensing radius of 0.3 m (drawn to scale). The environment is 42.44 times as large as the robot's sensing area. (b) Distributed layered architecture for multi-robot cooperation (MRTA = Multi-Robot Task Allocation, EKM = Extended Kohonen Map).

higher task allocation layer. It differs from existing layered architectures for multi-robot coordination ([8], [9]) by adopting a reactive method, rather than deliberative planning, for task allocation. Both layers employ concepts of self-organization that exhibit the following characteristics of autonomic systems:

1) *Self-Configuring:* Both the task allocation and execution schemes enable the sensor network to adapt to dynamically changing environments;
2) *Self-Optimizing:* Both schemes aim to maximize coverage and minimize robot interference;
3) *Self-Healing:* The task allocation scheme is robust to robot failures while the task execution scheme is able to self-repair unexpected damages to the robot formation;
4) *Self-Protecting:* The task execution scheme enables the robot to negotiate unforeseen complex obstacles.

These autonomic properties will be demonstrated in Section VI.

In the lower task execution layer, the robots use a reactive motion control strategy based on self-organizing neural networks [10] to coordinate their target tracking *within* a region without the need of communication (Section IV). This strategy is also responsible for their navigation between regions via beacons or checkpoints identified by a motion planner [11]. To perform these tasks, it has to coordinate multiple concurrent behaviors, which include target reaching, obstacle avoidance, and robot separation to minimize task interference. It differs from other Behavior Coordination Mechanisms (BCMs) ([12]) in the following ways:

Self-Organization of Continuous State and Motor Control Spaces: A high degree of smoothness and precision in motion control is essential for efficiently executing sophisticated tasks. This can only be achieved with *continuous response encoding* (i.e., infinite set of responses) of very low-level velocity/torque control of motor/joint actuators. Our proposed BCM uses self-organizing neural networks to map continuous state space to continuous motor control space. We have shown in ([11], [13]) via quantitative evaluation that such neural networks can produce fine, smooth, and efficient motion control. In contrast, BCMs that employ *discrete response encoding* (i.e., finite, enumerated set of responses) ([12], [14]) produce high-level motion commands (e.g., forward, left, right) that are usually too coarse for fine, smooth robot control. Consequently, the robot may fail to negotiate unforeseen complex obstacles.

Complexity of Robot Motion Tasks: Existing BCMs tend to under-utilize the sensory inputs that can potentially yield useful information for coordinating behaviors and choosing the most appropriate action. As a result, the robot is less capable of performing complex motion tasks such as negotiating unforeseen concave and closely spaced obstacles, and tracking multiple moving targets. Three classes of BCMs face this problem: behavior arbitration, action voting, and action superposition. Arbitration strategies ([14]) allow only one winning behavior among a group of competing ones to produce the action. This precludes the execution of several, possibly conflicting behaviors in parallel. In action voting schemes ([12]), each behavior can vote for various pre-defined discrete actions to different degrees and the action with the highest vote is performed. Both behavior arbitration and action voting methods suffer from the drawbacks of discrete response encoding discussed in the previous paragraph. Action superposition techniques (e.g., potential fields) ([11], [15]) combine all the potential actions, each generated by a behavior, using vector sum to produce a single action. They may cause the robot to fail in complex motion tasks [16] even though they utilize continuous response encoding. On the other hand, a robot endowed with our proposed BCM can achieve these tasks (Section VI-A).

In the higher task allocation layer, the robots use a dynamic ant-based scheme [17] to cooperatively self-organize their coalitions in a decentralized manner according to the target distributions *across* the regions (Section V). It contrasts with the other works of biologically-inspired robot swarms [15] that emphasize control- rather than task-level cooperation.

Our ant-based scheme addresses the following issues, which distinguish it from the other task allocation mechanisms:

Task Allocation for Multi-Robot Tasks: Existing Multi-Robot Task Allocation (MRTA) algorithms (i.e., auction- and utility-based) ([18], [19]) generally assume that a multi-robot task can be partitioned into several single-robot tasks. But this may not be always possible, or the multi-robot task can be more efficiently performed by coalitions of robots. Furthermore, the partitioned single-robot tasks are sometimes assumed to be independent, i.e., no interference would occur. However, the robots are bound to interfere with each other's ongoing activity either physically (e.g., space competition) or non-physically (e.g, shared radio bandwidth, conflicting goals). In the extreme

case, when too many robots are involved, little or no work gets done as they totally restrict each other's movement. Hence, task interference has an adverse effect on the overall system performance. Knowing that physical interference can be implied from robot density, our task allocation scheme dynamically distributes robots in real-time by estimating robot densities in different regions to minimize interference.

Coalition Formation for Minimalist Robots: Existing multi-agent coalition formation schemes [20] require complex planning, explicit negotiation, and precise estimation of coalitional cost. Hence, they may not be able to operate in real-time in a large-scale sensor network. Our task allocation method via self-organizing swarm coalitions is reactive, dynamic, and can operate with uncertain coalitional cost and resource-constrained robots.

Cooperation of Resource-Constrained Robots: Robots with limited communication and sensing capabilities can only extract local, uncertain information of the environment. As such, distributed methodologies are required to process and integrate the noisy, heterogeneous information to improve its quality so that it can be effectively utilized to estimate the coalitional cost and boost the task performance. Furthermore, if the robots have limited computational power, their cooperative strategies cannot involve complex planning or negotiation. Existing task allocation mechanisms ([18], [19], [21]) have either assumed perfect communications, high computational power, centralized coordination or global knowledge of the task and robots. For example, recent applications of sensor network coverage ([4]) and multi-robot systems [22] employ coalition leaders, one in each region, to negotiate with each other. This negotiation is conducted iteratively using an auction-based mechanism and attempts to balance the proportion of robots to that of the targets across all regions. To do so, each coalition leader must be able to obtain the exact number of robots and targets in its region as well as the task performance of these robots. Furthermore, it has to synchronize its negotiation with the coalition leaders in other regions via long-range communication. Note that this negotiation can be conducted entirely by a central coordinator running a centralized coalition formation scheme but it requires even more resources. In contrast, our proposed method does not require such expensive resources, thus catering to resource-constrained robots. The robots endowed with our ant-based scheme require only local sensing information and short-range communication. The robot coalitions can also be self-organized asynchronously without negotiation.

IV. Self-Coordinated Task Execution

A. Overview

Our proposed BCM, called *cooperative Extended Kohonen Maps* (EKMs), is implemented by connecting an ensemble of EKMs ([11], [13]), each of which is a neural network that extends the Kohonen Self-Organizing Map [23]. Its self-organization of the input space is similar to Voronoi tessellation such that each tessellated region is encoded by the input weights of an EKM neuron. In addition to encoding a set of input weights that self-organize the sensory input space,

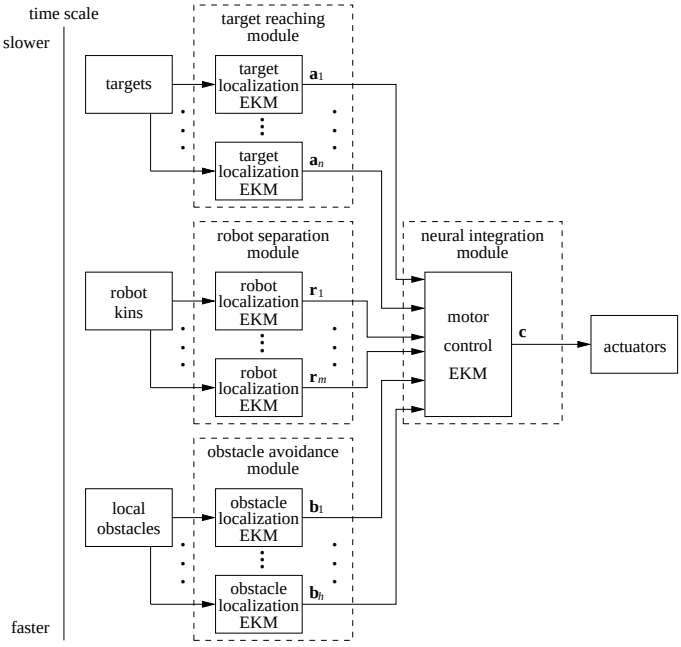


Fig. 2. A behavioral coordination mechanism that is implemented by an ensemble of Extended Kohonen Maps (EKMs).

the EKM neurons also produce outputs that vary with the incoming sensed inputs.

Our cooperative EKMs framework consists of four modules: target reaching, obstacle avoidance, robot separation, and neural integration (Fig. 2). The *target localization* EKMs in the *target reaching* module (Section IV-B) are activated by the presence of targets within the robot's target sensing range. Each EKM receives a sensed target location and outputs corresponding excitatory signals to the motor control EKM in the neural integration module at and around the locations of the sensed targets.

The *obstacle localization* EKMs in the *obstacle avoidance* module (Section IV-C) are activated by the presence of obstacles within the robot's obstacle sensing range. Each EKM receives a sensed obstacle location and outputs corresponding inhibitory signals to the motor control EKM in the neural integration module at and around the locations of the sensed obstacles. The *robot localization* EKMs in the *robot separation* module work in a similar fashion as the obstacle localization EKMs except that they process the sensed robot locations.

The *motor control* EKM in the *neural integration* module (Section IV-D) serves as the sensorimotor interface, which integrates the activity signals from the EKMs for cooperation and competition to produce an appropriate motor signal to the actuators.

The cooperative EKMs framework allows the modules to operate asynchronously at different rates, which is the key to preserving reactive capabilities. This contrasts with action voting and superposition BCMs, which require synchronization. For example, the target reaching and robot separation modules operate at about 256 ms between servo ticks while the obstacle avoidance module can typically operate faster at intervals of 128 ms. The neural integration module is activated as and when neural activities are received. One noteworthy aspect of

our framework is that no communication between robots is needed for the robots to cooperate in the tracking of multiple moving targets. In this paper, we demonstrate that robots, which are able to discriminate between targets, obstacles and robot kins, are adequate for achieving the cooperative task.

B. Target Reaching

The target reaching module adopts an egocentric representation of the sensory input vector $\mathbf{u}_p = (\alpha, d)^T$ where α and d are the direction and distance of a target relative to the robot's current location and heading. It uses the target localization EKM to self-organize the sensory input space $\mathcal{U}$. Each neuron i in the EKM has a sensory weight vector $\mathbf{w}_i = (\alpha_i, d_i)^T$ that encodes a region in $\mathcal{U}$ centered at $\mathbf{w}_i$. Each target that appears within the robot's sensory range activates a different target localization EKM. That is, n target localization EKMs will be activated for n targets. The same target can activate a different EKM at a different time. Based on each incoming sensory input $\mathbf{u}_p$ of the target location, the target localization EKM outputs excitatory signals to the motor control EKM in the neural integration module (Section IV-D). The target localization EKMs are activated as follows:

Target Localization

For each sensory input $\mathbf{u}_p$ of a target, $p = 1, \ldots, n$ (i.e., n targets),

1) Determine the winning neuron s in the p-th target localization EKM. Each winning neuron s is the one whose sensory weight vector $\mathbf{w}_s = (\alpha_s, d_s)^T$ is nearest to the input $\mathbf{u}_p = (\alpha, d)^T$:

$$D(\mathbf{u}_p, \mathbf{w}_s) = \min_{i \in \mathcal{A}(\alpha)} D(\mathbf{u}_p, \mathbf{w}_i). \tag{1}$$

The difference $D(\mathbf{u}_p, \mathbf{w}_i)$ is a weighted difference between $\mathbf{u}_p$ and $\mathbf{w}_i$:

$$D(\mathbf{u}_p, \mathbf{w}_i) = \beta_\alpha(\alpha - \alpha_i)^2 + \beta_d(d - d_i)^2 \tag{2}$$

where β_α and β_d are constant parameters. The minimum in Eq. 1 is taken over the set $\mathcal{A}(\alpha)$ of neurons encoding very similar angles as α:

$$\begin{gathered} |\alpha - \alpha_i| \leq |\alpha - \alpha_j|, \\ \text{for each pair } i \in \mathcal{A}(\alpha), j \notin \mathcal{A}(\alpha)\ . \end{gathered} \tag{3}$$

In other words, direction has priority over distance in the competition between EKM neurons. This method allows the robot to quickly orientate itself to face the target while moving towards it. An EKM contains a limited set of neurons, each of which has a sensory weight vector $\mathbf{w}_i$ that encodes a point in the sensory input space $\mathcal{U}$. The region in $\mathcal{U}$ that encloses all the sensory weight vectors of these neurons is called the *local workspace* $\mathcal{U}'$. Even if the target falls outside $\mathcal{U}'$, the nearest neuron can still be activated (Fig. 3a).

2) Compute output activity a_{pi} of neuron i in the p-th target localization EKM.

$$a_{pi} = G_a(\mathbf{w}_s, \mathbf{w}_i) \tag{4}$$

The function G_a is an elongated Gaussian:

$$G_a(\mathbf{w}_s, \mathbf{w}_i) = \exp\left(-\frac{(\alpha_s - \alpha_i)^2}{2\sigma_{a\alpha}^2} - \frac{(d_s - d_i)^2}{2\sigma_{ad}^2}\right). \tag{5}$$

Parameter σ_{ad} is much smaller than $\sigma_{a\alpha}$, making the Gaussian distance-sensitive and angle-insensitive. These parameter values elongate the Gaussian along the direction perpendicular to the target direction α_s (Fig. 3b). This elongated Gaussian is the *target field*, which plays an important role in avoiding local minima during obstacle avoidance.

The output activities of the neurons in the n target localization EKMs are aggregated in the motor control EKM to produce a motion that moves the robot towards the targets. This will be explained in Section IV-D. In the next section, we will present the obstacle and robot localization EKMs, which are activated in a similar manner as the target localization EKMs.

C. Obstacle Avoidance and Robot Separation

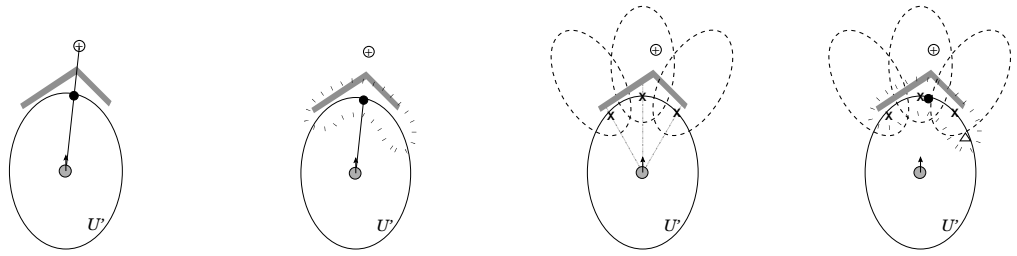
The obstacle avoidance module uses obstacle localization EKMs. The robot has h directed distance sensors around its body for detecting obstacles. Hence, each activated sensor encodes a fixed direction α_j and a variable distance d_j of the obstacle relative to the robot's heading and location. Each sensor's input $\mathbf{u}_j = (\alpha_j, d_j)^T$ induces an obstacle localization EKM. Note that the distance sensors operate differently from the target sensors. A target sensor (e.g., vision camera) can sense multiple targets whereas each distance sensor (e.g., laser) can only reflect the nearest obstacle in its sensing direction. Hence, unlike the target localization EKMs, the number of obstacle localization EKMs that are activated does not depend on the number of obstacles but rather, on the number of distance sensors. The obstacle localization EKMs have the same number of neurons and input weight values as the target localization EKMs, i.e., each neuron i in the obstacle localization EKM has the same input weight vector $\mathbf{w}_i$ as the neuron i in the target localization EKM. The EKMs output inhibitory signals to the motor control EKM in the neural integration module (Section IV-D). The obstacle localization EKMs are activated as follows:

Obstacle Localization

For each sensory input $\mathbf{u}_j$, $j = 1, \ldots, h$ (i.e., h distance sensors),

1) Determine the winning neuron s in the j-th obstacle localization EKM. The obstacle localization EKM is activated in the same manner as Step 1 of Target Localization (Section IV-B).
2) Compute output activity b_{ji} of neuron i in the j-th obstacle localization EKM:

$$b_{ji} = G_b(\mathbf{w}_s, \mathbf{w}_i) \tag{6}$$



(a) (b) (c) (d)

Fig. 3. Conceptual description of cooperative EKMs. (a) In response to the target $\oplus$, the nearest neuron (black dot) in the target localization EKM (ellipse) of the robot (gray circle) is activated. (b) The activated neuron produces a target field (dotted region) in the motor control EKM. (c) Three of the robot's sensors detect obstacles and activate three neurons (crosses) in the obstacle localization EKMs, which produce the obstacle fields (dashed ellipses). (d) Subtraction of the obstacle fields from the target field results in the neuron at $\triangle$ to become the winner in the motor control EKM, which moves the robot away from the obstacle.

where

$$G_b(\mathbf{w}_s, \mathbf{w}_i) = \exp\left(-\frac{(\alpha_s - \alpha_i)^2}{2\sigma_{b\alpha}^2} - \frac{(d_s - d_i)^2}{2\sigma_{bd}^2(d_s, d_i)}\right)$$

$$\sigma_{bd}(d_s, d_i) = \begin{cases} 2.475 & \text{if } d_i \geq d_s \\ 0.02475 & \text{otherwise.} \end{cases} \tag{7}$$

The function G_b is a Gaussian stretched along the obstacle direction α_s so that motor control EKM neurons beyond the obstacle locations are also inhibited to indicate inaccessibility (Fig. 3c). If no obstacle is detected, $G_b = 0$. In the presence of an obstacle, the neurons in the obstacle localization EKMs at and near the obstacle locations will be activated to produce *obstacle fields* (Eq. 6). The neurons nearest to the obstacle locations have the strongest activities.

The separation between a robot and its other kins is achieved with robot localization EKMs. These EKMs work in the same way as obstacle localization EKMs, i.e., each neuron i in the q-th robot localization EKM outputs an inhibitory activity r_{qi} to the motor control EKM in the neural integration module (Section IV-D). However, the robot localization EKMs produce wider *robot kin fields*. This has the effect of keeping a robot away from targets that are close to other robot kins. As a result, the overlap in the coverage of targets between robots is minimized. Unlike the distance sensors, a robot kin sensor (e.g., communication) can sense multiple robots. Hence, if there are m robots detected, m robot localization EKMs will be activated. The robot localization EKMs have the same number of neurons and input weight values as the target and obstacle localization EKMs.

D. Neural Integration and Motor Control

The neural integration module uses a motor control EKM to integrate the activities from the neurons in the target, obstacle and robot localization EKMs. The motor control EKM has the same number of neurons and input weight values as the target, robot, and obstacle localization EKMs. The neural integration is performed as follows:

Neural Integration

1) Compute activity e_i of neuron i in the motor control EKM.

$$e_i = \sum_{p=1}^{n} a_{pi} - \sum_{j=1}^{h} b_{ji} - \sum_{q=1}^{m} r_{qi} \tag{8}$$

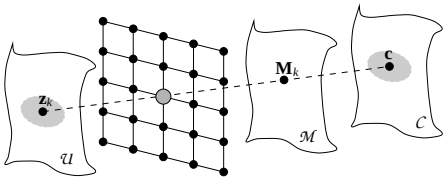


Fig. 4. Motor control EKM. The neurons map the sensory input space $\mathcal{U}$ indirectly to motor control space $\mathcal{C}$ through control parameter space $\mathcal{M}$.

where a_{pi} is the excitatory input from neuron i of the p-th target localization EKM (Section IV-B), b_{ji} is the inhibitory input from neuron i of the j-th obstacle localization EKM, and r_{qi} is the inhibitory input from neuron i of the q-th robot localization EKM (Section IV-C).

2) Determine the winning neuron k in the motor control EKM. Neuron k is the one with the largest activity:

$$e_k = \max_i e_i \ . \tag{9}$$

The motor control EKM also has a set of output weights, which encode the outputs produced by the neuron. It is trained to partition the sensory input space $\mathcal{U}$ into locally linear regions. Unlike existing direct-mapping methods ([24]) that perform discrete response encoding (Section I), the output weights $\mathbf{M}_i$ of neuron i of the motor control EKM represent control parameters in the parameter space $\mathcal{M}$ instead of the actual motor control vector (Fig. 4). The control parameter matrix $\mathbf{M}_i$ is mapped to the actual motor control vector $\mathbf{c}$ by a linear model (Eq. 10). Compared to direct-mapping EKM, indirect-mapping EKM can provide finer and smoother robot motion control. Detailed comparison and discussion have been reported in ([11], [13]). With indirect-mapping EKM, motor control is performed as follows:

Motor Control

Compute motor control vector $\mathbf{c}$:

$$\mathbf{c} = \mathbf{M}_k \mathbf{z}_k \tag{10}$$

where

$$\mathbf{z}_k = \frac{\sum_{i \in \mathcal{N}(k)} G(|e_i - e_k|)\mathbf{w}_i}{\sum_{i \in \mathcal{N}(k)} G(|e_i - e_k|)} \ . \tag{11}$$

$G(|e_i - e_k|)$ is a Gaussian with its peak located at neuron k and $\mathcal{N}(k)$ defines a small set of neurons in the neighborhood

of neuron k. At the goal state at time T, $\mathbf{z}_k(T) = (\alpha, 0)^T$ for any α.

In activating the motor control EKM (Fig. 3d), the obstacle fields are subtracted from the target field (Eq. 8). If the target lies within the obstacle fields, the activation of the motor control EKM neurons close to the target location will be suppressed. Consequently, another neuron at a location that is not inhibited by the obstacle fields becomes most highly activated (Fig. 3d). This neuron produces a control parameter that moves the robot away from the obstacle. While the robot moves around the obstacle, the target and obstacle localization EKMs are continuously updated with the current locations and directions of the target and obstacles. Their interactions with the motor control EKM produce fine, smooth, and accurate motion control of the robot to negotiate the obstacle and move towards the target until it reaches the goal state $\mathbf{z}_k(T)$ at time step T. In the case of multi-robot target tracking task, the robots act like obstacles to other robots, thus separating them from each other.

E. Self-Organization of EKMs

In contrast to offline learning methods, online training is adopted for the EKMs. Initially, the EKMs have not been trained and the motor control vectors $\mathbf{c}$ generated are inaccurate. Nevertheless, the EKMs self-organize, using these control vectors $\mathbf{c}$ and the corresponding robot displacements $\mathbf{v}$ produced by $\mathbf{c}$, to map $\mathbf{v}$ to $\mathbf{c}$ indirectly. As the robot moves around and learns the correct mapping, its sensorimotor control becomes more accurate. At this stage, the online training mainly fine tunes the indirect mapping. The self-organized training algorithm (in obstacle-free environment) is as follows:

Self-Organized Training

Repeat

1) Get sensory input $\mathbf{u}_p$.
2) Execute target reaching procedure and move robot.
3) Get new sensory input $\mathbf{u}'_p$ and compute actual displacement $\mathbf{v}$ as a difference between $\mathbf{u}'_p$ and $\mathbf{u}_p$.
4) Use $\mathbf{v}$ as the training input to determine the winning neuron k (same as Step 1 of Target Reaching except that $\mathbf{u}_p$ is replaced by $\mathbf{v}$).
5) Adjust the input weights $\mathbf{w}_i$ of neurons i in the neighborhood of the winning neuron k towards $\mathbf{v}$:

$$\Delta \mathbf{w}_i = \eta\, G(k,i)(\mathbf{v} - \mathbf{w}_i) \tag{12}$$

where $G(k,i)$ is a Gaussian function of the distance between the positions of neurons k and i in the EKM, and η is a constant learning rate.

6) Update the output weights $\mathbf{M}_i$ of neurons i in the neighborhood of the winning neuron k to minimize the error e:

$$e = \frac{1}{2} G(k,i) \|\mathbf{c} - \mathbf{M}_i \mathbf{v}\|^2 \ . \tag{13}$$

That is, apply gradient descent to obtain

$$\Delta \mathbf{M}_i = -\eta \frac{\partial e}{\partial \mathbf{M}_i} = \eta\, G(k,i)(\mathbf{c} - \mathbf{M}_i \mathbf{v})\mathbf{v}^T \ . \tag{14}$$

The target, obstacle, and robot localization EKMs self-organize in the same manner as the motor control EKM except that Step 6 is omitted. This will result in the same set of input weight vectors for all the localization and motor control EKMs after training. At each training cycle, the weights of the winning neuron k and its neighboring neurons i are modified. The amount of modification is proportional to the distance $G(k,i)$ between the neurons in the EKM. The input weights $\mathbf{w}_i$ are updated towards the actual displacement $\mathbf{v}$ and the control parameters $\mathbf{M}_i$ are updated so that they map the displacement $\mathbf{v}$ to the corresponding motor control $\mathbf{c}$. After self-organization has converged, the neurons will stabilize in a state such that $\mathbf{v} = \mathbf{w}_i$ and $\mathbf{c} = \mathbf{M}_i\mathbf{v} = \mathbf{M}_i\mathbf{w}_i$. For any winning neuron k, given that $\mathbf{z}_k = \mathbf{w}_k$, the neuron will produce a motor control output $\mathbf{c} = \mathbf{M}_k\mathbf{w}_k$ which yields a desired displacement of $\mathbf{v} = \mathbf{w}_k$. If $\mathbf{z}_k \neq \mathbf{w}_k$ but close to $\mathbf{w}_k$, the motor output $\mathbf{c} = \mathbf{M}_k\mathbf{z}_k$ produced by neuron k will still yield the correct displacement if linearity holds within the input region that activates neuron k. Thus, given enough neurons to produce an approximate linearization of the sensory input space $\mathcal{U}$, indirect-mapping EKM can produce finer and smoother motion control than direct-mapping EKM.

V. Self-Coordinated Task Allocation

Many multi-robot tasks, e.g., foraging [21], transportation, and exploration, have been inspired by social insects [25], in particular, ants. Our MRTA scheme encapsulates three concepts of ant behavior to self-organize the robot coalitions according to the target distributions across regions: (a) encounter pattern based on waiting time, (b) self-organization of social dominance, and (c) dynamic task allocation.

A. Encounter Pattern Based on Waiting Time

Encounter patterns provide a simple, local cue for ants with sensory and cognitive limitations to assess regional densities of ants and objects of interest, which are crucial to regulating the division of labor [26]. Instead of relying on global communication to relay target positions and density estimation [27], our scheme uses encounter patterns to predict target density via local sensing. Regional robot density is captured in a similar way using local communication.

An encounter pattern can be derived from a series of waiting time or interval between successive encounters. This simple form of information processing has accounted for the complex adaptive process of task allocation in ant colonies [28]. In our coverage task, the waiting time of a robot is defined in terms of its encounters with the other robots and targets. A robot encounter is defined as a reception of a message from another robot in the same region. A target encounter is defined as an increase in the number of targets tracked between the previous and the current time steps. For a robot i in region r, the waiting time for other robots $w_{ir}(k)$ and targets $w'_{ir}(k)$ is the time interval between the $(k-1)$th and kth encounters. Note that each waiting time is subject to stochastic variation. Hence, multiple samplings of waiting time have to be integrated to produce an accurate estimation of the regional density. The average waiting time $W_{ir}(k)$ between the $(k-1)$th and kth

robot encounters for a robot i in region r is computed as:

$$\begin{aligned} W_{ir}(k) = &\quad \frac{1}{n} w_{ir}(k) + \frac{n-1}{n} W_{ir}(k-1) \\ n = &\quad \min(k, n_{max}) \end{aligned} \tag{15}$$

where n_{max} is the maximum number of encounters that is monitored. This limit allows the robot to forget the early samplings of waiting time, which have become obsolete. The average target waiting time $W'_{ir}(k)$ is updated in the same manner. Both waiting times are updated according to the changing environment, and are inversely proportional to the robot and target densities in region r. The target density directly reflects the task demand of the region. The robot density reflects the amount of physical interference in the region, which is inversely proportional to the task demand. Therefore, the task demand $S_{ir}(k)$ of a region r can be determined by robot i as the ratio of the waiting times:

$$S_{ir}(k) = \frac{W_{ir}(k)}{W'_{ir}(k)} \ . \tag{16}$$

The task demand $S_{ir}(k)$ will be used in the self-organization of social dominance as well as in dynamic task allocation.

B. Self-Organization of Social Dominance

The division of labor in an ant colony is strongly influenced by its social dominance order [29], which self-organizes to match the task demands of the colony and the changing environment. Our scheme is inspired by this concept to move robots out of a region that has a lower target-to-robot density ratio than the other regions. Instead of fixing the dominance order [30], the social dominance of the robots in each coalition is self-organized according to their individual task performance. Robots in the same coalition engage in dominance contests at a regular interval τ if they are within communication range. The winner increases its tendency to stay in the current region while the loser increases its tendency to leave the current region and join another coalition in other regions. When robot i encounters robot j in region r, the probability of robot i winning a contest against robot j is defined as:

$$P(\text{robot } i \text{ winning}) = \frac{n_i^2 S_{ir}^2}{n_i^2 S_{ir}^2 + n_j^2 S_{jr}^2} \tag{17}$$

where S_{ir} and S_{jr} are respectively the task demand of region r determined by robot i and robot j, and n_i and n_j are the number of targets currently under observation by robot i and robot j respectively. Equation 17 implies that robot i would most likely win the contest if it observes more targets than robot j. However, if both are tracking the same number of targets, then their individual evaluation of the task demand can be used to differentiate them. This will distinguish a robot that has been observing the targets for a long time from another that just encounters the same number of targets.

To inject the influence of social dominance on the self-organization of robot coalitions, each time a robot i wins a contest (Eq. 17), it increases its tendency of staying in the current region, which is represented by the response threshold $\theta_i(t)$ to be used for dynamic task allocation:

$$\theta_i(t) = \theta_i(t-1) + \delta \tag{18}$$

where δ is small constant. Conversely, each time the robot loses, it decreases its tendency of staying in the region:

$$\theta_i(t) = \theta_i(t-1) - \delta \ . \tag{19}$$

θ_i varies in the range [0,1] to prevent robots from being overly submissive or dominating.

C. Dynamic Task Allocation

The distributed task allocation algorithm in ants can efficiently arrange the ants in proportion to the amount of work in the changing environment [31]. In a similar manner, our scheme aims to self-organize the robot coalitions according to the target distributions across the regions.

Our dynamic task allocation scheme is based on the notion of response thresholds [25]. In a threshold model, robots with low response thresholds respond more readily to lower levels of task demand than do robots with high response thresholds. Performing the task reduces the demand of the task. If robots with low thresholds perform the required tasks, the task demand will never reach the thresholds of the high-threshold robots. However, if the task demand increases, high-threshold robots will engage in performing the task.

MRTA strategies that utilize fixed response thresholds ([21], [27]) are incapable of responding effectively to dynamic environments [25]. In contrast, the thresholds in our scheme are continuously updated by the self-organizing process of social dominance.

To be effective in task allocation, a robot must at least have some knowledge of the task demands in its neighboring regions in order to make rational task decisions. To do so, robot i maintains a memory of the task demand S_{ir} of each region r (initialized to 0) and the amount of time T_{ir} that it previously spent in region r. T_{ir} can be used as a certainty measure of S_{ir}. In addition to computing S_{ir} using Equation 16, S_{ir} can also be updated when robot i receives a message from a neighboring robot j with S_{jr} less than S_{ir}. Then S_{ir} and T_{ir} are updated to take the values S_{jr} and T_{jr} respectively. In this manner, the task demands of the regions are kept in memory. Robot i can then predict which region has the greatest task demand and join that region. At every time interval of τ, if S_{ir} receives no update, the certainty value T_{ir} is decreased by τ while the task demand S_{ir} is increased by a small amount, such that its magnitude reflects the robot's motivation to explore.

Our distributed MRTA scheme uses a stochastic problem solving methodology. It is performed at intervals of τ to allow for multiple samplings of waiting time during each interval. The probability of a robot i to stay in its current region c is defined as:

$$P(\text{stay}) = \frac{S_{ic}^2}{S_{ic}^2 + (1-\theta_i)^2 + T_{ic}^{-2}} \ . \tag{20}$$

On the other hand, the probability of a robot i to leave region c to go to region r is defined as:

$$P(\text{leave}) = \frac{S_{ir}^2}{S_{ir}^2 + \theta_i^2 + T_{ir}^{-2} + d_{cr}^2} \tag{21}$$

where d_{cr} is the pre-computed collision-free distance between region c and region r, which can be viewed as the cost of task

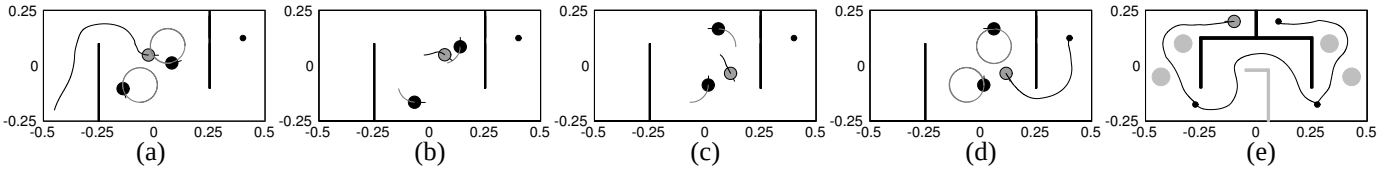


Fig. 5. (a–d) Motion of robot (gray) in an environment with two unforeseen obstacles (black) moving in anticlockwise circular paths. The robot successfully negotiated past the extended walls and the dynamic obstacles to reach the goal (small black dot). (e) Motion of robot (dark gray) in an environment with an unforeseen static obstacle (light gray). The robot successfully navigated through the checkpoints (small black dots) located at the doorways to reach the goal.

switching. Note that a robot that loses in the dominance contest in a coalition does not always leave the region. If it experiences a higher task demand in its region than in other regions, it will have a high tendency of remaining in its coalition.

From Equations 20 and 21, if the robot does not respond to any regions, it will not switch task and will remain in the current coalition. The robot may also respond to more than one region. This conflict is resolved with a method that is similar to Equation 17. The probability of a robot i choosing a region r that it has responded to is:

$$P(\text{choose}) = \frac{(S_{ir}\ln T_{ir})^2}{\sum_r (S_{ir}\ln T_{ir})^2} \ . \tag{22}$$

If the robot i chooses region r that is not the current region c, then it will employ cooperative EKMs to move through the checkpoints plotted by the planner to region r. The generation of checkpoints is performed by the approximate cell decomposition method for motion planning [11].

VI. Experiments and Discussion

A. Qualitative Evaluation of Cooperative EKMs

1) Robot Motion in Complex, Unpredictable Environments: This section presents a qualitative evaluation of the obstacle negotiation capabilities (i.e., self-protecting property) of a non-holonomic mobile robot endowed with cooperative EKMs in complex, unpredictable environments. The experiments were performed using Webots, a Khepera mobile robot simulator, which incorporated 10% noise in its sensors and actuators. 12 directed long-range sensors were also modelled around its body of radius 2.5 cm. Each sensor had a range of 17.5 cm, enabling the detection of obstacles at 20 cm or nearer from the robot's center, and a resolution of 0.5 cm to simulate noise.

Two tests were conducted to demonstrate the capabilities of cooperative EKMs in performing complex obstacle negotiation tasks. The environment for the first test consisted of three rooms connected by two doorways (Fig. 5(a)–(d)). The middle room contained two obstacles moving in anticlockwise circular paths. The robot began in the left-most room and was tasked to move to the right-most room. Test results show that the robot was able to negotiate past the extended walls and the dynamic obstacles to reach the goal.

The environment for the second test consisted of three rooms connected by two doorways and an unforeseen static obstacle (Fig. 5(e)). The robot began in the top corner of the left-most room and was tasked to move into the narrow corner of the right-most room via checkpoints identified by a motion planner [11]. The robot was able to move through the checkpoints to the goal by traversing between narrowly spaced convex obstacles in the first and the last room, and overcoming an unforeseen concave obstacle in the middle room. This result further confirms the effectiveness of cooperative EKMs in handling complex, unpredictable environments.

Similar tests have also been performed on robots that use potential fields. The robots were trapped by the extended walls and narrowly spaced obstacles in the first and second test respectively. This is because the obstacle avoidance behavior counteracted the target reaching behavior to cancel each other's effort.

These two tests show that for potential fields, though each behavior proposes an action that is optimal by itself, the vector sum of these action commands produces a combined action that may not satisfy the overall task. Cooperative EKMs, however, considers the preferences of each behavior and integrates them to determin an action that can satisfy each behavior to a certain degree. Such tightly coupled interaction between the behaviors and BCM enables the robot to achieve more complex tasks.

2) Cooperative Multi-Robot Tracking of Moving Targets: This section evaluates qualitatively the cooperative tracking capability of a team of robots, each fitted with cooperative EKMs, to maximize the coverage of multiple mobile targets (i.e., self-optimizing property). Two tests were conducted using Webots simulator with settings similar to those in Section VI-A.1. The first test (Fig. 6) was performed to highlight the advantages of cooperative EKMs over potential fields utilized by ([3], [6]) for the same task. The robot using potential fields got trapped by the static target while attempting to track all four targets. Eventually, the three mobile targets moved out of the robot's sensing range, causing the robot to observe only one out of four targets. In contrast, the robot fitted with cooperative EKMs was able to negotiate past the stationary target to track the three moving targets as well. All four targets were thus observed by the robot. The results of this test demonstrated that local minima situations could greatly decrease the coverage of targets by robots using potential fields. However, robots endowed with cooperative EKMs can still provide maximum coverage under these situations.

The next test (Fig. 7) illustrates how two robots endowed with cooperative EKMs cooperate to track four moving targets. When the targets were moving out of the robots' sensory range, the robot below chose to track the two targets moving to the bottom left while the robot above responded by tracking the two targets moving to the top right. In this manner, all targets could be observed by the robots. This test shows that the two robots can cooperate to track multiple moving targets without communicating with each other.

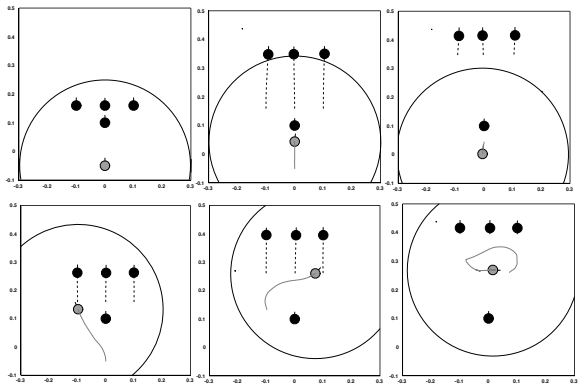


Fig. 6. (Top row) Robot (gray) using action superposition ASM got stuck at the stationary target. Eventually, the three mobile targets moved out of the robot's sensing range (circle). (Bottom row) Robot using cooperative EKMs could negotiate past the stationary target to track all the targets.

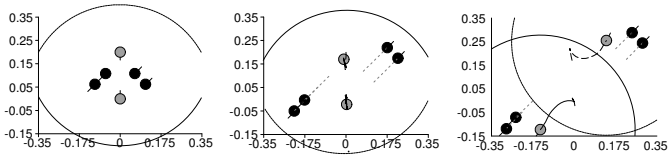


Fig. 7. Cooperative tracking of moving targets. When the targets were moving out of the robots' sensory range, the two robots moved in opposite directions to track the targets. In this way, all targets could be observed by the robots.

3) Self-Healing of Multi-Robot Formation: This section evaluates qualitatively the self-configuring capability of a team of robots, each fitted with cooperative EKMs, to repair unexpected damages to its formation. Fig. 8 shows the same environment in Fig. 1a covered by a robot team. When the robots in room 1 were removed (possibly due to bomb blast), the remaining 60 robots in the other rooms were able to self-configure and extend their coverage into this room. Hence, the formation is self-healed.

B. Quantitative Evaluation of Ant-Based MRTA and Cooperative EKMs

This section presents quantitative evaluations of the ant-based MRTA and cooperative EKMs schemes for distributed mobile sensor network coverage in a complex, unpredictable environment. The experiments were performed using Webots simulator with settings similar to those in Section VI-A.1. Each robot could also sense targets and kin robots at 0.3 m or nearer from its center and send messages to other robots that were less than 1 m away via short-range communication. A 4 m $\times$ 3 m environment (Fig. 1a) was used to house the Khepera robots and targets, which were randomly scattered initially. The number of robots varied from 5, 10 to 15, which corresponded to total robot sensing area of 11.8%, 23.6%, and 35.3% of the environment size. The mobile targets were forward-moving Braitenberg obstacle avoidance vehicles [32] that changed their direction and speed with 5% probability. The speed range of the robots and targets are 0-16 cm/s and 0-12 cm/s respectively.

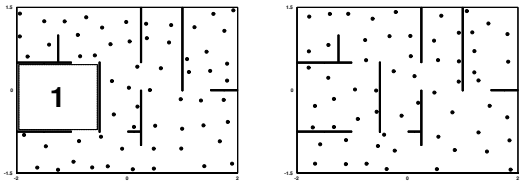


Fig. 8. Self-healing of multi-robot formation. When the robots in room 1 were removed (left), the remaining 60 robots were able to self-configure and extend their coverage into this room (right), thus repairing the damage.

1) Sensor Network Coverage: The first performance index determines the overall sensor network coverage of the robots:

$$\text{sensor network coverage} = \sum_{t=1}^{T} 100 \frac{n(t)}{NT} \tag{23}$$

where N is the total number of targets, n is the number of targets being tracked by the robots at time t, and the experiment lasts T amount of time. N and T are fixed respectively as 30 targets and 10000 time steps at intervals of 128 ms.

Using this index, a quantitative test was conducted to compare the network coverage of the robots adopting five distributed tracking strategies: (1) potential fields, (2) cooperative EKMs, (3) static placement, (4) auction-based negotiation, and (5) ant-based MRTA. Note that this index reflects the self-optimizing capability of the robots' tracking strategy. Unlike the latter three strategies, potential fields and cooperative EKMs are reactive motion control techniques that do not involve explicit task allocation. With static placement, static sensors are placed at least 0.6 m apart to ensure no overlap in coverage. With auction-based negotiation and ant-based MRTA, the robots are fitted with cooperative EKMs to coordinate their target tracking within a region, avoid obstacles, and navigate between regions.

Test results (Fig. 9a) show that ant-based MRTA provides better coverage than the other strategies. The differences in coverage between any two strategies have been verified using t-tests ($\alpha = 0.1$) to be statistically significant. Notice that 5 mobile robots endowed with our method can track better than 10 static sensors. Although auction-based negotiation uses complex negotiation, longer communication range, and more information about the robots and targets, it does not perform better than our ant-based scheme. This will be explained in the section of degree of specialization.

2) Total Coalitional Cost: The second performance index determines the total coalitional cost of the robots, which is inspired by the set partitioning problem [20]. Given a set of connected regions where coverage tasks are to be performed, and a set A of M robots, the task allocation algorithm assigns a robot coalition $C_r \subseteq A$ to the coverage task in region r such that (a) $\bigcup_r C_r = A$, (b) $\forall r \neq s, C_r \bigcap C_s = \emptyset$, and (c) each C_r has a positive cost $|(n_r/N) - (m_r/M)|$ where m_r and n_r are the number of robots and targets in region r respectively and N is the total number of targets. The objective is to minimize the total coalitional cost [20]:

$$\text{total coalitional cost} = \sum_r \left| \frac{n_r}{N} - \frac{m_r}{M} \right| . \tag{24}$$

This index varies within the range [0,2]. A coalitional cost of 0 implies that the robot distribution over all regions is exactly proportional to the target distribution. In this manner, interference between robots is at its minimum, which will improve overall coverage. High costs imply the opposite. Note

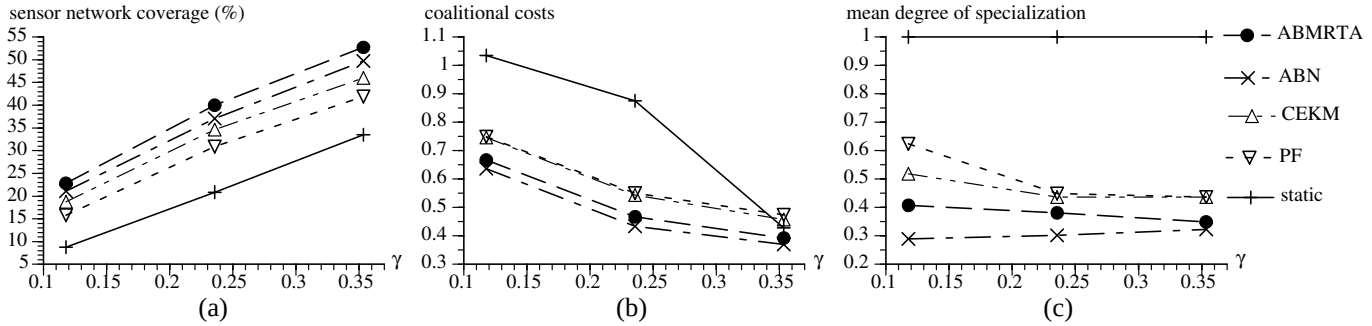


Fig. 9. Comparison of performance using different motion control and task allocation strategies: (a) Sensor network coverage, (b) total coalitional cost, and (c) mean degree of specialization. ABMRTA = Ant-Based MRTA, ABN = Auction-Based Negotiation, CEKM = Cooperative EKMs, PF = Potential Fields, and γ = total robot sensing area to environment size ratio.

that this index reflects the self-configuring capability of the robots' tracking strategy.

Test results (Fig. 9b) show that auction-based negotiation and ant-based MRTA have the lowest coalitional costs. Hence, we can conclude from Figures 9a and 9b that, with a lower cost, a higher coverage can be achieved. Although auction-based negotiation achieves slightly lower coalitional cost than ant-based MRTA, its coverage is lower. This will be explicated in the next section. Coalitional cost has been validated using t-tests ($\alpha = 0.1$) to be significantly different for various strategies except those without explicit task allocation (i.e., potential fields and cooperative EKMs). This is expected since they do not perform coalition formation, which account for their higher costs.

Coalitional cost is higher with fewer robots because with less robots, it is more difficult to achieve the same proportion of robots to that of the targets over all regions.

3) Degree of Specialization: To achieve low coalitional cost, the robot coalitions must be highly responsive, i.e., they can self-configure rapidly according to the changing distributions of targets across regions. In a temporally varying environment, an ant colony has to increase its responsiveness to cope with frequent changes in task demands by employing more generalist ants, which perform a range of tasks [33]. Similarly, we will like to examine the effect of our non-stationary task environment, induced by moving robots and targets, on the degree of specialization in the robots. Based on Shannon-Wiener information variable H, the third performance index quantifies the degree to which a robot specializes in a region:

$$\begin{aligned} \text{degree of specialization} = &\quad 1 - H \\ H = &\quad -\sum_{r} p_r \log_R p_r \end{aligned} \tag{25}$$

where p_r is the proportion of time a robot stays in region r for the task duration of T, and R is the total number of regions. This index varies within the range [0,1]. A degree of 1 implies the robot specializes in tracking only one region whereas a degree of 0 means the robot spends equal proportion of time tracking in all R regions.

Figure 9c shows the mean degree of specialization of all the robots, which is lower for auction-based negotiation and ant-based MRTA. Hence, we can conclude from Figures 9b and 9c that a larger number of generalist robots leads to a lower coalitional cost. Although auction-based negotiation achieves lower degree of specialization and coalitional cost than ant-based MRTA, its coverage is lower. This is because reducing the degree of specialization will incur more time in task switching and consequently decrease the time for performing the task [34]. In our test, this means that a robot endowed with auction-based negotiation will switch between several regions, thus incurring longer time in travelling between regions and searching for targets (Fig. 10). As a result, it spends less time in target tracking. This accounts for the poorer coverage of auction-based negotiation than ant-based MRTA.

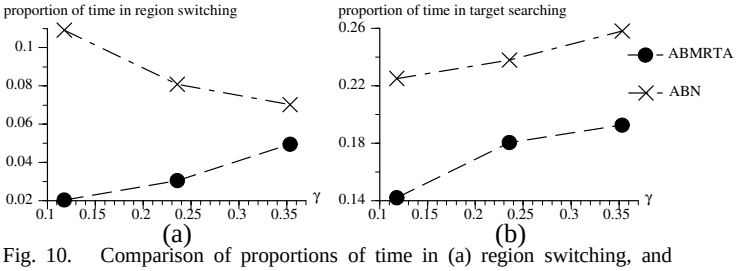


Fig. 10. Comparison of proportions of time in (a) region switching, and (b) target searching (i.e., not observing any targets) between explicit task allocation schemes. ABMRTA = Ant-Based MRTA, ABN = Auction-Based Negotiation, and γ = total robot sensing area to environment size ratio.

For ant-based MRTA, the mean degree of specialization is slightly higher with a smaller number of robots (Fig. 9c) because each robot receives fewer messages from the other robots. As a result, the robots are less certain about the task demands in other regions. This causes the robots to be more specialized and less inclined to explore other regions. Hence, they spend less time in region switching (Fig. 10a). On the other hand, the mean degree of specialization for auction-based negotiation is slightly lower with fewer robots because fewer robots are available for switching regions to minimize coalitional cost when the target distributions change. Therefore, each robot switches region more often (Fig. 10a). For explicit task allocation schemes, we can observe in Fig. 10b that a larger number of robots incurs longer target searching time. This is due to greater interference between robots. With cooperative EKMs or potential fields, fewer robots result in higher mean degree of specialization because the robots interfere less with each other and stay longer in a particular region.

The time spent in region switching and target searching (Fig. 10) can also reflect the amount of energy expended in robot motion that is not due to target tracking. As such, they can be used as metrics of energy efficiency. Even though ant-based MRTA provides better coverage than auction-based

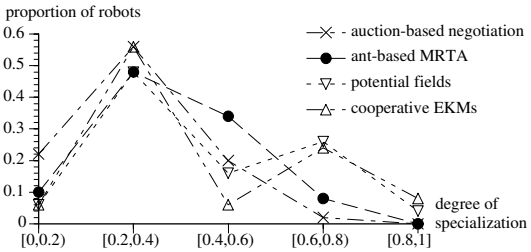


Fig. 11. Comparison of proportions of robots within different ranges of degrees of specialization.

negotiation, we can observe that it is more energy efficient as it spends less time in region switching and target searching.

Figure 11 shows the proportions of robots within different ranges of degrees of specialization for the case of 10 robots. Using ant-based MRTA and auction-based negotiation for explicit task allocation, most of the robots have degrees of specialization < 0.6. The other two methods without explicit task allocation have comparatively larger number of robots with degrees of specialization ≥ 0.6. Hence, the methods with explicit task allocation are less rigid to changes in regional task demands and incur lower coalitional cost.

4) Summary of Test Results: Compared to the other schemes, ant-based MRTA and auction-based negotiation have lower degree of specialization, coalitional cost, and higher coverage. But the degree of specialization cannot be too low as the cost of generalization (i.e., excessive time spent in region switching and target searching) would then exceed its benefits. This explains the higher coverage of ant-based MRTA over auction-based negotiation.

In the next few subsections, we will show more quantitative test results that address other important issues in a sensor network and its task environment.

5) Coverage of Evasive Targets: Our approach has been tested on the coverage of evasive targets that avoid the tracking robots. Compared with the tests of 15 robots tracking randomly moving targets (Fig. 9), our ant-based scheme can still maintain a 53% coverage. On the other hand, the coverage of static sensors dropped significantly from 34% to 10% whereas the coverage of the other schemes dropped slightly.

6) Robustness to Sensor Failures: Our scheme is robust to sensor failures (i.e., self-healing property), which is crucial for operating in dynamic, uncertain environments. For example, in the event that 5 mobile sensors fail completely, our scheme can still outperform a fully operational static sensor network (Fig. 9a).

Apart from sensor deaths, the sensors may also malfunction partially by experiencing faulty on-board sensing hardware or actuators. We have investigated the case of actuator failures in one-third of a network of 15 mobile sensors. This is similar to deploying a heterogeneous network of 10 mobile and 5 static sensors except that in our test, the sensors are not able to detect actuator malfunctions and be excluded from the task allocation process. Table I shows that when 5 sensors fail to move, the task allocation schemes achieve poorer coverage, higher coalitional cost, and higher mean degree of specialization. These 5 sensors that are unable to switch regions have degree of specialization of 1, which result in an overall increase

TABLE I

PERFORMANCE COMPARISON OF EXPLICIT TASK ALLOCATION SCHEMES.

	Ant-based MRTA			Auction-based negotiation		
Performance indices	FT	AF	ST	FT	AF	ST
Sensor network coverage (%)	52.74	47.79	62.80	49.75	43.79	58.31
Total coalitional cost	0.392	0.415	0.346	0.370	0.396	0.326
Mean degree of specialization	0.349	0.562	0.526	0.323	0.551	0.473

FT = Fast-moving Targets, AF = Actuator Failures, ST = Slow-moving Targets.

in the mean degree of specialization of the network. The loss of mobility in these sensors reduces the network's self-configuring capability, thus increasing the coalitional cost and consequently, decreasing the coverage (Section VI-B.2). Our ant-based scheme can still achieve better coverage than auction-based negotiation in the case of actuator failures. This has been explained in Section VI-B.3.

7) Varying Dynamism of Task Environment: The self-configuring capabilities of the explicit task allocation schemes have been evaluated under varying degrees of dynamism of the task environment. To do so, we vary the speed range at which the targets move. Slower-moving targets will change the regional target distributions less, thus making the task environment less dynamic. The speed range of the targets in the previous tests have been set to 0-12 cm/s. To compare with the previous results, we test the schemes with a reduced target speed range of 0-4 cm/s (i.e., less dynamic environment). As shown in Table I, when the targets move more slowly, the task allocation schemes achieve better coverage and lower coalitional cost but higher mean degree of specialization. Since the target distributions change slower, the robots do not need to switch regions so often. Hence, they tend to specialize in specific regions. The slow-changing target distributions also give the robots greater amount of time to self-configure their coalitions more proportionally, thus achieving lower coalitional cost. When the robot distributions are more proportional to that of the targets, a better coverage can be achieved (Section VI-B.2). Under different degrees of environmental dynamism, our ant-based scheme can provide better coverage than auction-based negotiation even though it has higher coalitional costs and mean degree of specialization (Section VI-B.3).

VII. CONCLUSION

The work in this paper describes a distributed layered architecture for resource-constrained cooperation of mobile sensors. This framework can be adapted to other autonomic multi-agent systems for distributed problem solving. By identifying the different granularities of coordination between agents (namely, for task decomposition, allocation, and execution), autonomic solutions can be devised for each of them. It has been demonstrated in Section VI how our task allocation and execution schemes employ self-organization techniques to achieve self-configuration, self-optimization, self-healing, and self-protection. Our ant-based scheme can be used to assign tasks optimally in an autonomic system. It requires a task demand/utility function to be specified (e.g., eq. 16), which can be used by autonomic agents for self-configuration to optimize task performance. Our cooperative EKMs strategy can be used by autonomic mobile agents to move towards their assigned

tasks, reduce interference, and maintain connectivity. To do so, the tasks and agents are modelled as targets or obstacles. Both schemes are robust to sensor failures and varying task dynamism. Automatic task decomposition will be considered in our future work on autonomic multi-agent systems.

REFERENCES

[1] V. Lesser, J. C. L. Ortiz, and M. Tambe, *Distributed Sensor Networks: A Multiagent Perspective*. Kluwer Acad. Publ., 2003.

[2] J. O'Rourke, *Art Gallery Theorems and Algorithms*. London, UK: Oxford Univ. Press, 1987.

[3] Z. Butler and D. Rus, "Event-based motion control for mobile sensor networks," *IEEE Pervasive Comput.*, vol. 2, no. 4, pp. 34–42, 2003.

[4] O. Yadgar, S. Kraus, and J. C. L. Ortiz, "Hierarchical information combination in large-scale multiagent resource management," in *Communications in Multiagent Systems*, M.-P. Huget, Ed. LNCS 2650, Springer-Verlag, 2003, pp. 129–145.

[5] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proc. IEEE Infocom*, 2001, pp. 1380–1387.

[6] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*, H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, Eds. NY: Springer, 2002, pp. 299–308.

[7] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Automat.*, vol. 20, no. 2, pp. 243–255, 2004.

[8] R. Alami, "On architectural and decisional issues for multi-robot cooperation," in *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*, A. C. Schultz, L. E. Parker, and F. E. Schneider, Eds. Kluwer Acad. Publ., 2003, pp. 3–14.

[9] D. Goldberg, V. Cicirello, M. B. Dias, R. Simmons, S. Smith, and A. Stentz, "Market-based multi-robot planning in a distributed layered architecture," in *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*, A. C. Schultz, L. E. Parker, and F. E. Schneider, Eds. Kluwer Acad. Publ., 2003, pp. 27–38.

[10] K. H. Low, W. K. Leow, and M. H. Ang, Jr., "Action selection for single- and multi-robot tasks using cooperative extended Kohonen maps," in *Proc. IJCAI*, 2003, pp. 1505–1506.

[11] ——, "A hybrid mobile robot architecture with integrated planning and control," in *Proc. AAMAS*, 2002, pp. 219–226.

[12] T. Huntsberger, P. Pirjanian, A. Trebi-Ollennu, H. D. Nayar, H. Aghazarian, A. J. Ganino, M. Garrett, S. S. Joshi, and P. S. Schenker, "CAMPOUT: A control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration," *IEEE Trans. Syst., Man, Cybern. A*, vol. 33, no. 5, pp. 550–559, 2003.

[13] K. H. Low, W. K. Leow, and M. H. Ang, Jr., "An ensemble of cooperative extended Kohonen maps for complex robot motion tasks," *Neural Comput.*, vol. 17, no. 6, pp. 1411–1445, 2005.

[14] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. 2, no. 1, pp. 14–23, 1986.

[15] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Automat.*, vol. 14, no. 6, pp. 926–939, 1998.

[16] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. IEEE ICRA*, 1991, pp. 1394–1404.

[17] K. H. Low, W. K. Leow, and M. H. Ang, Jr., "Task allocation via self-organizing swarm coalitions in distributed mobile sensor network," in *Proc. AAAI*, 2004, pp. 28–33.

[18] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Trans. Robot. Automat.*, vol. 18, no. 5, pp. 758–768, 2002.

[19] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multi-robot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, no. 2, pp. 220–240, 1998.

[20] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell.*, vol. 101, no. 1-2, pp. 165–200, 1998.

[21] M. J. B. Krieger and J.-B. Billeter, "The call of duty: Self-organized task allocation in a population of up to twelve mobile robots," *Robot. Auton. Syst.*, vol. 30, no. 1-2, pp. 65–84, 2000.

[22] M. B. Dias and A. Stentz, "Opportunistic optimization for market-based multirobot control," in *Proc. IEEE/RSJ IROS*, 2002, pp. 2714–2720.

[23] T. Kohonen, *Self-Organizing Maps*, 3rd ed. New York: Springer, 2000.

[24] C. Touzet, "Neural reinforcement learning for behavior synthesis," *Robot. Auton. Syst.*, vol. 22, no. 3-4, pp. 251–281, 1997.

[25] E. Bonabeau, M. Dorigo, and G. Théraulaz, Eds., *Swarm Intelligence: From Natural to Artificial Systems*. Oxford, UK: Oxford Univ. Press, 1999.

[26] D. M. Gordon, "Interaction patterns and task allocation in ant colonies," in *Information Processing in Social Insects*, C. Detrain, J. L. Deneubourg, and J. M. Pasteels, Eds. Basel, Switzerland: Birkhäuser Verlag, 1999, pp. 51–67.

[27] B. Jung and G. S. Sukhatme, "Tracking targets using multiple robots: The effects of environment occlusion," *Auton. Robots*, vol. 13, no. 3, pp. 191–205, 2002.

[28] A. E. Hirsh and D. M. Gordon, "Distributed problem solving in social insects," *Ann. Math. Artif. Intell.*, vol. 31, no. 1-4, pp. 199–221, 2001.

[29] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*. Princeton, NJ: Princeton Univ. Press, 2001.

[30] D. Goldberg and M. J. Matarić, "Interference as a tool for designing and evaluating multi-robot controllers," in *Proc. AAAI*, 1997, pp. 637–642.

[31] C. Tofts, "Algorithms for task allocation in ants (a study of temporal polyethism: Theory)," *Bull. Math. Biol.*, vol. 55, no. 5, pp. 891–918, 1993.

[32] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1984.

[33] D. S. Wilson and J. Yoshimura, "On the coexistence of specialists and generalists," *Am. Nat.*, vol. 144, no. 4, pp. 692–707, 1994.

[34] A. J. Spencer, I. D. Couzin, and N. R. Franks, "The dynamics of specialization and generalization within biological populations," *J. Complex Systems*, vol. 1, no. 1, pp. 115–127, 1998.