Swarm Coordination for Pursuit Evasion Games using Sensor Networks*

Luca Schenato, Songhwai Oh and Shankar Sastry
Dept. of Electrical Engineering and Computer Sciences
University of California at Berkeley
333 Cory Hall, Berkeley, CA 94720, U.S.A.
{*lusche|sho|sastry*}*@eecs.berkeley.edu*

Prasanta Bose
Modeling Simulation and Information Sciences Dept.
Adv. Tech. Center, Lockheed Martin Space Systems Company
3251 Hanover St., Palo Alto, CA, U.S.A.
prasanta.bose@lmco.com

***Abstract*— In this work we consider the problem of pursuit evasion games (PEGs) where a group of pursuers is required to detect, chase and capture a group of evaders with the aid of a sensor network in minimum time. Differently from standards PEGs where the environment and the location of evaders is unknown and a probabilistic map is built based on the pursuer's onboard sensors, here we consider a scenario where a sensor network, previously deployed in the region of concern, can detect the presence of moving vehicles and can relay this information to the pursuers. Here we propose a general framework for the design of a hierarchical control architecture that exploits the advantages of a sensor network by combining both centralized and decentralized real-time control algorithms. We also propose a coordination scheme for the pursuers to minimize the time-to-capture of all evaders. In particular, we focus on PEGs with sensor networks orbiting in space for artificial space debris detection and removal.**

***Index Terms*— Sensor networks, pursuit evasion games, vehicle coordination, space vehicles, space debris**

I. Introduction

Recent developments in integrated circuits, radio communication and sensor technology have provided us with a wealth of inexpensive, customizable, small, embedded sensor systems, computers and wireless radios. Therefore, deploying and maintaining a network of thousands of nodes, each provided with its own sensors, small computer and wireless radio, and capable to communicate with neighboring nodes, is becoming feasible from a technological as well as an economical perspective. Such systems, commonly known as Sensor Networks (SNs) are gaining a role of importance in the research community since they promise an unprecedented quantity and quality of information unobtainable with current technology [7].

In this work, we are interested in adopting SNs for Pursuit Evasion Games (PEGs). PEGs are a mathematical abstraction arising from numerous situations which addresses the problem of controlling a swarm of autonomous agents in the pursuit of one or more evaders [9] [11]. Typical examples are search and rescue operations, surveillance, localization and tracking of moving parts in a warehouse, and search and capture missions. In some cases, the evaders are actively avoiding detection as in capture missions, whereas in other cases their motion is approximately random as in rescue operations. In general PEGs the environment and the location of evaders is unknown. In this framework, an additional map-learning phase is required to precede the pursuit phase. In fact, pursuers have a relatively small detection range. They usually employ computer vision or ultrasonic sensors, providing only local observability over the area of interest [21]. This constraint makes designing a cooperative pursuit algorithm harder because lack of complete observability only allows for suboptimal pursuit policies. See Figure 1(left). Furthermore, a smart evaders makes the map-building process dynamic since their location changes over time. The map-learning phase is, by itself, time-consuming and computationally intensive even for simple two-dimensional rectilinear environments [5]. Moreover, inaccurate sensors complicate this process and a probabilistic approach is often required [21].

Fig. 1. Sensor Visibility in PEGs without SN (left) and with SN (right). Dots correspond to the SN nodes, each provided with a vehicle detection sensor. Courtesy of [20]

The use of a sensor network can greatly improve the overall performance of a PEG [20]. In fact, with sensor networks, complete visibility of the field and communication over a long radius is possible. See Figure 1(right). Global pursuit policies can then be used to efficiently find the optimal solution regardless of the level of intelligence of the evader. Also, with a sensor network, the number of pursuers needed is likely a function exclusively of the number of evaders and not to the size of the field.

However, SNs pose a series of novel problems that need to be considered to fully exploit its potential. These problems arise from the networked nature of SNs. In fact, sensor measurements within the network need to be relayed via multi-hop wireless communications to dedicated hubs or supernodes where high level signal processing and pursuing algorithms are implemented. As a consequence random packet delays, missing observations, false alarm, sensor noise, and imprecise local estimation of location of evaders is very common [20].

In this paper, we propose a general framework for analyzing and designing control algorithms for PEGs which takes into account the limitations and exploits the advantages of SNs. Although, the proposed framework and control architecture are sufficiently general to be applied to different PEGs scenarios, we developed and tested our algorithms on a specific instance of PEGs and SNs for Earth's space monitoring and debris detection. In the past decades,

*This work was partially performed as part of DyMND (Dynamic Meshes of Networked Devices) project of Lockheed Martin Space Systems Company funded by DARPA NEST [contract n. F33615-02-C4033]

military, scientific and commercial space missions have created a considerable amount of space debris which is like wandering bullets endangering future space missions. Today's radar-based surveillance systems can only detect and track objects that are larger than $10cm$ in size, which account for only few percents of total space debris weight. Even debris of just the size of few millimeters can wreck a mission [10]. Sensor networks could be efficaciously deployed in space to detect and track centimeter and sub-centimeter debris, and then dedicated space vehicles, acting as space vacuum cleaners, could chase and remove these debris before they enter safe regions for space missions (see Figure 2). This space scenario can be well formulated analyzed as a PEG with SN, where the debris are the evaders and the space vehicles are the pursuers.

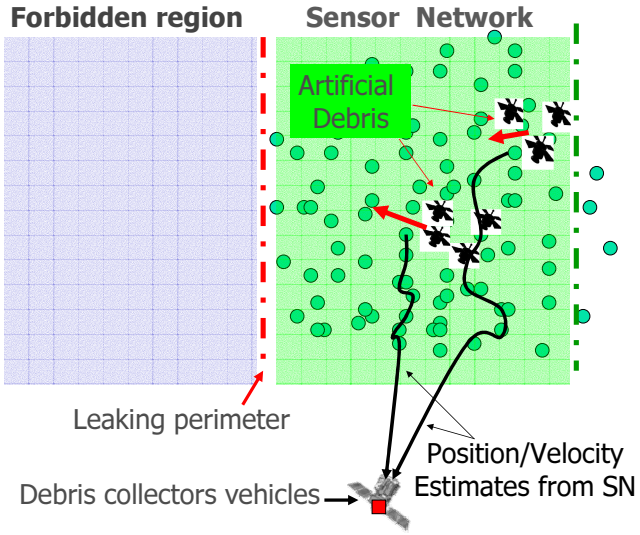

Fig. 2. PEGs scenario for Earth's space monitoring, artificial debris detection and collection.

II. System Modeling

In this section we introduce mathematical models for the sensor network and for the dynamics of the pursuers and evaders, which will be used to design and test control algorithms.

A. Sensor Networks

In this section, we describe the sensor network and sensor model used for simulations in Section VIII. Let N_{s} be the number of sensor nodes, including both supernodes and regular nodes, deployed over the surveillance region $\mathcal{R} \subset \mathbb{R}^2$. We assume that each supernode can communicate with its neighboring supernodes. Let $s_i \in \mathcal{R}$ be the location of the i-th sensor node and let $S = \{s_i : 1 \leq i \leq N_{\mathrm{s}}\}$. Let $R_{\mathrm{t}} \in \mathbb{R}$ be the transmission range of a regular sensor node. A pair of sensor nodes i and j can communicate to each other if the Euclidean distance $\|s_i - s_j\| \leq R_{\mathrm{t}}$. Let $G = (S, E)$ be a communication graph such that $(s_i, s_j) \in E$ if and only if $\|s_i - s_j\| \leq R_{\mathrm{t}}$. Let $N_{\mathrm{ss}} \ll N_{\mathrm{s}}$ be the number of supernodes and let $s_j^{\mathrm{s}} \in S$ be the position of the j-th supernode, for $j = 1, \ldots, N_{\mathrm{ss}}$. Let g be the assignment of each sensor to its nearest supernode such that $g(i) = j$ if $\|s_i - s_j^{\mathrm{s}}\| = \min_{k=1,\ldots,N_{\mathrm{ss}}} \|s_i - s_k^{\mathrm{s}}\|$. For a node i, if $g(i) = j$, then the shortest path from s_i to s_j^{s} in G is denoted by $sp(i)$.

Let $R_{\mathrm{s}} \in \mathbb{R}$ be the sensing range. If there is an object at $x \in \mathcal{R}$, each sensor within radius R_{s} from x detects the presence of the object with the detection probability p_{d}. The detection of an object by the sensor i is recorded by the sensor's signal strength, $z_i = \frac{\beta}{1+\gamma\|s_i - x\|^\alpha} + w_i$, where α, β and γ are constants specific to the sensor type and they are normalized such that w_i has the standard Gaussian distribution. This signal-strength based sensor model is general for sensors available in sensor networks, such as acoustic and magnetic sensors, and has been used frequently [12], [13], [14]. For each i, if $z_i \geq \eta$, where η 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 its $2R_{\mathrm{s}}$ neighborhood. Note that this approach is similar to the leader election scheme in [13] and we assume that $R_{\mathrm{t}} \geq 2R_{\mathrm{s}}$. However, this approach may cause some missing observations if there is more than one object in this disk of radius $2R_{\mathrm{s}}$. A better approach to fuse local data is required and we will address this issue in our future work. For the node i, if z_i is the 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 as $\hat{z}_i = \sum_{j=1}^{k} z_{i_j} s_{i_j} / \sum_{j=1}^{k} z_{i_j}$. Then $\hat{z}_i$ is transmitted to the supernode $g(i)$ via the shortest path $sp(i)$. If z_i is not the largest compared to the incoming messages, the node i does nothing and goes back to the sensing mode. Although each sensor cannot give an accurate estimate of object's position, as more sensors collaborate, the accuracy of estimates improves [17]. The collaboration of sensors makes the system more robust against node failures and we can increase the detection probability and decrease the false alarm rate by collaboration.

A transmission along the edge (s_i, s_j) fails independently with probability p_{te} and the message never reaches a supernode. So 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 supernode, the probability of sucessful 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 supernode. The probability of successful communication from the reporting node i to its supernode $g(i)$ can be computed as $1 - \left(1 - (1 - p_{\mathrm{te}})^k\right)^k$, where $k = |sp(i)|$.

The (additional) communication delay is modeled by the negative binomial distribution. We assume each node has the same probability p_{de} of delaying a message. If d_i is the number of delays occurred on the message originating from the sensor i, d_i is distributed as

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

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

B. Vehicle Dynamics

In this work we assume pursuers and evaders are spacecrafts orbiting at an almost constant altitude and velocity around the Earth. These vehicles are allowed to move on the surface of the imaginary sphere whose radius is given

by the mean altitude. If we consider a frame R_o attached to this imaginary sphere which orbits with the same mean velocity, then vehicles appear as they would move on a plane. Each vehicle is provided with four perpendicular monodirectional thrusters that generate forces along the longitudinal and lateral axes of the vehicle body, and four small differential thrusters that generate rotational torque about the vertical axis. The actuation of the thrusters is much faster than the dynamics of vehicle body, therefore it can be assumed that thrusters can be controlled almost instantaneously. The magnitude of the thruster output is bounded and the total amount of energy expenditure available on each vehicles is limited. Damping in space is negligible and therefore vehicle dynamics is purely inertial. Mathematically, the vehicle dynamics can be written as follows:

$$m\ddot{x} = u_x^b \cos(\theta) - u_y^b \sin(\theta) \quad (2)$$

$$m\ddot{y} = u_x^b \sin(\theta) + u_y^b \cos(\theta) \quad (3)$$

$$I_z\ddot{\theta} = u_\theta^b \quad (4)$$

where m is mass of the vehicle, I_z is the moment of inertia relative to the vertical axis, x, y, θ are the $x-y$ position and orientation of the vehicle relative to the inertial frame R_o, respectively. The inputs u_x^b, u_y^b, u_θ^b need to satisfy the following constraints:

$$|u_x^b| \le U_x, \quad |u_x^b| \le U_y, \quad |u_x^b| \le U_\theta \quad (5)$$

$$\int_0^{+\infty} \left(|u_x^b(t)|^2 + |u_y^b(t)|^2 + |u_\theta^b(t)|^2\right) dt \le E \quad (6)$$

where U_x, U_y, U_θ, E are constants that quantify the input magnitude and the energy bounds.

In the next sections, rather than the previous model of vehicle dynamics, we will use the following abstract model of vehicle dynamics for determining pursuer-to-evader assignment and coordinated path planning:

$$\ddot{x} = u_x^o, \qquad \ddot{y} = u_y^o \quad (7)$$

where the inputs satisfy the following constraints:

$$|u_x^o| \le U_x^o = \frac{U_x}{\sqrt{2}m}, \quad |u_x^b| \le U_y^o = \frac{U_y}{\sqrt{2}m} \quad (8)$$

$$\int_0^{+\infty} \left(|u_x^o(t)|^2 + |u_y^o(t)|^2\right) dt \le E^o = mE \quad (9)$$

It should be clear that any trajectory generated according to the abstract dynamics (7) is feasible for the exact dynamics (2). In fact, given the input pair (u_x^o, u_y^o) for the abstract dynamics, we can compute the input for the exact dynamics (u_x^b, u_y^b) using the following transformation:

$$u_x^b = m(u_y^o \sin(\theta) + u_x^o \cos(\theta)) \quad (10)$$

$$u_y^b = m(u_y^o \cos(\theta) - u_x^o \sin(\theta)) \quad (11)$$

assuming that the orientation angle θ is known. Only the energy constraint (6) can be violated by the abstract model. However in practice $|u_\theta^b(t)|^2 \ll |u_x^b(t)|^2 + |u_y^b(t)|^2$ since generating a rotational torque requires much less energy than generating a longitudinal or lateral thrust, and the vehicle does not need to change its orientation very often.

III. Control System Architecture

The goal of the control system architecture is to devise coordination and control algorithms to minimize the time required to capture multiple evaders by multiple pursuers with the aid of the information acquired through the sensor network. These algorithms need to take into account the limitations arising from the sensor networks, such as packet loss, random delay of observations, data ambiguity, false alarms, and the constraints of vehicles dynamics, such as limited thrust magnitude, energy budget, computational resources, digital quantized input, measurement noise and external perturbations.

We propose an architecture that relies on four key ideas. The first idea is to use a hierarchical modular structure, where each module is designed independently from the others to improve scalability and interpretability. The second idea is to use robust predictive control which tries to predict evaders motion to coordinate pursuers accordingly to minimize capture time. This approach can cope with random delays in the observations and packet loss, but it requires to take into account the uncertainties of the prediction which degrades as further in the future one tries to predict. Therefore, it needs to be robust to prediction errors. The third idea is to use centralized coordination performed by one of the pursuers or a specific unit in order to improve global performance, since the goal is to capture the "furthest" evader. Although this approach requires communication among pursuers and possibly high computational resources, it can greatly reduce capture time of evaders. The last idea is to implement a purely distributed collision avoidance path follower controller on each pursuer. Although the desired trajectory generated by the centralized path planner are optimal and collision-free, uncertainty and external disturbance can create critical situations where one pursuer can collide with another. This reactive controller will modify pursuer motion based on onboard sensors to avoid collision while still trying to follow the desired trajectory.

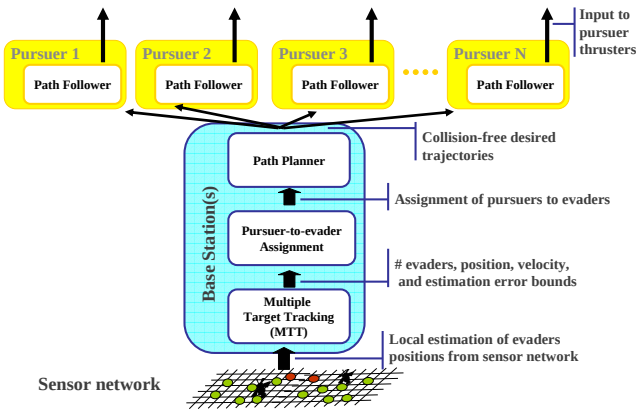

Fig. 3. Control Unit Architecture

Based on these ideas we propose an hierarchical control architecture composed of four layers: the *multiple target tracking (MTT)* module, the *pursuer-to-evader assignment* module, the *path planner* module, and the *path followers* modules, as shown in Fig. 3. The first three modules reside on one or few specialized structures with high computational power which collect global information about pursuers as well evaders position. These structures could

be some fixed base stations or some elected leaders from pursuers themselves, which monitor the whole sensor network or a very large fraction of it. Specifically, the MMT module collects the measurements from the sensor network which correspond to local estimates of evaders' position. However, some of these measurements are dropped before they arrive, most of them arrive with considerable delay and, most importantly, they are not associated to any specific evader. This module uses this information to estimate the number of evaders moving in the sensor network, their predicted current position and velocity, and the uncertainty on the prediction in statistical terms. This information is then passed along with the current position and velocity of each pursuer, to the pursuer-to-evader assignment module, which estimates the expected time to capture from each pursuer to each evader. Based on these estimates, this module assign one pursuer to one evader such that the estimated time to capture of the last evader is minimized. Once the assignment is determined, the path planner module determine the best trajectory for each pursuer to minimize capture time for the assigned evader, while avoiding possible collisions among pursuers, i.e. these trajectories are collision-free trajectories. Then each of these trajectories is transmitted to the corresponding pursuer. Each pursuer is provided with its own path follower controller that tries to track the desired trajectory. This controller receives information from onboard sensors and if on obstacle or another pursuer enter its sensing region, it modifies its own trajectory trying to follow the desired trajectory as close as possible, while maintaining a safe distance to avoid collisions.

Such a control architecture shares many similarities with those currently used for air traffic management (ATM) systems [15], where multiple airplanes needs to be routed to the destination airports in minimum time while avoiding collisions. These systems present an hierarchical approach similar to the one proposed here. The major difference is that in our scenario the evaders are equivalent to moving airports whose motion is unknown. This adds substantial uncertainty and a frequent update of pursuers routes (trajectories).

In the next sections we describe in detail the implementation of each module of the proposed architecture.

IV. Multiple Target Tracking (MTT)

We assume a sensor network is deployed over a bounded region $\mathcal{R} \in \mathbb{R}^n$ and provide observations about the moving evaders. We consider the most general setup in which the number of evaders and the states of evaders are unknown. In order to compute the control laws of the pursuers, it is of paramount importance to estimate precisely the number of evaders and their states. The estimation problem of multiple moving targets is known as multiple target tracking. Multiple target tracking has been extensively studied in radar-based tracking and vision-based tracking [1], [4]. Under the most general setup, a varying number of indistinguishable targets is moving around in a region with continuous motions and the positions of moving targets are sampled at random intervals. The measurements about the positions are noisy, with detection probability less than one, and there is a noise background of spurious position reports, i.e., false alarms. Targets arise at random in space and time. Each target persists independently for a random length of time and ceases to exist. A track of a target is defined as a path in space-time traveled by the target. In most cases, there is no clear association between targets and observations, requiring a solution to the data association problem to associate observations to targets.

In sensor networks, we seek for an autonomous tracking algorithm which does not require a continuous monitoring by a human operator. We also need to consider the following constraints on sensor networks. Due to the limited supply of power, the multi-hop wireless ad-hoc communication is used in sensor networks. In many cases, the communication bandwidth is low and the communication links are not reliable, causing transmission failures. In addition, due to the low communication bandwidth and a limited amount of memory, communication delays can occur frequently. It is well known that communication is costlier than computation in sensor networks in terms of power usage [6]. Hence it is essential to fuse local observations before the transmission. Since the data association problem is NP-hard [3], [18], we cannot expect to solve it with only local information. But, at the same time, we cannot afford to have a centralized algorithm since such solution cannot be scalable. In summary, we need a simple and efficient tracking algorithm that is robust against the low detection probability and high false alarm rates; capable of initiating and terminating tracks; uses less memory; combines local information to reduce the communication load; and is scalable. Also it must be robust against transmission failures and communication delays. But at the same time we want an algorithm that can provide a good solution and improve its solution toward the optimal solution given an enough computation time.

The algorithm developed in our companion paper [17] is a general multiple-target tracking algorithm for sensor networks which can systematically track an unknown number of targets in the presence of false alarms and missing observations and robust against sensor localization error, transmission failures and communication delays. The algorithm is based on the efficient Markov chain Monte Carlo (MCMC) data association algorithm which is capable of tracking a varying number of targets [16]. It has been demonstrated that the algorithm achieves remarkable performance compared to MHT under the extreme conditions, such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates [16]. In [17], the MCMC data association algorithm is extended in a hierarchical manner so that the algorithm becomes scalable and its robustness against sensor localization error, transmission failures and communications delays is demonstrated.

V. Pursuer-to-Evader Assignment

In a scenario 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 of all evaders. However, other criteria might be possible, such as minimization of pursuers energy while guaranteeing capture of all evaders or maximization of number of captured evaders within a certain amount of time. Since evaders motion is not known, exact time to capture is not known either, therefore we need to define a metric to estimate the time-to-capture. We will use the following definition of time-to-capture:

Definition 1. *Let $(p_e(t_0), v_e(t_0)) \in \mathbb{R}^2 \times \mathbb{R}^2$ the position and velocity of a evader at time $t = t_0$, and $(p_p(t_1), v_p(t_1)) \in \mathbb{R}^2 \times \mathbb{R}^2$ the position and velocity of a pursuer at time $t = t_1 \geq t_0$. We define* ***time-to-capture*** *the minimum time T_c necessary for the pursuer to reach the evader with the same velocity, assuming that the evader will keep moving with constant velocity, i.e.,*

$$T_c \triangleq \min_T \{T \mid p_p(t_1+T) = p_e(t_1+T), v_p(t_1+T) = v_e(t_1+T)\}$$

where $p_e(t_1+T) = p(t_0)+(t_1+T-t_0)v(t_0)$, $v_e(t_1+T) = v(t_0)$, and the pursuer moves according to its dynamics.

This definition allow us to quantify time-to-capture in an unambiguous way and, although evader can change trajectories over time, it is a more accurate estimate than, for example, some metric based on the distance between the evader and the pursuer, since time-to-capture incorporates the dynamics of pursuers. Moreover, it is well-defined for any arbitrary time delay $t_d = t_1 - t_0$ in the estimate of evader position and velocity relative to current time t_1. Given this definition and the constraints on the dynamics of the pursuer, it is possible to calculate explicitly the time-to-capture T_c, as well as the input to the actuators of the pursuers as described in the next section.

A. Minimum Time-to-Capture Control

The computation of time-to-capture will be estimated using the abstract model of pursuer dynamics given by (7) and (8). If we define the position error between the pursuer and the evader as $e(t) \triangleq p_p(t) - p_e(t)$, then the time-to-capture problem is equivalent to the following optimization problem:

$$\begin{array}{ll} \min_{u_x^o(t), u_y^o(t)} & T \\ \text{subject to} & \left\{ \begin{array}{l} \ddot{e}_x(t) = u_x^o(t), \quad \ddot{e}_y(t) = u_y^o(t) \\ |u_x^o(t)| \leq U_x^o, \quad |u_y^o(t)| \leq U_y^o \\ e_x(T) = \dot{e}_x(T) = e_y(T) = \dot{e}_y(T) = 0 \end{array} \right. \end{array}$$

Since the pursuer dynamics is decoupled along the two axes, the solution of this problem can be obtained directly from the well known problem of minimum time control of a double integrator. The solution is given by a bang-bang control that can be written in feedback form as follows:

$$u_x^o(t) = \left\{ \begin{array}{cl} -U_x^o & \text{If} \quad 2U_x^o \dot{e}_x > -e_x|e_x| \\ +U_x^o & \text{If} \quad 2U_x^o \dot{e}_x < -e_x|e_x| \\ -U_x^o \text{sign}(e_x) & \text{If} \quad 2U_x^o \dot{e}_x = -e_x|e_x| \\ 0 & \text{If} \quad \dot{e}_x = e_x = 0 \end{array} \right. \tag{12}$$

The minimum time can be also be written in terms of the position and velocity error as follows:

$$T_{c,x}(e_x, \dot{e}_x) = \left\{ \begin{array}{ll} \frac{-\dot{e}_x + \sqrt{2\dot{e}_x^2 - 4U_x^o e_x}}{U_x^o} & \text{If} \quad 2U_x^o \dot{e}_x \geq -e_x|e_x| \\ \frac{\dot{e}_x + \sqrt{2\dot{e}_x^2 + 4U_x^o e_x}}{U_x^o} & \text{otherwise} \end{array} \right. \tag{13}$$

Similar equations can be written for the y-axis, therefore the minimum time to capture is given by:

$$T_c = \max(T_{c,x}, T_{c,y}) \tag{14}$$

Despite its simplicity and apparent efficacy minimum-time control is rarely used since it very sensitive to modeling errors in the dynamics and in the implementation. In fact, even the apparently innocuous digital implementation of the feedback given in (12) can exhibit poor performance and input chattering even in absence of any kind of noise. Therefore, although in principle minimum time control is attractive since it gives the best performance, it needs to be modified to be able to cope with practical issues such as digital implementation of control feedback, quantization of inputs, measurement and process noise. Let us first consider a more realistic model of error dynamics. Since the dynamics along the x- and y-axes are decoupled, we simply consider the dynamics along one of the axis:

$$\begin{aligned} e_{k+1} &= e_k + v_k T_d + 0.5 u_k^o T_d^2 + 0.5 w_k T_d^2 \quad (15) \\ v_{k+1} &= v_k + u_k^o T_d + w_k T_d \quad (16) \end{aligned}$$

where $e_k = e_x(kT_d)$, $v_k = \dot{e}_x(kT_d)$, $u_k^o = u_x^o(kT_d)$, T_d is the discretization time interval, and w_k is an unknown external disturbance that models the fact that the evader has an unknown trajectory. Also we assume there is measurement noise so that the estimated error position and velocity are given by:

$$\begin{aligned} \hat{e}_k &= e_k + z_k \quad (17) \\ \hat{v}_k &= v_k + n_k \quad (18) \end{aligned}$$

where z_k and n_k are the measurement noise for the error position and velocity respectively.

As explained above, the implementation of feedback given by (12) can give poor performance, even in ideal scenario where no external disturbance and no measurement noise is present. It is true that the smaller the discretization time T_d is, the smaller the error between the theoretical and the true performance is. However, this is still unsatisfactory and, even for very small $T_d \ll T_c$ relative to the capture time, the performance degradation is considerable. Recently, [8] and [22] solved the problem of minimum time optimal control for the discretized double and triple intergrator, respectively. In this case the difference relative to the ideal case is negligible and it is possible to show that $T_{c,\text{digital}} \leq T_c + 2T_d$. However, these control still perform poorly if implemented in a scenario where even small external disturbances and measurement noise are present (see Figure 4). Here we propose a robust minimum time feedback control that takes into account uncertainty about exact motion and position of evader.

First, we consider bounded external disturbance $|w_k| \leq W$, and measurement noise, $|z_k| \leq Z, |n_k| \leq N$, where W, Z, N are positive scalars. The feedback control input will be chosen based on the following min-max optimization problem

$$u_k^* = \min_{|u_k^o| \leq U} \left(\max_{|w_k| \leq W, |z_k| \leq Z, |n_k| \leq N} T_{c,x}(e_{k+1}, v_{k+1}) \right) \tag{19}$$

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

$$\begin{aligned} u_k^* &= \min_{|u_k^o| \leq U} \max \left(T_{c,x}(e_{k+1}^+, v_{k+1}^+), T_{c,x}(e_{k+1}^-, v_{k+1}^-) \right) \\ e_{k+1}^\pm &\triangleq \hat{e}_k + \hat{v}_k T_d + 0.5 u_k^o T_d^2 \pm (0.5 W T_d^2 + N T_d + Z) \\ v_{k+1}^\pm &\triangleq \hat{v}_k + u_k^o T_d \pm (W T_d + N) \end{aligned}$$

In the interest of space, derivations are not included in this work and will be presented in a forthcoming technical paper. The solution of the previous equation can be obtained analytically by solving a quartic polynomial equation, however, in realistic scenarios, the input u_k^o is quantized

and takes values from finite set $U_q = \{U_1, \ldots, U_p\}$. In this case, the solution of the previous problem could be obtained by simply evaluating the time-to-capture for those p input values and then choosing the minimizer. This is computationally efficient since the computation of time-to-capture can be done in parallel for each value of the input and it involves only the evaluation of sums, multiplications and square roots.

Figure 4 shows the performance of the robust minimum time-to-capture control feedback for tracking of an evader which does not move on a straight line with constant velocity and position and velocity estimates of evader is affected by noise. It is compared with the discrete-time minimum time controller proposed in [22] and [8]. Our controller feedback design outperforms the discrete-time minimum time controller since the latter one does not take into account process and measurement noise. Note how both controllers do not direct pursuers toward the actual position of evader, but try to estimate future location and therefore minimize the time-to-capture.

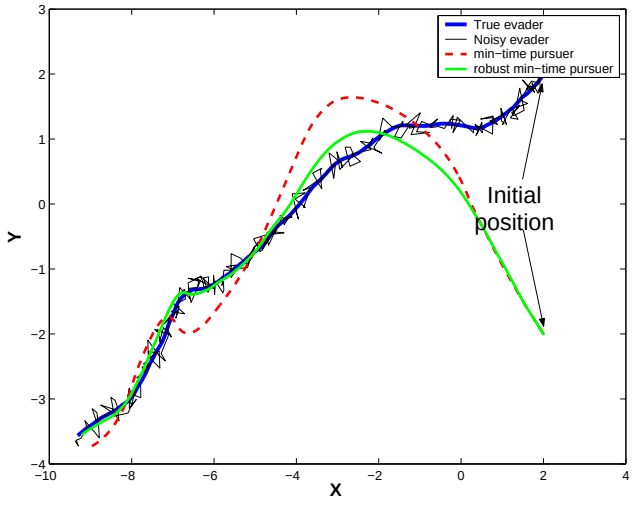

Fig. 4. Trajectories of pursuers and evaders on x-y plane. Feedback control is based on noisy estimate (*thin solid line*) of true evader position (*thick solid line*). The robust minimum time-to-capture feedback proposed in this paper (*dot-solid line*) is compared with the discrete-time minimum time-to-capture feedback (*dashed line*) proposed in [22] .

B. Assignment Algorithms

In the previous section, we presented a definition to compute the expected time-to-capture for a pursuer-evader pair and, as a byproduct, we also computed the robust minimum-time control feedback. Therefore, given positions and velocities of all pursuers and evaders, it is possible to compute the time-to-capture matrix $C = [c_{i,j}] \in \mathbb{R}^{N_p \times N_e}$, where N_p and N_e are the total number of pursuers and evaders, respectively, and the entry $c_{i,j}$ 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 select an assignment. Our objective is to select an assignment that minimizes the expected time-to-capture of *all* evaders. Let us assume for now that we have the same number of pursuers and evaders, i.e. $N_p = N_e$.

An assignment can be represented as a matrix $X = [x_{i,j}] \in \mathbb{R}^{N_p \times N_e}$, where the entry $x_{i,j}$ of the matrix X 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 follows:

$$\begin{array}{l} \min_{x_{i,j} \in \{0,1\}} \max_{i,j=1,\ldots,N} (c_{i,j} \cdot x_{i,j}) \\ \text{subject to } \sum_{i=1}^{N} x_{i,j} = 1, \sum_{j=1}^{N} x_{i,j} = 1 \end{array} \tag{20}$$

As formulated in the previous equation, the assignment problem appears as a combinatorial problem.

One simple *greedy assignment* algorithm that tries to solve the optimization problem above, is to look for the smallest time-to-capture entry in the matrix C, assign the corresponding pursuer-evader pair, and remove the corresponding row and column from the matrix C which now becomes of dimension $(N-1) \times (N-1)$, and repeat the same process until each pursuer is assigned to an evader. Although it is straightforward and easy to implement, this is a suboptimal algorithm, since there are cases when the greedy assignment gives the worst solution. Consider the time-to-capture matrix $C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. The greedy assignment would assign pursuer 1 to evader 1 and pursuer 2 to evader 2, with time-to-capture of last evader equal to $T_{max} = 4$. However, the assignment that minimizes the time-to-capture of last evader is $(1, 2)$ and $(2, 1)$, which gives $T_{max} = 3$.

The optimization problem given in (20) can be reformulated as a *linear bottleneck assignment* problem and can be solved with polynomial-time algorithm based on network flow theory. The actual implementation of these algorithms goes beyond the scope of this paper and we address the interested reader to the survey [2] and references therein.

Figure 5 compares the greedy assignment with the optimal linear bottleneck assignment for a scenario with three pursuers and three evaders. The greedy assignment assign the closest pursuer-evader pair and in fact the first evader is "captured" already at time $t = 50$ (top-right plot), while according to the linear bottleneck assignment the same pursuer is assigned to the farthest evader. However, at time $t = 75$ all three evaders are captured by the pursuers employing the optimal assignment, while only two evaders are captured by the pursuers employing the greedy assignment (bottom-left plot).

When the number of pursuers and evaders is different, different options are available. For example, if pursuers are less numerous than the evaders, one could try to capture the N_p closest evaders and neglect the other $N_e - N_p$ evaders. If pursuers are more numerous than the evaders, then extra pursuers can be assigned to some evaders or the unassigned $N_p - N_e$ pursuers could stay in stand-by in case additional evaders appear in a later time. The optimal choice in these cases depends on the specific application and it will be investigated in a future work.

VI. Collision-free Coordinated Path Planning

Once an assignment is chosen by the pursuer-to-evader assignment module, the trajectories generated by the robust minimum time controller described in the previous section are not guaranteed to be collision free, i.e. $||x_i(t) - x_j(t)|| \geq \epsilon$ for all time t, where i, j are two different pursuers and ϵ is related to the minimum safe distance. Although the distributed path follower controller module described in the next section can overcome this problem by replanning on-line the trajectory to avoid collision, it possible to use global information about the location

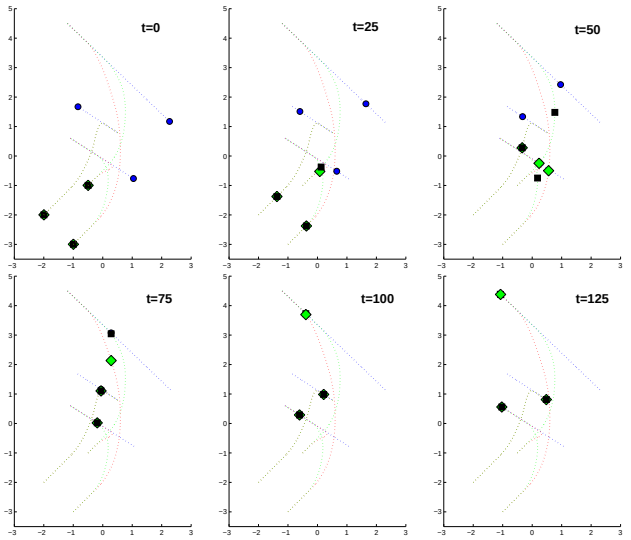

Fig. 5. Movie of evaders tracking on the x-y plane. Evaders (*circles*) move along straight lines with constant velocity. Pursuers using greedy assignment algorithm (*diamonds*) and optimal linear bottleneck assignment (*squares*) are superimposed.

of pursuers and evaders available at the base stations or supernodes to generate pursuer *off-line* trajectories that are collision-free. Finding the minimum time-to-capture, collision-free trajectories given a specified assignment is a hard problem, since it requires the solution of N_p coupled optimization problems. One possible approach would be to use dynamic programming to *simultaneously* design these trajectories [15]. Unfortunately, the complexity of this approach grows exponentially with the number of pursuers, i.e. $O(A^{N_p})$, where A is an appropriate constant, thus making it impractical for a large swarm of pursuers.

One alternative approach is to solve this optimization problem *sequentially*. Given an assignment, it is possible to compute the expected time-to-capture and the expected trajectory for each pursuer-evader pair. These trajectories are not necessarily collision free. However, it is possible to order this assignment based on the expected time-to-capture. The sequential algorithm works as follows. First, the pursuer with the largest expected time-to-capture will follow the trajectory generated by the minimum time-to-capture controller. Second, this trajectory is frozen and the pursuer with the second largest expected time-to-capture is required to capture its assigned evader while avoiding the first pursuer. This problem can be solved via dynamic programming [15] or via model predictive control [19] and it has complexity $O(A)$, since only one pursuer is involved in this optimization problem. Then also this trajectory is frozen, and the pursuer with the third largest time-to-capture is required to chase its assigned evader while avoiding the previous two pursuers. Once again, this is an optimization problem with complexity $O(A)$, since only one pursuer is involved in this optimization problem. The process is then continued similarly by incrementally freezing pursuers trajectories till all pursuers' trajectories are generated. Although the sequential optimization algorithm is only suboptimal, it is likely to give a good performance since the pursuers that need to change the most their trajectory to avoid the other pursuers, are those which had the smallest expected time-to-capture. Therefore, the time-to-capture of last evader, after the collision-free replanning algorithm, should not be much greater than the ideal time-to-capture when collision free trajectories are not enforced. Moreover, the complexity of this algorithm is linear in the number of pursuer, i.e. $O(N_p A)$, since it requires the solution of $N-1$ optimization problem where only one pursuer's trajectory is designed at one time.

We are currently implementing the simultaneous and sequential algorithms and results will be presented in a future paper.

VII. Distributed Path Follower Controller

The minimum time-to-capture control generates desirable trajectory path $y_i(t)$ for each pursuer i. In pursuit of the assigned evader, the pursuer may encounter other pursuers or obstacles. We need to make sure that the pursuer can reactively avoid collisions while following the desirable trajectory path. We use the decentralized model predictive control technique developed in [19] to generate collision-avoiding path follower control laws. Let $x_i(t)$ be the state of the i-th pursuer and $u_i(t)$ be the control for pursuer i. Then the decentralized discrete-time optimal control problem is to find the optimal control $\{u_i^*(t)\}_{k=1}^T$, where T is the time horizon, such that

$$\{u_i^*(t)\}_{t=1}^T = \arg\min \sum_{t=1}^{T} q_i(x_i(t), u_i(t)) + q_{if}(x_i(T+1)) \tag{21}$$

subject to the dynamics of pursuer i. $q_i(\cdot)$ is the cost-to-go and $q_{if}(\cdot)$ is the terminal cost. For each time step k, we solve for $\{u_i^*(t)\}_{t=k}^T$ from the state $x_i(k)$ and apply $u_i^*(k)$. For the next step $k+1$, we repeat these steps from the state $x_i(k+1)$. The terminal cost is defined as

$$q_{if}(x_i(T+1)) = \frac{1}{2}\left(y_i(t) - Cx_i(t)\right)^T P_0 \left(y_i(t) - Cx_i(t)\right)$$

for a symmetric positive-definite matrix P_0. The cost-to-go is decomposed into two parts

$$q_i(x_i(t), u_i(t)) = q_i^t(x_i(t), u_i(t)) + q_i^c(x_i(t), u_i(t)),$$

where $q_i^t(x_i(t), u_i(t))$ is the cost of being away from the desired path and $q_i^c(x_i(t), u_i(t))$ is the cost of being close to obstacles. q_i^t is defined in the usual form

$$q_i^t(x_i(t), u_i(t)) = \tfrac{1}{2}\Big[(y_i(t) - Cx_i(t))^T Q (y_i(t) - Cx_i(t)) + u_i(t)^T R u_i(t)\Big],$$

where Q and R are symmetric positive-definite matrices. Suppose that there are N obstacles, including other pursuers. We use the potential field method to compute q_i^c as

$$q_i^c(x_i(t), u_i(t)) = \sum_{j=1}^{N} \frac{K_j \mathbb{I}(j \neq i)}{\left(x_i(t) - x_j(t)\right)^T Q_c \left(x_i(t) - x_j(t)\right)},$$

where Q_c is a symmetric positive-definite matrix and K_j a constant determining the shape of the potential function.

The optimization problem of Equation (21) is nonlinear, therefore its solution is not guaranteed by any algorithm. However, iterative gradient descent algorithms initialized with the initial desired trajectory $y_i(t)$ has been found to quickly converge to a solution with very good performance. More details can be found in [19].

VIII. Simulations

For simulations below, we consider the surveillance over a rectangular region on a plane, $\mathcal{R} = [0, 100]^2$. The state vector is $x = [x, y, \dot{x}, \dot{y}]^T$ where (x, y) is a position in $\mathcal{R}$ along the usual x and y axes and $(\dot{x}, \dot{y})$ is a velocity vector. The following linear dynamic and measurement models are used

$$\begin{array}{rcl} x_{t+1} & = & A(\delta)x_t + G(\delta)w_t \\ y_t & = & Cx_t + v_t, \end{array} \tag{22}$$

where δ is a sampling interval, w_t and v_t are white Gaussian noises with zero mean and covariance $Q = \text{diag}(.15^2, .15^2)$ and $R = \text{diag}(\frac{R_s^2}{16}, \frac{R_s^2}{16})$, respectively, and

$$A(\delta) = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad G(\delta) = \begin{bmatrix} \frac{\delta^2}{2} & 0 \\ 0 & \frac{\delta^2}{2} \\ \delta & 0 \\ 0 & \delta \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T$$

The transmission and sensing ranges are $R_t = 20$ and $R_s = 5$, respectively, which correspond to a SN of 400 nodes. For the sensor model, we use $\alpha = 2$, $\gamma = 1$ and $\beta = 2(1 + \gamma R_s^\alpha)$. We used $p_e^t = .05$ and $p_e^d = .05$. The false alarms are uniformly distributed over $\mathcal{R}$ and its rate is one false alarm per time. For each sensor, the detection probability is .95.

Figure 6 shows snapshots of the PEGs scenario with 5 evaders and 5 pursuers. Note how the assignment is dynamic and it is pursuers are reassigned as new evaders appear or disappear in order to minimize the time-to-capture of all evaders.

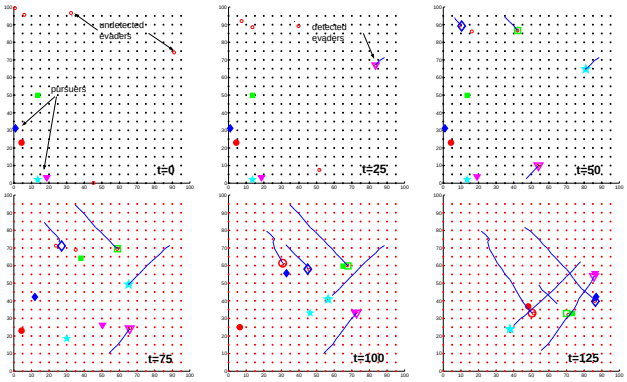

Fig. 6. Snapshots of PEGs using SN x-y plane. Evaders (*large hollow icons*) move within a sensor network (*small circles*). Pursuers (*small filled icons*) using linear bottleneck assignment algorithm and optimal linear bottleneck assignment (*squares*) chase evaders. The pursuer-evader assignment is indicated using the same icon.

IX. Conclusion and Future Work

In this paper we presented a framework to analyze and design algorithms for Pursuit Evasion Games with Sensor Networks. We presented a mathematical formulation of sensor network and vehicle dynamics and a hierarchical control architecture that exploits the benefits of using a sensor network. We also proposed a series of algorithms to combine both coordinated maneuvering and distributed control of pursuers at different stages in order to minimize time-to-capture of all evaders while guaranteeing safety and collision free maneuvers of pursuers.

Future work would include a more extensive comparison between probabilistic PEGs with PEGs using SNs in order to evaluate the trade-offs between the two approaches. Also, the algorithms presented here can be extended more rigorously when there are more pursuers than evaders and coordinated maneuvering of pursuers allow the capture of "fast and smart" evaders similarly as observed in mobs of lions hunting an agile pray.

References

[1] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Mathematics in Science and Engineering Series 179 Academic Press, San Diego, CA, 1988.

[2] R.E. Burkard and R. Çela. Linear assignment problem and extensions. Technical Report 127, Karl-Franzens University of Graz, Graz, Austria, 1998.

[3] J.B. Collins and J.K. Uhlmann. Efficient gating in data association with multivariate distributed states. In *IEEE Trans. Aerospace and Electronic Systems*, volume 28(3), 1992.

[4] I.J. Cox. A review of statistical data association techniques for motion correspondence. In *International Journal of Computer Vision*, volume 10(1), pages 53–66, 1993.

[5] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *Journal of the ACM*, 45(2):215–245, 1998.

[6] L. Doherty, B. A. Warneke, B. Boser, and K. S. J. Pister. Energy and performance considerations for smart dust. In *International Journal of Parallel and Distributed Sensor Networks*, Dec 2001.

[7] 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.

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

[9] J.P. Hespanha, H.J. Kim, and S.S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *IEEE Int. Conf. on Decision and Control*, pages 2432–2437, 1999.

[10] N.L. Johnson. Monitoring and controlling debris in space. *Scientific American*, pages 62–68, August 1998.

[11] H.J. Kim, R. Vidal, D.H. Shim, O. Shakernia, and S.S. Sastry. A hierarchical approach to probabilistic pursuit evasion games with unmanned ground and aerial vehicles. In *IEEE Int. Conf. on Decision and Control*, pages 634–639, 1901.

[12] J.J. Liu, J. Liu, M. Chu, J.E. Reich, and F. Zhao. Distributed state representation for tracking problems in sensor networks. In *Proc. of 3nd workshop on Information Processing in Sensor Networks (IPSN)*, April 2004.

[13] Juan Liu, Jie Liu, James Reich, Patrick Cheung, and Feng Zhao. Distributed group management for track initiation and maintenance in target localization applications. In *Proc. of 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, April 2003.

[14] Seapahn Meguerdichian, Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. Exposure in wireless ad hoc sensor networks. In *Procs. of 7th Annual International Conference on Mobile Computing and Networking*, pages 139–150, July 2001.

[15] A. Nilim and L. El Ghaoui. Algorithms for air traffic flow management under stochastic environments. In *Proc. of American Control Conference*, 2004.

[16] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain monte carlo data association for general multiple target tracking problems. In *43rd IEEE Conference on Decision and Control (to appear)*, Paradise Island, Bahamas, December 2004.

[17] Songhwai Oh, Luca Schenato, and Shankar Sastry. A hierarchical multiple-target tracking algorithm for sensor networks. In *International Conference on Robotics and Automation (submitted)*, Barcelona, Spain, 2005.

[18] A.B. Poore. Multidimensional assignment and multitarget tracking. In *Partitioning Data Sets. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 19, pages 169–196, 1995.

[19] D.H. Shim, H.J. Kim, and S.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.

[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 Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots (joint issue)*, 31(5):1–25, 1998.

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