Tracking and coordination of multiple agents using sensor networks: system design, algorithms and experiments

Songhwai Oh, Luca Schenato, Phoebus Chen, and Shankar Sastry

***Abstract*— This paper considers the problem of pursuit evasion games (PEGs), where the objective of a group of pursuers is to chase and capture a group of evaders in minimum time with the aid of a sensor network. The main challenge in developing a real-time control system using sensor networks is the inconsistency in sensor measurements due to packet loss, communication delay, and false detections. We address this challenge by developing a real-time hierarchical control system, named *LochNess*, which decouples the estimation of evader states from the control of pursuers via multiple layers of data fusion. The multiple layers of data fusion convert noisy, inconsistent, and bursty sensor measurements into a consistent set of fused measurements. Three novel algorithms are developed for *LochNess*: multi-sensor fusion, hierarchical multi-target tracking, and multi-agent coordination algorithms. The multi-sensor fusion algorithm converts correlated sensor measurements into position estimates, the hierarchical multi-target tracking algorithm based on Markov chain Monte Carlo data association (MCMCDA) tracks an unknown number of targets, and the multi-agent coordination algorithm coordinates pursuers to chase and capture evaders using robust minimum-time control. The control system *LochNess* is evaluated in simulation and successfully demonstrated using a large-scale outdoor sensor network deployment.**

***Index Terms*— Sensor networks, networked control systems, pursuit evasion games, multi-target tracking, multi-agent coordination, multi-sensor fusion**

I. Introduction

Recently we have been witnessing dramatic advances in micro-electromechanical sensors (MEMS), digital signal processing (DSP) capabilities, computing, and low-power wireless radios which are revolutionizing our ability to build massively distributed, easily deployed, self-calibrating, disposable, wireless sensor networks [1, 2, 3]. Soon, the fabrication and commercialization of inexpensive millimeter-scale autonomous electromechanical devices containing a wide range of sensors, including acoustic, vibration, acceleration, pressure, temperature, humidity, magnetic, and biochemical sensors, will be readily available [4]. These potentially mobile devices, called "nodes", are provided with their own power supply [5] and can communicate with neighboring sensor nodes via low-power wireless communication to form a wireless ad-hoc sensor network with up to 100,000 nodes [6, 7]. Sensor networks can offer access to an unprecedented quantity of information about our environment, bringing about a revolution in the amount of control an individual has over his environment. The ever-decreasing cost of hardware and steady improvements in software will make sensor networks ubiquitous in many aspects of our lives [8] such as building comfort control [9], environmental monitoring [10], traffic control [11], manufacturing and plant automation [12], service robotics [13], and surveillance systems [14, 15].

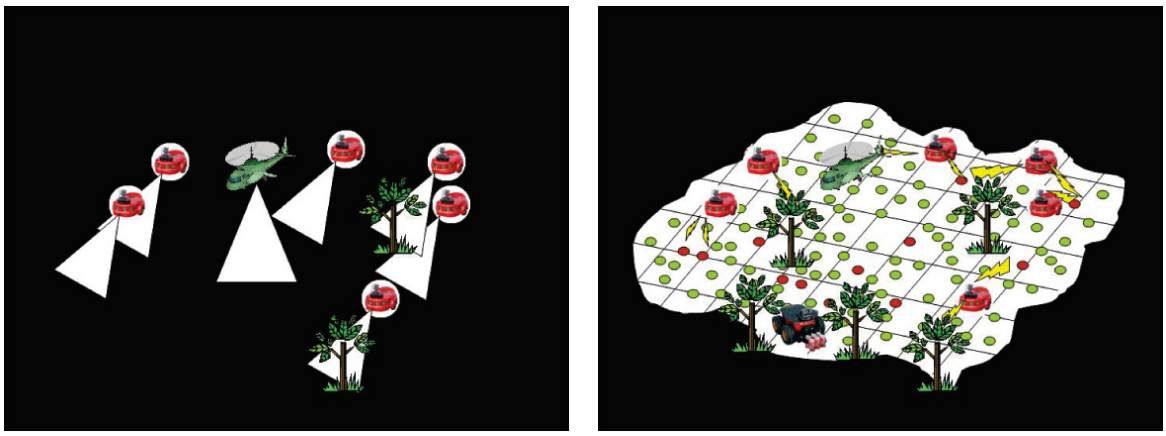
In particular, wireless sensor networks are useful in applications that require locating and tracking moving targets and real-time dispatching of resources. Typical examples include search-and-rescue operations, civil surveillance systems, inventory systems for moving parts in a warehouse, and search-and-capture missions in military scenarios. The analysis and design of such applications are often reformulated within the framework of pursuit evasion games (PEGs), a mathematical abstraction which addresses the problem of controlling a swarm of autonomous agents in the pursuit of one or more evaders [16, 17]. The locations of moving targets (evaders) are unknown and their detection is typically accomplished by employing a network of cameras or by searching the area using mobile vehicles (pursuers) with on-board high resolution sensors. However, networks of cameras are rather expensive and require complex image processing to properly fuse their information. On the other hand, mobile pursuers with their on-board cameras or ultrasonic sensors with a relatively small detection range can provide only local observability over the area of interest. Therefore, a time-consuming exploratory phase is required [18, 19]. This constraint makes the task of designing a cooperative pursuit algorithm harder because partial observability results in suboptimal pursuit policies (see Figure 1(a)). An inexpensive way to improve the overall performance of a PEG is to use wireless ad-hoc sensor networks [20]. With sensor networks, global observability of the field and long-distance communication are possible (see Figure 1(b)). Global pursuit policies can then be used to efficiently find the optimal solution regardless of the level of intelligence of the evaders. Also, with a sensor network, the number of pursuers needed is a function exclusively of the number of evaders and not the size of the field.

In this paper, we consider the problem of pursuit evasion games (PEGs), where the objective of a group of pursuers is to chase and capture a group of evaders in minimum time with the aid of a sensor network. The evaders can either move randomly to model moving vehicles in search-and-rescue and traffic control applications, or can adopt evasive maneuvers to model search-and-capture missions in military scenarios.

While sensor networks provide global observability, they cannot provide high quality measurements in a timely manner due to packet loss, communication delay, and false detections. This has been the main challenge in developing a real-time

Songhwai Oh, Phoebus Chen, and Shankar Sastry are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, Emails: {sho,phoebusc,sastry}@eecs.berkeley.edu.

Luca Schenato is with the Department of Information Engineering, University of Padova, Via Gradenigo 6/B, 31100 Padova, Italy. Email: schenato@dei.unipd.it.



(a) (b)

Fig. 1. (a) Sensor visibility in PEGs without sensor network. (b) Sensor visibility in PEGs with sensor network. Dots correspond to sensor nodes, each provided with a vehicle detection sensor. Courtesy of [20].

control system using sensor networks. In this paper, we address this challenge by developing a real-time hierarchical control system called *LochNess* (Large-scale "on-time" collaborative heterogeneous Networked embedded systems). *LochNess* decouples the estimation of evader states from the control of pursuers via multiple layers of data fusion. Although a sensor network generates noisy, inconsistent, and bursty measurements, the multiple layers of data fusion convert raw sensor measurements into fused measurements in a compact and consistent representation and forward the fused measurements to the pursuers' controllers in a timely manner.

The main contributions of this paper are (1) a real-time hierarchical control system *LochNess* for tracking and coordination using sensor networks; (2) a demonstration of the system on a large-scale sensor network deployment; and (3) three new algorithms developed for *LochNess*:

- A multi-sensor fusion algorithm that combines noisy and inconsistent sensor measurements locally. The algorithm produces coherent evader position reports and reduces the communication load on the network.
- A multi-target tracking algorithm that tracks an unknown number of targets (or evaders). The algorithm is a hierarchical extension of the Markov chain Monte Carlo data association (MCMCDA) [21] algorithm for sensor networks to add scalability. MCMCDA is a true approximation scheme for the *optimal Bayesian filter*; *i.e.*, when run with unlimited resources, it converges to the Bayesian solution [22]. MCMCDA is computationally efficient and robust against measurement noise and inconsistency (including packet loss and communication delay) [23]. In addition, MCMCDA operates with no or incomplete classification information, making it suitable for sensor networks. In fact, the performance of the algorithm can be improved given additional measurements to help identify the targets.
- A multi-agent coordination algorithm that assigns one pursuer to one evader such that the estimated time to capture the last evader is minimized based on the estimates computed by the multi-target tracking algorithm.

Our control system *LochNess* was successfully demonstrated using a large-scale sensor network. The system correctly found the number of evaders and their tracks and coordinated the pursuers to capture the evaders. Only a handful of the tracking algorithms in the literature that are designed for sensor networks have been demonstrated on a real sensor network deployment. Of these demonstrations, the algorithms are usually used to track a single target [14,24,25,26] or track multiple targets using classification [15]. To our knowledge, this paper presents the first demonstration of multi-target tracking using a sensor network without relying on classification.

The remainder of this paper is structured as follows. The overall architecture of *LochNess* for a PEG using a sensor network and formulations of multi-target tracking and multi-agent coordination are described in Section III. The components of *LochNess* are described in Section IV. The experimental results from the sensor network deployment are given in Section V.

II. Related Work: Target Tracking in Sensor Networks

One of the main applications of wireless ad-hoc sensor networks is surveillance. However, considering the resource constraints on each sensor node, the well known multi-target tracking algorithms such as joint probabilistic data association filter (JPDAF) [27] and multiple hypothesis tracker (MHT) [28, 29] are not feasible for sensor networks due to their exponential time and space complexities. As a result, many new tracking algorithms have been developed recently.

Most of the algorithms developed for sensor networks are designed for single-target tracking [30,15,14,24,25,31,26,32, 33,34,35,36] and some of these algorithms are applied to track multiple targets using classification [30, 15, 36] or heuristics, such as the nearest-neighbor filter (NNF[1]) [14]. A few algo-

[1]The NNF [27] processes the new measurements in some predefined order and associates each with the target whose predicted position is closest, thereby selecting a single association. Although effective under benign conditions, the NNF gives order-dependent results and breaks down under more difficult circumstances.

rithms are designed for multi-target tracking [37,38,39] where the complexity of the data association problem[2] inherent to multi-target tracking is avoided by classification [37, 39] or heuristics [38]. When tracking targets of a similar type or when reliable classification information is not available, the classification-based tracking algorithm behaves as the NNF. Considering the fact that the complexity of the data association problem is NP-hard [41,42], a heuristic approach breaks down under difficult circumstances. Furthermore, the measurement inconsistencies common in sensor networks, such as false alarms and missing measurements (due to missing detection or packet loss), are not fully addressed in many algorithms. On the contrary, the multi-target tracking algorithm developed in this paper is based on a rigorous probabilistic model and based on a true approximation scheme for the optimal Bayesian filter.

Tracking algorithms for sensor networks can be categorized according to their computational structure: centralized [15, 24, 33], hierarchical [34, 35], or distributed [30, 14, 25, 31, 26, 32, 36, 37, 38, 39]. However, since each sensor has only local sensing capability and its measurements are noisy and inconsistent, measurements from a single sensor and its neighboring sensors are not sufficient to initiate, maintain, disambiguate, and terminate tracks of multiple targets in the presence of clutter; it requires measurements from distant sensors. Considering the communication load and delay when exchanging measurements between distant sensors, a completely distributed approach to solve the multi-target tracking problem is not feasible for real-time applications. On the other hand, a completely centralized approach is not robust and scalable. In order to minimize the communication load and delay while being robust and scalable, a hierarchical architecture is considered in this paper.

III. Problem Formulation and Control System Architecture

In this paper, we consider the problem of pursuing multiple evaders over a region of interest (or the surveillance region). Evaders (or targets) arise at random in space and time, persist for a random length of time, and then cease to exist. When evaders appear, a group of pursuers is required to detect, chase and capture the group of evaders in minimum time with the aid of a sensor network. In order to solve this problem, we propose a hierarchical real-time control system *LochNess* which is shown in Figure 2. *LochNess* is composed of seven layers: the *sensor network*, the *multi-sensor fusion (MSF)* module, the *multi-target tracking (MTT)* modules, the *multi-track fusion (MTF)* module, the *multi-agent coordination (MAC)* module, the *path planner* module, and the *path follower* modules.

Sensors are spread over the surveillance region and form an ad-hoc network. The sensor network detects moving objects in the surveillance region and the MSF module converts the sensor measurements into target position estimates (or reports) using spatial correlation. This paper considers a hierarchical sensor network. In addition to regular sensor nodes ("Tier-1" nodes), we assume the availability of "Tier-2" nodes which have long-distance wireless links and more processing power. We assume that each Tier-2 node can communicate with its neighboring Tier-2 nodes. Examples of a Tier-2 node include high-bandwidth sensor nodes such as iMote and BTnode [43], gateway nodes such as Stargate, Intrinsyc Cerfcube, and PC104 [43], and the Tier-2 nodes designed for our experiment [44]. Each Tier-1 node is assigned to its nearest Tier-2 node and the Tier-1 nodes are grouped by Tier-2 nodes. We call the group of sensor nodes formed around a Tier-2 node a "tracking group". When a node detects a possible target, it listens to its neighbors for their measurements and fuses the measurements to forward to its Tier-2 node. Each Tier-2 node receives the fused measurements from its tracking group and the MTT module in each Tier-2 node estimates the number of evaders, the positions and velocities of the evaders, and the estimation error bounds. Each Tier-2 node communicates with its neighboring Tier-2 nodes when a target moves away from the region monitored by its tracking group. Lastly, the tracks estimated by the Tier-2 nodes are combined hierarchically by the MTF module at the base station.

The estimates computed by the MTF module are then used by the MAC module to estimate the expected capture times of all pursuer-evader pairs. Based on these estimates, the MAC module assigns one pursuer to one evader by solving the bottleneck assignment problem [45] such that the estimated time to capture the last evader is minimized. Once the assignments are determined, the path planner module computes a trajectory for each pursuer to capture its assigned evader in the least amount of time without colliding into other pursuers. Then, the base station transmits each trajectory to the path following controller of the corresponding pursuer. The path following controller modifies the pursuer's trajectory on the fly to avoid any obstacles sensed by the pursuer's on-board sensors. The path planning and path follower modules can be implemented using dynamic programming [46] or model predictive control [47]. In the paper, we focus on MSF, MTT, MTF, and MAC modules and they are described in Section IV. In the remainder of this section, we describe the sensor network model and the problem formulations of multi-target tracking and multi-agent coordination.

A. Sensor Network and Sensor Models

In this section, we describe the sensing models — the signal-strength and binary sensor models — and the sensor network model considered in this paper. A signal-strength sensor reports the range to a nearby target while a binary sensor reports only a binary value indicating whether an object is detected near the reporting sensor. The signal-strength sensor model is used for the development and analysis of our system while the binary sensor model is used in our experiments. While the signal-strength sensors provide better accuracy, our evaluation of the sensors developed for the experiments showed that the variability in the signal strength of the sensor reading prohibited extraction of ranging information. However, we found that the sensors were still effective as binary sensors.

[2]In multi-target tracking, the associations between measurements and targets are not completely known. The data association problem is to work out which measurements were generated by which targets; more precisely, we require a partition of measurements such that each element of a partition is a collection of measurements generated by a single target or clutter [40].

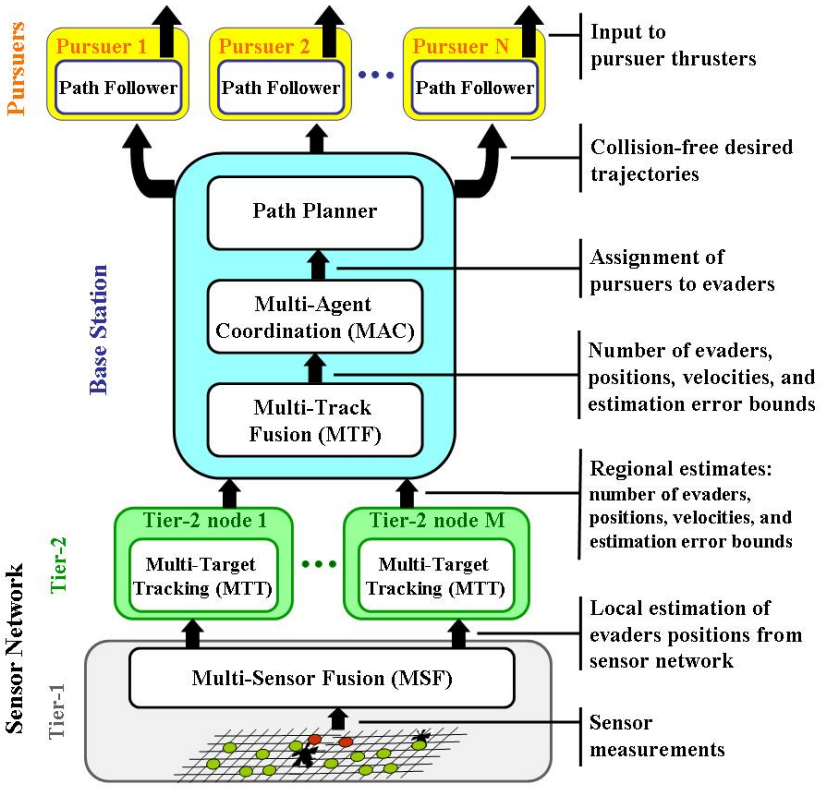


Fig. 2. *LochNess*: a hierarchical real-time control system architecture using sensor networks for multi-target tracking and multi-agent coordination.

We also found that binary sensors were much less sensitive to time synchronization errors than signal-strength sensors.

Let $N_{\rm s}$ be the number of sensor nodes, including both Tier-1 and Tier-2 nodes, deployed over the surveillance region $\mathcal{R} \subset \mathbb{R}^2$. Let $s_i \in \mathcal{R}$ be the location of the i-th sensor node and let $S = \{s_i : 1 \le i \le N_{\rm s}\}$. Let $N_{\rm ss} \ll N_{\rm s}$ be the number of Tier-2 nodes and let $s^{\rm s}_j \in S$ be the position of the j-th Tier-2 node, for $j = 1, \ldots, N_{\rm ss}$.

Signal-Strength Sensor Model

Let $R_{\rm s} \in \mathbb{R}$ be the sensing range. If there is an object at $x \in \mathcal{R}$, a sensor can detect the presence of the object. Each sensor records the sensor's signal strength,

$$z_i = \begin{cases} \frac{\beta}{1+\gamma\|s_i - x\|^\alpha} + w^{\rm s}_i, & \text{if } \|s_i - x\| \le R_{\rm s} \\ w^{\rm s}_i, & \text{if } \|s_i - x\| > R_{\rm s}, \end{cases} \quad (1)$$

where α, β and γ are constants specific to the sensor type, and we assume that z_i are normalized such that $w^{\rm s}_i$ has the standard Gaussian distribution. This signal-strength based sensor model is a general model for many sensors available in sensor networks, such as acoustic and magnetic sensors, and has been used frequently [14, 25, 26, 39].

Binary Sensor Model

For each sensor i, let R_i be the sensing region of i. R_i can have an arbitrary shape but we assume that it is known to the system. Let $z_i \in \{0, 1\}$ be the detection made by sensor i, such that sensor i reports $z_i = 1$ if it detects a moving object in R_i, and $z_i = 0$ otherwise. Let p_i be the detection probability and q_i be the false detection probability of sensor i.

Sensor Network Model

Let $G = (S, E)$ be a communication graph such that $(s_i, s_j) \in E$ if and only if node i can communicate directly to node j. Let $g : \{1, \ldots, N_{\rm s}\} \to \{1, \ldots, N_{\rm ss}\}$ be the assignment of each sensor to its nearest Tier-2 node such that $g(i) = j$ if $\|s_i - s^{\rm s}_j\| = \min_{k=1,\ldots,N_{\rm ss}} \|s_i - s^{\rm s}_k\|$. For a node i, if $g(i) = j$, the shortest path from s_i to $s^{\rm s}_j$ in G is denoted by $sp(i)$. In this paper, we assume that the length of $sp(i)$, *i.e.*, the number of communication links from node i to its Tier-2 node, is smaller when the physical distance between node i and its Tier-2 node is shorter. But if this is not the case, we can assign a node to the Tier-2 node with the fewest communication links between them.

Local sensor measurements are fused by the MSF module described in Section IV-A. Let $\hat{z}_i$ be a fused measurement originated from node i. Node i transmits the fused measurement $\hat{z}_i$ to the Tier-2 node $g(i)$ via the shortest path $sp(i)$. A transmission along an edge (s_i, s_j) on the path fails independently with probability $p_{\rm te}$ and the message never reaches the Tier-2 node. Transmission failures along an edge (s_i, s_j) may include failures from retransmissions from node i to node j. We can consider transmission failure as another form of a missing observation. If k is the number of hops required to relay data from a sensor node to its Tier-2 node, the probability of successful transmission decays exponentially as k increases. To overcome this problem, we use k independent paths to relay data if the reporting sensor node is k hops away from its Tier-2 node. The probability of successful communication $p_{\rm cs}$ from the reporting node i to its Tier-2 node

$g(i)$ can be computed as $p_{\text{cs}}(p_{\text{te}}, k) = 1 - \left(1 - (1 - p_{\text{te}})^k\right)^k$, where $k = |sp(i)|$ and $|sp(i)|$ denotes the cardinality of the set $sp(i)$.

We assume each node has the same probability p_{de} of delaying a message. If d_i is the number of (additional) delays on a message originating from the sensor i, then d_i is distributed as

$$p(d_i = d) = \binom{|sp(i)| + d - 1}{d} (1 - p_{\text{de}})^{|sp(i)|} (p_{\text{de}})^d. \quad (2)$$

We are modeling the number of (additional) delays by the negative binomial distribution. A negative binomial random variable represents the number of failures before reaching a fixed number of successes from Bernoulli trials. In our case, it is the number of delays before $|sp(i)|$ successful delay-free transmissions.

If the network is heavily loaded, the independence assumptions on transmission failure and communication delay may not hold. However, the model is realistic under moderate conditions and we have chosen it for its simplicity.

B. Multi-Target Tracking

The MTT and MTF modules of *LochNess* estimate the number of targets, positions and velocities of targets, and estimation error bounds. Since the number of targets is unknown and time-varying, we need a general formulation of the multi-target tracking problem. This section describes the multi-target tracking problem and two possible solutions.

Let $T_{\text{s}} \in \mathbb{Z}^+$ be the duration of surveillance. Let K be the number of targets that appear in the surveillance region $\mathcal{R}$ during the surveillance period. Each target k moves in $\mathcal{R}$ for some duration $[t_{\text{i}}^k, t_{\text{f}}^k] \subset [1, T_{\text{s}}]$. Notice that the exact values of K and $\{t_{\text{i}}^k, t_{\text{f}}^k\}$ are unknown. Each target arises at a random position in $\mathcal{R}$ at t_{i}^k, moves independently around $\mathcal{R}$ until t_{f}^k, and disappears. At each time, an existing target persists with probability $1 - p_{\text{z}}$ and disappears with probability p_{z}. The number of targets arising at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $\lambda_{\text{b}} V$ where λ_{b} is the birth rate of new targets per unit time, per unit volume, and V is the volume of $\mathcal{R}$. The initial position of a new target is uniformly distributed over $\mathcal{R}$.

Let $F^k : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ be the discrete-time dynamics of the target k, where n_x is the dimension of the state variable, and let $x^k(t) \in \mathbb{R}^{n_x}$ be the state of the target k at time t for $t = 1, \ldots, T_{\text{s}}$. The target k moves according to

$$x^k(t+1) = F^k(x^k(t)) + w^k(t), \text{ for } t = t_{\text{i}}^k, \ldots, t_{\text{f}}^k - 1, \quad (3)$$

where $w^k(t) \in \mathbb{R}^{n_x}$ are white noise processes. The white noise process is included to model non-rectilinear motions of targets. When a target is present, a noisy observation (or measurement[3]) of the state of the target is measured with a detection probability p_{d}. Notice that, with probability $1 - p_{\text{d}}$, the target is not detected and we call this a missing observation. There are also false alarms and the number of false alarms has a Poisson distribution with a parameter $\lambda_{\text{f}} V$,

[3]Note that the terms *observation* and *measurement* are used interchangeably in this paper.

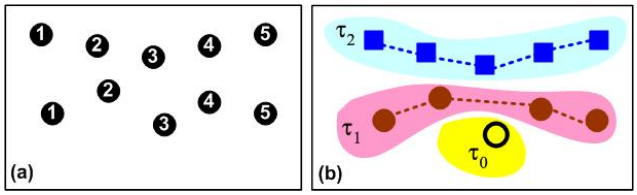


Fig. 3. (a) An example of observations Y (each circle represents an observation and numbers represent observation times). (b) An example of a partition ω of Y.

where λ_{f} is the false alarm rate per unit time, per unit volume. Let $n(t)$ be the number of observations at time t, including both noisy observations and false alarms. Let $y^j(t) \in \mathbb{R}^{n_y}$ be the j-th observation at time t for $j = 1, \ldots, n(t)$, where n_y is the dimension of each observation vector. Each target generates a unique observation at each sampling time if it is detected. Let $H^j : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ be the observation model. Then the observations are generated as follows:

$$y^j(t) = \begin{cases} H^j(x^k(t)) + v^j(t) & \text{if } y^j(t) \text{ is from } x^k(t) \\ u_{\text{f}}(t) & \text{otherwise,} \end{cases} \quad (4)$$

where $v^j(t) \in \mathbb{R}^{n_y}$ are white noise processes and $u_{\text{f}}(t) \sim \text{Unif}(\mathcal{R})$ is a random process for false alarms. We assume that the targets are indistinguishable in this paper, but if observations include target type or attribute information, the state variable can be extended to include target type information as done in [48].

The main objective of the multi-target tracking problem is to estimate K, $\{t_{\text{i}}^k, t_{\text{f}}^k\}$ and $\{x^k(t) : t_{\text{i}}^k \le t \le t_{\text{f}}^k\}$, for $k = 1, \ldots, K$, from noisy observations.

Let $Y(t) = \{y^j(t) : j = 1, \ldots, n(t)\}$ be all measurements at time t and $Y = \{Y(t) : 1 \le t \le T_{\text{s}}\}$ be all measurements from $t = 1$ to $t = T_{\text{s}}$. Let Ω be a collection of partitions of Y such that, for $\omega \in \Omega$, $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$, where τ_0 is a set of false alarms and τ_k is a set of measurements from target k for $k = 1, \ldots, K$. Note that ω is also known as a *joint association event* in literature. More formally, ω is defined as following.

1) $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$;
2) $\bigcup_{k=0}^{K} \tau_k = Y$ and $\tau_i \cap \tau_j = \emptyset$ for $i \neq j$;
3) τ_0 is a set of false alarms;
4) $|\tau_k \cap Y(t)| \le 1$ for $k = 1, \ldots, K$ and $t = 1, \ldots, T_{\text{s}}$; and
5) $|\tau_k| \ge 2$ for $k = 1, \ldots, K$.

An example of a partition is shown in Figure 3. Here, K is the number of tracks for the given partition $\omega \in \Omega$. We call τ_k a track when there is no confusion although the actual track is the set of estimated states from the observations τ_k. This is because we assume there is a deterministic function that returns a set of estimated states given a set of observations. A track is assumed to contain at least two observations since we cannot distinguish a track with a single observation from a false alarm, assuming $\lambda_{\text{f}} > 0$. For special cases in which $p_{\text{d}} = 1$ or $\lambda_{\text{f}} = 0$, the definition of Ω can be adjusted accordingly.

Let $n_{\text{e}}(t-1)$ be the number of targets at time $t-1$, $n_{\text{z}}(t)$ be the number of targets terminated at time t and $n_{\text{c}}(t) = n_{\text{e}}(t-1) - n_{\text{z}}(t)$ be the number of targets from time $t-1$

that have not terminated at time t. Let $n_{\rm b}(t)$ be the number of new targets at time t, $n_{\rm d}(t)$ be the number of actual target detections at time t and $n_{\rm u}(t) = n_{\rm c}(t) + n_{\rm b}(t) - n_{\rm d}(t)$ be the number of undetected targets. Finally, let $n_{\rm f}(t) = n(t) - n_{\rm d}(t)$ be the number of false alarms. Using the Bayes rule, it can be shown that the posterior of ω is [22]:

$$\begin{aligned} P(\omega|Y) &\propto P(\omega)\cdot P(Y|\omega) \\ &\propto \prod_{t=1}^{T_{\rm s}} p_{\rm z}^{n_{\rm z}(t)}(1-p_{\rm z})^{n_{\rm c}(t)} p_{\rm d}^{n_{\rm d}(t)}(1-p_{\rm d})^{n_{\rm u}(t)} \\ &\times \prod_{t=1}^{T_{\rm s}} (\lambda_{\rm b}V)^{n_{\rm b}(t)}(\lambda_{\rm f}V)^{n_{\rm f}(t)}\cdot P(Y|\omega), \end{aligned} \tag{5}$$

where $P(Y|\omega)$ is the likelihood of observations Y given ω, which can be computed based on the chosen dynamic and measurement models[4]. For example, the computation of $P(Y|\omega)$ for the linear dynamic and measurement models can be found in [21].

There are two major approaches to solve the multi-target tracking problem [22]: *maximum a posteriori* (MAP) and Bayesian approaches. The MAP approach finds a partition of observations such that $P(\omega|Y)$ is maximized and estimates the states of the targets based on this partition. A Bayesian approach called *minimum mean square error* (MMSE) finds an estimate which minimizes the expected square error. For instance, $\mathbb{E}(x^k(t)|Y)$ is the MMSE estimate for the state $x^k(t)$ of target k. However, when the number of targets is not fixed, a unique labeling of each target is required to find $\mathbb{E}(x^k(t)|Y)$ under the MMSE approach. In this paper, we take the MAP approach to the multi-target tracking problem for its convenience.

C. Agent Dynamics and Coordination Objective

In a situation where multiple pursuers and evaders are present, several assignments are possible and some criteria need to be chosen to optimize performance. In this work, we focus on minimizing the time to capture all evaders. However, other criteria might be possible, such as minimizing the pursuer's energy consumption while guaranteeing capture of all evaders or maximizing the number of captured evaders within a certain amount of time. Since the evaders' motions are not known, an exact time to capture a particular evader is also not known. Therefore, we need to define a metric to estimate the time to capture the evaders. Let us define the state vector of a vehicle as $x = [x_1, x_2, \dot{x}_1, \dot{x}_2]^T$, where (x_1, x_2) and $(\dot{x}_1, \dot{x}_2)$ are the position and the velocity components of the vehicle along the x and y axes, respectively. We denote by $x^{\rm p}$ and $x^{\rm e}$ the state of a pursuer and an evader, respectively. We will use the following definition of time-to-capture:

Definition 3.1 (Time-to-capture): Let $x^{\rm e}(t_0)$ be the position and velocity vector of an evader in a plane at time t_0, and $x^{\rm p}(t)$ be the position and velocity vector of a pursuer at the current time $t \geq t_0$. We define the (constant speed) *time-to-capture* as the minimum time $T_{\rm c}$ necessary for the pursuer to reach the evader with the same velocity, assuming that the evader will keep moving at a constant velocity, *i.e.*,

$$T_{\rm c} := \min\Big[T \mid x^{\rm p}(t+T) = x^{\rm e}(t+T)\Big],$$

where $x^{\rm e}_{1,2}(t+T) = x^{\rm e}_{1,2}(t_0) + (t+T-t_0)\dot{x}^{\rm e}_{1,2}(t_0)$, $\dot{x}^{\rm e}_{1,2}(t+T) = \dot{x}^{\rm e}_{1,2}(t_0)$, and the pursuer moves according to its dynamics.

This definition allows us to quantify the time-to-capture in an unambiguous way. Although an evader can change trajectories over time, it is a more accurate estimate than, for example, some metric based on the distance between an evader and a pursuer, since the time-to-capture incorporates the dynamics of the pursuer.

Given Definition 3.1 and the constraints on the dynamics of the pursuer, it is possible to calculate explicitly the time-to-capture $T_{\rm c}$, as well as the optimal trajectory $x^{\rm e*}(t)$ for the pursuers as shown in Section IV-C.

We assume the following dynamics for both pursuers and evaders:

$$x(t+\delta) = A_\delta x(t) + G_\delta u(t) \tag{6}$$

$$\eta(t) = x(t) + v(t) \tag{7}$$

where δ is the sampling interval, $u = [u_1, u_2]^T$ is the control input vector, $\eta(t)$ is the estimated vehicle state provided by the MTF module, $v(t)$ is the estimation error, and

$$A_\delta = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G_\delta = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix},$$

which correspond to the discretization of the dynamics of a decoupled planar double integrator. Although this model appears simplistic for modeling complex motions, it is widely used as a first approximation in path-planning [51, 52, 53]. Moreover, there exist methodologies to map such a simple dynamic model into a more realistic model via consistent abstraction as shown in [54, 55]. Finally, any possible mismatch between this model and the true vehicle dynamics can be compensated for by the path-follower controller implemented on the pursuer [47].

The observation vector $\eta = [\eta_1, \eta_2, \dot{\eta}_1, \dot{\eta}_2]^T$ is interpreted as a measurement, although in reality it is the output from the MTF module shown in Figure 2. The estimation error $v_t = [v_1, v_2, \dot{v}_1, \dot{v}_2]^T$ can be modeled as a Gaussian noise with zero mean and covariance Q or as an unknown but bounded error, *i.e.*, $|v_1| < V_1, |v_2| < V_2, |\dot{v}_1| < \dot{V}_1, |\dot{v}_2| < \dot{V}_2$, where V_1, V_2, $\dot{V}_1$ and $\dot{V}_2$ are positive scalars that are possibly time-varying. Both modeling approaches are useful for different reasons. Using a Gaussian noise approximation allows a closed-form optimal filter solution such as the well-known Kalman filter [56]. On the other hand using the unknown but bounded error model allows for the design of a robust controller such as the robust minimum-time control of pursuers proposed in Section IV-C.

[4]Our formulation of (5) is similar to MHT [49] and the derivation of (5) can be found in [50]. The parameters $p_{\rm z}$, $p_{\rm d}$, $\lambda_{\rm b}$ and $\lambda_{\rm f}$ have been widely used in many multi-target tracking applications [27, 49]. Our experimental and simulation experiences show that our tracking algorithm is not sensitive to changes in these parameters in most cases. In fact, we used the same set of parameters for all our experiments.

We also assume that the control input to a pursuer is bounded, *i.e.*,

$$|u_1^{\mathrm{p}}| \leq U_{\mathrm{p}}, \ |u_2^{\mathrm{p}}| \leq U_{\mathrm{p}} \tag{8}$$

where $U_{\mathrm{p}} > 0$. We consider two possible evader dynamics:

$$u_1^{\mathrm{e}} \sim \mathcal{N}(0, q_{\mathrm{e}}), u_2^{\mathrm{e}} \sim \mathcal{N}(0, q_{\mathrm{e}}) \qquad \text{(random motion)} \tag{9}$$

$$|u_1^{\mathrm{e}}| \leq U_{\mathrm{e}}, |u_2^{\mathrm{e}}| \leq U_{\mathrm{e}} \qquad \text{(evasive motion)}, \tag{10}$$

where $\mathcal{N}(0, q_{\mathrm{e}})$ is a Gaussian distribution with zero mean and variance $q_{\mathrm{e}} \in \mathbb{R}^+$. Equation (9) is a standard model for the unknown motion of vehicles, where the variation in a velocity component is a discrete-time white noise acceleration [57]. Equation (10) allows for evasive maneuvers but places bounds on the maximum thrust. The multi-agent coordination scheme proposed in Section IV-C is based on dynamics (10) as pursuers choose their control actions to counteract the best possible evasive maneuver of the evader being chased. However, in our simulations and experiments, we test our control architecture using the dynamics (9) for evaders where we set $q_{\mathrm{e}} = 2U_{\mathrm{e}}$.

Since the definition of the time-to-capture is related to relative distance and velocity between the pursuer and the evader, we consider the state space error $\xi = x^{\mathrm{p}} - x^{\mathrm{e}}$ which evolves according to the following error dynamics:

$$\begin{array}{rcl} \xi(t+\delta) & = & A_\delta \xi(t) + G_\delta u^{\mathrm{p}}(t) - G_\delta u^{\mathrm{e}}(t) \\ \eta^\xi(t) & = & \xi(t) + v^\xi(t) \end{array} \tag{11}$$

where the pursuer thrust $u^{\mathrm{p}}(t)$ is the only controllable input, while the evader thrust $u^{\mathrm{e}}(t)$ acts as a random or unknown disturbance, and $v^\xi(t)$ is the measurement error which takes into account the uncertainties on the states of both the pursuer and the evader. According to the definition above, an evader is captured if and only if $\xi(t) = 0$, and the time-to-capture T_{c} corresponds to the time necessary to drive $\xi(t)$ to zero assuming $u^{\mathrm{e}}(t) = 0$ for $t \geq t_0$. However, this assumption is relaxed in Section IV-C.

According to the definition of time-to-capture above and the error dynamics (11), given the positions and velocities of all the pursuers and evaders, it is possible to compute the time-to-capture matrix $C = [c_{ij}] \in \mathbb{R}^{N_{\mathrm{p}} \times N_{\mathrm{e}}}$, where N_{p} and N_{e} are the total number of pursuers and evaders, respectively, and the entry c_{ij} of the matrix C corresponds to the expected time-to-capture between pursuer i and evader j. When coordinating multiple pursuers to chase multiple evaders, it is necessary to assign pursuers to evaders. Our objective is to select an assignment that minimizes the expected time-to-capture of *all* evaders, which correspond to the *global worst case time-to-capture*. In this paper, we focus on a scenario with the same number of pursuers and evaders, *i.e.*, $N_{\mathrm{p}} = N_{\mathrm{e}}$. When there are more pursuers than evaders, then only a subset of all the pursuers can be dispatched and the others are kept on alert in case more evaders appear. Alternatively, more pursuers can be assigned to a single evader. When there are more evaders than pursuers, one approach is to minimize the time to capture the N_{p} closest evaders. Obviously, many different coordination objectives can be formulated as they are strongly application-dependent. We have chosen the definition of global worst case time-to-capture as it enforces strong global coordination to achieve high performance.

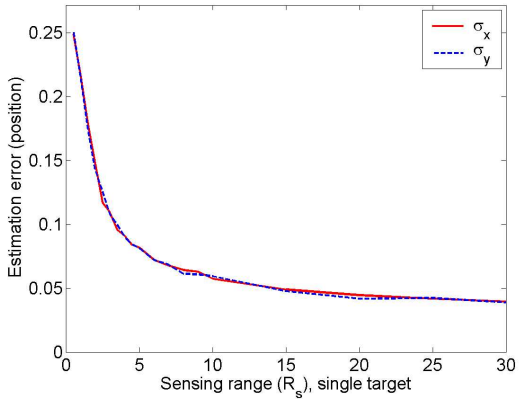


Fig. 4. Single target position estimation error as a function of sensing range. See Section IV-B.3 for the sensor network setup used in simulations (Monte Carlo simulation of 1000 samples, unity corresponds to the separation between sensors).

IV. Control System Implementation

A. Multi-Sensor Fusion Module

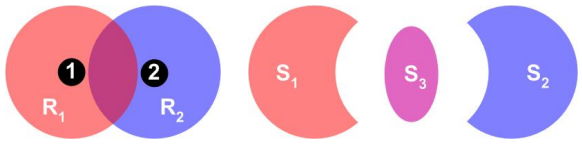
1) Signal-Strength Sensor Model: Consider the signal-strength sensor model described in Section III-A. Recall that z_i is the signal strength measured by node i. For each node i, if $z_i \geq \theta$, where θ is a threshold set for appropriate values of detection and false-positive probabilities, the node transmits z_i to its neighboring nodes, which are at most $2R_{\mathrm{s}}$ away from s_i, and listens to incoming messages from neighboring nodes within a $2R_{\mathrm{s}}$ radius. We assume that the communication range of each node is larger than $2R_{\mathrm{s}}$. For a node i, if z_i is larger than all incoming messages, $z_{i_1}, \ldots, z_{i_{k-1}}$, and $z_{i_k} = z_i$, then the position of an object is estimated by

$$\hat{z}_i = \frac{\sum_{j=1}^{k} z_{i_j} s_{i_j}}{\sum_{j=1}^{k} z_{i_j}}. \tag{12}$$

The estimate $\hat{z}_i$ corresponds to a center of mass of the node locations weighed by their measured signal strengths. Node i transmits $\hat{z}_i$ to the Tier-2 node $g(i)$. If z_i is not the largest compared to the incoming messages, node i simply continues sensing. Although each sensor cannot give an accurate estimate of the object's position, as more sensors collaborate, the accuracy of the estimates improves as shown in Figure 4.

2) Binary Sensor Model: In order to obtain finer position reports from binary detections, we use spatial correlation among detections from neighboring sensors. The idea behind the fusion algorithm is to compute the likelihood of detections assuming there is a single target. This is only an approximation since there can be more than one target. However, any inconsistencies caused by this approximation are fixed by the tracking algorithm described in Section IV-B using spatio-temporal correlation.

Consider the binary sensor model described in Section III-A. Let x be the position of an object. For the purpose of illustration, suppose that there are two sensors, sensor 1 and sensor 2, and $R_1 \cap R_2 \neq \emptyset$ (see Figure 5(a)). The overall sensing region $R_1 \cup R_2$ can be partitioned into a set of non-overlapping cells (or blocks) as shown in Figure 5(b). The



(a) (b)

Fig. 5. (a) Sensing regions of two sensors 1 and 2. R_i is the sensing region of sensor i. (b) A partition of the overall sensing region $R_1 \cup R_2$ into non-overlapping cells S_1, S_2 and S_3, where $S_1 = R_1 \setminus R_2$, $S_2 = R_2 \setminus R_1$ and $S_3 = R_1 \cap R_2$.

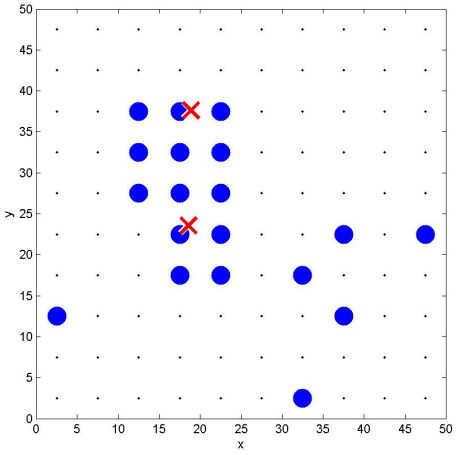


Fig. 6. Detections of two targets by a 10×10 sensor grid (targets in $\times$, detections in disks, and sensor positions in small dots).

likelihoods can be computed as follows:

$$\begin{aligned} P(z_1, z_2 | x \in S_1) &= p_1^{z_1}(1-p_1)^{1-z_1} q_2^{z_2}(1-q_2)^{1-z_2} \\ P(z_1, z_2 | x \in S_2) &= q_1^{z_1}(1-q_1)^{1-z_1} p_2^{z_2}(1-p_2)^{1-z_2} \\ P(z_1, z_2 | x \in S_3) &= p_1^{z_1}(1-p_1)^{1-z_1} p_2^{z_2}(1-p_2)^{1-z_2}, \end{aligned} \tag{13}$$

where $S_1 = R_1 \setminus R_2$, $S_2 = R_2 \setminus R_1$ and $S_3 = R_1 \cap R_2$ (see Figure 5(b)). Hence, for any deployment we can first partition the surveillance region into a set of non-overlapping cells. Then, given detection data, we can compute the likelihood of each cell as shown in the previous example.

An example of detections of two targets by a 10×10 sensor grid is shown in Figure 6. In this example, the sensing region is assumed to be a disk with radius of 7.62m (10 ft). We have assumed $p_i = 0.7$ and $q_i = 0.05$ for all i. These parameters are estimated from measurements made with the passive infrared (PIR) sensor of an actual sensor node described in Section V. From the detections shown in Figure 6, the likelihood can be computed using equations similar to (13) for each non-overlapping cell (see Figure 7). Notice that it is a time-consuming task to find all non-overlapping cells for arbitrary sensing region shapes and sensor deployments. Hence, we quantized the surveillance region and the likelihoods are computed for a finite number of points as shown in Figure 7.

There are two parts in this likelihood computation: the detection part (terms involving p_i) and the false detection part (terms involving q_i). Hereafter, we call the detection part of the

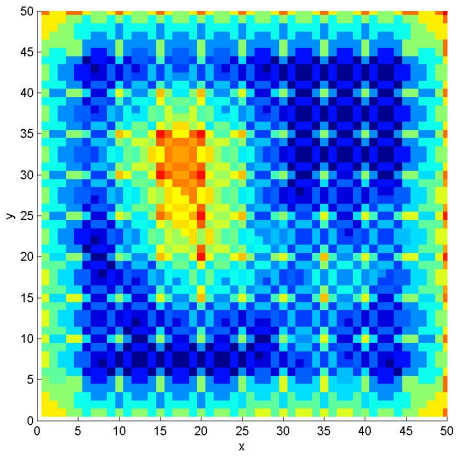


Fig. 7. Likelihood of detections from Figure 6.

likelihood as the *detection-likelihood* and the false detection part of the likelihood as the *false-detection-likelihood*. Notice that the computation of the false-detection-likelihood requires measurements from all sensors. However, for a large wireless sensor network, it is not feasible to exchange detection data with all other sensors. Instead, we use a threshold test to avoid computing the false-detection-likelihood and distribute the likelihood computation. The detection-likelihood of a cell is computed if there are at least θ_{d} detections, where θ_{d} is a user-defined threshold. Using $\theta_{\mathrm{d}} = 3$, the detection-likelihood of the detections from Figure 6 can be computed as shown in Figure 8. The computation of the detection-likelihood can be done in a distributed manner. Assign a set of non-overlapping cells to each sensor such that no two sensors share the same cell and each cell is assigned to a sensor whose sensing region includes the cell. For each sensor i, let $\{S_{i_1}, \ldots, S_{i_{m(i)}}\}$ be a set of non-overlapping cells, where $m(i)$ is the number of cells assigned to sensor i. Then, if sensor i reports a detection, it computes the likelihoods of each cell in $\{S_{i_1}, \ldots, S_{i_{m(i)}}\}$ based on its own measurements and the measurements from neighboring sensors. A neighboring sensor is a sensor whose sensing region intersects the sensing region of sensor i. Notice that no measurement from a sensor means no detection.

Based on the detection-likelihoods, we compute target position reports by clustering. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a set of cells whose detection-likelihoods are computed, *i.e.*, the number of detections for each S_i is at least θ_{d}. First, randomly pick S_j from $\mathcal{S}$ and remove S_j from $\mathcal{S}$. Then cluster around S_j the remaining cells in $\mathcal{S}$ whose set-distance to S_j is less than the sensing radius. The cells clustered with S_j are then removed from $\mathcal{S}$. Now repeat the procedure until $\mathcal{S}$ is empty. Let $\{C_k : 1 \leq k \leq K_{\mathrm{cl}}\}$ be the clusters formed by this procedure, where K_{cl} is the total number of clusters. For each cluster C_k, its center of mass is computed to obtain a a fused position report, *i.e.*, an estimated position of a target. An example of position reports is shown in Figure 8.

The multi-sensor fusion algorithm described above runs on

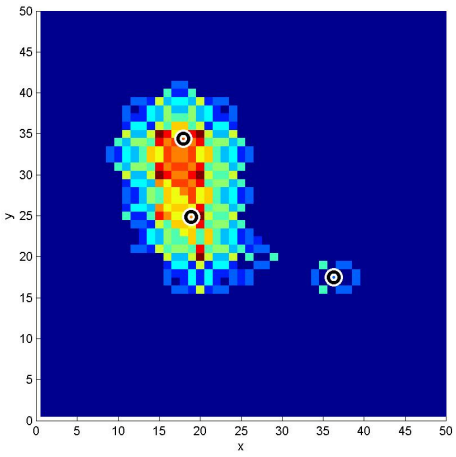


Fig. 8. Detection-likelihood of detections from Figure 6 with threshold $\theta_{\mathrm{d}} = 3$. Estimated positions of targets are shown in circles.

two levels: Algorithm 1 on the Tier-1 nodes and Algorithm 2 on the Tier-2 node. Each Tier-1 node combines detection data from itself and neighboring nodes using Algorithm 1 and computes detection-likelihoods. The detection-likelihoods are forwarded to its Tier-2 node and the Tier-2 node generates position reports from the detection-likelihoods using Algorithm 2. The position reports are then used by the MTT module described in Section IV-B to track multiple targets.

Algorithm 1 Multi-Sensor Fusion: Sensor i

Input: detections from sensor i and its neighbors
Output: detection-likelihoods

for each S_{i_j}, $j = 1, \ldots, m(i)$ **do**
  if number of detections for $S_{i_j} \geq \theta_{\mathrm{d}}$ **then**
    compute detection-likelihood $\hat{z}_i(j)$ of S_{i_j}
    forward $\hat{z}_i(j)$ to Tier-2 node $g(i)$
  end if
end for

Algorithm 2 Multi-Sensor Fusion: Tier-2 Node

Input: detection-likelihoods $\mathcal{Z} = \{\hat{z}_i(j)\}$ received from its tracking group
Output: position reports y

$\mathcal{S} = \{S_{i_j} : \hat{z}_i(j) \in \mathcal{Z}\}$
$y = \emptyset$
find clusters $\{C_k : 1 \leq k \leq K_{\mathrm{cl}}\}$ from $\mathcal{S}$ as described in the text
for each C_k, $k = 1, \ldots, K_{\mathrm{cl}}$ **do**
  compute the center of mass y_k of C_k
  $y = y \cup y_k$
end for

B. Multi-Target Tracking and Multi-Track Fusion Modules

Our tracking algorithms are based on Markov chain Monte Carlo data association (MCMCDA) [21]. We first describe the MCMCDA algorithm and then describe the MTT and MTF modules of *LochNess*.

Markov chain Monte Carlo (MCMC) plays a significant role in many fields such as physics, statistics, economics, and engineering [58]. In some cases, MCMC is the only known general algorithm that finds a good approximate solution to a complex problem in polynomial time [59]. MCMC techniques have been applied to complex probability distribution integration problems, counting problems such as #P-complete problems, and combinatorial optimization problems [59, 58].

MCMC is a general method to generate samples from a distribution π on a space Ω by constructing a Markov chain $\mathcal{M}$ with states $\omega \in \Omega$ and stationary distribution $\pi(\omega)$. We now describe an MCMC algorithm known as the Metropolis-Hastings algorithm [60]. If we are at state $\omega \in \Omega$, we propose $\omega' \in \Omega$ following the proposal distribution $q(\omega, \omega')$. The move is accepted with an acceptance probability $A(\omega, \omega')$ where

$$A(\omega, \omega') = \min\left(1, \frac{\pi(\omega')q(\omega', \omega)}{\pi(\omega)q(\omega, \omega')}\right), \tag{14}$$

otherwise the sampler stays at ω, so that the detailed balance is satisfied. If we make sure that $\mathcal{M}$ is irreducible and aperiodic, then $\mathcal{M}$ converges to its stationary distribution by the ergodic theorem [61].

The MCMC data association (MCMCDA) algorithm is described in Algorithm 3. MCMCDA is an MCMC algorithm whose state space is Ω, as described in Section III-B, and whose stationary distribution is the posterior (5). The proposal distribution for MCMCDA consists of five types of moves (a total of eight moves). They are (1) a birth/death move pair; (2) a split/merge move pair; (3) an extension/reduction move pair; (4) a track update move; and (5) a track switch move. The MCMCDA moves are illustrated in Figure 9. We index each move by an integer such that $m = 1$ for a birth move, $m = 2$ for a death move and so on. The move m is chosen randomly from the distribution $q_K^{\mathrm{m}}(m)$ where K is the number of tracks of the current partition ω. When there is no track, we can only propose a birth move, so we set $q_0^{\mathrm{m}}(m = 1) = 1$ and $q_0^{\mathrm{m}}(m = 1) = 0$ for all other moves. When there is only a single target, we cannot propose a merge or track switch move, so $q_1^{\mathrm{m}}(m = 4) = q_1^{\mathrm{m}}(m = 8) = 0$. For the other values of K and m, we assume $q_K^{\mathrm{m}}(m) > 0$. For a detailed description of each move, see [21]. The inputs for MCMCDA are the set of all observations Y, the number of samples n_{mc}, the initial state ω_{init}, and a bounded function $X : \Omega \to \mathbb{R}^n$. At each step of the algorithm, ω is the current state of the Markov chain. The acceptance probability $A(\omega, \omega')$ is defined in (14) where $\pi(\omega) = P(\omega|Y)$ from (5). The output $\hat{X}$ approximates the MMSE estimate $\mathbb{E}_\pi X$ and $\hat{\omega}$ approximates the MAP estimate $\arg\max P(\omega|Y)$. The computation of $\hat{\omega}$ can be considered as simulated annealing at a constant temperature. Notice that MCMCDA can provide both MAP and MMSE solutions to the multi-target tracking problem. In this paper, we use the

MAP estimate $\hat{\omega}$ to estimate the states of the targets[5].

Algorithm 3 MCMCDA

Input: $Y, n_{\text{mc}}, \omega_{\text{init}}, X : \Omega \to \mathbb{R}^n$
Output: $\hat{\omega}, \hat{X}$

1: $\omega = \omega_{\text{init}}$; $\hat{\omega} = \omega_{\text{init}}$; $\hat{X} = 0$
2: **for** $n = 1$ to n_{mc} **do**
3: propose ω' based on ω (see below)
4: sample U from Unif$[0, 1]$
5: $\omega = \omega'$ if $U < A(\omega, \omega')$
6: $\hat{\omega} = \omega$ if $p(\omega|Y)/p(\hat{\omega}|Y) > 1$
7: $\hat{X} = \frac{n}{n+1}\hat{X} + \frac{1}{n+1}X(\omega)$
8: **end for**

It has been shown that MCMCDA is an *optimal Bayesian filter* in the limit [22]. In addition, in terms of time and memory, MCMCDA is more computationally efficient than MHT and outperforms MHT with heuristics (*i.e.*, pruning, gating, clustering, N-scan-back logic and k-best hypotheses) under extreme conditions, such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates [21].

1) Multi-Target Tracking Module: At each Tier-2 node, we implement the online MCMCDA algorithm with a sliding window of size w_{s} using Algorithm 3 [21]. This online implementation of MCMCDA is suboptimal because it considers only a subset of past measurements. But since the contribution of older measurements to the current estimate is much less than recent measurements, it is still a good approximation. At each time step, we use the previous estimate to initialize MCMCDA and run MCMCDA on the observations belonging to the current window. Each Tier-2 node maintains a set of observations $Y = \{y^j(t) : 1 \le j \le n(t), t_{\text{curr}} - w_{\text{s}} + 1 \le t \le t_{\text{curr}}\}$, where t_{curr} is the current time. Each $y^j(t)$ is either a fused measurement $\hat{z}_i$ from some signal-strength sensor i or an element of the fused position reports y from some binary sensors. At time $t_{\text{curr}} + 1$, the observations at time $t_{\text{curr}} - w_{\text{s}} + 1$ are removed from Y and a new set of observations is appended to Y. Any delayed observations are inserted into the appropriate slots. Then, each Tier-2 node initializes the Markov chain with the previously estimated tracks and executes Algorithm 3 on Y. Once a target is found, the next state of the target is predicted. If the predicted next state belongs to the surveillance area of another Tier-2 node, the target's track information is passed to the corresponding Tier-2 node. These newly received tracks are then incorporated into the initial state of MCMCDA for the next time step. Lastly, each Tier-2 node forwards its track information to the base station.

2) Multi-Track Fusion Module: Since each Tier-2 node maintains its own set of tracks, there can be multiple tracks from a single target maintained by different Tier-2 nodes. To make the algorithm fully hierarchical and scalable, the MTF module performs the track-level data association at the base station to combine tracks from different Tier-2 nodes. Let ω_j be the set of tracks maintained by Tier-2 node $j \in \{1, \ldots, N_{\text{ss}}\}$. Let $Y_{\text{c}} = \{\tau_i(t) \in \omega_j : 1 \le t \le t_{\text{curr}}, 1 \le i \le |\omega_j|, 1 \le j \le N_{\text{ss}}\}$ be the combined observations only from the established tracks. We form a new set of tracks ω_{init} from $\{\tau_i \in \omega_j : 1 \le i \le |\omega_j|, 1 \le j \le N_{\text{ss}}\}$ while making sure that the constraints defined in Section III-B are satisfied. Then, we run Algorithm 3 on this combined observation set Y_{c} with the initial state ω_{init}. An example in which the multi-track fusion corrects mistakes made by Tier-2 nodes due to missing observations at the tracking group boundaries is shown in Section IV-B.3.

[5]The states of the targets can be easily computed by any filtering algorithm since, given $\hat{\omega}$, the associations between the targets and the measurements are completely known.

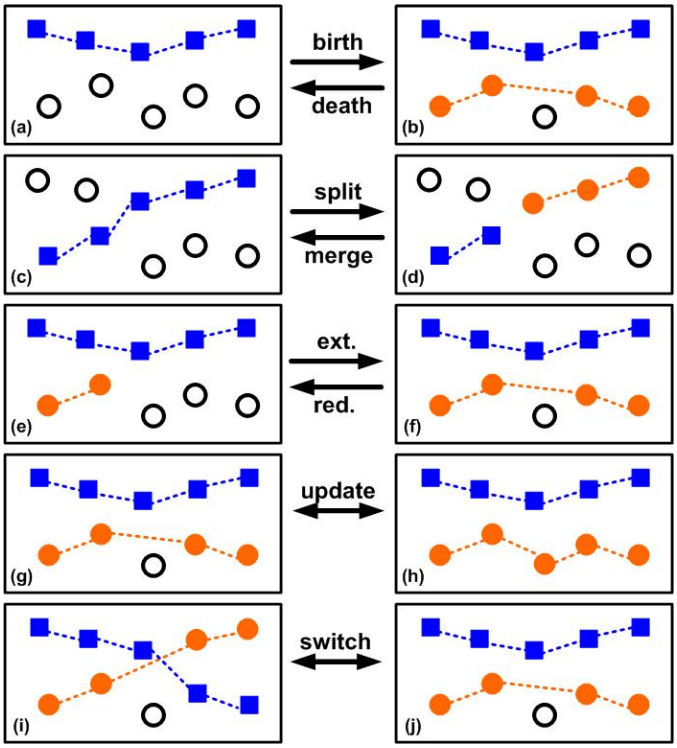


Fig. 9. Graphical illustration of MCMCDA moves (associations are indicated by dotted lines and hollow circles are false alarms). Each move proposes a new joint association event ω' which is a modification of the current joint association event ω. The birth move proposes ω' by forming a new track from the set of false alarms ($(a) \to (b)$). The death move proposes ω' by combining one of the existing tracks into the set of false alarms ($(b) \to (a)$). The split move splits a track from ω into two tracks ($(c) \to (d)$) while the merge move combines two tracks in ω into a single track ($(d) \to (c)$). The extension move extends an existing track in ω ($(e) \to (f)$) and the reduction move reduces an existing track in ω ($(f) \to (e)$). The track update move chooses a track in ω and assigns different measurements from the set of false alarms ($(g) \leftrightarrow (h)$). The track switch move chooses two track from ω and switches some measurement-to-track associations ($(i) \leftrightarrow (j)$).

The algorithm is autonomous and shown to be robust against packet loss, communication delay and sensor localization error. In simulation, there is no performance loss up to an average localization error of 0.7 times the separation between sensors, and the algorithm tolerates up to 50% lost-to-total packet ratio and 90% delayed-to-total packet ratio [23].

3) An Example of Surveillance using Sensor Networks: Here, we give a simulation example of surveillance using sensor networks. The surveillance region $\mathcal{R} = [0, 100]^2$ was

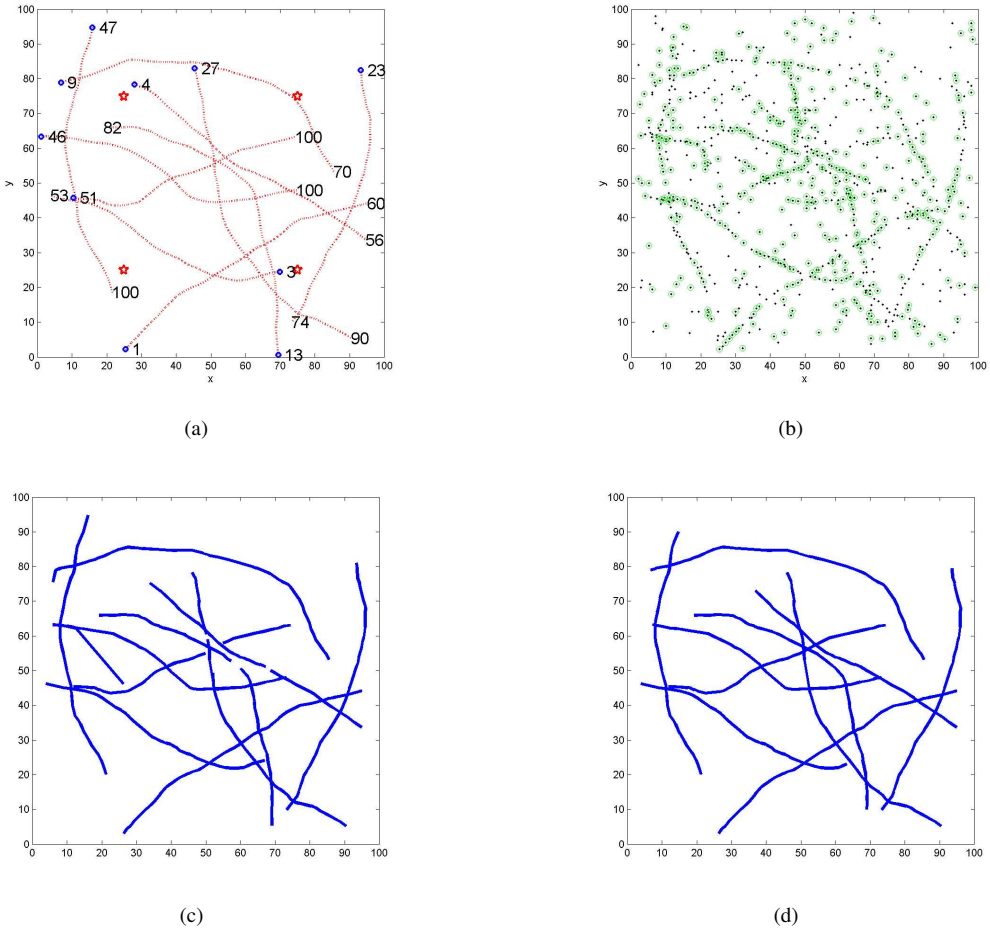


Fig. 10. (a) Tracking scenario, where the numbers are target appearance and disappearance times, the initial positions are marked by circles, and the stars are the positions of Tier-2 nodes. (b) Accumulated observations received by Tier-2 nodes with delayed observations circled. (c) Tracks estimated locally by the MTT modules at Tier-2 nodes, superimposed. (d) Tracks estimated by the MTF module.

divided into four quadrants and sensors in each quadrant formed a tracking group, where a Tier-2 node was placed at the center of each quadrant. The scenario is shown in Figure 10(a). We assumed a 100×100 sensor grid, in which the separation between sensors was normalized to 1. Thus, the unit length in simulation was the length of the sensor separation. For MCMCDA, $n_{\text{mc}} = 1000$ and $w_{\text{s}} = 10$. The signal-strength sensor model was used with parameters $\alpha = 2$, $\gamma = 1$, $\theta = 2$, and $\beta = 3(1 + \gamma R_{\text{s}}^{\alpha})$. In addition, $p_{\text{te}} = .3$ and $p_{\text{de}} = .3$. The surveillance duration was $T_{\text{s}} = 100$.

The state vector of a target is $x = [x_1, x_2, \dot{x_1}, \dot{x_2}]^T$ as described in Section III-C. The simulation used the dynamic model in (6) and the evader control inputs were modeled by the random motion (9) with $q_{\text{e}} = .15^2$ and Q set according to Figure 4. Since the full state is not observable, the measurement model (7) was modified as follows:

$$y(t) = Dx(t) + v(t), \quad \text{where } D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (15)$$

and y is a fused measurement computed by the MSF module in Section IV-A.

Figure 10(b) shows the observations received by the Tier-2 nodes. There were a total of 1174 observations and 603 of these observations were false alarms. A total of 319 packets out of 1174 packets were lost due to transmission failures and 449 packets out of 855 received packets were delayed. Figure 10(c) shows the tracks estimated locally by the MTT modules on the Tier-2 nodes while Figure 10(d) shows the tracks estimated by the MTF module using track-level data association. Figure 10(d) shows that the MTF module corrected mistakes made by Tier-2 nodes due to missing observations at the tracking group boundaries. The algorithm is written in C++ and MATLAB and run on PC with a 2.6-GHz Intel Pentium 4 processor. It takes less than 0.06 seconds per Tier-2 node, per simulation time step.

C. Multi-Agent Coordination Module

The time-to-capture is estimated using the abstract model of pursuer and evader dynamics given in Section III-C. Let us consider the error between the pursuer and the evader $\xi = [\xi_1, \xi_2, \dot{\xi}_1, \dot{\xi}_2]^T$ whose dynamics is given in (11). The time-to-capture problem is equivalent to the following opti-

mization problem:

$$\begin{array}{ll}\min_{u_1^{\mathrm{p}}(t),u_2^{\mathrm{p}}(t)} & T \\ \text{subject to} & \left\{\begin{array}{l}\xi(t+\delta)=A_\delta\xi(t)+G_\delta u^{\mathrm{p}}(t)\\ |u_1^{\mathrm{p}}(t)|\le U_{\mathrm{p}},\quad |u_2^{\mathrm{p}}(t)|\le U_{\mathrm{p}}\\ \xi(t+T)=0.\end{array}\right.\end{array} \tag{16}$$

Recently, Gao *et al.* [62] solved the previous problem as an application of minimum-time control for the discretized double integrator. An extension to minimum-time control for the discretized triple integrator is also available [63]. Despite its simplicity and apparent efficacy, minimum-time control is rarely used in practice, since it is highly sensitive to small measurement errors and external disturbances. Although, in principle, minimum-time control gives the best performance, it needs to be modified to cope with practical issues such as the quantization of inputs, measurement and process noise, and modeling errors. We propose an approach that adds robustness while preserving the optimality of minimum-time control.

Since the state error dynamics is decoupled along the x and y-axes, the solution of the optimization problem (16) can be obtained by solving two independent minimum-time problems along each axis. When $\delta \to 0$ in (11), the minimum-time control problem restricted to one axis reduces to the well known minimum-time control problem of a double integrator in continuous time, which can be found in many standard textbooks on optimal control such as [64, 65]. The solution is given by a bang-bang control law and can be written in state feedback form as follows:

$$u_1^{\mathrm{p}} = \left\{\begin{array}{ccc} -U_{\mathrm{p}} & \text{If} & 2U_{\mathrm{p}}\dot\xi_1 > -\xi_1|\xi_1| \\ +U_{\mathrm{p}} & \text{If} & 2U_{\mathrm{p}}\dot\xi_1 < -\xi_1|\xi_1| \\ -U_{\mathrm{p}}\,\mathrm{sign}(\xi_1) & \text{If} & 2U_{\mathrm{p}}\dot\xi_1 = -\xi_1|\xi_1| \\ 0 & \text{If} & \dot\xi_1=\xi_1=0. \end{array}\right. \tag{17}$$

The minimum time required to drive ξ_1 to zero in the x-axis can be also written in terms of the position and velocity error as follows:

$$T_{\mathrm{c},1}(\xi_1,\dot\xi_1) = \left\{\begin{array}{ll} \frac{-\dot\xi_1+\sqrt{2\dot\xi_1^2-4U_{\mathrm{p}}\xi_1}}{U_{\mathrm{p}}} & \text{if } \ 2U_{\mathrm{p}}\dot\xi_1 \ge -\xi_1|\xi_1| \\ \frac{\dot\xi_1+\sqrt{2\dot\xi_1^2+4U_{\mathrm{p}}\xi_1}}{U_{\mathrm{p}}} & \text{otherwise.} \end{array}\right. \tag{18}$$

Figure 11 shows the switching curve $2U_{\mathrm{p}}\dot\xi_1 = -\xi_1|\xi_1|$ and the level curves of the time-to-capture T_{c} for different values.

Similar equations can be written for the control u_2^{p} along the y-axis. Therefore the minimum time-to-capture is given by:

$$T_{\mathrm{c}} = \max(T_{\mathrm{c},1}, T_{\mathrm{c},2}) \tag{19}$$

According to the previous analysis, given the state error $\xi(t)$ at current time t, we can compute the corresponding constant velocity time-to-capture T_{c}, the optimal input sequence $u^{\mathrm{p}*}(t')$ and the optimal trajectory $\xi^*(t')$ for $t' \in [t, t+T_{\mathrm{c}}]$.

However, the optimal input (17) is the solution when $\delta \to 0$ in (11) with no measurement errors and no change in the evader's trajectory. In order to add robustness to take into account the quantization in the digital implementation, the measurement errors, and the evasive maneuvers of the evader, we analyze how the time-to-capture can be affected by these

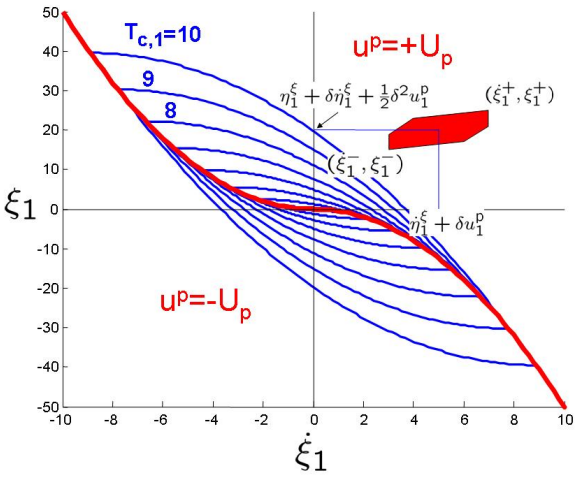


Fig. 11. Optimal switching curve for the continuous minimum-time control of the double integrator (*thick solid line*) and curves of constant time-to-capture (*thin solid lines*) in the phase space $(\xi_1, \dot\xi_1)$. The hexagon represents the set of all possible locations of the true error state $(\xi_1(t+\delta), \dot\xi_1(t+\delta))$ at the next time step $t+\delta$ given measurement $(\eta_1, \dot\eta_1)$ and pursuer control input u_1^{p} at time t.

terms. Let us first rewrite the error dynamics given by (11) explicitly for the x-axis:

$$\begin{array}{lcl}\xi_1(t+\delta) &=& \xi_1(t)+\delta\,\dot\xi_1(t)+\frac{1}{2}\delta^2u_1^{\mathrm{p}}(t)+\frac{1}{2}\delta^2u_1^{\mathrm{e}}(t)\\ \dot\xi_1(t+\delta) &=& \dot\xi_1(t)+\delta\,u_1^{\mathrm{p}}(t)+\delta u_1^{\mathrm{e}}(t)\\ \eta_1^\xi(t) &=& \xi_1(t)+v_1^\xi(t)\\ \dot\eta_1^\xi(t) &=& \dot\xi_1(t)+\dot v_1^\xi(t)\end{array}$$

If we substitute the last two equations into the first two we get:

$$\begin{aligned}\xi_1(t+\delta) &= \eta_1^\xi(t)+\delta\dot\eta_1^\xi(t)+\frac{1}{2}\delta^2u_1^{\mathrm{p}}(t)- \\ &\quad -v_1^\xi(t)-\delta\dot v_1^\xi(t)+\frac{1}{2}\delta^2u_1^{\mathrm{e}}(t) \qquad (20)\\ \dot\xi_1(t+\delta) &= \dot\eta_1^\xi(t)+\delta\,u_1^{\mathrm{p}}(t)-\dot v_1^\xi(t)+\delta u_1^{\mathrm{e}}(t) \qquad (21)\end{aligned}$$

where $(\eta_1^\xi, \dot\eta_1^\xi)$ are output estimates from the MTF module, u_1^{p} is the controllable input, and $(u_1^{\mathrm{e}}, v_1^\xi, \dot v_1^\xi)$ play the role of external disturbances. Our goal now is to choose u_1^{p}, *i.e.*, the thrust of the pursuer, in such a way as to minimize the time-to-capture under the worst possible choice of $(u_1^{\mathrm{e}}, v_1^\xi, \dot v_1^\xi)$, which are not known in advance but are bounded. Figure 11 illustrates this approach graphically: the hexagon in the figure represents the possible position of the true state error $(\xi_1, \dot\xi_1)$ at the next time step $t+\delta$ which accounts for all possible evasive maneuvers of the evader, *i.e.*, $|u_1^{\mathrm{e}}| < U_e$, and accounts for the estimation errors on the position and velocity of the pursuer and the evader, *i.e.*, $|v_1^\xi| < V_1, |\dot v_1^\xi| < \dot V_1$, for a given choice of u_1^{p}. Since the center of the hexagon $(\eta_1^\xi+\delta\dot\eta_1^\xi+\frac{1}{2}\delta^2u_1^{\mathrm{p}}, \dot\eta_1^\xi+\delta u_1^{\mathrm{p}})$ depends on the pursuer control u_1^{p}, one could try to choose u_1^{p} in such a way that the largest time-to-capture $T_{\mathrm{c},1}$ of the hexagon is minimized. This approach is common in the literature for non-cooperative games [66]. More formally, the feedback control input will be chosen based

on the following min-max optimization problem

$$u_1^{\mathrm{p}*}(t) = \arg \min_{|u_1^{\mathrm{p}}| \le U_{\mathrm{p}}} \left(\max_{\substack{|v_1^{\xi}| \le V_1, |\dot{v}_1^{\xi}| \le \dot{V}_1, \\ |u_1^{\mathrm{e}}| \le U_{\mathrm{e}}}} T_{\mathrm{c},1}\big(\xi_1(t+\delta), \dot{\xi}_1(t+\delta)\big) \right) \tag{22}$$

This is, in general, a nonlinear optimization problem. However, thanks to the specific structure of the time-to-capture function $T_{\mathrm{c},1}$, it is possible to show that (22) is equivalent to:

$$\begin{aligned} u_1^{\mathrm{p}*} &= \arg \min_{|u_1^{\mathrm{p}}| \le U_{\mathrm{p}}} \max \Big(T_{\mathrm{c},1}(\xi_1^+, \dot{\xi}_1^+), T_{\mathrm{c},1}(\xi_1^-, \dot{\xi}_1^-) \Big) \\ \xi_1^{\pm} &:= \eta_1^{\xi} + \delta \dot{\eta}_1^{\xi} \pm V_1 \pm \delta \dot{V}_1 \pm \frac{1}{2}\delta^2 U_{\mathrm{e}} + \frac{1}{2}\delta^2 u_1^{\mathrm{p}} \\ \dot{\xi}_1^{\pm} &:= \dot{\eta}_1^{\xi} \pm \dot{V}_1 \pm \delta U_{\mathrm{e}} + \delta u_1^{\mathrm{p}}, \end{aligned} \tag{23}$$

i.e., it is necessary to compute only the time-to-capture of the top right and the bottom left corner of the hexagon in Figure 11 since all points inside the set always have smaller values of $T_{\mathrm{c},1}$. Once the expected minimum time-to-capture control input $u^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ is computed, then the corresponding optimal trajectory for the pursuer $x^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ can be easily obtained by substituting $u^{\mathrm{p}*}(t')$ into the pursuer dynamics (6). The robust minimum-time path planning algorithm is summarized in Algorithm 4.

Algorithm 4 Robust Minimum-Time Path Planning

Input: $x^{\mathrm{p}}(t), x^{\mathrm{e}}(t)$, and bounds $V_1, V_2, \dot{V}_1, \dot{V}_2, U_{\mathrm{e}}, U_{\mathrm{p}}$
Output: optimal trajectory $x^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$
1: compute $u^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ using (23)
2: compute $x^{\mathrm{p}*}(t'), t' \in [t, t+T_{\mathrm{c}}]$ given $u^{\mathrm{p}*}(t')$ using (6)

Figure 12 shows the performance of the proposed robust minimum time-to-capture control feedback for a scenario where the evader moves with random motion and the evader's position and velocity estimates are noisy. It is compared with the discrete-time minimum-time controller proposed in [63] and [62]. Our controller feedback design outperforms the discrete-time minimum-time controller since the latter one does not take into account process and measurement noises. Note how both controllers do not direct pursuers toward the actual position of evader, but to the estimated future location of the evader to minimize the time-to-capture.

As introduced in Section III-C, given the positions and velocities of all pursuers and evaders and bounds on the measurement error and evader input, it is possible to compute the expected time-to-capture matrix $C = [c_{ij}] \in \mathbb{R}^{N_{\mathrm{p}} \times N_{\mathrm{e}}}$ using the solution to the optimal minimum-time control problem. The entry c_{ij} of the matrix C corresponds to the expected time for pursuer i to capture evader j, $T_{\mathrm{c}}(i,j)$, that can be computed as described in (18) and (19). As motivated in Section III-C, we assume the same number of pursuers as the number of evaders, *i.e.*, $N_{\mathrm{p}} = N_{\mathrm{e}} = N$.

An assignment can be represented as a matrix $\Phi = [\phi_{ij}] \in \mathbb{R}^{N \times N}$, where the entry ϕ_{ij} of the matrix Φ is equal to 1 if pursuer i is assigned to evader j, and equal to 0 otherwise. The assignment problem can therefore be written formally as

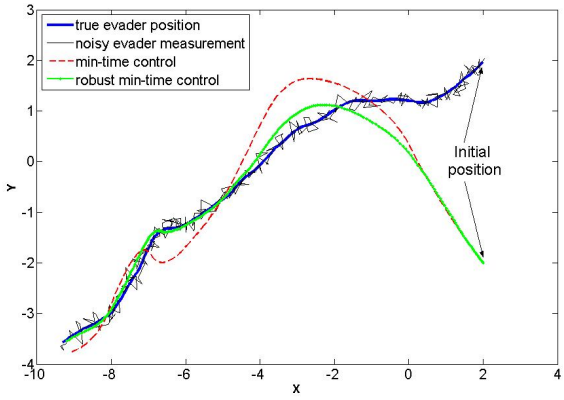


Fig. 12. Trajectories of pursuers and evaders on the x-y plane. The feedback control is based on noisy measurements (*thin solid line*) of the true evader positions (*thick solid line*). The robust minimum time-to-capture feedback proposed in this paper (*dot-solid line*) is compared with the discrete-time minimum time-to-capture feedback (*dashed line*) proposed in [63].

follows:

$$\begin{array}{ll} \min_{\phi_{ij} \in \{0,1\}} & \max_{i,j=1,\ldots,N} (c_{ij} \cdot \phi_{ij}) \\ \text{subject to} & \sum_{i=1}^{N} \phi_{ij} = 1, \ \forall i \\ & \sum_{j=1}^{N} \phi_{ij} = 1, \ \forall j. \end{array} \tag{24}$$

As formulated in (24), the assignment problem is a combinatorial optimization problem.

The optimization problem given in (24) can be reformulated as a *linear bottleneck assignment* problem and can be solved by any of the polynomial-time algorithms based on network flow theory. Here we give a brief description of one algorithm and we direct the interested reader to the survey [45] for a detailed review of these algorithms. For our implementation, we use a randomized threshold algorithm that alternates between two phases. In the first phase, we list the cost elements c_{ij} in increasing order and we choose a cost element c^*, *i.e.*, a threshold. Then we construct the matrices $\bar{C}(c^*) = [\bar{c}_{ij}] \in \mathbb{R}^{N \times N}$ and $C_{\text{Tutte}}(c^*) \in \mathbb{R}^{2N \times 2N}$ as follows:

$$\bar{c}_{ij} = \begin{cases} a_{ij} & \text{if} \quad c_{ij} > c^* \\ 0 & \text{if} \quad c_{ij} \le c^* \end{cases}, \quad C_{\text{Tutte}} = \begin{bmatrix} 0 & \bar{C} \\ -\bar{C} & 0 \end{bmatrix} \tag{25}$$

where a_{ij}'s are independent random numbers sampled from a uniform distribution in the interval $[0,1]$, *i.e.*, $a_{ij} \sim \mathcal{U}([0,1])$. Using Tutte's Theorem [45], it is possible to show that if $\det(C_{\text{Tutte}}(c^*)) \neq 0$, then there exists an assignment that achieves c^*[6]. Therefore, we search for the smallest $c^*_{\min}$ in the ordered list of costs c_{ij} which guarantees an assignment. Once we find $c^*_{\min}$, we find the pursuer-evader pair corresponding to that cost. Then, we remove its row and column from the cost matrix C and repeat the procedure until all pursuers are assigned. The assignment algorithm is summarized in Algorithm 5.

[6]In reality, since the algorithm is randomized, there is a small probability equal to $(1/N)^r$ that there exists a feasible assignment if $\det(C_{\text{Tutte}}) = 0$ for r random Tutte's matrices C_{Tutte}. In the rare cases when this event happens, the algorithm simply gives a feasible assignment with a higher cost to capture.

Algorithm 5 Pursuers-to-evaders Assignment

Input: $x_i^{\mathrm{p}}, x_j^{\mathrm{e}},\ i,j = 1,\dots,N$
Output: assignment $(i \to j)$ for $i = 1,\dots,N$
1: compute matrix $C = [c_{ij}]$, $c_{ij} = T_{\mathrm{c}}(i,j)$
2: **for** $n = 1$ to N **do**
3: $[i^*, j^*] = \mathrm{argmin}_{ij}\{c_{ij} \mid \det(C_{\mathrm{Tutte}}(c_{ij})) \neq 0\}$, using (25)
4: assign pursuer i^* to evader j^*, *i.e.*, $(i^* \to j^*)$
5: $C \leftarrow \{C \,|\, \text{remove row } i^* \text{ and column } j^*\}$
6: **end for**

It is important to note that an assignment based on the solution to the global optimization problem described above is necessary for good performance. For example, let us consider the *greedy assignment* algorithm. This algorithm looks for the smallest time-to-capture entry in the matrix C, assigns the corresponding pursuer-evader pair, and removes the corresponding row and column from the matrix C. The dimensions of the resulting matrix C become $(N-1) \times (N-1)$ and the algorithm repeats the same process until each pursuer is assigned to an evader. This algorithm is very simple and can be implemented in a fully distributed fashion. However, it is a suboptimal algorithm since there are cases where the greedy assignment finds the worst solution. Consider the time-to-capture matrix $C = \begin{bmatrix} 1 & 2 \\ 3 & 100 \end{bmatrix}$. The optimal assignment that minimizes the time-to-capture of all evaders for this matrix is $(1 \to 2)$ and $(2 \to 1)$, which gives $T_{\mathrm{c,max}} = 3$, where $T_{\mathrm{c,max}}$ is the time-to-capture of all evaders. The greedy assignment would instead assign pursuer 1 to evader 1 and pursuer 2 to evader 2, with the time-to-capture of all evaders equal to $T_{\mathrm{c,max}} = 100$.

V. Experiments

Multi-target tracking and a pursuit evasion game using the control system *LochNess* were demonstrated at the Defense Advanced Research Projects Agency (DARPA) Network Embedded Systems Technology (NEST) final experiment on August 30, 2005. The experiment was performed under warm sunny conditions on a large-scale, long-term, outdoor sensor network testbed deployed on a short grass field at U.C. Berkeley's Richmond Field Station (see Figure 13). A total of 557 sensor nodes were deployed and 144 of these nodes were allotted for the tracking and PEG experiments. However, six out of the 144 nodes used in the experiment were not functioning on the day of the demo, reflecting the difficulties of deploying large-scale, outdoor systems.

The 144 nodes used for the tracking and PEG experiments were deployed at approximately 5 meter spacing in a 12×12 grid (see Figure 14). Each node was elevated using a camera tripod to prevent the passive infrared (PIR) sensors from being obstructed by grass and uneven terrain (see Figure 13(a)). The locations of the nodes were measured during deployment using differential GPS and stored in a table at the base station for reference and for generating Figure 14. However, in the experiments the system assumed the nodes were placed exactly on a 5 meter spacing grid to highlight the robustness of the system with respect to localization error.

(a) (b)

Fig. 13. Hardware for the sensor nodes. (a) Trio sensor node on a tripod. On top is the microphone, buzzer, solar panel, and user and reset buttons. On the sides are the windows for the passive infrared sensors. (b) A live picture from the 2 target PEG experiment. The targets are circled.

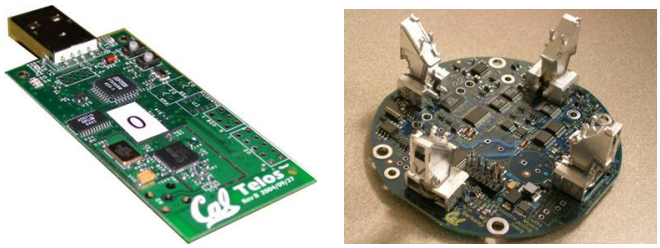
Fig. 14. Sensor network deployment (not all deployed sensor nodes are shown). The disks and circles represent the positions of the sensor nodes. The network of 144 nodes used in the multi-target tracking and PEG experiments is highlighted.

The deployment of *LochNess* contained some modifications to the architecture described in Section III. Due to the time constraint, the Tier-2 nodes were not fully functional on the day of the demo. Instead, we used a mote connected to a personal computer as the Tier-2 node. Only one such Tier-2 node was necessary to maintain connectivity to all 144 nodes used for the tracking experiment. In the experiment, simulated pursuers were used since it was difficult to navigate a ground robot in the field of tripods.

A. Platform

A new sensor network hardware platform called the *Trio* mote was designed by Dutta *et al.* [44] for the outdoor testbed. The Trio mote is a combination of the designs of the *Telos B* mote, eXtreme Scaling Mote (XSM) sensor board [67], and Prometheus solar charging board [68], with improvements. Figure 15 shows the Trio node components and Figure 13(a) shows the assembled Trio node in a waterproof enclosure sitting on a tripod.

The Telos B mote [69] is the latest in a line of wireless sensor network platforms developed by U.C. Berkeley for the



(a) (b)

Fig. 15. (a) Telos B. (b) Trio sensor board, based off the XSM sensor board and Prometheus solar power circuitry. See [44] for details.

NEST project. It features an 8MHz Texas Instruments MSP430 microcontroller with 10kB of RAM and 48kB of program flash and a 250kbps, 2.4GHz, IEEE 802.15.4 standard compliant, Chipcon CC2420 radio. The Telos B mote provides lower power operation than previous motes (5.1 μA sleep, 19 mA on) and a radio range of up to 125 meters, making it the ideal platform for large-scale, long-term deployments.

The Trio sensor board includes a microphone, a piezoelectric buzzer, x-y axis magnetometers, and four passive infrared (PIR) motion sensors. For the multi-target tracking application, we found that the PIR sensors were the most effective at sensing human subjects moving through the sensor field. The magnetometer sensor had limited range even detecting targets with rare earth magnets and the acoustic sensor required complex signal processing to pick out the various acoustic signatures of a moving target from background noise. The PIR sensors provided an effective range of approximately 8 meters, with sensitivity varying depending on weather conditions and time of day. The variability in the signal strength of the PIR sensor reading prohibited extraction of ranging information from the sensor, so the PIR sensors were used as binary detectors.

The software running on the sensor nodes are written in NesC [70] and run on TinyOS [71], an event-driven operating system developed for wireless embedded sensor platforms. The core sensor node application is the *DetectionEvent* module, a multi-mode event generator for target detection and testing node availability. The sensor node application relies on a composition of various TinyOS subsystems and services that facilitate management and interaction with the network (see Figure 16).

The *DetectionEvent* module provides four modes of event generation from the node – events generated periodically by a timer; events generated by pressing a button on the mote; events generated by the raw PIR sensor value crossing a threshold; and events generated by a three-stage filtering, adaptive threshold, and windowing detection algorithm for the PIR sensor signal developed by the University of Virginia [75]. The timer generated events were parsed and displayed at the base station to help visualize which nodes in the network were alive. The three-stage PIR detection filter code was used during the development cycle. While it had potential to be more robust to different environmental conditions, during the day of the demo we reverted to the simple threshold PIR detector because

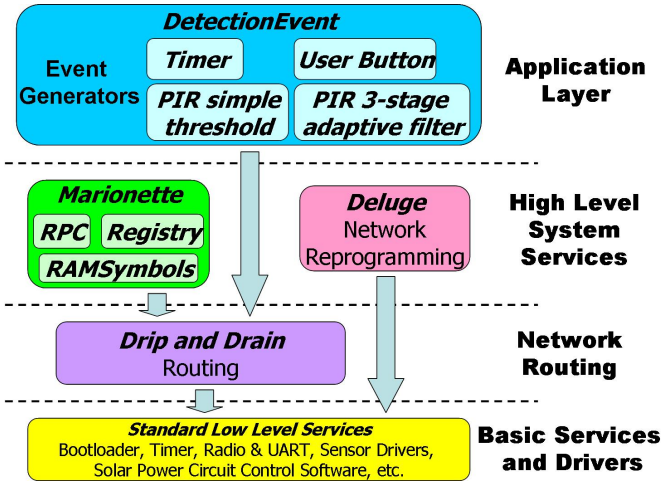


Fig. 16. Software services on the sensor network platform. The core network management services are *Deluge* for network reprogramming [72] and *Marionette* for fast reconfiguration of parameters on the nodes [73]. The *DetectionEvent* application relies on the *Drip* and *Drain* routing layer for dissemination of commands and collection of data [74]. For more details on the software architecture used in the outdoor testbed, see [44, 73].

the simple threshold detector was easy to tune and performed well.

The algorithms for the MSF, MTT, MTF, and MAC modules are all written in MATLAB and C++ and run on the base station in real-time. The same implementation of the tracking algorithm and the robust minimum time controller used in the simulations shown in Figure 10 and Figure 12 are used in the experiments. The data was timestamped at the base station.

B. Live Demonstration

The multi-target tracking algorithm was demonstrated on one, two, and three human targets, with targets entering the field at different times. In all three experiments, the tracking algorithm correctly estimated the number of targets and produced correct tracks. Furthermore, the algorithm correctly disambiguated crossing targets in the two and three target experiments without classification labels on the targets, using the dynamic models and target trajectories before crossing to compute the tracks.

Figure 17 shows the multi-target tracking results with three targets walking through the field. The three targets entered and exited the field around time 10 and 80, respectively. During the experiment, the algorithm correctly rejected false alarms and compensated for missing detections. There were many false alarms during the span of the experiments, as can be seen from the false alarms before time 10 and after time 80 in Figure 18. Also, though not shown in the figures, the algorithm dynamically corrected previous track hypotheses as it received more sensor readings. Figure 18 also gives a sense of the irregularity of network traffic. The spike in traffic shortly after time 50 was approximately when two of the targets crossed. It shows that the multi-target tracking algorithm is robust against missing measurements, false measurements, and the irregularity of network traffic.

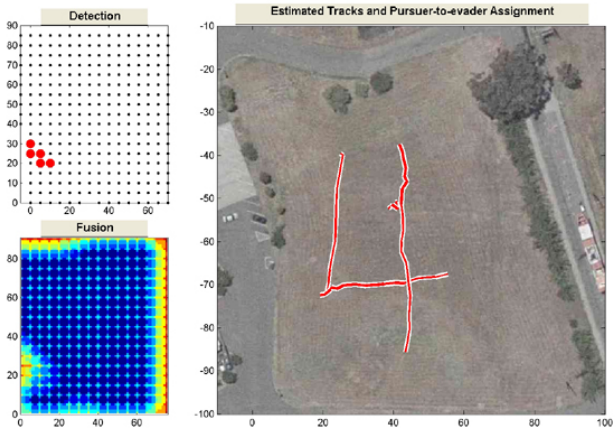


Fig. 17. Estimated tracks of targets at time 70 from the experiment with three people walking in the field. (upper left) Detection panel. Sensors are marked by small dots and detections are shown in large disks. (lower left) Fusion panel shows the fused likelihood. (right) Estimated Tracks and Pursuer-to-evader Assignment panel shows the tracks estimated by the MTT module, estimated evader positions (stars) and pursuer positions (squares).

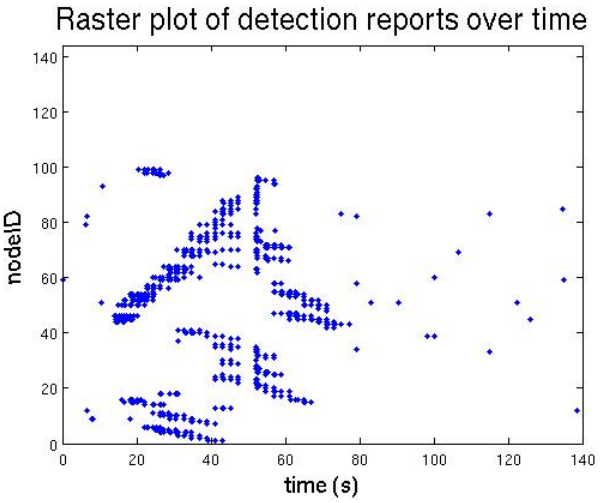
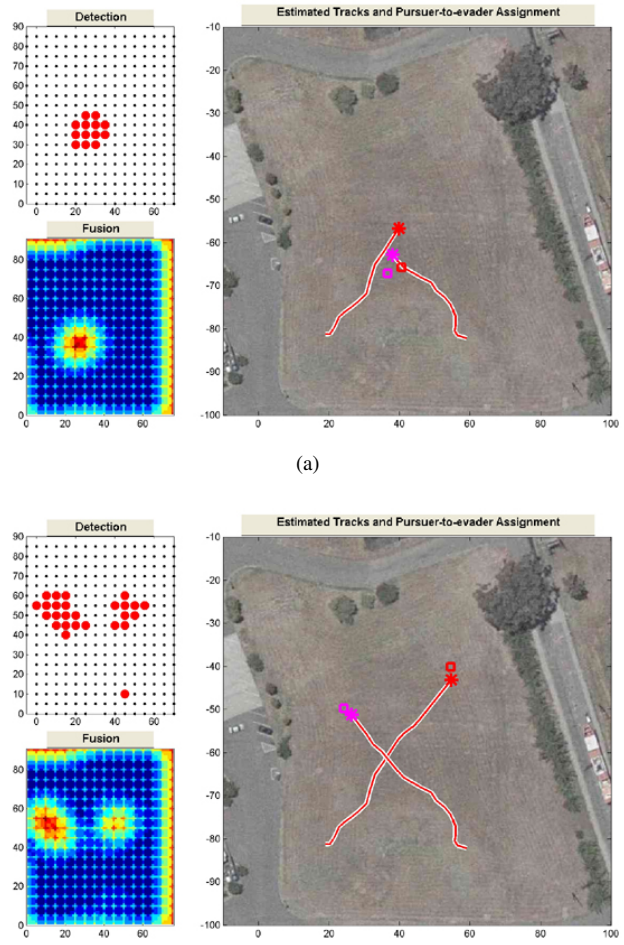


Fig. 18. Raster plot of the binary detection reports from the three target tracking demo. Dots represent detections from nodes that were successfully transmitted to the base station.

In the last demonstration, two simulated pursuers were dispatched to chase two crossing human targets. The pursuer-to-target assignment and the robust minimum time-to-capture control law were computed in real-time, in tandem with the real-time tracking of the targets. The simulated pursuers captured the human targets, as shown in Figure 19. In particular, note that the MTT module is able to correctly disambiguate the presence of two targets (right panel of Figure 19(a)) using past measurements, despite the fact that the MSF module reports the detection of a single target (upper left panel of Figure 19(a)). A live picture of this experiment is shown on the right of Figure 13.



(a)

(b)

Fig. 19. Estimated tracks of evaders and pursuer positions from the pursuit evasion game experiment. (a) Before crossing. (b) After crossing.

VI. Conclusions and Future Work

This paper described *LochNess*, a hierarchical real-time control system for sensor networks. *LochNess* is applied to pursuit evasion games, in which a group of evaders are tracked using a sensor network and a group of pursuers are coordinated to capture the evaders. Although sensor networks provide global observability, they cannot provide high quality measurements in a timely manner due to packet loss, communication delay, and false detections. These factors have been the main challenge to developing a real-time control system using sensor networks.

This paper proposes a possible solution for closing the loop around wireless ad-hoc sensor networks. The hierarchical real-time control system *LochNess* decouples the estimation of evader states from the control of pursuers by using multiple layers of data fusion, including the multi-sensor fusion (MSF) module, the multi-target tracking (MTT) module, and the multi-track fusion (MTF) module. While a sensor network generates noisy, inconsistent, and bursty measurements, the three layers of data fusion convert raw sensor measurements into fused measurements in a compact and consistent repre-

sentation and forward the fused measurements to the pursuers' controllers in a timely manner.

In order to coordinate multiple pursuers, the multi-agent coordination (MAC) module is developed. The assignments of pursuers to evaders are chosen such that the time to capture all evaders is minimized. The controllers for the pursuers are based on minimum-time control but were designed to account for the worst-case evader motions and to add robustness to the quantization of inputs, measurement and process noises, and modeling errors.

Simulation and experimental results have shown that *LochNess* is well suited for solving real-time control problems using sensor networks and that a sensor network is an attractive solution for the surveillance of a large area.

In this work, we assumed a stationary hierarchy, *i.e.*, the Tier-2 nodes and base station are fixed. However, a stationary hierarchy is not robust against malicious attacks. In our future work, we will address this issue by introducing redundancy, distributing the coordination tasks among Tier-2 nodes, and dynamically managing the hierarchy of the system. Our immediate goal is to quantify the robustness of the system against false measurements and packet loss and to identify the sensor network parameters such as maximum delay rate, maximum packet loss rate, and maximum false detection rate, necessary for seamless operation of the control system.

ACKNOWLEDGEMENT

We thank the anonymous reviewers for their helpful comments and suggestions. The experiments were made possible by contributions from David Culler, Prabal Dutta, Eric Fraser, Mike Howard, Jonathan Hui, Jaein Jeong, August Joki, Sukun Kim, Philip Levis, Michael Manzo, Joseph Polastre, Travis Pynn, Peter Ray, Tanya Roosta, Shawn Schaffert, Cory Sharp, Bruno Sinopoli, Jay Taneja, Gilman Tolle, David Shim, Robert Szewczyk, Kamin Whitehouse, and Bonnie Zhu. This material is based upon work supported by the Defense Advanced Research Projects Agency under Grant No. F33615-01-C-1895 (NEST), the National Science Foundation under Grant No. EIA-0122599, the Team for Research in Ubiquitous Secure Technology (TRUST), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: Cisco, ESCHER, HP, IBM, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia and United Technologies, and the European Community Research Information Society Technologies under Grant No. RECSYS IST-2001-32515 and No. SENSNET-MIRG-6-CT-2005-014815, and by the Italian Ministry of Education, University and Research (MIUR).

REFERENCES

[1] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Connecting the physical world with pervasive networks," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 59–69, January 2002.

[2] I. A. W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–116, August 2002.

[3] "Sensor networks and applications," *Proceedings of the IEEE, Special Issue*, vol. 91, no. 8, pp. 1151–1256, August 2003.

[4] B. Warnake, M. Scott, B. Leibowitz, L. Zhou, C. Bellew, J. Chediak, J. Kahn, and B. B. K. Pister, "An autonomous $16mm^3$ solar-powered node for distributed wireless sensor networks," in *IEEE International Conference on Sensors 2002*, Orlando, FL, USA, June 2002, pp. 1510–15.

[5] S. Roundy, D. Steingart, L. Frchette, P. Wright, and J. Rabaey, "Power sources for wireless networks," in *Proc. 1st European Workshop on Wireless Sensor Networks (EWSN '04)*, Berlin, Germany, January 2004, pp. 1–17.

[6] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, Salt Lake City, UT, May 2001.

[7] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer, Special Issue in Sensor Networks*, Aug. 2004.

[8] "10 emerging tecnology that will change the world," *Technol. Rev.*, vol. 106, no. 1, pp. 33–49, February 2003.

[9] M. Kintner-Meyer and R. Conant, "Opportunities of wireless sensors and controls for building operation," *Energy Engineering Journal*, vol. 102, no. 5, pp. 27–48, 2005.

[10] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. M. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communication of the ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[11] M. Nekovee, "Ad hoc sensor networks on the road: the promises and challenges of vehicular ad hoc networks," in *Workshop on Ubiquitous Computing and e-Research*, Edinburgh, UK, May 2005.

[12] A. Willig, K. Matheus, and A.Wolisz, "Wireless technology in industrial networks," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1130–1151, June 2005.

[13] A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. B. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello, "Making sensor networks practical with robots," in *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, London, UK, 2002, pp. 152–166.

[14] R. R. Brooks, D. Friedlander, J. Koch, and S. Phoha, "Tracking multiple targets with self-organizing distributed ground sensors," *J. Parallel Distrib. Comput.*, vol. 64, pp. 874–884, 2004.

[15] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, , and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks*, vol. 46, no. 5, pp. 605–634, Dec. 2004.

[16] J. Hespanha, H. Kim, and S. Sastry, "Multiple-agent probabilistic pursuit-evasion games," in *IEEE Int. Conf. on Decision and Control*, 1999, pp. 2432–2437.

[17] R. Vidal, O. Shakernia, J. Kim, D. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, October 2002.

[18] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning and Autonomous Robots (joint issue)*, vol. 31, no. 5, pp. 1–25, 1998.

[19] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry and Applications*, vol. 9, no. 4/5, pp. 471–493, 1999.

[20] B. Sinopoli, C. Sharp, S. Schaffert, L. Schenato, and S. Sastry, "Distributed control applications within sensor networks," *IEEE Proceedings Special Issue on Distributed Sensor Networks*, November 2003.

[21] S. Oh, S. Russell, and S. Sastry, "Markov chain Monte Carlo data association for general multiple-target tracking problems," in *Proc. of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, Dec. 2004.

[22] ——, "Markov chain Monte Carlo data association for multiple-target tracking," Univ. of California, Berkeley, Tech. Rep. UCB//ERL M05/19, 2005.

[23] S. Oh, L. Schenato, P. Chen, and S. Sastry, "A scalable real-time multiple-target tracking algorithm for sensor networks," Univ. of California, Berkeley, Tech. Rep. UCB//ERL M05/9, 2005.

[24] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," in *Proc. of the 2nd European Workshop on Wireless Sensor Networks*, January 2005, pp. 93–107.

[25] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed group management for track initiation and maintenance in target localization applications," in *Proc. of the 2nd workshop on Information Processing in Sensor Networks*, April 2003.

[26] J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *J. of Applied Signal Processing*, April 2003.

[27] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*. San Diego, CA: Academic Press, 1988.

[28] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automatic Control*, vol. 24, no. 6, pp. 843–854, December 1979.

[29] C. Chong, S. Mori, and K. Chang, "Distributed multitarget multisensor tracking," in *Multitarget-Multisensor Tracking: Advanced Applications*, Y. Bar-Shalom, Ed. Artech House: Norwood, MA, 1990, pp. 247–295.

[30] D. Li, K. Wong, Y. H. Hu, and A. Sayeed, "Detection, classification and tracking of targets," *IEEE Signal Processing Magazine*, vol. 17-29, March 2002.

[31] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: An information directed approach," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1999–1209, Aug. 2003.

[32] D. McErlean and S. Narayanan, "Distributed detection and tracking in sensor networks," in *Proc. of the 36th Asilomar Conference on Signal, System and Computers*, November 2002.

[33] J. Aslam, Z. Butler, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *ACM International Conference on Embedded Networked Sensor Systems*, 2003.

[34] W. Chen, J. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," in *Proc. of the 11th IEEE International Conference on Network Protocols*, November 2003.

[35] M. Coates, "Distributed particle filters for sensor networks," in *Proc. of the 3nd workshop on Information Processing in Sensor Networks*, April 2004.

[36] S. Oh and S. Sastry, "Tracking on a graph," in *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, Los Angeles, CA, April 2005.

[37] J. Shin, L. Guibas, and F. Zhao, "A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks," in *Proc. of the 2nd workshop on Information Processing in Sensor Networks*, April 2003.

[38] M. Chu, S. Mitter, and F. Zhao, "Distributed multiple target tracking and data association in ad hoc sensor networks," in *Proc. of the 6th International Conference on Information Fusion*, July 2004.

[39] J. Liu, J. Liu, M. Chu, J. Reich, and F. Zhao, "Distributed state representation for tracking problems in sensor networks," in *Proc. of the 3nd workshop on Information Processing in Sensor Networks*, April 2004.

[40] R. Sittler, "An optimal data association problem on surveillance theory," *IEEE Trans. Military Electronics*, vol. MIL-8, pp. 125–139, April 1964.

[41] J. Collins and J. Uhlmann, "Efficient gating in data association with multivariate distributed states," *IEEE Trans. Aerospace and Electronic Systems*, vol. 28, no. 3, pp. 909–916, July 1992.

[42] A. Poore, "Multidimensional assignment and multitarget tracking," *Partitioning Data Sets. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 19, pp. 169–196, 1995.

[43] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 41–46, 2004.

[44] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments," in *Proc. of the International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors*, 2006.

[45] R. Burkard and R. Çela, "Linear assignment problem and extensions," Karl-Franzens University of Graz, Graz, Austria, Tech. Rep. 127, 1998.

[46] A. Nilim and L. E. Ghaoui, "Algorithms for air traffic flow management under stochastic environments," in *Proc. of American Control Conference*, 2004.

[47] D. Shim, H. Kim, and S. Sastry, "Decentralized reflective model predictive control of multiple flying robots in dynamic environment," in *Proc. of IEEE Conf. on Decision and Control*, Las Vegas, 2003.

[48] H. Pasula, S. J. Russell, M. Ostland, and Y. Ritov, "Tracking many objects with many sensors," in *Proc. of the International Joint Conference on Artificial Intelligence*, Stockholm, 1999.

[49] T. Kurien, "Issues in the design of practical multitarget tracking algorithms," in *Multitarget-Multisensor Tracking: Advanced Applications*, Y. Bar-Shalom, Ed. Artech House, Norwood, MA, 1990.

[50] S. Oh, "Multiple target tracking for surveillance," Univ. of California, Berkeley, Tech. Rep. UCB/ERL MO3/54, 2003.

[51] M. Lepetic, G. Klancar, I. Skrjanc, D. Matko, and B. Potocnic, "Time optimal path planning considering acceleration limits," *Robotics and Autonomous Systems*, vol. 45, pp. 199–210, 2003.

[52] A. Saccon, "Minimum time maneuver for nonholonomic car with acceleration constraints: Preliminary results," in *13th Mediterranean Conference on Control and Automation (MED)*, Limassol, Cyprus, 2005.

[53] E. Velenis and P. Tsiotras, "Optimal velocity profile generation for given acceleration limits: Receding horizon implementation," in *American Control Conference (ACC05)*, Portland, OR, USA, June 2005, pp. 2147–2152.

[54] C. Belta, V. Isler, and G. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, October 2005.

[55] P. Tabuada and G. Pappas, "Hierarchical trajectory refinement for a class of nonlinear systems," *Automatica*, vol. 41, no. 4, pp. 701–708, April 2005.

[56] T. Kailath, A. Sayed, and B. Hassibi, *State Space Estimation*. Prentice-Hall, 1999.

[57] D. Lerro and Y. Bar-Shalom, "Interacting multiple model tracking with target amplitude feature," *IEEE Trans. Aerospace and Electronic Systems*, vol. 29, pp. 494–509, 1993.

[58] I. Beichl and F. Sullivan, "The Metropolis algorithm," *Computing in Science and Engineering*, vol. 2, no. 1, pp. 65–69, 2000.

[59] M. Jerrum and A. Sinclair, "The Markov chain Monte Carlo method: An approach to approximate counting and integration," in *Approximations for NP-hard Problems*, D. Hochbaum, Ed. PWS Publishing, Boston, MA, 1996.

[60] W. Gilks, S. Richardson, and D. Spiegelhalter, Eds., *Markov Chain Monte Carlo in Practice*, ser. Interdisciplinary Statistics Series. Chapman and Hall, 1996.

[61] G. Roberts, "Markov chain concepts related to sampling algorithms," in *Markov Chain Monte Carlo in Practice*, ser. Interdisciplinary Statistics Series, W. Gilks, S. Richardson, and D. Spiegelhalter, Eds. Chapman and Hall, 1996.

[62] Z. Gao, "On discrete time optimal control: A closed-form solution," in *Proceeding of the 2004 American Control Conference (ACC)*, Boston, Massachusetts, U.S.A., June 2004, pp. 52–58.

[63] R. Zanasi and R. Morselli, "Discrete minimum time tracking problem for a chain of three integrators with bounded input," *Automatica*, vol. 39, pp. 1643–1649, 2003.

[64] E. Lee and L. Markus, *Foundations of optimal control theory*. New York: Wiley, 1967.

[65] E. Ryan, *Optimal relay and saturation control synthesys*. London: Peter Peregrinus Ltd., 1982.

[66] T. Basar and G. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. London and San Diego: Academic Press, 1995.

[67] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," in *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, April 2005.

[68] X. Jiang, J. Polastr, and D. Culler, "Perpetual environmentally powered sensor networks," in *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, April 2005.

[69] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. of the Fourth International Conference on Information Processing in Sensor Networks*, April 2005.

[70] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesc language: a holistic approach to networked embedded systems," in *Proc. of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, June 2003, pp. 1–11.

[71] "TinyOS," http://www.tinyos.net/.

[72] J. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004.

[73] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler, "Marionette: Providing an interactive environment for wireless debugging and development," in *Proc. of the International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors*, 2006.

[74] G. Tolle, "A network management system for wireless sensor networks," Master's thesis, Univ. of California, Berkeley, 2005.

[75] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, T. He, A. Tirumala, Q. Cao, J. Stankovic, T. Abdelzaher, and B. Krogh, "Lightweight detection and classification for wireless sensor networks in realistic environments," in *SenSys*, November 2005.