RECONNAISSANCE AND SURVEILLANCE IN URBAN TERRAIN WITH UNMANNED AERIAL VEHICLES

Tamir Hegazy [*] **Ben Ludington** [*]
George Vachtsevanos [*]

[*] *Department of Electrical and Computer Engineering, Georgia Institute of Technology*

Abstract: Unmanned vehicles endowed with attributes of intelligence are finding applications in reconnaissance, surveillance, and rescue operations in both military and civilian domains. We are introducing a methodology to locate a number of such Unmanned Aerial Vehicles (UAVs) optimally over an urban environment and identify and track potential ground targets using a particle filtering framework. The particle filtering framework, when used in conjunction with novel initialization and adaptation techniques, is a robust, reliable state estimation tool that avoids many of the drawbacks of classical techniques. Simulation results support the efficacy of the proposed approach and validate the effectiveness of the algorithmic developments. *Copyright ©2005 IFAC*

Keywords: Target tracking, Manoeuvring target, Agents, Autonomous vehicles

1. INTRODUCTION

The term *target tracking* is often used to refer to the task of finding/estimating the motion parameters (mainly the location and direction) of a moving target in a time sequence of measurements. This task is achievable as long as the target is within the sensor's field of view (FOV). If it happens that the target keeps moving away to the point it runs off the FOV, the target tracking task will fail to track the moving target until the target re-enters the sensor's FOV. To address such problems, the sensor is mounted on a moving platform such as a UAV. We call the new setup (the sensor plus the UAV) an agent. Thus, we can start a second task, other than the target tracking task, to (reactively or proactively) move the sensor to guarantee that the target stays in view. That second task is what we call the *agent placement* task. The work presented in this paper is of the *active sensing-based target tracking* variety, in which both tasks discussed above are integrated.

The problem of interest to this paper has been the focus of attention of some researchers in the past few decades. Such agent placement approaches as A-CMOMMT (Parker, 2002), CEP (Hegazy and Vachtsevanos, 2004), and *spreading out* (Batalin and Sukhatme, 2002) have addressed the agent placement problem for a general region of interest. The main difference between those approaches and the approach presented in this paper is that this paper is mainly interested in surveillance and reconnaissance in urban environments. Therefore, the problem is formulated accordingly to better accommodate the nature of urban environments.

2. AGENT PLACEMENT

2.1 Problem Statement

<u>Introduction</u> We use a formulation of the variety of *Weighted Cooperative Multi-robot Observation of Multiple Moving Targets* (W-CMOMMT)

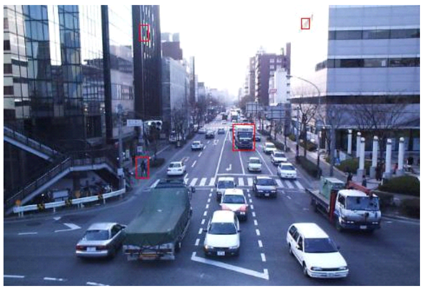


Fig. 1. A possible view from a mini-UAV hovering over a street intersection.

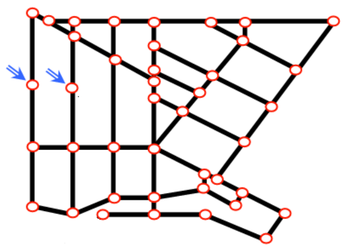


Fig. 2. Inserting excessive observation points.

(Werger and Mataric, 2001) since it captures the multiple-observer-multiple-target scenario with target prioritization. W-CMOMMT can be shown to be NP-hard (Hegazy and Vachtsevanos, 2004).

In this paper, the choice is made to limit the observation points (OPs) to street intersections. The rationale behind this choice is illustrated in Figure 1, which shows a possible view from a mini-UAV hovering over a street intersection. As evident from the picture, the agent can easily observe all the moving vehicles on the street as well as the building surfaces all the way to the next street intersection. All street segments around an agent can be visually observed simultaneously using either multiple cameras or a single panoramic camera. Thus, the region of interest (ROI) is modelled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices; each vertex v_i represents an OP, and $\mathcal{E}$ is a set of edges; each edge e_i represents a street segment. Starting with a street map, $\mathcal{G}$ can be created by placing vertices at the street intersections. Extra OPs can be inserted as needed. For example, when a street segment is too long to be covered by a single agent's FOV, one or more OPs can be inserted along the segment to split it into two or more smaller segments. This idea is illustrated in Figure 2.

Problem Formulation Given:

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: a graph representing the ROI,
- R: a team of m agents that are located on $\mathcal{U} \subseteq \mathcal{V}$ and have observation sensors with a FOV that can cover the longest street segment represented in $\mathcal{G}$,
- $O(t)$: a set of n targets, such that, at time t, target $o_j(t)$ is located on some edge $e_k \in \mathcal{E}$, and
- $\mathbf{U}$: a vector of utility values such that u_j reflects the utility value gained by observing o_j.

Define an $m \times n$ Boolean matrix $\mathbf{\Gamma}(t)$, where $\gamma_{ij}(t)$ is equal to 1 if and only if o_j is located on an edge that is incident to the vertex where r_i is located.

The goal is to provide a dynamic agent placement policy to maximize the normalized global utility function

$$G = \frac{\sum_{t=0}^{T} \sum_{j=1}^{n} u_j \bigvee_{i=1}^{m} \gamma_{ij}(t)}{T \sum_{j=1}^{n} u_j}, \tag{1}$$

where $\bigvee$ is the logical disjunction operator, and T is the time interval during which the tracking mission is carried out.

2.2 Approach

Coarse Motion Model A target currently residing on an edge e connecting the vertices v_1 and v_2 may stay on the same edge e or move to any other edge incident to v_1 or v_2. Target transitions among street segments follow a stochastic model described by an M^{th}-order Markov chain, which can be parsimoniously represented using the generalized mixture transition distribution (GMTD) model (Berchtold and Raftery, 2002) as

$$\mathbf{X}(t) = \sum_{i=1}^{M} \lambda_i \mathbf{Q}_i \mathbf{X}(t-i), \tag{2}$$

where λ_i's add up to 1, $\mathbf{X}(t)$ is a vector representing the probability distribution of target location over $\mathcal{E}$ at time t, $\mathbf{Q}_i$ is the i-step transition matrix.

Agents can use the model to predict the target locations at future time instant h as a probability distribution using

$$\hat{\mathbf{X}}_j(t+h) = \sum_{i=1}^{M} \lambda_i \mathbf{Q}_i \hat{\mathbf{X}}_j(t+h-i), \tag{3}$$

where $\hat{\mathbf{X}}_j$ is the best estimate of the target location as a probability distribution. The lags $\hat{\mathbf{X}}_j(t+h-i)$ are obtained in the light of the output of the target tracking task.

Distributed Algorithm Every sampling period t, each agent r_i broadcasts its location $v_i(t) \in \mathcal{V}$, the set of locally observed targets, and their locations. Each agent then builds a weighted version of the graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, w)$, where $w : \mathcal{E} \to \mathbb{R}$ can be described by *probable edge outcomes* for a future time horizon H, which we define as

$$w(e, H) = \sum_{o_j \in O} u_j \left(\sum_{h=1}^{H} v_h \left(\hat{\mathbf{X}}_j (t+h) \cdot \mathbf{Y}_e \right) \right) \quad (4)$$

where v_h describes the contribution of the future prediction at time $t+h$ to the placement decision (v_h's add up to 1), and $\mathbf{Y}_c$ is a binary vector with all zeros and a one at the location corresponding to e. Equation 4 sums up the *probability* $\times$ *utility* quantities over all observed targets over a future horizon H. Note that since targets change their locations over time, the edge weights w are time-dependent, and so is $\mathcal{H}$. However, the time-dependence notation is left out for clarity.

The algorithm attempts to maximize the global utility defined in Equation 1 by searching for a set of OPs $\mathcal{U} \subseteq \mathcal{V}$ that maximizes the coverage at each time step, which can be defined as

$$g = \sum_{e \in \mathcal{E}} adj_{\mathcal{U}}(e)\, w(e), \quad (5)$$

where $adj_{\mathcal{U}} : \mathcal{E} \rightarrow \{0, 1\}$, such that $adj_{\mathcal{U}}(e)$ is 1 if and only if e has at least one end in $\mathcal{U}$. This is known as the *max vertex cover problem*, which is an NP-hard problem (Han *et al.*, 2002). Consequently, we are only interested in approximate solutions.

An iterative greedy algorithm (Hochbaum, 1995), provides a performance guarantee of $1-e^{-1}$ of the that obtained by an optimal policy [1]. Each iteration of the algorithm starts by assigning weights to the vertices such that the weight of v is the sum of the weights of all incident edges. Then, it picks the vertex v_{max} with the maximum weight and adds it to the cover set. Finally, v_{max} and the incident edges are removed. The process is repeated in the next iterations until m vertices are obtained.

The algorithm can be decentralized by dividing the set of vertices $\mathcal{V}$ into m disjoint subsets; each agent r_i is assigned a subset $\mathcal{V}_i$. Each agent r_i computes the weights of each vertex $v \in \mathcal{V}_i$ as described above. r_i broadcasts the pair $(\hat{v}_i, \hat{w}_i)$ containing the best local candidate $\hat{v}_i$ and its corresponding weight $\hat{w}_i$. r_i waits on similar broadcast messages from other agents. The global best candidate $\hat{v}$ is determined and added to the solution set. Then, All edges incident to $\hat{v}$ are removed. If $\hat{v} \in \mathcal{V}_i$, the agent removes that vertex from $\mathcal{V}_i$. Thus the distributed greedy algorithm entails sending $O(m)$ messages and guarantees a solution that is at least $1 - e^{-1} \approx 63\%$ of the optimal solution. Figure 3 shows a pseudo-code representation of the distributed greedy algorithm. A clock symbol "⏱" placed next to a step number indicates that a timeout is used to avoid waiting indefinitely for a message from a failed agent.

```
    0. Initialize $L = \phi$.
    1. For each local vertex $v \in \mathcal{V}_i$, compute vertex
       weight $w(v) = \sum_{adj_{\{v\}}(e)=1} w(e)$.
    2. Pick the vertex $\hat{v}_i$ with the maximum weight
       $\hat{w}_i = w(\hat{v}_i)$ (if exists).
    3. Broadcast the pair $(\hat{v}_i, \hat{w}_i)$ (if exists, otherwise
       send a NULL message).
⏱  4. Wait on a message from each agent $r_j$ containing
       the pair $(\hat{v}_j, \hat{w}_j)$.
    5. Choose $(\hat{v}, \hat{w})$ as the pair with the highest $\hat{w}$.
    6. Remove each edge $e$ (if exists) that satisfies
       both $adj_{\{\hat{v}\}}(e) = 1$ and $adj_{\{\mathcal{V}_i\}}(e) = 1$.
    7. Add $\hat{v}$ to the solution set $L$.
    8. If $\hat{v} \in \mathcal{V}_i$, remove $v$ from $\mathcal{V}_i$.
    9. If $|L| = m$, end.
   10. Goto Step 1.
```

Fig. 3. Pseudo-code representation of the distributed greedy algorithm.

3. VIDEO TARGET TRACKING

Each agent is equipped with video cameras that deliver videos of street segments it is currently observing. Those videos need to be processed and important targets are to be tracked. Particle filters (PFs) have recently been successful in tracking mobile targets in video (Perez *et al.*, 2004), (Perez *et al.*, 2002). Information such as size, color, and motion characteristics of targets, is known a priori. This information is used to initialize the PF in the first few frames. Thereafter, using a dynamic model, the state of each particle is updated as the video progresses. At each step, color and motion data is collected for each particle to determine which particles have a high probability of correctly tracking the target. On the next iteration, particles are drawn according to this probability. Thus, successful particles "survive", while the other particles "die."

3.1 Particle Filtering in a Bayesian Framework

The objective of Bayesian state estimation is to estimate the posterior pdf $p(\mathbf{x}_k | \mathbf{z}_{1:k})$ of a state $\mathbf{x}_k$ based on all previous measurements $\mathbf{z}_{1:k}$. This pdf can be determined in two steps: prediction and update. In the prediction step, the state update model is used to determine the prior pdf $p(\mathbf{x}_k | \mathbf{x}_{k-1})$. If a first-order Markov model is assumed, then the prior is given as

$$p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1})\, d\mathbf{x}_{k-1}. \quad (6)$$

[1] e here refers to the natural logarithm base (≈ 2.7183).

After the measurement, $\mathbf{z}_k$ is made, the prior is updated using Bayes' rule:

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = \frac{p\left(\mathbf{x}_k|\mathbf{z}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)}{p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right)} \tag{7}$$

Generally, the above equations cannot be determined analytically. A well-known exception is the Kalman filter, which can handle linear systems with Gaussian distributions. PFs are one way to estimate the above equations for nonlinear, non-Gaussian processes.

A PF iteratively approximates the posterior pdf as a set

$$S_k = \left\{ \langle \mathbf{x}_k^{(i)}, w_k^{(i)} \rangle | i = 1, ..., n \right\}, \tag{8}$$

where $\mathbf{x}_k^{(i)}$ is a point in the state space, and $w_k^{(i)}$ is the weight associated with this point. $w_k^{(i)}$'s are non-negative and sum to unity. At each iteration, the particles are updated using the system dynamics and sampling from $p\left(\mathbf{x}_k^{(i)}, \mathbf{x}_{k-1}^{(i)}\right)$.

Measurements are then taken at each particle and the weights are updated [2] using

$$w_k^{(i)} \propto w_{k-1}^{(i)} p\left(\mathbf{z}_k | \mathbf{x}_k^{(i)}\right). \tag{9}$$

The posterior pdf estimated using particle filtering converges to the actual pdf as the number of particles increases (Arulampalam and Maskell, 2002).

3.2 Tracking using Particle Filtering

Initialization In much of the previous work (Perez *et al.*, 2004), (Perez *et al.*, 2002), the filter is initialized by manually selecting a region of interest in one of the early frames of a video sequence. Since the PF is used on an autonomous vehicle, an automatic initialization routine is developed, where the prior knowledge of the targets' color, motion, and size, is used. First, two adjacent frames are selected and sent through a brick wall chromatic filter such that only pixels that are near the color of the target are kept. The filtered frames are subtracted pixel by pixel to gain motion information. The remaining regions in this difference image are grouped. Particles are assigned to the regions based on size; regions with a sizes close to the target size are assigned more particles than others.

Fine Motion Model A motion model similar to (Perez *et al.*, 2004) is used. This model describes the target motion at a finer scale than the one used for the agent placement. Although, the model assumes that targets move smoothly from one frame to the next, a term is included to allow the model to "lean" towards motion regions. This accounts for instances where the filter loses lock. The model is described as

$$\begin{aligned} p\left(\mathbf{x}_k\right) = {} & \beta_{RW} N\left(\mathbf{x}_{k-1}, \sigma^2\right) \\ & + \left(1 - \beta_{RW}\right) N\left(\mathbf{y}_k, \sigma_y^2\right), \end{aligned} \tag{10}$$

where $0 < \beta_{RW} < 1$ is an adaptable term to allow the random-walk portion to become a contributor to the model, $\mathbf{x}_k$ is the state estimate at time k, $N\left(\mathbf{x}, \sigma^2\right)$ is a Gaussian distribution with mean $\mathbf{x}$ and variance σ^2, $\mathbf{y}_k$ is a region where motion is detected, and σ_y^2 is the associated variance.

Measurements After each particle is updated using Equation 10, color and motion measurements are taken. The color data is collected in a manner similar to (Perez *et al.*, 2002). A histogram is populated using only pixels whose saturation and value (in the HSV sense) are above certain thresholds. The histogram is compared to a reference histogram, which is created using the prior color information. The comparison is done by multiplying the reference histogram by the particle histogram bin by bin. A distance measurement $D_C\left(\mathbf{h}, \mathbf{h}_r\right)$ is then defined as a function of the product

$$D_C\left(\mathbf{h}, \mathbf{h}_r\right) = \left(1 - \sum_{i=1}^{B} \sqrt{h_i h_{i,r}}\right)^{\frac{1}{2}}. \tag{11}$$

$D_C\left(\mathbf{h}, \mathbf{h}_r\right)$ is then used to determine the color model

$$p\left(\mathbf{y}^C | \mathbf{x}\right) = \exp\left(-\frac{D_C\left(\mathbf{h}, \mathbf{h}_r\right)}{\sigma_C^2}\right), \tag{12}$$

where σ_C^2 is an adaptable term that determines how "spread" $p\left(\mathbf{y}^C|\mathbf{x}\right)$ is.

Motion data is collected at each particle by subtracting the pixel values (in the HSV sense) minus the corresponding values from the previous frame. The difference is summed and normalized by the area. This sum is finally used to determine a motion distance

$$D_M\left(\mathbf{x}\right) = \left(1 - \frac{M}{A}\right)^{\frac{1}{2}}, \tag{13}$$

where M is the sum of the difference pixels within the particle, and A is the area of the particle. $D_M\left(\mathbf{x}\right)$ is used to determine the motion measurement model

$$p\left(\mathbf{y}^M | \mathbf{x}\right) \propto \exp\left(-\frac{D_M\left(\mathbf{x}\right)}{\sigma_M^2}\right), \tag{14}$$

where σ_M^2 is analogous to σ_C^2.

[2] If the particles are resampled at each iteration, the previous weights may be neglected.

It is assumed that the color measurements are independent from their motion counterparts. Therefore, the total measurement model may be factored as

$$p(\mathbf{y}|\mathbf{x}) = p\left(\mathbf{y}^C|\mathbf{x}\right)^{\lambda_C} p\left(\mathbf{y}^M|\mathbf{x}\right)^{\lambda_M}, \qquad (15)$$

where $\lambda_{C,M}$ are adaptable terms that are adjusted throughout the filtering process.

Adaptation The iterative nature of the PF allows for various parameters to be updated throughout the filtering process. (Kwok *et al.*, 2003) and (Fox, 2003) have shown that the number of particles may be updated during the filtering process in order to increase efficiency. In this case, the number of particles as well as model and measurement parameters are automatically adjusted. For example, when a target is in a cluttered environment, it is advantageous for the motion measurements to make a larger contribution to the measurement model and to allow the state update model to lean more towards regions of high motion. The adaptation is determined by examining the goodness of the"lock" that the filter had on the target, which can be measured using (1) the weights that would be obtained if only color data was collected and (2) the spatial dispersion of the best particles. In a frame with a good lock, the standard deviation of the color weights and the maximum color weight are both higher than in frames with a poor lock.

Confidence Level Determination In order to use the PF as part of a decision support tool, it is essential to know how well the targets are being tracked. This can be accomplished using a neural network (NN), which takes data from the filter such as the maximum particle weight, standard deviation of particle weight, the spatial dispersion of the best particles, and the number of particles, and outputs a confidence level as a discrete value. In our case, 4 confidence levels are used. A confidence level of 1 corresponds to having a poor lock on the target where none of the top ten particles are on the target. A confidence level of four corresponds to having a strong lock on the target where all of the ten best particles are on the target. The NN was trained off-line using data collected while running a PF and manually assigning confidence levels.

4. PERFORMANCE EVALUATION

The integrated approach is evaluated on an urban zone whose street map is similar to the one shown in Figure 2. A first-order Markov chain is defined to determine how targets move from a street segment to another within the urban zone. As an example of how a single target is tracked on a street segment, a PF is used to track a soldier as he manoeuvered on a street segment form an urban environment [3]. Frames are grabbed from a movie at a rate of 30 Hz. The results of the integrated approach are shown in Figure 4

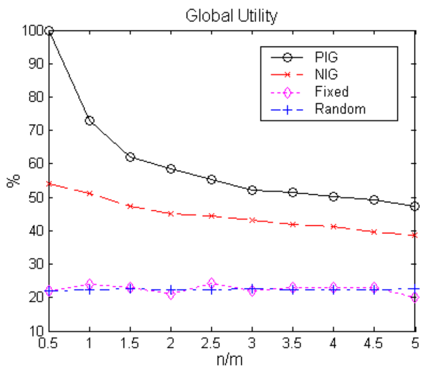
Figure 4(a) shows the simulation results of applying different algorithms to the scenario described above. In addition to the predictive greedy algorithm introduced above, a random dynamic policy and a fixed policy are tested. A nonpredictive version of the greedy algorithm is also tested, where the weight of an edge is computed simply by adding the utility values of the targets without predicting future target locations. It is clear that the predictive iterative greedy algorithm exhibits the best performance among all of the algorithms under evaluation. The conclusions to be drawn from the figure confirm two intuitions. First, carefully designed policies, such as the distributed greedy algorithms, significantly outperform simple random and fixed policies. Second, a nonpredictive policy, where agents base their placement decisions on current target locations only, is outperformed by a predictive policy.

Figure 4(b) shows the initialization of the PF given the prior knowledge about the soldier mentioned above. Sixty particles (shown as boxes) are assigned, and 51 of them are placed on the target. It is noteworthy that the filter is initialized properly even when the target is camouflaged. Figure 4(c) shows 6 sample frames of tracking the soldier. The box represents the weighted average of the ten best particles. The set of "lights" in the upper left corner of each frame are used to indicate the output of the NN. If the lowest "light" is "illuminated," the NN has output the lowest confidence level. If the second lowest is illuminated, the NN has output the second lowest confidence level. If the middle two are illuminated, the NN has output the second highest confidence level. If the top three are illuminated, the NN has output the highest confidence level.

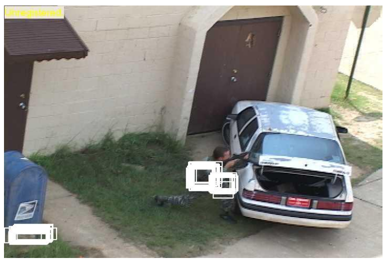
5. CONCLUSION

This paper introduces an integrated approach to reconnaissance and surveillance in urban environments. The approach relies on a multi-level target motion model and entails algorithms for autonomous dynamic UAV placement and visual target tracking. Algorithms are developed and evaluated using urban warfare scenarios. Experimental results reveal the effectiveness of the approach and its applicability to real-life scenarios. Therefore, this paper contributes (1) an integrated approach to UAV placement and target tracking

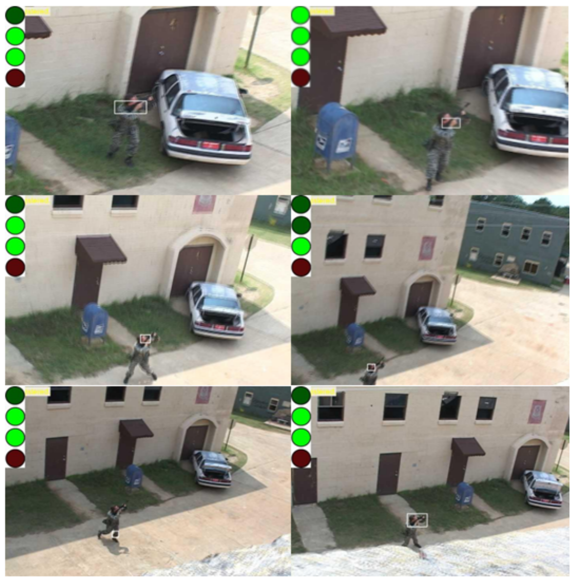
[3] McKenna Military Operations in Urban Training (MOUT) site at Fort Benning.



(a) Performance of different agent placement policies as the target-to-agent ratio increases. (PIG: Predictive iterative greedy, NIG: Nonpredictive iterative greedy.)



(b) Automatic particle initialization.



(c) Tracking a single target in an urban environment.

Fig. 4. Results of applying the integrated approach in an urban environment.

for urban reconnaissance and surveillance, (2) a distributed UAV placement algorithm for target tracking applications and (3) a particle filter-based target tracking algorithm.

ACKNOWLEDGEMENT

This work has been partially funded by DARPA under the Software-Enabled Control program and NSF under grant number CCR-0209179. We are gratefully acknowledging the support from these two organizations.

REFERENCES

Arulampalam, A. and S. Maskell (2002). A tutorial on particle filters for online nonlinear/ non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing* **50**, 174–188.

Batalin, M. and G. Sukhatme (2002). Spreading out: a local approach to multi-robot coverage. In: *Proc. of the Sixth International Symposium on Distributed Autonomous Robotic Systems*. Fukuoka, Japan. pp. 373–382.

Berchtold, A. and A. Raftery (2002). The mixture transition distribution model for high-order Markov chains and non-gaussian time series. *Statistical Science* **17**(3), 328–356.

Fox, D. (2003). Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research* pp. 985–1002.

Han, Q., Y. Ye, H. Zhang and J. Zhang (2002). On approximation of max-vertex-cover. *European Journal of Operational Research* (2), 207–220.

Hegazy, T. and G. Vachtsevanos (2004). Dynamic agent deployment for tracking moving targets. In: *Proc. of the Twelfth Mediterranean Conference on Control and Automation*. Kusadasi, Aydin, Turkey.

Hochbaum, D. (1995). *Approximate Algorithms for NP-Hard Problems*. PWS Publishing Company.

Kwok, C., D. Fox and M. Meila (2003). Adaptive real-time particle fitlers for robot locatlization. In: *Proc. of the IEEE International Conference on Robotices & Automation*. pp. 2836–2841.

Parker, L. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots* **12**(3), 231–255.

Perez, P., C. Hue and J. Vermaak (2002). Color-based probabilistic tracking. In: *Proc. of the European Conference on Computer Vision*. pp. 134–149.

Perez, P., J. Vermaak and A. Blake (2004). Data fusion for visual tracking with particles. In: *Proc. of the IEEE*. Vol. 92. pp. 495–513.

Werger, B. and Maja J. Mataric (2001). From insect to internet: Situated control for networked robot teams. *Annals of Mathematics and Artificial Intelligence* **31**(1-4), 173–197.