An Integrated Particle Filter & Potential Field Method for Cooperative Robot Target Tracking

Roozbeh Mottaghi
School of Engineering Science
Simon Fraser University
rmottagh@sfu.ca

Richard Vaughan
School of Computing Science
Simon Fraser University
vaughan@sfu.ca

***Abstract*— A fundamental challenge for robotic target-tracking systems is to cope with cases in which the target is not seen for long periods of time. An additional challenge in multiple-robot systems is to coordinate robot activity to best track targets with limited visibility. We describe a novel technique that combines a particle filter target model with a potential field robot controller. Robots are attracted to points sampled from the particle cloud that models the probability distribution over the target's position, subject to environmental constraints. We show how this method can be used as a coordination strategy whereby a team of robots cooperatively minimize the uncertainty in the pose of a tracked target. Simulation results are presented.**

I. Introduction

Probabilistic approaches to automatic target tracking have recently become popular due to their robustness in tracking in the presence of uncertainty. In particular, multi-modal representations have proved successful when targets are partially or briefly occluded.

In addition to extensive work on target tracking in the computer vision community, several authors have described tracking systems using autonomous mobile robots. As an example of a recent work Schulz et al. [1] have introduced sample-based joint probability data association filters to track multiple moving objects. Montemerlo et al. [2] present a probabilistic algorithm called the conditional particle filter to track a large distribution of people locations conditioned upon the robot poses. These approaches are based on the decisions of a single robot which results in non-robustness against the failure of that robot. Target tracking can be improved by deploying multiple robots but a strategy is needed for coordination of the robots to improve the efficiency of tracking. Jung and Sukhatme [3] have proposed an example cooperative system. Their approach, in common with the other approaches mentioned above, work well when the target lies in the sensors' field of view or it has a short-term occlusion but they do not address the long-term occlusions which cause large uncertainty for the tracker.

The formal class of pursuit-evasion problems guarantees that even in the worst cases in which the evaders move arbitrarily fast, any evader would be found by a group of pursuers. Gerkey et al. [4] recently introduced a new class of searcher, the ϕ-searcher where each pursuer has a ϕ radian field of view.

A practical deficiency of the known solutions to the pursuit-evader problem is that they are highly computational intensive and do not scale well in application to multiple-robot systems. For example, in Gerkey's approach, the joint information and action space grows exponentially in the number of searchers. A further limitation is that these methods do not address the continuous tracking of intruders once they are found.

We propose a new tracking and coordination technique which requires a minimal amount of communication among the agents to coordinate multiple robots in order to minimize the uncertainty about the location of a moving intruder. As mentioned above, existing tracking methods (excluding the pursuit-evasion methods) focus on combining sensor measurements to track the object of interest. The problem arises when the object of interest goes out of the field of view for a long period. Our method simultaneously addresses this problem and the problem of coordinating multiple robot trackers.

A. Task and Approach

Consider the task of a mobile robot tracking a person through a building. Fig. 1 shows the situation where a robot has followed its target down a corridor to a T-junction, and the target has left the robot's sensor field of view. Assume that the robot could not detect whether the person moved to the right or the left. The robot has a probabilistic model of the target's future movements, and the two modes of the probability distribution over target location can be seen in the split particle cloud. If a particle falls within the sensor's field of view, but no target is detected, the particle can be eliminated. Considering this model, we can state a simple rule that maximizes the probability of observing the target: the robot must visit the location of all particles with its sensors.

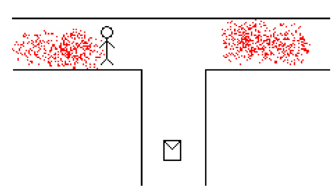


Fig. 1. A robot (bottom) tracking a person (stick figure) who has disappeared from view. The point-cloud represents a set of hypotheses about the person's current position, generated by a probabilistic model of his movements.

The task of our robot controller therefore is to minimize the uncertainty by maximizing the number of visited particles. This approach was taken by Rosencrantz et al. [5] to locate the opponents in a laser tag game in which the opponents might be under pervasive occlusions. However, their work mostly addressed the improvement of the tracker for multiple opponent tracking, rather than coordination of multiple trackers.

In this paper, a particle filtering method has been implemented to represent arbitrary multi-modal densities for the location of the intruder. Then we apply a potential field method on top of the particle filtering for coordinating multiple agents to reduce the uncertainty in the environment. Each agent decreases the uncertainty in target position estimate by sweeping as many particles as it can. Coordination between agents is achieved by each robot selecting a subset of the particles to observe.

Section II presents an outline of the probabilistic tracking method, robot mapping and localization. Section III introduces a novel technique where a particle cloud and map are combined to create a potential field robot controller for a single robot. In section IV, we present cooperative action selection and optimization strategies for searching the environment by multiple robots. An experimental section then compares the performance of a pair of robots tracking a target without coordination, and with two alternative coordination methods.

II. Probabilistic Tracking

A. Prerequisites: Localization and Mapping

We assume a map of the robot's workspace is available. Such a map of can provided *a priori*, or acquired automatically online or offline using tools such as those described by Thrun [6]. For the experiments in this paper, we supply *a priori* maps. Also, the robots need an estimate of their location and orientation. Using the map, odometry and laser scan data, good pose estimates are obtained using Monte Carlo localization [7], where the belief about the position of the robot at the current time step, $Bel(x_t)$, can be estimated by recursive update of the following equation:

$$Bel(x_t) = \eta p(z_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}) Bel(x_{t-1})\, dx_{t-1}. \tag{1}$$

where, $p(z_t|x_t)$ is the sensor model and $p(x_t|x_{t-1}, a_{t-1})$ is the next state density or motion model and a_{t-1} is the action performed in the last time step. The separation of mapping stage from localization can somewhat help us to reduce the computational burden of the whole system.

B. Particle Filter Tracking Method

Since our target of interest is moving autonomously, and may be invisible to our sensors for extended periods, our probabilistic estimate of its position may have a multi-modal density. Thus we use particle filtering to track it. For instance, such a multi-modal distribution is caused when the particles arrive at a 3-way junction of corridors as shown in Fig. 1. According to a pre-defined motion model of the target, combined with the map, the pose-estimate particles spread into the available space. The state that we want to estimate consists of the location and orientation of the object. So our state vector has the form $x_t = [x, y, \theta]$ where x and y are 2D Cartesian coordinates of the object on the map and θ represents its orientation.

In the cases when we have an observation of the object, a probability distribution over the state space is found according to the measurement, $p(x_t|z_1, z_2, ..., z_t)$, that is the probability that the state at time t is equal to x_t provided that the measurements from time 1 up to time t are equal to z_1,z_2,...,z_t respectively. Using Bayes' rule, $p(x_t|z_1, z_2, ..., z_t)$ is computed as follows:

$$p(x_t|z_1, ..., z_{t-1}, z_t) = \frac{p(x_t|z_1, ..., z_{t-1}) p(z_t|x_t, z_1, ..., z_{t-1})}{p(z_t|z_1, ..., z_{t-1})} \tag{2}$$

Since the measurement at time t is independent of the previous measurements, according to the above rules $p(z_t|x_t, z_1, ..., z_{t-1}) = p(z_t|x_t)$. Also $p(z_t|z_1, ..., z_{t-1})$ is a constant. Therefore:

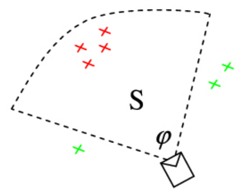
$$p(x_t|z_1, ..., z_t) = kp(z_t|x_t)p(x_t|z_1, ..., z_{t-1}) \tag{3}$$

We can compute $p(x_t|z_1, ..., z_{t-1})$ by applying the dynamic model of the object motion to $p(x_{t-1}|z_1, ..., z_{t-1})$ which is known from the previous time step. The dynamic is a known motion model of the object and it is approximated before the start of tracking and it relates the state vector at current time step to that of previous time step and it depends on the intrinsic properties of the object and the environment in which the object moves. The dynamic model can be defined as:

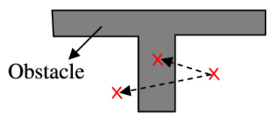
$$x_t = f(x_{t-1}) + stochastic\ part \tag{4}$$

where the stochastic part is a vector of independent standard normal variables which are scaled by a factor that is determined according to the noise. f also can be any function which relates the current state of the samples to the previous state. Because the movement of the object is unpredictable from the point of view of the tracker, we add a stochastic part to the dynamic model to add unpredictability to the deterministic model. For example, the intruder can stop or move backward instead of moving forward which is forced by the deterministic model. The next step would be the reweighting of the samples to find the probability distribution of the object over the state space. Three cases have been considered for the reweighting step:

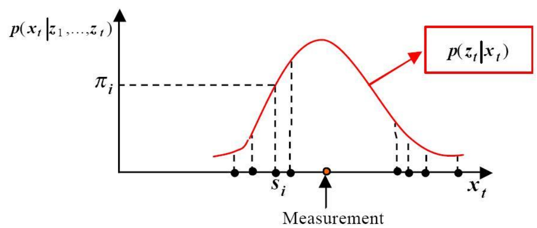
1) We define the area S which is a circular segment as the sensor visibility of the agents. This area is centered at the robot position and its central angle, φ, is in the range $\theta - \frac{F}{2}$ to $\theta + \frac{F}{2}$ where θ is the robot orientation and F is the field-of-view angle subtended by the sensor. Finally, the radius of the circular segment shows the sensor range. If the i^{th} sample $s_i \in S$ and the line that connects the sample to the robot does not intersect an obstacle, w_i the weight of that sample will be zero. This case is shown in Fig. 2a. Thus the robot deletes



(a) The robot assigns low weights to the visited particles.



(b) The particles which cross an obstacle will be given low weight.



(c) Reweighing the particles according to the sensor model.

Fig. 2. Weighing schemes.

the particles that it now knows do not correspond to the real position of the target.

2) If $\overline{s_i^t s_i^{t-1}} \cap C_{obs} \neq \emptyset$, w_i the weight of i^{th} sample would be zero. $\overline{s_i^t s_i^{t-1}}$ is the line segment that connects the position of sample i at the current time step to its position in the previous time step and C_{obs} is the space of all of the obstacles present in the map. Intuitively, it means if the particles go inside an obstacle or through a wall, their weight becomes zero. Fig. 2b shows this case. This models the constraints on the target that it can only move through free space.
3) If we have an observation of the object, the Factored Sampling method [8] is used to find the new weight of the samples. If the number of samples goes to infinity the distribution of samples from $p(z_t|x_t)p(x_t|z_1, ..., z_{t-1})$ tends to be that of $p(x_t|z_1, ..., z_{t-1}, z_t)$. A reasonable assumption for the sensor model, $p(z_t|x_t)$, is to be a Gaussian function. Fig. 2c shows a one dimensional update model.

If none of the above cases happened, the sample keeps its previous weight or we can assign an equal weight to all of the samples. We describe below how these particles and their weights are used to track the object. In the traditional particle filter tracking system, some clustering method is used to decide what the current 'actual' estimate is. Usually the particle with the mean or median of the weights is considered

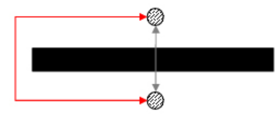


Fig. 3. The grey arrow shows the Euclidean distance between two points which is not useful for calculating the attraction force because of the presence of the obstacle.

to be the target. We avoid this clustering step, and thus avoid the artifacts it can introduce. As we may not have an observation during the tracking, we try to maximize the number of visible particles while simultaneously optimizing the joint motion of the robots.

III. Tracking using potential fields

Our goal is to minimize the uncertainty by maximizing the number of visited particles. Also, the likelihood of finding the target is found is maximized if all the particles are swept off. A problem with existing methods such as POMDP (Partially Observable Markov Decision Processes) which seems well-suited to solve this kind of problem, is that their computational complexity or memory needs grow exponentially with the number of robots. The problem also can be simplified to find a polygon in the environment to cover as many particles as possible and move the robot with a limited field of view to that area to cover a large number of particles. But there is no polynomial algorithm for performing these calculations.

We implement a potential field method for doing this task which is $O(N_p n)$ in the worst case where n is the number of cells if we represent the map by grid cells and N_p is the number particles used in the tracking algorithm. For speed, the number of particles used for the calculation of the forces can be decreased by selecting at random a subset of the particles. Since the particle filtering results in producing more particles in the high probability areas, the chance of choosing particles in those areas would be higher and the distribution of the particles is approximately the same. So we perform the update stage for the whole set of particles but calculate the forces based on the randomly chosen particle subset. In the following subsection we explain the method for finding the distances on a map (instead of Euclidean distance) and after that, the application of this method for tracking is described.

A. Finding map distances

For maximizing the number of visited particles, the idea is that each particle exerts a force on the robot to attract it. So the greater number of particles in an area, the larger the force imposed on the robot. The magnitude of this force is inversely proportional to the distance from the particle to the robot. This means that the robot tends to sweep the nearest particles first. But when the robot's mobility is limited by obstacles, as shown in Fig. 3, the Euclidean distance from particle to robot does not indicate how quickly the robot can reach the particle. Instead we must calculate the shortest traversible path using the map.

Shortest-traversible-path calculations are done using a simple occupancy-grid flood fill method, though any equivalent method could be substituted. The distance algorithm outputs a value which is assigned to each grid cell and shows the distance of that cell from the cell where the robot is located. The flood-fill works as follows: First, we assign a zero value to the cell in which the robot is located and an infinite number to the obstacles. Then, we pick one of the unoccupied four-connected cells around the robot cell (top, right, bottom and left cell; the order is important) and increment its value by one and put that cell in a queue and pick another neighbour cell until there is no cell around the current cell without an assigned value. After that, we pop the first cell in the queue and do the same thing for its surrounding cells. This algorithm is continued until there is no cell in the queue. This method returns the minimum map distance of a point to the current position of the robot and its time complexity is $O(n)$ if it is implemented by a queue where n is the number of cells on the map. Fig. 4 shows an output of this method for measuring the map distance of a cell of the map.

Thus we find the map distance of each particle from the robot as required for the calculation of the forces exerted by the particles. These calculations are explained in detail in the next subsection. For simplicity, from now on we represent the map distance of a cell, which is located at row i and column j, from the robot cell by $\Delta(i, j)$.

B. Computation of potential forces

The navigation of our robots is based on the total force which is exerted on the robots by randomly selected particles. That means at each time step, we apply the normalized total force to the robot to find its next target position. An underlying robot controller based on the extended Vector Field Histogram (VFH+) [9] performs the task of avoiding local obstacles while moving according to the potential field. To compute the total potential acting on a robot, we find the force vector for each particle. Then, we sum the vectors to find the magnitude and direction of the resultant total force.

To find the approximate direction of the particle force, we start from the cell where the particle is located and we check its surrounding cells, the cell with the minimum value will be selected. We continue performing the same procedure for the minimum-value cell until we reach a certain distance from the robot cell (this distance is approximated by the circle in Fig. 4). The direction of the force is approximated by the direction of the vector from the robot to the cell that is reached through the above procedure. The reason that we do not use directly the vector from the robot to the particle, is that the vector may intersect obstacles that block the robot's way. Fig. 4 shows an example of finding the force direction. The dashed line shows one of the paths from the red cell to the robot cell and the vector from the robot to the cell with value 3 can be considered as the force direction.

If m_i and n_i are the row and column index of particle i in the map grid, the magnitude of the force exerted by that

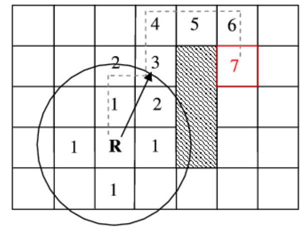
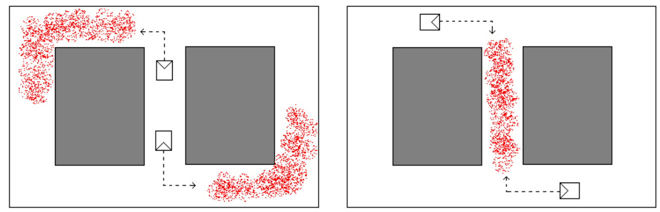


Fig. 4. The vector shows the direction of the force which is exerted from a particle located in the red cell.



(a) (b)

Fig. 5. Simple cases of minimizing the uncertainty by two robots.

particle, F_i, is calculated by the following Gaussian model:

$$F_i = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{\Delta^2(m_i,n_i)}{\sigma^2}} \tag{5}$$

where the σ is assumed to be a constant or it can be determined according to the data. This equation means that the closer particles exert a larger force and the first priority of the robot is to sweep the nearest particles. Nevertheless, if the number of particles is large in an area the robot will be attracted to that area neglecting the nearest particles. The magnitude and direction of the attractive force is determined by the vector summation of the forces from all of the particles which were selected randomly from the whole set of particles. The robot will be driven around according to the direction of this force.

The main criticism of potential field methods in general is that rapidly changing local optima can cause an oscillatory behaviour in the navigation of the robot. However, because of the random nature of the particle filtering method and clearing of the particles during the navigation, the symmetry breaks and we have not observed adverse oscillations in the robot movements.

This method is very easily extended to rationally coordinate multiple tracking agents, as described in the next section.

IV. Coordination Strategies

The expected uncertainty minimization and tracking performance of the system can be improved by simply adding more robots, but to maximize performance, the robots' actions should be coordinated in some way. In this section we describe how multiple robots cooperate to perform the assigned task according to the potential fields which has been formed by the particles. The assumption for the coordination method is that each robot has an estimate of the location of the other robots. Each robot can send its global position information to the teammates through communication or it can localize the other robots in its coordinate frame. Both of these constraints are feasible using current methods. In our simulations, we use communication among the robots. The communication can be direct communication between two robots or in the case of limited communication range, a robot can get the location information of one robot through communication with a third robot.

As stated before, we want to minimize the uncertainty by maximizing the number of visited particles. So our goal is to cover an area that is occupied by larger number of particles and to prevent the particles from further spreading. Two simple cases are shown in Fig. 5. The first figure (a) shows the case where we have two high density regions that means the chance of finding the intruder is high in those two regions. The best action to minimize the uncertainty is that one robot goes toward one cloud of particles and the other robot goes toward the other cloud. The next figure (b) shows the case that there is one high density area. The best action to shrink the particles' area and prevent it from further growing is that the robots approach the covered area from different directions. Our coordination method tries to achieve the above goals while minimizing the length of path that a robot navigates.

For the coordination of the motion of the robots, we compute the cooperative forces which are exerted by the set of particles to each robot. These forces will determine the navigation direction of the robots. First, we assign a value to each particle according to density and distance of the robots. The more negative the value, more desirable for the agent to go toward that particle. This value which is represented by $V_{n,j}$ for particle n relative to agent j is determined by:

$$V_{n,j} = \sum_{i=1, i\neq j}^{N} \begin{cases} -w_i F_{nj} & \Delta_i > \Delta_j; \\ w_i F_{ni} & \Delta_i \leq \Delta_j. \end{cases} \tag{6}$$

where F_{nj} and F_{ni} are the forces that particle n imposes on agent j and agent i, respectively and are computed according to Eq 5. N is the number of agents used for tracking and w_i is a priority factor which is used to assign higher priorities to some agents. Also, Δ_i and Δ_j are the map distance of the n^{th} particle from agent i and agent j. Intuitively, this equation means that the parameter V will be more positive for a selected particle and a specific robot if the density of the other robots around the particle is high. That means the other robots will take care of the particles nearest to them. We normalize these values to get positive force magnitudes. The normalization is done by an exponential function again. So, the force magnitude that particle n exerts to agent j in presence of the other robots, $F_{n,j}$, is calculated as follows (note that $F_{n,j}$ is different from F_{nj} since F_{nj} is that force without the presence of the other robots):

$$F_{n,j} = e^{-(V_{n,j} - V_{min})^2} \tag{7}$$

where V_{min} is the most negative value. The direction of the force is also found by the procedure described in section III.B. Now, we find the vector sum of the forces which are exerted on one robot by the set of particles, thus:

$$F_j^{tot} = \sum_{n=1}^{N_s} \vec{F_{n,j}} \tag{8}$$

where N_s is the number of randomly selected particles. As mentioned before, for the sake of efficiency, we use a small set of particles for force calculations and only the update step (in particle filtering) is done for the whole set of particles. The direction of navigation of robot j is dependent on F_j^{tot}. In the next section, the simulation results of this coordination method are shown.

V. Experimental Results

Our experiments are done in simulation using the Player/Stage robot development and simulation system [10]. Our Stage models approximate ActivMedia Pioneer-3DX robots with SICK LMS-200 laser range finders for localization and navigation. We also use a Fiducial-finder to detect the objects of interest, modeling a feature detector on a camera or other sensor. Fig. 6 shows a run of the system on simulated robots. There are two teammates (red robots) that try to catch the intruder (the blue robot) by cooperative uncertainty minimization. The robots exhibit desirable behaviour: the searcher robots go toward the particles from two different directions. Figs. 6(a), 6(b) and 6(c) show the Stage snapshots in different time steps and the distribution of the particles on the map at the corresponding simulation time are shown in Figs. 6(d), 6(e) and 6(f).

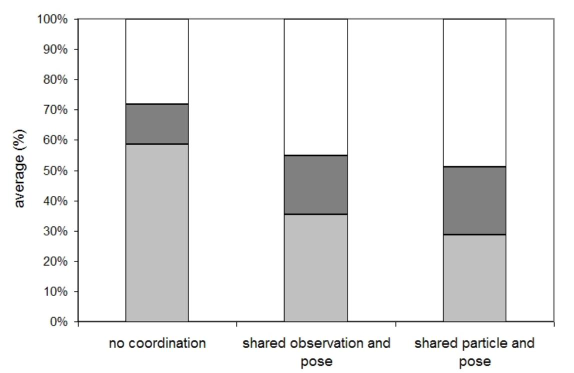
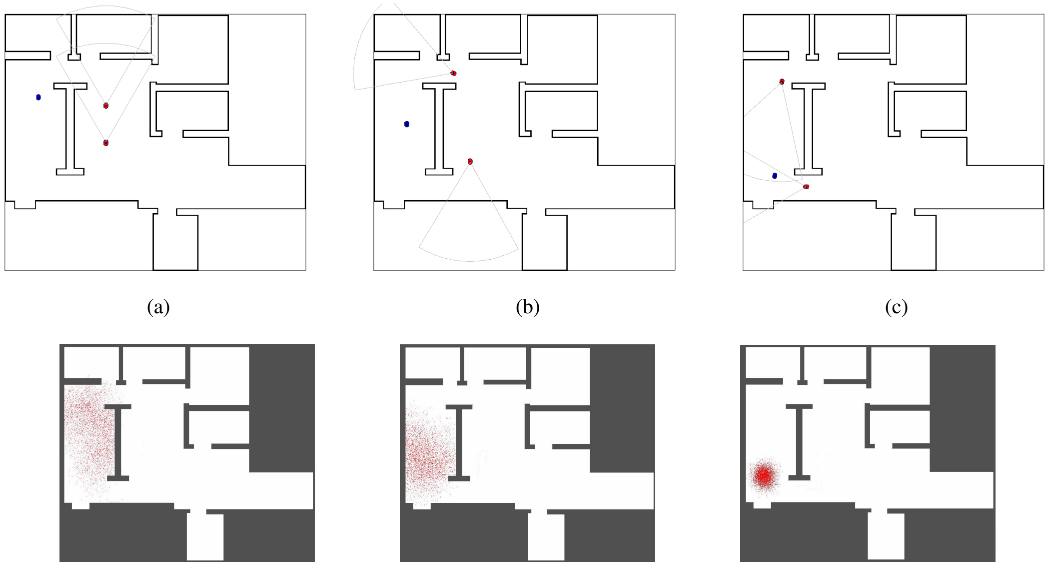


Fig. 7. No coordination, Shared observation and pose and Shared particles (from left to right) are three cases shown in this diagram. The light gray area shows the average percentage of time an agent has spent before visiting the object for the first time. The dark gray area is the average percentage of time that the robots had no observation after they see the intruder for the first time.

We performed three types of experiments in an indoor office environment whose map is represented in Fig. 6 to show the performance of the coordination methods compared to the case when the robots do not cooperate. In the first case, we tested two searcher robots to find the intruder without any coordination strategy. In the second case, the teammate robots communicate their pose and observation and in the last experiment the teammate robots share the set of randomly selected particles used in force calculations as well as the pose



(a) (b) (c)

(d) (e) (f)

Fig. 6. Simulation Results. There are two teammates (red robots) which try to catch the intruder (the blue robot). (a), (b) and (c) Stage snapshots in different simulation times (from left to right). (d), (e) and (f) The particles' distribution in the corresponding simulation time.

and observation communicated through a TCP connection. For these experiments, the range of the robot sensors is assumed to be 8 meters with an angle of view of 120 degrees while the environment dimension is $24m \times 20.5m$. The results that are shown in Fig. 7 were gathered from 10 trials of five minutes of tracking one randomly moving intruder by two searchers (the start position of the intruder was also random). The light gray area shows the average percentage of time a searcher spends before first locating the intruder. That means, how successful were the teammate robots in decreasing the uncertainty of initially uniformly distributed particles. The dark gray area is the average percentage of time that the searchers had no observation after they see the intruder for the first time. The diagram shows the total time that the robots have no observation (sum of the values of light and dark gray areas) in the shared-particle and non-shared particle case is less than that of no-coordination case, indicating that the performance is improved on average by cooperation.

VI. Conclusion and Future Work

We have described a method for a team of mobile robots to cooperatively track a moving target. This approach addresses the main limitation of previous approaches in that it actively minimizes the uncertainty caused when the target is occluded for long periods. A particle filtering method represents the multi-modal uncertainty in the estimated pose of the target. Then a potential field is generated using the location of particles directly as input - no clustering of particles is performed. The potential field guides the robots to visit as many particles as they can to reduce the uncertainty in the environment and to prevent the uncertainty area from further growing. The algorithm is extended to multiple robots by allocating subsets of particles to each robot. We used a simple nearest-robot filter to achieve this.

We are currently implementing this system on real robots tracking humans. In collaboration with vision researchers, we will use activity recognition methods (recognizing running, walking, etc.) to estimate online the parameters of a motion model for humans. Also, by using methods from the literature cited above, method can be easily extended to multi-target problems.

References

[1] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. People Tracking with a Mobile Robot Using Sample-based Joint Probabilistic Data Association Filters. *International Journal of Robotics Research (IJRR)*, 22(2), 2003.

[2] M. Montemerlo, W. Whittaker, and S. Thrun. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *IEEE International Conference on Robotics and Automation*, 2002.

[3] B. Jung and G. S. Sukhatme. A Generalized Region-based Approach for Multi-target Tracking in Outdoor Environments In *IEEE International Conference on Robotics and Automation*, pp. 2189–2195, 2004.

[4] B. P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 2004)*, pp. 20–27, San Jose, California, July 25–29, 2004.

[5] M. Rosencrantz, G. Gordon, and S. Thrun. Locating moving entities in dynamic indoor environments with teams of mobile robots. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2003.

[6] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002.

[7] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2), 2001.

[8] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal on Computer Vision*, 29(1):5–28, 1998.

[9] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceeding of the IEEE International Conference on Robotics and Automation*, pp. 1572–1577, 1998.

[10] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, G. S. Sukhatme, and M. J. Mataric. Most valuable player: A robot device server for distributed control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1226–1231, 2001.