Reconfiguration Methods for Mobile Sensor Networks

AMAN KANSAL
Microsoft Research
WILLIAM KAISER, GREGORY POTTIE, and MANI SRIVASTAVA
University of California Los Angeles
and
GAURAV SUKHATME
University of Southern California

Motion may be used in sensor networks to change the network configuration for improving the sensing performance. We consider the problem of controlling motion in a distributed manner for a mobile sensor network for a specific form of motion capability. Mobility itself may have a high resource overhead, hence we exploit *motility*, a constrained form of mobility, which has very low overheads but provides significant reconfiguration potential. We present an architecture that allows each node in the network to learn the medium and phenomenon characteristics. We describe a quantitative metric for sensing performance that is concretely tied to real sensor and medium characteristics, rather than assuming an abstract range based model. The problem of determining the desirable network configuration is expressed as an optimization of this metric. We present a distributed optimization algorithm which computes a desirable network configuration, and adapts it to environmental changes. The relationship of the proposed algorithm to simulated annealing and incremental subgradient descent based methods is discussed. A key property of our algorithm is that convergence to a desirable configuration can be proved even though no global coordination is involved. A network protocol to implement this algorithm is discussed, followed by simulations and experiments on a laboratory test bed.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]; C.2.4 [**Computer Communication Networks**]: Distributed Systems

General Terms: Performance, Reliability, Measurement

Additional Key Words and Phrases: Mobile or actuator systems, network protocols, coverage, actuation, motion coordination, spatial resolution, mobility control

ACM Reference Format:

Kansal, A., Kaiser, W., Pottie, G., Srivastava, M., and Sukhatme, G. 2007. Reconfiguration methods for mobile sensor networks. ACM Trans. Sens. Netw. 3, 4, Article 22 (October 2007), 28 pages. DOI = 10.1145/1281492.1281497 http://doi.acm.org/10.1145/1281492.1281497

Author's address: A. Kansal; email: kansal@microsoft.com.

1. INTRODUCTION

We consider a sensor network in which limited motion capabilities are available for physical network reconfiguration. Our objective is to understand the various steps needed for selecting an optimal configuration and to design distributed algorithms for the same.

The optimization metric we consider is based on the sensing performance achieved by the sensor network. The sensing performance affects most sensor network applications directly. It is thus important to collect data at the highest fidelity possible within the given resource constraints. Motion can help achieve this in at least three ways:

(1) *Phenomenon Adaptivity:* Mobile sensors can move to sample the phenomenon precisely where highest resolution coverage is required.
(2) *Medium Adaptivity:* Mobile sensors can reposition themselves to overcome medium obstacles and anisotropies.
(3) *Increased Sensor Range:* Mobile sensors can cover a larger volume, depending on the acceptable motion delay.

The sensing advantages outweigh the cost of motion for appropriate system design choices.

An interesting analogy of using motion to leverage limited sensing resources occurs in human eyes. The resolution is highest in the center of the retina and gradually degrades toward the periphery [Perry and Geisler 2002]. This is an attempt to minimize sensing resources (in this case, neurons) and at the same time maximize spatial resolution and field of view. Rapid eye movements are used to shift the focus of attention toward a peripheral region when some event, such as motion, is detected by the low resolution peripheral view [Bichot et al. 2005].

1.1 Key Contributions

We design and implement a network of cameras with limited mobility (pan, tilt, and zoom), and use the motion for improving sensing performance. In particular we focus on developing a distributed algorithm that determines a desirable network configuration.

We begin with a system architecture that modularizes the various functions of the motion coordination task. A constrained form of mobility is used, to minimize the resource cost of motion itself. A realistic sensing performance metric is developed that models the actual coverage characteristics of the sensors, rather than an abstract disk-based model. The metric quantifies the resultant network coverage.

We describe a distributed algorithm to optimize this global metric using only local communication in a defined neighborhood. This allows scalability in the number of nodes and reduces the computational complexity of the optimization.

We prove that our algorithm converges to a desirable network configuration. Convergence is proved without assumptions on the differentiability or smoothness of the performance metric, but using only its dependence on sensor coverage characteristics. This is important because the realistic camera

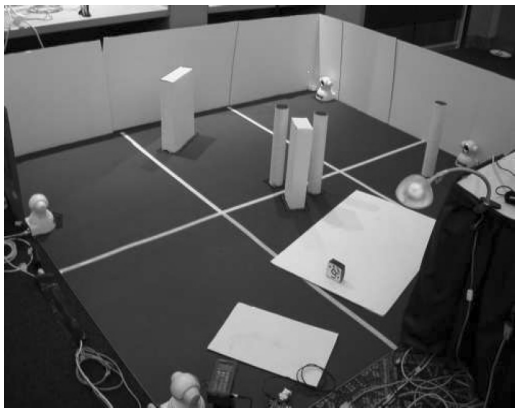


Fig. 1. Prototype system with motile cameras and supporting resources.

coverage model, along with the presence of obstacles in the medium, does not yield a closed-form expression for the objective function, and it is not immediately obvious if typical optimization procedures will converge when used with this function.

We present a network protocol that implements our distributed motion control algorithm, and addresses practical details such as the message passing required for local coordination, and the termination of the motion without global coordination. The effect of environment dynamics on the motion strategy is considered. The performance of the protocol is studied through simulations that model multiple sensor network deployments with obstacles.

Finally, we present an implementation of our proposed methods on a sensor network consisting of network cameras and the supporting resources used to learn the presence of obstacles in the medium.

1.2 Prototype System

The algorithms and methods discussed in this article are developed in the context of a network of cameras with limited motion capabilities. The sensed data consists of video streams from these cameras, which are processed frame by frame to detect events of interest. Detected events are then sensed at high resolution to provide higher-fidelity data for more sophisticated data processing algorithms, such as those for recognition or classification of the events. The medium contains physical obstacles that cause occlusions in the covered region. The test bed is shown in Figure 1. Apart from the cameras, it also has other supporting resources such as laser ranges for medium mapping and processing platforms.

2. SYSTEM ARCHITECTURE

It is important to understand the various tasks required to reconfigure the network in response to medium and phenomenon demands. We modularize these tasks into the blocks shown in Figure 2. Each of the blocks shown is considered below, except block III, which is discussed at length in subsequent sections.

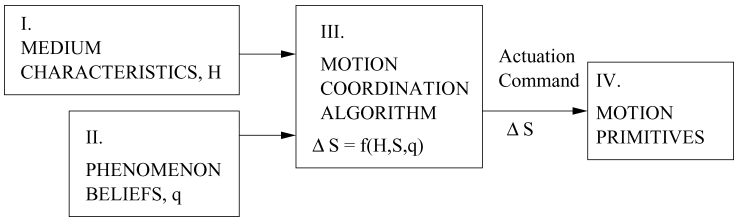


Fig. 2. A framework for managing actuation for sensing uncertainty control.

(a) Laser Ranger Node. (b) Event tag.

Fig. 3. System design (a) our prototype laser ranger node with pan and tilt capability, used to map the obstacles in the sensing medium, and (b) a sample tag, used as an event to be detected and recognized in our test bed.

2.1 Acquiring Medium Characteristics

Block I in the figure represents any means available to the network for learning the locations of the obstacles. If available, this knowledge helps the motion strategy to minimize occlusions. Such methods may be based on the use of range sensors [Harle and Hopper 2003] or in the case of cameras, the use of stereo-vision.

In our testbed, we use laser ranging to map the obstacles, as discussed in Kansal et al. [2005]. A Leica Disto Pro 4 laser ranging device is used, mounted on a pan-tilt platform and controlled using an Intel X-scale based Stargate processor board, shown in Figure 3(a). A key distinction from laser ranging used in robot navigation is that the range data is not being used for real time navigation and this allows using slower ranging devices, with a lower overhead in system cost.

2.2 Learning the Phenomenon Distribution

The system also needs means to discover events of interest, as represented by block II. This information can be used to direct high resolution sensing at the regions where events have occurred while coverage is maintained over a much larger area at a lower resolution.

In our test system, a low resolution image is used to detect targets. The target pattern consists of a set of markings that identify each target (Figure 3(b), from

Ipina and Lo [2001]), while we use color to detect the presence of a target in field of view. The detection and identification algorithms are not studied as part of this work; we use an off-the-shelf method [Ipina and Lo 2001] for identification and well-known color-finding methods for detection. The key feature of interest to our system design is that the detection task can succeed at a much lower resolution than required for reliable identification. Another example where low-resolution coverage may be used to direct the high-resolution sensing is a surveillance scenario, where low-resolution data may be used for motion detection and then high-resolution data helps classify the cause of motion.

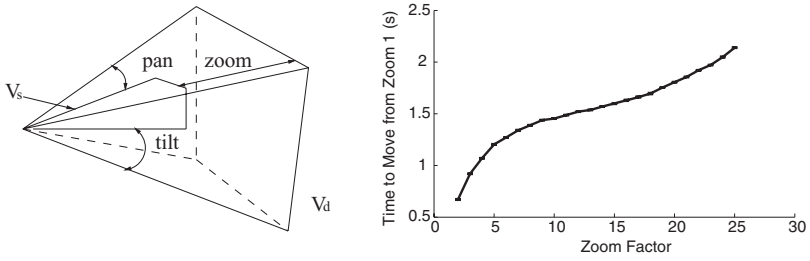
The locations of detected events over time may be used to approximate the distribution of events in space. This information is beneficial in choosing network configurations that are biased toward regions of high event probability. In our system, each sensor is required to know the locations of events only within a local neighborhood. Thus, while the event distribution is not centrally available, a distributed representation of it is built up, as we will describe in our motion coordination algorithm (block III).

2.3 Choice of Motion Primitives

Motion algorithms must be designed for the specific motion primitives (block IV) available to the system. The selection of the motion capabilities is affected by several considerations. It is also important to determine whether motion is useful at all compared to an alternative strategy of deploying a high density of static sensors, because motion itself has resource overheads. We consider a constrained form of motion, referred to as *motility*: the capability for pan, tilt, and zoom in a camera. Motility is well suited for sensor networks because it has the following advantages:

(1) Low energy. Only the sensor transducer has to be moved while the bulkier parts such as the motors, the battery, and the processor board, remain stationary.
(2) Minimal navigation support. Since motility does not depend on unreliable or arbitrary terrain characteristics, no extra sensors are needed to obtain terrain feedback.
(3) Infrastructural support, such as localization beacons or trajectory markings, is not required.
(4) Feasible in tethered nodes. Since the node itself does not move, motility is also feasible in power intensive or high bandwidth sensors, such as video sensors.

We observe that even such a limited form of motion has significant sensing advantages. To quantify the advantage, consider the following model to use motion: the cameras are used at low resolution to detect events and then reoriented to provide high resolution coverage at the location of the detected event. The extent of the advantage depends on two factors: (1) the difference between the resolution at which detection can take place and the resolution required for the final application task and (2) the time available for motion.



(a) Covered Volumes (b) Zoom Delay

Fig. 4. (a) Covered volumes for detection and sensing phases, with the pan, tilt, and zoom required to provide sensing resolution within the detection volume. (b) The time to change zoom measured as a function of the zoom step. The communication delay in sending the zoom command was separately characterized and has been subtracted from the motion time.

Suppose the resolution required for the final sensing objective, such as tag identification in our system, is such that an object dimension l_s in space should map to at least one pixel-length in the image. Given the angular field of view in horizontal direction, $\theta_{fov,h}$, of the camera and the image size in pixels, $L \times H$, we can compute the maximum distances R_s up to which a camera can reliably sense, using the following relationship:

$$tan(\theta_{fov,h}/2) = l_s L/2R_s. \tag{1}$$

The volume covered by the camera can be modeled as a pyramid of height R_s with the base area $Ll_s \times Hl_s$, and its vertex at the camera.[1] This yields the volume over which the desired sensing resolution is provided, say, V_s.

On the other hand, suppose detection can occur when a distance l_d is mapped to a single pixel-length, which similarly leads to a volume V_d over which detection is feasible. Figure 4(a) shows the volumes V_s and V_d. The camera can provide the resolution l_s at points outside V_s, but within V_d, by using its pan, tilt, and zoom motion. The maximum advantage that may occur in coverage due to motion is $G = V_d/V_s$, if the camera has motion capabilities to reach any point in V_d at the resolution l_s and sufficient time is available for that motion to occur. Figure 4(a) also shows the pan, tilt, and zoom ranges required to cover the entire V_d. Suppose the time taken for that motion, referred to as the actuation delay, is τ. The advantage G may be reduced if the tolerable actuation delay does not permit reaching some fraction of V_d. The tolerable delay τ depends on the event dynamics. As an example, consider the PTZ camera [Sony 2004] used in our test bed. This camera has a field of view $\theta_{fov,h} = 45^\circ$ in the horizontal direction and $\theta_{fov,v} = 30^\circ$ in the vertical direction, which determine the horizontal and vertical vertex angles respectively for V_d. It has pan and tilt ranges of 340° and 115° respectively, which are more than sufficient to cover the entire V_d. The zoom range is 25, which means that the maximum ratio R_d/R_s supported is 25. The pan and tilt speeds are $170^\circ s^{-1}$ and $57.5^\circ s^{-1}$ respectively.

[1]A small region near the vertex may be too close for focus limitations but its volume is assumed insignificant compared to the entire pyramid.

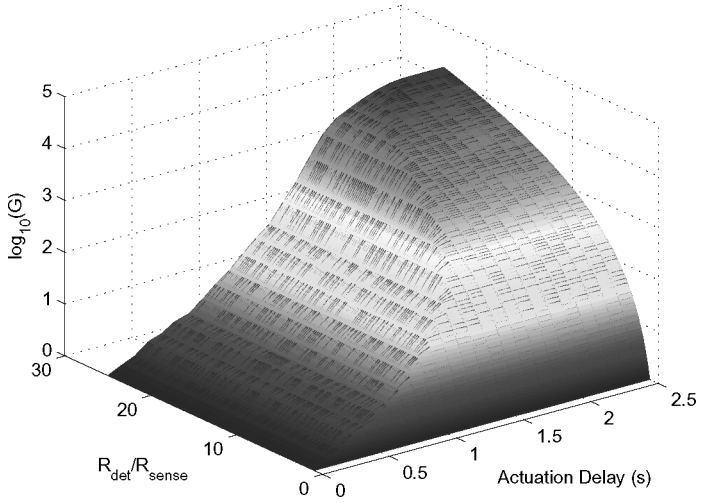
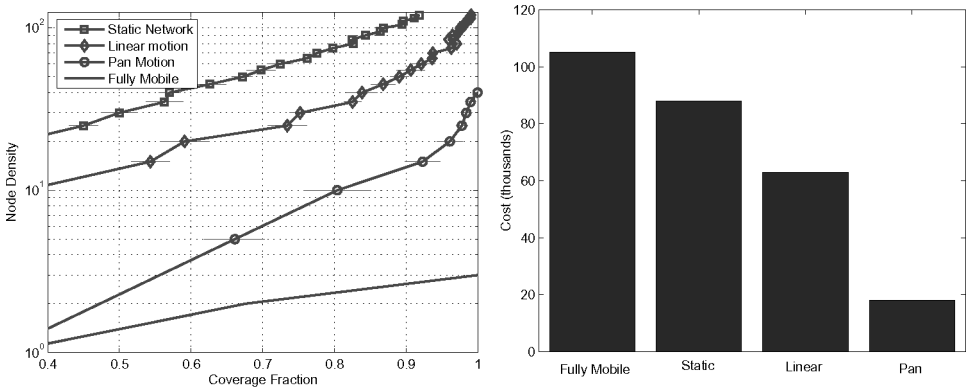


Fig. 5. Evaluating coverage gain due to motion with varying actuation delay and difference in detection and identification resolution.

The time taken for zoom is nonlinear and was measured experimentally for the entire zoom range (Figure 4(b)).

The detailed calculation of the advantage in covered volume due to motion, G, for these capabilities is skipped for brevity. The derived value of G is plotted in Figure 5 for various values of τ ranging from zero to the maximum time required to exploit the entire motion range and various values of R_d/R_s within the zoom range of the camera. Note that G is quite large and hence plotted on a log scale. The region in the figure corresponding to actuation delay above $1s$ and the ratio $R_d/R_s > 2$ shows more than an order of magnitude advantage in coverage. Applications with sensing requirements that lie in this region will benefit significantly from the algorithms discussed in this paper.

In addition to the multiple orders of magnitude advantage seen above by providing high-resolution coverage only on demand rather than at all times, motion also provides another advantage when multiple cameras are used in a network. Consider the area over which coverage is provided for detection using low resolution. An initial random deployment may have wasted sensing resources due to overlapped regions of coverage among cameras or certain cameras being blocked by obstacles. Motion can be used to reconfigure the camera poses to improve the detection coverage. The exact calculation of such advantage depends on the precise location of obstacles and the exact deployment in any given setting. To evaluate an expected advantage we simulate multiple random network configurations with random obstacle positions. We consider four different scenarios: cameras with no motion capability; cameras with only pan capability; cameras with a capability to move a small distance linearly, such as along a fixed track; and cameras with unconstrained robotic motion capability. The motion time is not considered a limitation here since such reconfiguration is assumed to occur only at infrequent intervals when there are significant changes in the obstacle configuration. Figure 6(a) shows the densities of cameras with these different



(a) Density Comparison (b) Cost Comparison

Fig. 6. Alternative deployments: (a) Coverage fraction with and without actuation at different deployment densities. Node density is the number of nodes in a $20m \times 20m$ square region, where each sensor covers a sector of 45° with radius $Rs = 7.3m$. The error bars show the standard deviation across 10 random topologies simulated. (b) Converting the node density to total network cost, at 90% coverage point.

capabilities required to achieve the same coverage. Clearly, mobile and motile cameras require much lower density. For scenarios where a small number of static cameras suffice, this may not be a big advantage, but in scenarios where the number of static cameras required is large, say 1000, the reduction in the number of cameras needed to, say 50, using pan motion for instance, may have huge benefits in ease of deployment, maintenance, and operation.

Also, since motile nodes, mobile nodes, and static nodes have different costs, it may be relevant to compare the overall system cost rather than node density. The cost needs to be evaluated for the particular sensors of interest for a system, and these costs may change over time. However, the factors responsible for the costs can be considered here. The cost of a static node includes the node production cost, c_n; the installation cost (placing it and providing it power and network connectivity), c_i; maintenance cost, c_m; and the back-end data processing cost, c_p, to process the partially processed data received from that node. The total system cost is $N(c_n + c_i + c_m + c_p)$. It is reasonable to assume that only c_n varies across the three types of nodes, while the other costs are similar. Let us make three further assumptions that make the comparison harsher on the motile nodes. First, we do not account for $(c_i + c_m + c_p)$. Installation costs as high as 75% of the system cost have been reported [Farrar 2001]. Only c_n is considered, which means that the savings in $N(c_i + c_m + c_p)$ due to a lower N for the motile nodes compared to static ones are being ignored. Second, the off-the-shelf fully mobile nodes do not have built-in reliable navigation and localization subsystems; rather, they have a traction platform with appropriate interfaces for adding deployment specific navigational support. We ignore the costs of these additions, thus losing the significant cost advantage that motile nodes have in that they do not need such navigation support. Third, the cost of motile node considered is with three motile capabilities—pan, tilt,

and zoom—even though only the pan capability is considered in the above density calculation. Obviously, the cost of a pan-only camera will be lower than one with all three motility primitives. Figure 6(b) shows this cost trade-off at the densities required for achieving 90% coverage, and even with the harsher comparison, the motile nodes offer a significant advantage.[2]

These considerations motivate our choice of using motility in our camera network. It may happen for other applications and sensor types that using static nodes, or fully mobile nodes, or a mix of the two, is a more optimal choice.

3. MOTION COORDINATION

We now consider block III of Figure 2: algorithms to achieve a desirable network configuration, and adapt it to the spatio-temporal dynamics of the environment. Our focus is on distributed methods, due to scalability concerns.

3.1 Sensing Performance

The first important consideration in motion coordination is the choice of the sensing performance metric to quantify the value of a given network configuration. Our camera coverage model depends on three parameters. The camera can view any point not closer than its minimum focal distance R_{min}, and not farther than R_{max} beyond which distance the spatial resolution is too poor to be of interest. The camera has an angular field of view θ_{fov}. The three parameters R_{max}, R_{min}, and θ_{fov}, change with the zoom setting of the camera. Given this model, the sensing performance is characterized as follows.

The first quantity of interest is the actuation delay achieved by the network configuration. As mentioned before, complete coverage is only provided at a resolution lower than required for the final sensing application, and the cameras are zoomed in to the relevant location when an event of interest is detected. Thus, it is important to characterize the delay in providing the required high resolution coverage after an event is detected. Let the region to be covered by the network be denoted by $\mathcal{A}$ and let p be a point in $\mathcal{A}$. Suppose an event is detected at a point p. For the motion capabilities chosen for our system, the actuation delay depends on the time taken by the camera to pan, tilt, and zoom for focusing at the point p. Denote this time by $\tau(p)$. It can be measured in terms of the motor capabilities of the node. Since the pan and zoom are driven by separate motors and can occur in parallel, the time required is:

$$\tau(p) = max\{\delta\theta_p * \omega_p, t_p(\delta_z)\}, \tag{2}$$

where $\delta\theta_p$ is the pan angle to be moved to direct the sensor at p, ω_p is the angular pan speed, δ_z is the change in zoom required for achieving the required classification resolution at p, and $t_p(\delta_z)$ is the time required for changing the zoom (Figure 4(b)). The tilt time can be considered similarly as the pan time,

[2]The costs considered are USD 800 for a static network camera [SNC-CS3N 2004], 1300 for a pan-tilt-zoom network camera [Sony 2004], and 35000 for a fully mobile node [Packbot 2004]. For linear actuation, we assumed the cost to be 1200, assuming the static node cost plus an additional track, motor, and motor controller. The specific static camera used in the comparison was chosen to match the PTZ camera in all other respects, such as optics, network connectivity and brand value.

but our system deployment is two-dimensional and hence tilt is ignored. When multiple sensors can observe a point p, for simplicity we assume no coordination among sensors for detection, and thus $\tau(p)$ is based on the best sensor observing p.

Second, the coverage metric must also characterize the detection performance. Several possible metrics exist for this, beginning with a purely geometric coverage metric which quantifies the area in line of sight of the sensors, to more sophisticated estimation theoretic metrics which quantify the probability of detection based on the noise models and collaboration among sensors. Let $c(p)$ denote the detection performance at point p. Since detection depends on resolution, we model $c(p)$ as proportional to the resolution at which a sensor s views p, measured in the number of pixels per unit area. This is a more accurate model than a purely distance based one, since the resolution depends not only on the distance but also the zoom setting of the camera and obstacles in the medium. When multiple sensors can observe a point p, as before we assume no coordination among sensors for detection, and thus $c(p)$ is based on the best sensor observing p.

To aggregate the above factors into a single metric, we consider a linear combination of the two metrics to determine the sensing performance, $f(p)$, at a point:

$$f(p) = w * c(p) + (1 - w)\{1/\tau(p)\}. \tag{3}$$

The choice of the weighting parameter w, where $0 \leq w \leq 1$, determines the proportion of the contribution of actuation delay and detection terms to the performance. The value $1/\tau(p)$ is capped to a maximum when $\tau(p) = 0$.

Third, apart from maximizing the probability of detection, and minimizing the actuation delay, it may also be of interest to maximize the area covered. Let $1(p)$ be a binary function which takes the value 1 at covered points and 0 at others.

Fourth, the network may have to be geared toward regions where more events are expected. Suppose that the event distribution function is known at all points $p \in \mathcal{A}$ and is denoted $q(p)$. Considering the above factors, we define the coverage utility at a point as:

$$u(p) = (1 - \alpha) f(p) * q(p) + \alpha * 1(p), \tag{4}$$

where α is a constant weight, $0 \leq \alpha \leq 1$, determining the significance of coverage alone. Note that the second term in Equation (4) is not multiplied by $q(p)$. This allows maximizing the area covered, even when events are concentrated in a small region, by setting the value of α to be high.

Suppose the state, or pose, of a sensor node i is denoted by $\mathbf{s}_i = \{\theta_p, z\}$, where θ_p and z represent the pan and zoom settings respectively. The network configuration for a network of N nodes is then denoted by $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N\}$. Given the pan and zoom ranges of a node, the node state $\mathbf{s}_i$ lies in a set $\mathcal{S}_i = \{[\theta_p^{min}, \theta_p^{max}], [1, z^{max}]\}$, where the superscripts min and max represent the minimum and maximum possible values respectively for the superscripted parameters. Let $\mathcal{S}$ represent the resultant set of possible network configurations.

It is given by:

$$\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_N. \tag{5}$$

Using the metric defined above for each point, we now express the overall utility of a network configuration, $\mathbf{S}$, as:

$$U(\mathbf{S}) = \sum_{p \in \mathcal{A}} u(p). \tag{6}$$

The summation requires that the region $\mathcal{A}$ be discretized. An obvious granularity to discretize is the value of l_s, the spatial distance that maps to one pixel, since the image is discretized at the pixel level. However, a coarser granularity may be used to reduce computation overhead.

The network reconfiguration problem can now be expressed as an optimization problem, where the goal is to determine the network configuration which maximizes the coverage utility:

$$\mathbf{S}_{opt} = \max_{\mathbf{S} \in \mathcal{S}} U(\mathbf{S}). \tag{7}$$

3.2 Distributed Motion Coordination Algorithm

Note that the performance metric as defined in the preceeding is a global metric for the entire network and depends on the state of multiple sensors. The phenomenon distribution and medium characteristics are are learnt locally. Thus, the performance metric is not known at individual sensors given their local views and only partial knowledge about $q(p)$.

Even if all the information could be made available centrally, it is not immediately obvious how the optimization problem can be solved efficiently. We first note that the optimization problem in (7) is NP-hard. Consider a special case of the problem: $\alpha = 1$. In this case the utility function is reduced to maximizing the coverage due to N sensors over a polygonal region with obstacles. So, if the problem in (7) can be solved in polynomial time, then a polynomial time solution exists to maximize coverage in a polygonal region with obstacles. However, that problem is known to be NP-hard [Chin and Ntafos 1988]. Thus, the problem in (7) is clearly NP-hard.

Further, the nature of $c(p)$ and $\tau(p)$ implies that $U(\mathbf{S})$ is not only non-linear but also not known in a closed form—it can only be evaluated numerically for each $\mathbf{S}$. As seen from equation (5), the search space is exponential in N and a brute force search over $\mathcal{S}$ is clearly out of the question.

A distributed solution will thus help ameliorate both communication overheads and computational complexity. Here, we define an algorithm to be distributed, if under its execution, each node exchanges messages only within a local neighborhood rather than with a central coordination entity or the entire network. The desirable local neighborhood for a node should minimize the number of communication hops required for coordination, and its size should not increase with the network size N.

Consider the utility function $U(\mathbf{S})$ defined in Equation (6). This function is inherently distributed in that the utility value at a point p can only be influenced by a subset of the sensors: the ones which have p in their line of sight and

within the maximum range of view from one or more of their possible poses. Denote the set of sensors that influence the utility value at p by $\beta(p)$. Collect together into a subregion $\mathcal{B}_j$ all those points which share the common influencing sensor set $\beta(p)$, and denote this sensor subset by β_j. Suppose the entire region $\mathcal{A}$ can be divided into K such subregions. By definition the $\{\mathcal{B}_j\}_{j=1}^{K}$ are disjoint (the corresponding β_j are not). There may be points in region $\mathcal{A}$ are not be covered by any of the sensors; as a matter of notation, these may be denoted by $\mathcal{B}_K$ corresponding to $\beta_K = \Phi$, the null set.

The utility function can now be decomposed into a summation over these disjoint subregions:

$$U(\mathbf{S}) = \sum_{j=1}^{K} \sum_{p \in \mathcal{B}_j} u(p). \tag{8}$$

Since each $\mathcal{B}_j$ corresponds to a sensor subset β_j and a set of N sensors may have 2^N subsets, the number of terms in the above summation is potentially exponential. We can however map the above decomposition to one where the number of terms is linear in N. Let γ_i denote the subset of sensors that is the union of all subsets β_j to which sensor i belongs:

$$\gamma_i = \{j | \exists \beta_k \quad \text{such} \quad \text{that} \quad (i, j) \in \beta_k\}. \tag{9}$$

The utility in the region within range of sensor i can be affected only by the sensors in γ_i. Given the state of each sensor in γ_i, sensor i can compute the set of points Γ_i at which i gives the highest $f(p)$ value. The subregions Γ_i are disjoint (though the sensor subsets γ_i are not) and $\{\Gamma_i\}_{i=1}^{N} \cup \mathcal{B}_K = \mathcal{A}$. Hence the utility function can be decomposed as:

$$U(\mathbf{S}) = \sum_{i=1}^{N} \sum_{p \in \Gamma_i} u(p). \tag{10}$$

The interesting property of this decomposition is that each sensor only needs to communicate with sensors in γ_i to determine $U(\mathbf{S})$ over Γ_i. We call γ_i the neighborhood of sensor i. The set γ_i includes all sensors that may affect the computation of Γ_i from any pose selected at those sensors. Suppose the region influenced by sensor i from all its poses is $\mathcal{V}_i$. Then the set γ_i includes potentially all sensors influencing any point in $\mathcal{V}_i$. The expected cardinality of this set depends on the deployment density and the maximum distance up to which a sensor may influence coverage. For most practical sensors, the distance up to which they may sense is bounded, which implies that the cardinality of γ_i stays constant even as the network size, N, increases, at a given deployment density.

We now describe a distributed algorithm that requires each sensor to coordinate only with sensors within γ_i and yet improve the global utility function. Suppose a mechanism is available at sensor i to ensure that when it is changing its pose, all other sensors in γ_i will remain static. The communication protocol to realize this mechanism and other implementation details will be discussed in the next section. The algorithm then proceeds as follows.

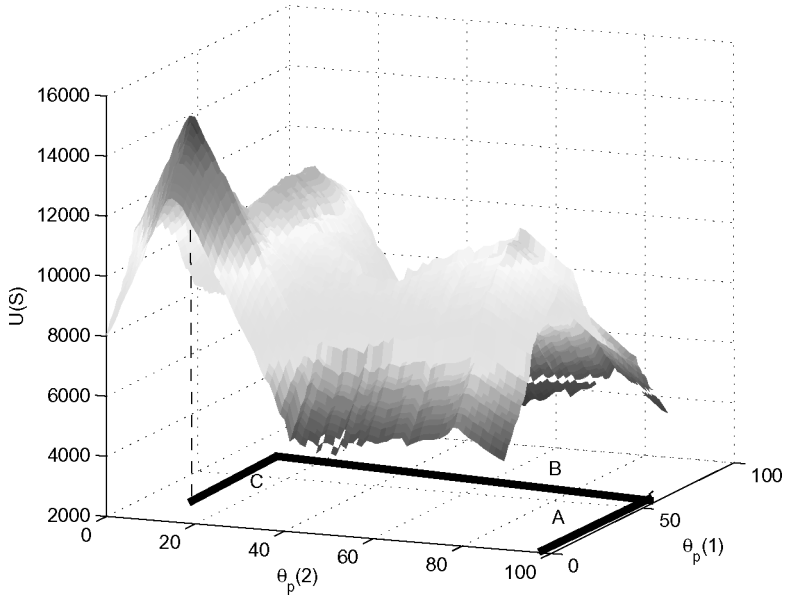


Fig. 7. Utility function for various possible pan pose combinations. The θ_p values are in degrees, measured counterclockwise from initial camera pose. The utility function has multiple local maxima, even for this simple scenario.

Sensor Pose Update: Sensor i searches over all its possible pan settings, and chooses the one for which the utility evaluated over $\mathcal{V}_i$ is maximum:

$$\theta_p(i) = \max_{\theta_p \in [\theta_p^{min}, \theta_p^{max}]} \sum_{p \in \mathcal{V}_i} u(p). \tag{11}$$

This is a one-dimensional search and thus computationally tractable. After selecting its best pan setting, the sensor similarly selects its best zoom setting. After this pose update, sensor i stops and some other sensor j waiting on i may update its pose, again making sure that sensors in γ_j stay static during its update step.

This change in pose at i affects all Γ_j for $j \in \gamma_i$. However, it does not affect sensors not in γ_j, and they may be changing their poses in parallel with sensor i using the same method. This property makes the algorithm distributed over neighborhoods γ_i. The sensors update their poses in an arbitrary order.

While the proposed algorithm is not guaranteed to find the global optimum, it has several desirable properties discussed in subsequent sections, including convergence to a stable state. Before analyzing the convergence and other properties of interest, let us visualize the above update procedure for a simple example with two sensors. Consider a rectangular region with a few obstacles and two cameras placed at diagonally opposite corners. Each camera can pan 90°, in its corner. To reduce the dimensionality of the search space to two for easier visualization, let us ignore zoom. The utility function achieved for all combinations of pan angles at each camera is shown in Figure 7. The pose selection algorithm described above proceeds as follows: sensor C1 first selects

a pan, $\theta_p(1)$, that maximizes the utility in its region of influence. This change in configuration is seen as line segment A in Figure 7. Then, sensor C2 selects a pan pose, $\theta_p(2)$, for this given value of $\theta_p(1)$, and line segment B shows the change in network configuration. In the next iteration, again C1 selects a new pan $\theta_p(1)$ moving along line segment C. The process continues until no camera is able to select a better pose than its existing pose. A distributed termination condition will be discussed in a later section. At each step, our algorithm searches along a line in the $2N$-dimensional search space, $\mathcal{S}$, and so we will refer to the above optimization procedure as incremental line search (ILS).

Note that in addition to the above method for configuring the network, another motion control algorithm is also required to move the cameras after an event is detected for providing the required high resolution coverage. For this task, we use a simple method that zooms in to the pixel location in the image where an event is detected. One advantage of this approach is that the spatial location of the detected event is not required. More sophisticated methods which may move multiple cameras in response to detected events are also of interest, but not studied as part of this work.

4. CONVERGENCE AND OTHER PROPERTIES OF ILS

An important property of this optimization procedure is that it does not require a closed-form expression for the utility function, but may proceed using only a numerical computation. This makes the above procedure amenable to several realistic performance metrics, in the presence of nonlinearities in actuation and the effect of obstacles.

Another important property is the convergence of the network configuration to a stable state. Even though the sensors may update their local poses without any global coordination, the network utility function will not oscillate. Further, the utility will reach a local maximum,[3] though not necessarily the global one, which is hard to ensure without global coordination. We prove this convergence property below.

THEOREM 4.1 (MONOTONICITY OF ILS). *Under ILS execution, the global network utility increases monotonically.*

PROOF. Assume first that only one sensor i in the entire network updates its pose at a time. Suppose the network configuration after iteration t is denoted $\mathbf{S}_t$. The region $\mathcal{A}$ can be viewed as consisting of two disjoint subregions $\mathcal{V}_i$ and $\mathcal{A} - \mathcal{V}_i$, and the utility can be expressed as:

$$U(\mathbf{S}_t) = \sum_{p \in \mathcal{V}_i} u(p) + \sum_{p \in \mathcal{A} - \mathcal{V}_i} u(p). \tag{12}$$

The update at sensor i does not affect the second term, and it thus stays constant at this iteration. The first term only depends on the region over which utility is computed by i, and since i choses a pose to maximize this term, this term can only increase or stay constant. Thus when only one sensor updates at a given

[3]As seen in Figure 7, the maximum found by ILS is not necessarily the local maximum closest to its starting point, unlike gradient descent based methods.

iteration, monotonicity holds:

$$U(\mathbf{S}_{t+1}) \geq U(\mathbf{S}_t). \tag{13}$$

Now, suppose multiple sensors update in parallel. However, the coordination over γ_i ensures that no sensor that affects $\mathcal{V}_i$ will update its pose when i is moving. Thus, we can consider the region $\mathcal{A} - \mathcal{V}_i$ independently. For another moving sensor j, we can decompose $\mathcal{A} - \mathcal{V}_i$ into two disjoint sets $\mathcal{A} - \mathcal{V}_i - \mathcal{V}_j$ and $\mathcal{V}_j$. An equation similar to (12) can then be written over these two subregions. Recursively applying the same argument, all sensors j that update in parallel within the coordination constraint over respective γ_j can only lead to a monotonic increase in the global network utility regardless of update order. □

Corollary: The network configuration converges to a stable state using ILS.

The proof of the corollary is easy to see from the fact that a monotonically increasing function can either grow unbounded or stabilize to a fixed value. Since the definition of $u(p)$ always yields a finite value and a summation over a finite region $\mathcal{A}$ ensures that U cannot grow infinitely, the ILS algorithm will cause the network configuration to converge to a stable state.

Note that the monotonicity is valid as long as other environmental parameters such as the obstacles and phenomenon distribution remain stationary. We will see later how environmental dynamics affect ILS and lead to a graceful performance degradation. The distributed operation ensures that such dynamics are incorporated into the algorithm's actions as soon as learned locally.

As long as the decomposition in Equation (6) holds, ILS can be used for arbitrary performance metrics, other than the detection and classification one discussed above. Generalizations include examples where the objective is not classification but tracking the movement of events after detection. Instead of time $\tau(p)$, the metric of interest would be the localization quality which might depend on the angle and distance from the nearest two sensors for any point p.

4.1 Relationship to Other Heuristics

Several heuristics have been explored in literature for optimization over combinatorially large search spaces. An important class of such heuristics is that based on stochastic local search methods, such as simulated annealing. While the optimization problem for network reconfiguration can be solved centrally using simulated annealing, it is not straightforward to extend it to a distributed algorithm. First, a global temperature state may have to be maintained. Second, the monotonicity property discussed in the preceeding does not hold for the annealing process, and convergence proofs are not available for distributed operation. The ILS algorithm always guarantees that any reconfiguration actions it takes will not worsen the utility compared to a static network. Parallelized simulated annealing algorithms, also sometimes referred to as distributed, are significantly different from our approach. They are typically designed for implementation on multiprocessor systems or computer clusters, for reducing the processing time compared to a single processor implementation. The global objective function is still available to each node, and each node may anneal the entire set of state variables. The multiple processors are mainly used to search

multiple random paths in parallel. The distributed nature of the objective function itself is not considered. Extension of stochastic local search methods to distributed settings, exploiting the decomposition of the optimization objective function over a relevant neighborhood set, is an open research problem.

Another potentially useful optimization heuristic is incremental subgradient descent. Its similarity to ILS lies in the fact that at each iteration step only a subset of parameters is updated in order to change the global objective function. While the utility function in our system is not differentiable and subgradients do not exist, such methods may be applied to our objective function at least centrally, by computing its derivatives using the method of finite differences. However, convergence is an issue when distributed operation is desired. Monotonicity does not hold. Convergence proofs of incremental subgradient descent, such as discussed in Solodov [1995], require the utility function to be continuous and differentiable. Extending such methods to operate with limited neighbor coordinations is still an open problem.

5. COORDINATION PROTOCOL IMPLEMENTATION

In this section we develop a motion coordination protocol to implement the distributed optimization algorithm described above. This includes determining a distributed mechanism to ensure coordination among the neighbors, exchanging the information required for calculating the motion steps, and determining the appropriate stopping criterion for the optimization in a distributed manner. This also involves learning the function $q(p)$ from local data without global normalization. Note that our protocol requires a sensor i to coordinate with other sensors in γ_i and since these have overlapping sensing regions, they are expected to be reachable within short network routes. They may not necessarily be on the same subnet in a LAN such as in case of two cameras mounted on two different buildings looking at the same street.

5.1 Protocol Specification

The first step is the determination of a mechanism to ensure that for each node i changing its pose, the set of nodes in γ_i stay static, and the number of nodes that can update their pose at any given time is maximized. This can be viewed as a graph coloring problem where for a given node i, all nodes in γ_i are treated to be adjacent to it. Each node must be allotted a color such that no two adjacent nodes have the same color while the number of colors is minimized. With this, nodes of one color can update their poses simultaneously and the number of iterations required to allow each node update once would be equal to the number of colors used. Efficient graph coloring heuristics are known, and hence if the entire network graph is known at a central location, any such heuristic may be used. However, we do not assume that a central view of the network graph is known and propose the following heuristic for achieving a graph coloring in a distributed manner. This procedure is inspired from common RTS-CTS based wireless MAC protocols for avoiding interference, though we do not need the communication channel to be wireless for its execution. The protocol at each node i is described in Figure 8.

(1) **Contention:** Wait for a random back-off period, and send Request To Update (RTU) to each node in γ_i. Wait for $T_{timeout}$. If no CONFLICT message is received, begin updating as per step 2. Each node which receives an RTU packet, refrains from sending an RTU until it receives an Update Finished (UF) packet from each node which had sent it an RTU. If a CONFLICT message is received, wait for UF message from the sender of CONFLICT message, and retry the RTU after random back-off. This ensures that within γ_i, only one node moves at any instant.

(2) **Update:** If an RTU is received anytime during this phase, send a CONFLICT message to the sender of the RTU.
   (a) **Medium and Phenomenon Information:** Use the medium mapping service to get information about obstacles within $\mathcal{V}_i$. Retrieve the $q(p)$ over the region $\mathcal{V}_i$ as acquired locally in the phenomenon learning phase.
   (b) **Neighbor Pose Information:** Look up the list of UF packets received up to now. UF packets contain the sender's chosen {pan, zoom} pose after an update. Thus the poses of all nodes within γ_i which have updated up to now are known. Using these and the medium information, the utility metric over $\mathcal{V}_i$ can be calculated.
   (c) **Pose Selection:** Select the pan and zoom settings as per the ILS algorithm. Transmit packet UF={identity, chosen pose}. Other nodes may now contend to update their poses. After a node receives a UF packet from every node from which it had received an RTU, it goes to Step 1 unless the termination condition (Step 3) is met.

(3) **Termination:** When a node discovers that the pose computed using the ILS algorithm does not change the utility over $\mathcal{V}_i$ by more than a threshold Δ compared to its current pose, it need not update its pose for that iteration. Each node maintains the past two UF packets received from its neighbors. Now, if the poses of all the neighbors have remained constant in the past two UF packets received, the node no longer needs to update its pose. If it has itself transmitted at least two UF packets, it terminates its motion algorithm until such time as another neighbor updates its pose or the information regarding $q(p)$ or the medium changes.

Fig. 8. Distributed motion coordination protocol for optimizing network configuration.

The protocol requires a random wait before sending a Request to Update (RTU) packet in order to ensure that over a long duration, most nodes will get a change to update their poses rather than a just few well-positioned nodes with smaller number of neighbors repeatedly winning the contention. No sequential ordering within a neighborhood is imposed, and it is not required that each node get equal number of chances to update its pose. This is compatible with the distributed termination condition, which may cause certain nodes to stop the update before others and hence this will not cause an undue interruption of updates at other nodes which are still changing their pose.

The protocol assumes a medium information service to learn the obstacles. An example implementation of this service using laser ranging, as shown in Section 2.1, is used in our experiments.

5.1.1 *Phenomenon Learning Phase.* The protocol also requires a service to acquire $q(p)$. This is carried out in a distributed manner as follows. Node i maintains locations of all events detected by itself or its neighbors within $\mathcal{V}_i$. We assume that when a node detects an event, it informs all its neighbors about

it. Since only a subregion of $\mathcal{V}_i$ is covered by selected poses of the sensor and its neighbors, events that occur in uncovered parts of $\mathcal{V}_i$, are not detected and do not affect the learning process. Since the total number of events in the network is not known to node i, $q(p)$ is not normalized but the count of events at a point p is incremented whenever an event is detected there. Thus, $q(p)$ acquired locally need not be a consistent probability density function in a global sense, but it serves its purpose of providing higher weight in the utility function, to points where more events occur. To prevent the count variables from overflowing, the $q(p)$ data is periodically aged as follows. After every time duration T_q, the sensor i scales the $q(p)$ at each point in $\mathcal{V}_i$ by 0.5. The aging step is:

$$q(p) \leftarrow 0.5 * q(p) + q'(p) \quad \forall \quad p \in \mathcal{V}_i, \tag{14}$$

where the temporary variable $q'(p)$ stores the event count since the previous aging step. This aging procedure thus gives higher weight to more recent events as the contribution of older events degrades by 0.5 every T_q. The duration T_q is expected to be a long duration over which sufficient events may be accumulated. The phenomenon distribution learning thus continues perennially and every T_q, the update of $q(p)$ may trigger a network reconfiguration to occur.

5.1.2 *Termination Condition.* The termination condition in the protocol is chosen for ease of distributed implementation. In centralized optimizations, a stopping criterion that can determine when to stop the iterations is to check when the change in the estimate of the optima has become insignificant over the past few iterations. However, in the distributed setting, the global utility metric is not available to any node. Even if the parameters in control of a particular node are not changing from one iteration to the next, the parameters at other nodes may be changing, and determining the stopping time would thus require global coordination. To avoid this global coordination, the termination condition proposed in Figure 8 allows each node to stop based only on local information. The value of Δ is set to 0.1% of the previously measured utility over $\mathcal{V}_i$ and may vary from sensor to sensor.

If a node receives an RTU packet from any neighbor followed by an UF packet that shows a change in pose, it is likely that this node has a further opportunity to optimize its pose and hence it will restart its motion coordination protocol. The node will also restart pose updates if the medium information affecting its covered region is updated or when the phenomenon density learning module indicates a change in $q(p)$ in this node's covered region. Thus a node stops pose updates based on local conditions and may reenter the update process when required. No node may know when the entire network has reached a stable state. Each node may stop its motion and restart it multiple times before reaching a stable condition. This approach is valid because of the monotonicity property of our algorithm. Due to monotonicity, the stable state is reached even if different parts of the network stop and restart motion at different times. This stopping criterion automatically ensures that updates to node poses due to medium changes lead to relevant pose changes at affected nodes without the need for globally sharing such information.

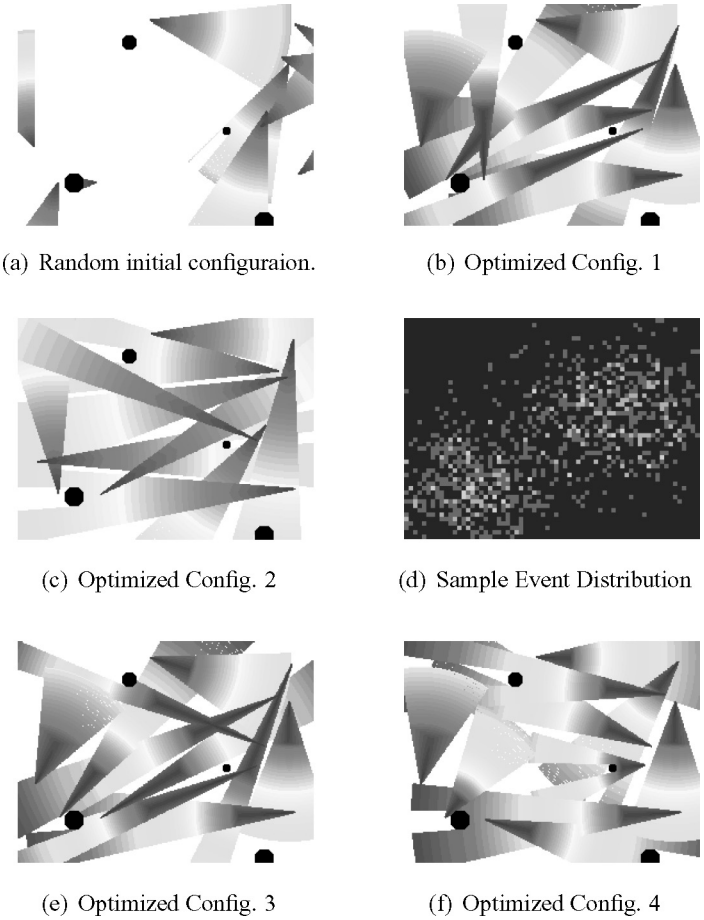
5.1.3 *Graceful Degradation.* The monotonicity property yields another desirable behavior in the distributed motion coordination protocol: graceful degradation in the presence of rapid environment dynamics. Under such conditions, the network configuration may continue to evolve without ever reaching a globally static configuration, due to frequent medium and phenomenon changes. Between any two instances of an environmental change, the monotonicity property holds. This implies that even when the environmental dynamics are faster than the convergence time of ILS, the network will only try to improve its configuration for the current situation. The improvement in configuration may be smaller than that which the ILS algorithm could have achieved in a static environment, thus degrading the performance gracefully with increasingly faster environment dynamics.

6. EVALUATION OF PROPOSED METHODS

There are three main performance considerations for a distributed optimization algorithm: whether it converges to a stable configuration, is this configuration a good one, and how long does it take to converge?

We have already proved convergence to a stable state. Since we used a realistic sensing performance model, our assumptions are valid under practical implementation constraints and hence the algorithm will converge to a stable state. As an illustration, let us visualize the operation of the algorithm for an example scenario and a randomly generated event distribution, for various choices of design parameters. Figure 9(a) shows a random initial deployment of 10 sensors with a few obstacles. The shade at a point represents the utility at that point due to the best-suited sensor for it, the white regions are uncovered and the black circles represent obstacles. Suppose the event distribution has not been acquired as yet. Also, we set $\alpha = 0$ first: ILS will try to maximize network utility—which may depend on a few points with high utility, rather than due to more covered points. The final configurations found by executing ILS with $w = 0.1$, corresponding to low weight for the detection performance and a higher weight for time to recognition, and at $w = 0.9$ are shown in Figures 9(b) and 9(c) respectively. The shades within covered regions vary since in one case the utility is dominated by time to access a point while in the other by the resolution of coverage. The exact choice of the weight parameter depends on desired behavior, which is application specific, and we do not dwell on it further here. Suppose next that the event distribution shown in Figure 9(d) is acquired. The figure axes represent the spatial extent and the shade at each location corresponds to the event density at that location. In the simulation this is generated as a sum of two Gaussian distributions. The optimized configuration is shown in Figure 9(e) with $w = 0.1$—the prominence of higher zoom (narrower field of view) in the diagonal region with higher event density is apparent. Suppose however that the area covered is also to be kept high and thus α is set to 0.8. This yields the configuration in Figure 9(f).

The second important issue is whether the stable configuration discovered above is a good one. More specifically, we wish to know if it is better than the configuration achieved without using ILS but by allowing each node to take



(a) Random initial configuraion. (b) Optimized Config. 1

(c) Optimized Config. 2 (d) Sample Event Distribution

(e) Optimized Config. 3 (f) Optimized Config. 4

Fig. 9. ILS optimization (a) random initial network configuration (b) optimized configuration with $w = 0.1$, showing significantly improved coverage than (a), (c) optimized coverage using $w = 0.9$, showing the different stable state than (b), (d) a sample event distribution, where brighter areas show higher event density (e) optimized configuration after learning the event distribution, and (f) optimized configuration with α set to a high value.

its individual decision, and if it is close to the global optimum. Few analytical proofs are available for convergence to a global optimum for distributed operation; rather we simulate how ILS behaves in random scenarios. Different orders in which the sensors update their poses lead to different search paths in the search space. Also, in the contention phase of the network protocol, certain sensors may win the contention more often than others, and update their pose more often. We simulate the operation of ILS for ten random contention success patterns, and plot the network utility in Figure 10. Note that the network utility is not available to any individual node, and is made available in the simulations by considering the poses of all nodes centrally for the purpose of evaluation. The solid curves show the evolution of utility with iteration for different contention success patterns. The dotted line shows the utility achieved

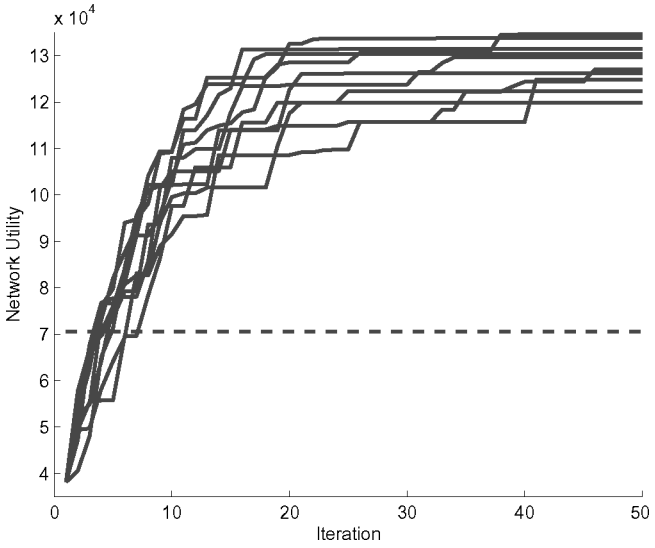


Fig. 10. Optimization accuracy: convergence with ten random search paths. Each curve shows a different random contention success pattern. The various converged states have a utility within 10.9% of the best case.

by a naïve method: updating the pose of each sensor locally without any coordination with its neighbors. The graph shows that while the convergence time varies among multiple contention success orders, the final utility with ILS is always better than a naïve local method.

The graph also shows that the final utilities, which are local minima, found in multiple random runs are within 10.9% of the best case. This suggests that ILS behaves well for several random instances. This is also expeted due to the relationship of ILS to incremental subgradient methods. It has been known that incremental methods are less likely to get stuck in poor local optima or stationary points, compared to gradient descent methods, since they do not suffer from the gradient descent effect of finding the local maxima closest to the starting point [Solodov 1995]. Based on such arguments and our simulation results, we conjecture that ILS discovers a good quality stable configuration with high probability. The monotonicity property always guarantees that ILS will never worsen a configuration, and the graph shows that using ILS has a significant advantage compared to a naïve approach.

We now address the third important consideration: time to converge. This is plotted in Figure 11 for a larger network, with $N = 25$ sensors spread across 100×50 meters, and maximum camera range $R_{max} = 8m$ at widest zoom, corresponding to $l = 1cm$ for the prototype camera as shown in section 2.3. Each curve corresponds to a randomly generated initial topology. To consider the worst case execution time, we consider each sensor's update as a separate iteration, even though several of these occur in parallel. Each iteration consists of the contention phase wait of $T_{timeout}$. This setting depends on the communication layer used. For instance, for the 802.11 MAC used in a WiFi-based LAN, the per hop delay is expected to be less than 10ms [Tickoo and Sikdar 2004;

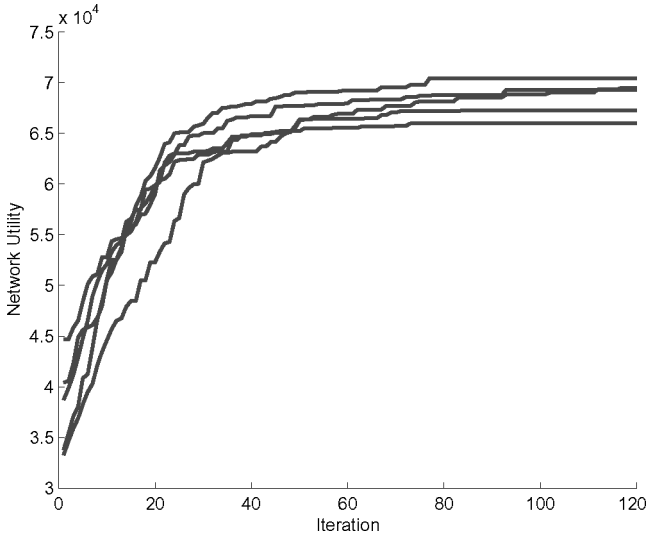
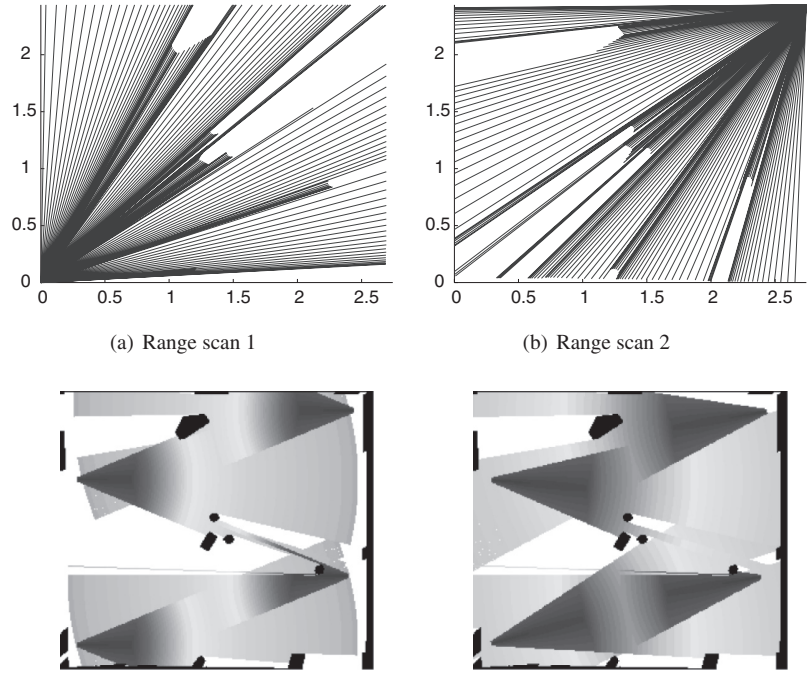


Fig. 11. Convergence: evolution of utility with increasing number of iterations. Each curve corresponds to a random location topology and random initial poses of the sensors.

Gupta and Kumar 2004] and even lower when an-ethernet based LAN is used. If γ_i spans 10 hops as a worst case estimate, the total round trip delay would be limited to $200ms$. At least one sensor wins the contention and computes its optimal pose. This has the delay of pan motion and zoom motion. The longest pan step may be 340° degrees, which takes 2 seconds. The longest zoom step (see Figure 4(b)) allowing a maximum zoom setting of 5 in detection mode (leaving the remaining zoom range of 20 available for providing high resolution coverage for recognition), is $1.3s$. A single update time is thus $2 + 1.3 + 0.2 = 3.5s$. The number of iterations required to achieve the final utility is about 80, as seen in the graph for each random topology, and even in the worst case this requires 4 minutes 40 seconds. The number of iterations is about 3 times the number of sensors, implying that on an average each sensor updates three times. Even with increasing network size, due to parallelism in updates, the convergence time will stay at the same order. Thus the stable state is discovered in only a few minutes for the hardware capabilities in our system. Since the phenomenon learning phase (which must wait for several events to occur in order to estimate or update the $q(p)$) and medium dynamics are expected to be much slower, convergence time is not an important consideration for our system, and ILS can repeatedly be used to update the network configuration whenever new environmental information becomes available. Time to converge may be an important concern in other applications of ILS where the system must deal with rapid dynamics.

6.1 Experiments

Let us now consider the execution of ILS on the test bed shown in Figure 1. The test bed has four cameras. Rather than beginning with a random configuration,



(a) Range scan 1 (b) Range scan 2

(c) Obstacles and Initial Config. (d) Final config.

Fig. 12. ILS Optimization. The shaded sectors represent camera coverage, and the shade at a location corresponds to the spatial resolution achieved by the best camera for that location. (a) Range scan from laser in bottom left corner (b) range scan from laser in top right corner, (c) map of obstacles in the medium and the initial network configuration, (d) optimized configuration found by ILS—it changes both the pan and zoom settings of the cameras to increase sensing performance characterized by time required for recognition.

we arrange the cameras along the edges to form a regular tessellation covering the entire rectangular region of interest. However, the test bed also has some obstacles in the rectangular region, which render the regular tessellation ineffective.

The first requirement is the information about medium obstacles. We collect this using laser ranging. Range scans from the bottom left corner and the top right are shown in Figures 12(a) and 12(b) repectively. Note that these scans exploit the adaptive scanning enhancements discussed in Kansal et al. [2005], and hence the scan step size is seen to be greater when scanning a structured environmental feature, such as a straight line, since it matches the adaptive algorithm's predicted structure. A backtracking procedure allows the algorithm to reduce the scan step size and rescan the missed portion in case a large deviation from the predicted structure is observed. Combining the two scans yields the obstacle map shown in Figure 12(c) by the dark patches; efficient communication methods for sharing the laser range data from multiple laser devices are discussed in Kansal et al. [2005]. Figure 12(c) also shows the initial network configuration.

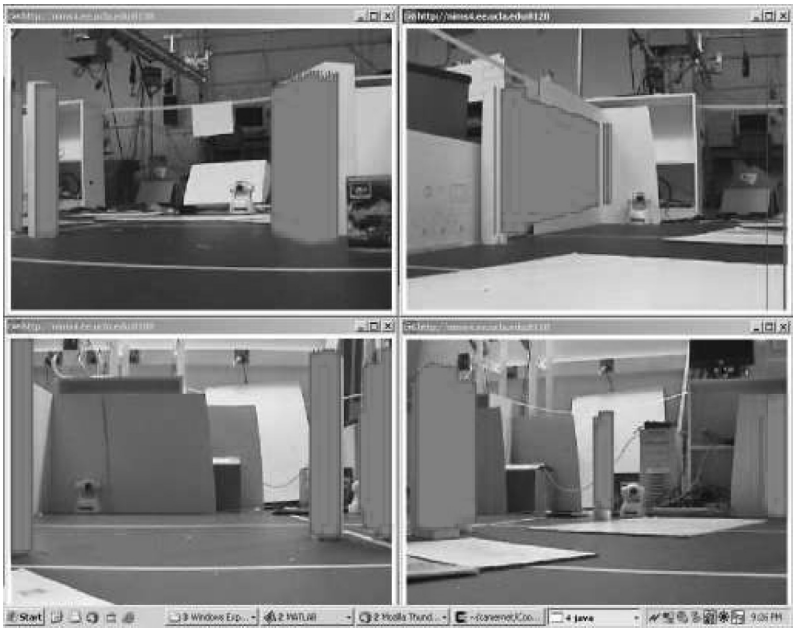


Fig. 13. GUI snapshots of the four processes controlling the network cameras in the testbed.

In this configuration, each sensor node has all other nodes as its neighbors. The test bed can thus be viewed as one neighborhood set in a larger network. The final network configuration found by the ILS algorithm is shown in Figure 12(d). The experiment used $w = 0.1$, thus giving more weight to the time required for recognition and the parameter α was set high to maximize covered area. These results not only verified that the ILS algorithm behaves as designed but also helped us ensure that all the algorithms and methods proposed can be implemented in a real system.

The test bed software also includes certain supporting components to enable visualization of the experiment's progress. The images captured by the network cameras are compressed JPEG files for ease of transfer over the network. Our camera controller software decompresses each image. It also calculates the occluded pixels based on the laser ranger readings. The occluded pixels are plotted superimposed on the raw image for visualizing the camera's field of view as it evolves under ILS. A snapshot of the visualization of four separate processes, each controlling one of the network cameras is shown in Figure 13.

7. RELATED WORK

Motion coordination algorithms have been proposed in mobile sensor networks for the purpose of improving communication performance [Goldenberg et al. 2004], deployment [Howard et al. 2002], and coverage [Butler and Rus 2003; Chu et al. 2004; Jung and Sukhatme 2002; Merrill et al. 2002] among other tasks. We consider a different coverage metric that combines detection performance and motion delay. This metric is better suited to characterize sensing

when motion is an integral part of the system. Another distinction of our work is that we have considered realistic sensor and motion models rather than assuming a circular disk-based coverage, isotropic medium, and the availability of reliable navigational systems for the mobile sensors.

There is considerable work on methods to determine appropriate network configurations spanning various fields of study including computer vision, sensor networks, and robotics.

Beginning with the art gallery problem, several efforts have tried to determine an optimal configuration of sensors to cover a given region [Chvatal 1975; O'Rourke 1987; Fisk 1978]. A variant that allows the use of mobile sensors is known as the watchmen tours problem [Carlsson et al. 1993; Efrat et al. 2000; LaValle et al. 1997]. However, this research is directed at finding the optimal solution in a central manner. Further, their methods are designed for abstract sensor models where a sensor has infinite range and is limited only by line of sight limitations due to the shape of the area to be covered. Some work has considered more realistic sensor models [Erdem and Sclaroff 2004], but discussion is limited to simple polygons since linear time algorithms are available to find the visible subregion of a simple polygon [Gindy and Avis 1981] (coverage regions with obstacles are thus not addressed). While these solutions have several interesting properties [Nilsson 1994], they are not directly applicable to our problem.

Distributed geometric optimization methods [Cortes et al. 2005; Olfati-Saber and Murray 2004] have also been used for mobile sensor network reconfiguration. However, that approach is developed for a differentiable optimization metric with idealized motion models. A gradient descent method is used that may converge to poor local optima. Extension to our specific problem with the more sophisticated sensor model and constrained motion capability is not obvious.

Another related class of methods is found in the use of estimation-theoretic optimization metrics and the application of information filters to coordinate network wide motion [Grocholsky et al. 2003; Geman and Jedynak 1996; Aranda et al. 2005]. However, most of these methods involve sharing some global information among nodes and are again designed with abstract sensing models.

There are certain other distributed optimization methods which use a distributed control law and show that it optimizes a global metric of interest, such as using a potential field or other linear control law based only on local neighbor interactions [Jadbabaie et al. 2003; Gazi and Passino 2004; Howard et al. 2002]. Such methods have several useful properties but are hard to design for given performance metrics. Also, convergence analysis used in these works, such as using matrix Laplacian or Lyapunov function based techniques may not directly apply when the motion control law is nonlinear, which is the case in our system.

The most closely related works focusing on the control of cameras with pan tilt, and zoom capabilities are Chu et al. [2004] and Collins et al. [2001]. In Chu et al. [2004], the cameras are constantly moved to track observed targets, using a factor graph [Kschischang et al. 2001]. Constant motion is also used to increase the covered area and search for potential targets. On the other hand, our algorithm does not constantly move cameras but proactively positions

cameras for making the network better suited to detect and classify targets as they emerge. A low-resolution view is used to detect targets, and high-resolution coverage is provided on demand where needed. Thus, the factor graph based tracking method could potentially be used as a higher layer method on top of our algorithm to execute the tracking functionality when targets are present. In Collins et al [2001] too, a set of cameras are controlled to optimize a sensing performance metric. However, the algorithm used is not distributed but is based on a centralized greedy heuristic. Also, the motion of the cameras is reactive and cameras are moved in response to existing tasks.

Another interesting work for sensor node location control [Butler and Rus 2003] has also considered the problem of configuring the network in accordance with event distributions. The event distribution, however, was tracked in a central manner and each sensor was required to know the entire distribution to map its location. The methods were designed for an isotropic medium, range-based coverage models, and sensor nodes with unconstrained motion capability. Implementation of such a method is not obvious for our system constraints.

Most previous systems attempt to maximize the spatial extent of coverage at a given resolution. Our goal to maximize resolution while balancing the opposing demand of maximizing the spatial extent of coverage. One of our system design assumptions is that phenomena of interest can be detected at lower resolution but high-resolution coverage is required for the specific sensing application using the data. We can use this assumption to provide higher-resolution sensing in regions with detected events, by reducing resolution in uninteresting regions. In a fixed-resolution system, on the other hand, when a camera orients toward one region, coverage in another region may be totally lost.

8. CONCLUSIONS

We have presented a framework for optimizing the configuration of a mobile sensor network with limited motion capabilities. The nature of motion capabilities was motivated by the resource constraints and the scale of the network. We presented practical methods to learn the environmental parameters which affect the network configuration and then designed a distributed optimization protocol to help adapt the network configuration to the sensing demands. A key convergence property of our algorithm was proved. A significant point to note is that all methods were designed in the context of realistic sensor models with due consideration of deployment constraints such as obstacles. Our approach enables using significantly reduced sensing resources to achieve an equivalent high-resolution coverage at the expense of a small actuation delay.

We explored the behavior of our algorithm with respect to some performance considerations, in addition to convergence, in simulations, and also demonstrated its implementation on a prototype sensor network. The sensing performance metric used here characterized the detection performance and the actuation delay. Future work includes integration of the exact motion coordination strategies that may be used for the sensing task after event detection and also extensions of our current methods to include the case of tracking mobile events. The ILS algorithm may further be extended to exploit the temporal

dimension and determine sequences of configurations to be visited over a given time period rather than finding a single optimal configuration.

REFERENCES

ARANDA, S., MARTINEZ, S., AND BULLO, F. 2005. On optimal sensor placement and motion coordination for target tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Barcelona, Spain.

BICHOT, N. P., ROSSI, A. F., AND DESIMONE, R. 2005. Parallel and serial neural mechanisms for visual search in macaque area v4. *Science 308*, 5721 (April), 529–534.

BUTLER, Z. AND RUS, D. 2003. Event-based motion control for mobile-sensor networks. *IEEE Pervas. Comput. 2*, 4, 34–42.

CARLSSON, S., NILSSON, B. J., AND NTAFOS, S. C. 1993. Optimum guard covers and *m*-watchmen routes for restricted polygons. *Inte. J. Computat. Geom. Appl. 3*, 1, 85–105.

CHIN, W. P. AND NTAFOS, S. 1988. Optimum watchman routes. *Inform. Process. Letters 28*, 39–44.

CHU, M., REICH, J., AND ZHAO, F. 2004. Distributed attention for large video sensor networks. In *Intelligent Distributed Surveillance Systems Seminar*. London, UK.

CHVATAL, V. 1975. A combinatorial theorem in plane geometry. *J. Combinat. Theory Series 18*, 39–41.

COLLINS, R., LIPTON, A., FUJIYOSHI, H., AND KANADE, T. 2001. Algorithms for cooperative multisensor surveillance. In *Proceedings of the IEEE 89*, 10, 1456–1477.

CORTES, J., MARTINEZ, S., AND BULLO, F. 2005. Analysis and design tools for distributed motion coordination. In *American Control Conference*. Portland, OR.

EFRAT, A., GUIBAS, L. J., HAR-PELED, S., LIN, D. C., MITCHELL, J. S. B., AND MURALI, T. M. 2000. Sweeping simple polygons with a chain of guards. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'00). San Francisco, CA. 927–936.

ERDEM, U. M. AND SCLAROFF, S. 2004. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *Omnivis2004, The 5th Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*. Prague, Czech Republic.

FARRAR, C. 2001. *Lecture Notes on Structural Health Monitoring Using Statistical Pattern Recognition* (Chapter Historical Overview of Structural Health Monitoring). Los Alamos Dynamics, Los Alamos, NM.

FISK, S. 1978. A short proof of Chvatal's watchmen theorem. *J. Combinat. Theory Series 24*, 374.

GAZI, V. AND PASSINO, K. M. 2004. Stability analysis of social foraging swarms. *IEEE Trans. Syst., Man, and Cybern (Part B: Cybernetics). 34*, 1.

GEMAN, D. AND JEDYNAK, B. 1996. An active testing model for tracking roads in satellite images. *IEEE Trans. Patt. Anal. Mach. Intell. 18*, 1, 1–14.

GINDY, H. E. AND AVIS, D. 1981. A linear algorithm for computing the visibility polygon from a point. *J. Algor. 2*, 186–197.

GOLDENBERG, D., LIN, J., MORSE, A. S., ROSEN, B., AND YANG, Y. R. 2004. Towards mobility as a network control primitive. In *ACM MobiHoc*. Tokyo, Japan.

GROCHOLSKY, B., MAKARENKO, A., KAUPP, T., AND DURRANT-WHYTE, H. 2003. Scalable control of decentralised sensor platforms. In *Information Processing in Sensor Networks 2nd International Workshop (IPSN'03)*. 96–112.

GUPTA, N. AND KUMAR, P. 2004. A performance analysis of the 802.11 wireless LAN medium access control. http://www.ece.rice.edu/čamp/MAC/MAC_Analysis.pdf. Submitted Internationals *Communications in Information and Systems*.

HARLE, R. K. AND HOPPER, A. 2003. Building world models by ray tracing within ceiling mounted positioning systems. In *UbiComp*. Seattle, WA. 1–17.

HOWARD, A., MATARIC, M., AND SUKHATME, G. 2002. An incremental self-deployment algorithm for mobile sensor networks. *Autonom. Robots J.* (Special Issue on Intelligent Embedded Systems). *13*, 2, 113–126.

IPINA, D. L. D. AND LO, S. L. 2001. Sentient computing for everyone. In *Proceedings of the IFIP TC6/WG6.1 3rd International Working Conference on New Developments in Distributed Applications and Interoperable Systems*. Kluwer, B.V., Deventer, The Netherlands, 41–54.

JADBABAIE, A., LIN, J., AND MORSE, A. S. 2003. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Autom. Control 48*, 6, 988–1001.

JUNG, B. AND SUKHATME, G. S. 2002. Tracking targets using multiple robots: the effect of environmental occlusion. *Autonom. Robots 13*, 3, 191–205.

KANSAL, A., CARWANA, J., KAISER, W. J., AND SRIVASTAVA, M. B. 2005. Acquiring medium models for sensing performance estimation. In *IEEE SECON*.

KSCHISCHANG, F., FERRY, B., AND LOELINGER, H.-A. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory 47*, 2, 498–519.

LAVALLE, S. M., LIN, D., GUIBAS, L. J., LATOMBE, J. C., AND MOTWANI, R. 1997. Finding an unpredictable target in a workspace with obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Albuquerque, NM, 737–742.

MERRILL, W., GIROD, L., ELSON, J., SOHRABI, K., NEWBERG, F., AND KAISER, W. 2002. Autonomous position location in distributed, embedded, wireless systems. In *Proceedings of the IEEE CAS Workshop on Wireless Communications and Networking* Pasadena, CA.

NILSSON, B. J. 1994. Guarding art galleries, methods for mobile guards. Ph.D. thesis, Department of Computer Science, Lund University.

OLFATI-SABER, R. AND MURRAY, R. M. 2004. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Auto. Control 49*, 9, 1520–1533.

O'ROURKE, J. 1987. *Art Gallery Theorems and Algorithms*. Oxford University Press, Oxford, UK.

PACKBOT 2004. Packbot, the next step in unmanned tactical mobile robots. http://www.packbot.com.

PERRY, J. AND GEISLER, W. 2002. Gaze-contingent real-time simulation of arbitrary visual fields. In *SPIE Proceedings of the Human Vision and Electronic Imaging*. San Jose, CA.

SNC-CS3N 2004. Sony SNCCS3N. http://bssc.sel.sony.com/Professional/. CS Mount Fixed Network Color Camera.

SOLODOV, M. V. 1995. Nonmonotone and perturbed optimization. Ph.D. thesis, University of Wisconsin, Madison, WI.

SONY 2004. Sony SNC-RZ30N data sheet. http://bssc.sel.sony.com/.

TICKOO, O. AND SIKDAR, B. 2004. Queueing analysis and delay mitigation in IEEE 802.11 random access Mac based wireless networks. In *IEEE Infocom*. Hong Kong, China.

Received December 2005; accepted March 2007