Object Tracking in the Presence of Occlusions via a Camera Network

Ali Ozer Ercan
Department of Electrical Engineering
Stanford University
Stanford, CA USA 94305
aliercan@stanford.edu

Abbas El Gamal
Department of Electrical Engineering
Stanford University
Stanford, CA USA 94305
abbas@ee.stanford.edu

Leonidas J. Guibas
Department of Computer Science
Stanford University
Stanford, CA USA 94305
guibas@cs.stanford.edu

ABSTRACT

This paper describes a sensor network approach to tracking a single object in the presence of static and moving occluders using a network of cameras. To conserve communication bandwidth and energy, each camera first performs simple local processing to reduce each frame to a scan line. This information is then sent to a cluster head to track a point object. We assume the locations of the static occluders to be known, but only prior statistics on the positions of the moving occluders are available. A noisy perspective camera measurement model is presented, where occlusions are captured through an occlusion indicator function. An auxiliary particle filter that incorporates the occluder information is used to track the object. Using simulations, we investigate (i) the dependency of the tracker performance on the accuracy of the moving occluder priors, (ii) the tradeoff between the number of cameras and the occluder prior accuracy required to achieve a prescribed tracker performance, and (iii) the importance of having occluder priors to the tracker performance as the number of occluders increases. We generally find that computing moving occluder priors may not be worthwhile, unless it can be obtained cheaply and to a reasonable accuracy. Preliminary experimental results are provided.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Probabilistic Algorithms; I.4.9 [**Image Processing and Computer Vision**]: Applications.

General Terms

Algorithms.

Keywords

Tracking, occlusion, auxiliary particle filter, wireless sensor network, camera network, noisy perspective camera model.

1. INTRODUCTION

There is a growing need to develop low cost wireless networks of cameras with automated detection capabilities [1]. The main challenge in building such networks is the high data rate of video cameras. On the one hand sending all the data, even after performing standard compression, is very costly in transmission energy, and on the other, performing sophisticated vision processing at each node to substantially reduce transmission rate requires high processing energy. To address these challenges, a task-driven approach, in which simple local processing is performed at each node to extract the essential information needed for the network to collaboratively perform the task, has been proposed and demonstrated [2, 3].

In this paper, we adopt such a task-driven approach for tracking a single object (e.g., a suspect) in a structured environment (e.g., an airport or a mall) in the presence of static and moving occluders using a wireless camera network. Most previous work on tracking with multiple cameras has focused on tracking *all* the objects and does not deal directly with static occluders, which are often present in structured environments (see brief survey in Section 2). Tracking all the objects clearly provides a solution to our problem, but may be infeasible to implement in a wireless camera network due to its high computational cost. Instead, our approach is to track only the target object treating all other objects as occluders. We assume complete knowledge of static occluder (e.g., partitions, large pieces of furniture) locations and some prior statistics on the positions of the moving occluders (e.g., people) which are updated in time. Simple local processing whereby each image is reduced to a horizontal scan line is performed at each camera node. If the camera sees the object, it provides a measurement of its position in the scan line to the cluster head, otherwise it reports that it cannot see the object. A noisy perspective camera measurement model is assumed, where occlusions are captured through an occlusion indicator function. Given the camera measurements and the occluder position priors, an auxiliary particle filter (PF) [4] is used at the cluster head to track the object. The occluder information is incorporated into the measurement likelihood, which is used in the weighting of the particles.

Even if one wishes to track only one object treating other moving objects as occluders, a certain amount of information about the positions of the occluders may be needed to achieve high tracking accuracy. Since obtaining more accurate occluder priors would require expending more processing and/or communication energy, it is important to understand the tradeoff between the accuracy of the occluder information and that of tracking. Do we need *any* prior occluder information? If so, how much accuracy is sufficient? A goal of this paper is to investigate this important tradeoff.

We develop a measure of the moving occluder prior accuracy and use simulations to explore the dependency of the tracker performance on this measure. We also explore the tradeoff between the number of cameras used, the number of occluders present, and the amount of occluder prior information needed to achieve a prescribed tracker performance. We generally find that:

- Obtaining moving occluder prior information may not be worthwhile in practice, unless it can be obtained cheaply and to a reasonable accuracy.

- There is a tradeoff between the number of cameras used and the amount of occluder prior information needed. As more cameras are used, the accuracy of the prior information needed decreases. Having more cameras, however, means incurring higher communications and processing cost. So, in the design of a tracking system, one needs to compare the cost of deploying more cameras to that of obtaining more accurate occluder priors.

- The amount of prior occluder position information needed depends on the number of occluders present. When there are very few moving occluders, prior information does not help (because the object is not occluded most of the time). When there is a moderate number of occluders, prior information becomes more useful. However, when there are too many occluders, prior information becomes less useful (because the object becomes occluded most of the time).

It is important to note that these conclusions are based only on our simulation setting, and that additional explorations by simulation and experiments are needed to validate them.

The rest of the paper is organized as follows. A brief survey of previous work on tracking using multiple cameras is presented in the next section. In Section 3, we describe the setup of our tracking problem and introduce the camera measurement model used. The tracker is described in Section 4. Simulation and experimental results are presented in Sections 5 and 6, respectively.

2. PREVIOUS WORK

Tracking has been a popular topic in sensor network research (e.g., [5–11]). Most of this work assumes low data rate range sensors. By comparison, our work assumes cameras, which are bearing sensors and have high data rate. The most related work to ours is [10] and [11]. Pahawalatta et al. [10] use a camera network to track and classify multiple objects on the ground plane. This is done by detecting feature points on the objects and using a Kalman Filter (KF) for tracking. By comparison, we use a PF, which is more suitable for non-linear camera measurements and track only a single object treating others as occluders. Funiak et al. [11] use a Gaussian model obtained by reparametrizing the camera coordinates together with KF. This method is fully distributed and requires less computational power than PF. However, because the main goal of the system is camera calibration and not tracking, occlusions are not considered. Also, this work requires minimal overlap of the camera FOVs, which is not a requirement for our work.

Tracking has also been a very popular topic in computer vision (e.g., [12–16]). Most of the work, however, has focused on tracking objects in a single camera video sequence [12, 13]. Tracking using multiple camera video streams has also been considered [14–16]. Individual tracking is performed for each video stream and the objects appearing in the different streams are associated. More recently, there has been work on tracking multiple objects in world coordinates using multiple cameras [17–19]. Utsumi et al. [17] extract feature points on the objects and use a KF to track the objects. They perform camera selection to avoid occlusions. By comparison, in our work occlusions are treated as part of the tracker. Otsuka et al. [18] describe a double loop filter to track multiple objects, where objects can occlude each other. One of the loops is a PF that updates the states of the objects in time using the object dynamics, the likelihood of the measurements, and the occlusion hypotheses. The other loop is responsible for generating these hypotheses and testing them using the object states generated by the first loop, the measurements, and a number of geometric constraints. Although this method also performs a single object tracking in the presence of moving occluders, the hypothesis generation and testing is computationally prohibitive for a sensor network implementation. The work also does not consider static occlusions that could be present in structured environments. Dockstader et al. [19] describe a method for tracking multiple people using multiple cameras. Feature points are extracted from images locally and corrected using the 3-D estimates of the feature point positions that are fed back from the central processor to the local processor. These corrected features are sent to the central processor where a Bayesian network (BN) is employed to deduce a first estimate of the 3-D positions of these features. A KF follows the BN to maintain temporal continuity. This approach requires that each object is seen by some cameras at all times. This is not required in our approach. Also, performing motion vector computation at each node is computationally costly in a wireless sensor network.

We would like to emphasize that our work is focused on tracking a *single object* in the presence of *static and moving occluders* in a wireless sensor network setting. When there are no occluders, one could adopt a less computationally intensive approach similar to [11]. When all the objects need to be tracked simultaneously, the above mentioned methods ([18, 19]) or a filter with joint-state for all the objects [20] can be used.

3. SETUP, MODELS, AND ASSUMPTIONS

We consider the setup illustrated in Fig. 1 in which N cameras are aimed roughly horizontally around a room. Although an overhead camera would have a less occluded view than a horizontally placed one, it generally has a more limited view of the scene and may be impractical to deploy. Additionally, targets may be easier to identify in a horizontal view. The cameras are assumed to be fixed and their locations and orientations are known to some accuracy to the cluster head. The camera network's task is to track an object in the presence of static occlusions and other moving objects. We assume that the object to track to be a point object. This is reasonable because the object may be distinguished from occluders by some specific point feature. We assume there are M other moving objects, each modeled as a cylinder of diameter D. The position of each object is assumed to be the center of its cylinder. From now on, we shall refer to the object to track as the "object" and the other moving objects as "moving occluders."

We assume the positions and the shapes of the static occluders in the room to be completely known in advance. This is not unreasonable since this information can be easily provided to the network. On the other hand, only some prior statistics of the moving occluder positions are known at each time step. In Subsection 4.4, we discuss how these priors may be obtained.

As in [2], we assume that simple background subtraction is performed locally at each camera node. We assume that the camera nodes can distinguish between the object and the occluders. This can be done, for example, through feature detection, e.g., [21]. Since the horizontal position of the object in each camera's image plane is the most relevant information to 2-D tracking, the back-

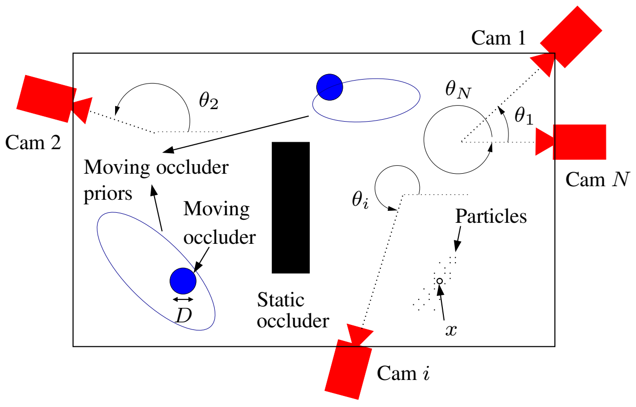


Figure 1: Illustration of the setup.

ground subtracted images are vertically summed and thresholded to obtain a "scan line" (see Fig. 2). Only the center of the object in the scan line is sent to the cluster head.

3.1 Camera Measurement Model

If a camera "sees" the object, its measurement is described by a noisy projective camera model. If the camera cannot see the object because of occlusions or limited FOV, it reports a "NaN" (using MATLAB syntax) to the cluster head. Mathematically, for camera $i = 1, \dots, N$, we define the *occlusion indicator function*

$$\eta_i \triangleq \begin{cases} 1, & \text{if camera } i \text{ sees the object} \\ 0, & \text{otherwise.} \end{cases}$$

Note that the η_i random variables are not in general independent from each other. The camera measurement model including occlusions is then defined as

$$z_i = \begin{cases} f_i \frac{h_i(x)}{d_i(x)} + v_i, & \text{if } \eta_i = 1 \\ \text{NaN}, & \text{otherwise,} \end{cases} \tag{1}$$

where x is the position of the object, f_i is the focal length of camera i, and $d_i(x)$ and $h_i(x)$ are defined through Figure 3. The random variable v_i models the read noise and the errors in the camera position and angle θ_i. (see Figure 1). Assuming that these noise sources are zero mean and uncorrelated, the variance of v_i is given by

$$\sigma_{v_i}^2 = f_i^2 \left(1 + \frac{h_i^2(x)}{d_i^2(x)}\right)^2 \sigma_\theta^2 + f_i^2 \frac{h_i^2(x) + d_i^2(x)}{d_i^4(x)} \sigma_{\text{pos}}^2 + \sigma_{\text{read}}^2, \tag{2}$$

where σ_{pos}^2 is the variance of the camera position and σ_{read}^2 is the variance of the read noise (See Appendix A for derivation of this formula). We further assume that given x, the noise from the different cameras $v_1, v_2, \dots, v_N$ are independent, identically distributed Gaussian random variables. Note that the camera nodes report only the observations $\{z_i\}$ to the cluster head, and the cluster head derives the values of the η_is from the z_is.

4. TRACKING

As the measurement model in (1) is nonlinear in the object position, using a linear filter, e.g., Kalman Filter (KF), for tracking would yield poor results. As discussed in [22], using an Extended Kalman Filter (EKF) with measurements from bearing sensors, which are similar to cameras with the aforementioned local processing, is not very successful. Although the use of an Unscented Kalman Filter (UKF) is more promising, its performance

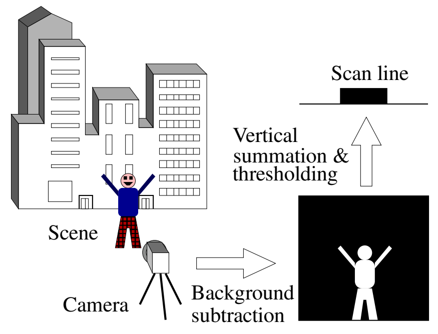


Figure 2: Local processing at each camera node.

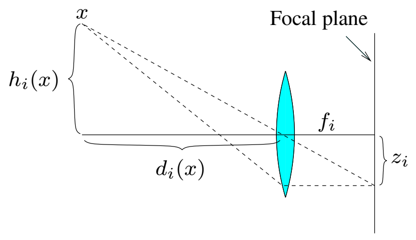


Figure 3: The camera measurement model.

degrades quickly when the static occluders and limited FOV constraints are considered. Because of the discreteness of the occlusions and FOV and the fact that UKF uses only a few points from the prior of the object state, most of these points may get discarded. We also experimented with a Maximum A-Posteriori (MAP) estimator combined with a KF, which is similar to the approach in [8]. This approach, however, failed at the optimization stage of the MAP estimator, as the feasible set is highly disconnected due to the static occluders and limited camera FOV. Given these considerations, we decided to use a particle filter (PF) tracker [4].

We denote by $u(t)$ the state of the object at time t, which includes its position $x(t)$ and other relevant information. The positions of the moving occluders $m \in \{1, \dots M\}$, $x_m(t)$ are assumed to be Gaussian with mean $\mu_m(t)$ and covariance matrix $\Sigma_m(t)$. These priors are available to the tracker. The state of the object and positions of moving occluders are assumed to be mutually independent. Note that if the objects move in groups, one can still apply the following tracker formulation by defining a "super-object" for each group and assuming that the super-objects move independently. The tracker maintains the probability density function (pdf) of the object state $u(t)$, and updates it at each time step using the new measurements. Given the measurements up to time $t-1$, $\{Y(\tau)\}_{\tau=1}^{t-1}$, the particle filter approximates the pdf of $u(t-1)$ by a set of L weighted particles as

$$f(u(t-1) | \{Y(\tau)\}_{\tau=1}^{t-1}) \approx \sum_{\ell=1}^{L} w_\ell(t-1) \delta\left(u(t-1) - u_\ell(t-1)\right),$$

where $\delta(\cdot)$ is the Dirac delta function. u_ℓ is the state of particle ℓ, (i.e., a sample of $u(t)$). At each time step, given these L weighted particles, the camera measurements $Z(t) = \{z_1(t), \dots, z_N(t)\}$ and $\eta(t) = \{\eta_1(t), \dots, \eta_N(t)\}$, the moving occluder priors $\{\mu_m(t), \Sigma_m(t)\}$, $m \in \{1, \dots, M\}$, information about the static occluder positions and the camera FOV, the tracker incorporates

```
Algorithm: ASIR
Inputs: $\{u_\ell(t-1), w_\ell(t-1)\}_{\ell=1}^L$; $\{\mu_m(t), \Sigma_m(t)\}_{m=1}^M$;
$Z(t) = \{z_1(t), \dots, z_N(t)\}$; $\eta(t) = \{\eta_1(t), \dots, \eta_N(t)\}$;
Shapes and positions of static occluders;
Camera positions and orientations ($\theta_i$, $i \in \{1, \dots, N\}$);
FOV of the cameras.
Output: $\{u_\ell(t), w_\ell(t)\}$, $\ell \in \{1, \dots, L\}$.
01.  for $\ell = 1, \dots, L$
02.      $\kappa_\ell := \mathrm{E}[u(t)|u(t-1)]$
03.      $\tilde{w}_\ell(t) \propto f(Z(t), \eta(t)|\kappa_\ell) w_\ell(t-1)$
04.  end for
05.  $\{w_\ell(t)\}_{\ell=1}^L$ = Normalize $(\{\tilde{w}_\ell(t)\}_{\ell=1}^L)$
06.  $\{\cdot, \cdot, i^\ell\}_{\ell=1}^L$ = Resample $(\{\kappa_\ell, w_\ell(t)\}_{\ell=1}^L)$
07.  for $\ell = 1, \dots, L$
08.      Draw $u_\ell(t) \sim f(u(t)|u_{i^\ell}(t-1))$
09.      $\tilde{w}_\ell(t) = \frac{f(Z(t), \eta(t)|u_\ell(t))}{f(Z(t), \eta(t)|\kappa_{i^\ell})}$
10.  end for
11.  $\{w_\ell(t)\}_{\ell=1}^L$ = Normalize$(\{\tilde{w}_\ell(t)\}_{\ell=1}^L)$
```

Figure 4: The auxiliary sampling importance resampling algorithm.

the new information obtained from the measurements at time t to update the particles (and their associated weights).

We use the auxiliary sampling importance resampling (ASIR) filter described in [4]. The outline of one step of our implementation of this filter is given in Fig. 4. In this figure, $\mathrm{E}[\cdot]$ represents the expectation operator, and the procedure "$\{w_\ell\}_{\ell=1}^L$ = Normalize ($\{\tilde{w}_\ell\}_{\ell=1}^L$)" normalizes the weights so that they sum to one. The procedure "$\{u_\ell, w_\ell, i^\ell\}_{\ell=1}^L$ = Resample ($\{u_\ell, w_\ell\}_{\ell=1}^L$)" takes L particle-weight pairs and produces L equally weighted particles ($w_\ell = 1/L$), preserving the original distribution. This amounts to particles with small initial weights being killed and the ones with high weights reproducing. The third output of the procedure (i^ℓ) refers to the index of particle ℓ's parent. The ASIR algorithm approximates the optimal importance density function $f(u(t)|u_\ell(t-1), Z(t), \eta(t))$, which is not feasible to compute in general [4].

In the following, we explain the implementation of the importance density function $f(u(t)|u_\ell(t-1))$ and the likelihood $f(Z(t), \eta(t)|u_\ell(t))$.

4.1 Importance Density Function

The particles are advanced in time by drawing a new sample $u_\ell(t)$ from the "importance density function" $f(u(t)|u_\ell(t-1))$:

$$u_\ell(t) \sim f(u(t)|u_\ell(t-1)), \ell \in \{1, \dots, L\}.$$

This is similar to the "time update" step in a KF. After all L new particles are drawn, the distribution of the state is forwarded one time step. Therefore, the dynamics of the system should be reflected as accurately as possible in the importance density function. In KF, a constant velocity assumption with a large variance on the velocity is assumed to account for direction changes. Although assuming that objects move at constant velocity is not a realistic assumption, the linearity constraint of the KF forces this choice. In the PF implementation, we do not have to choose linear dynamics. We use the more realistic "random waypoints model," where the objects choose a target and try to move toward the target with constant speed plus noise, until they reach the target. When they reach it, they choose a new target.

We implemented a modified version of this model in which the state of the particle consists of its current position $x_\ell(t)$, target $\tau_\ell(t)$, speed $s_\ell(t)$ and regime $r_\ell(t)$. Note that the time step here is 1 and thus s_ℓ represents the distance travelled in a unit time. The model is given by

$$u_\ell^T(t) = [x_\ell^T(t)\ \tau_\ell^T(t)\ s_\ell(t)\ r_\ell(t)].$$

The regime can be one of the following:

1. *Move toward target* (MTT): A particle in this regime tries to move toward its target with constant speed plus noise:

$$x_\ell(t) = x_\ell(t-1) + s_\ell(t-1) \frac{\tau_\ell(t-1) - x_\ell(t-1)}{\|\tau_\ell(t-1) - x_\ell(t-1)\|_2} + \nu(t),$$

   where $\nu(t)$ is zero mean Gaussian white noise with $\Sigma_\nu = \sigma_\nu^2 I$, I denotes the identity matrix and σ_ν is assumed to be known. The speed of the particle is also updated according to

$$s_\ell(t) = (1-\phi)s_\ell(t-1) + \phi\|x_\ell(t) - x_\ell(t-1)\|_2.$$

   Updating the speed this way smooths out the variations due to added noise. We chose $\phi = 0.7$ for our implementation. The target is left unchanged.

2. *Change Target* (CT): A particle in this regime first chooses a new target randomly (uniformly) in the room and performs an MTT step.

3. *Wait* (W): A particle in this regime does nothing.

Drawing a new particle from the importance density function involves the following. First, each particle chooses a regime according to their current position and their target. If a particle reached its target, it chooses the regime according to

$$r_\ell(t) = \begin{cases} \text{MTT}, & \text{w.p. } \beta_1, \\ \text{CT}, & \text{w.p. } \lambda_1, \\ \text{W}, & \text{w.p. } (1-\beta_1-\lambda_1). \end{cases}$$

The target is assumed "reached" when the distance to it is less than the particle's speed. If a particle does not reach its target, the probabilities β_1 and λ_1 are replaced by β_2 and λ_2, respectively. We chose $\beta_1 = 0.05, \lambda_1 = 0.9, \beta_2 = 0.9, \lambda_2 = 0.05$.

4.2 Likelihood

Updating the weights in the ASIR algorithm requires the computation of the likelihood of the measurements, $f(Z(t), \eta(t)|u_\ell(t))$. For brevity, we shall drop the time index from now on. We can use the chain rule for probabilities to decompose the likelihood and obtain

$$f(Z, \eta|u_\ell) = p(\eta|u_\ell) f(Z|\eta, u_\ell). \tag{3}$$

Now, given x_ℓ, which is part of u_ℓ, and η, $z_1, \dots, z_N$ become independent Gaussian random variables and we have

$$f(Z|\eta, u_\ell) = \prod_{i;\eta_i=1} \mathcal{N}\left\{z_i; f_i \frac{h_i(x_\ell)}{d_i(x_\ell)}, \sigma_{v_i}^2\right\},$$

where $\mathcal{N}\{r; \xi, \rho^2\}$ denotes a univariate Gaussian function of r with mean ξ and variance ρ^2, $\sigma_{v_i}^2$ is given in (2) and $d_i(x)$ and $h_i(x)$ are defined in Fig. 3.

The first term in (3), however, cannot be expressed as a product, as the occlusions are not independent given u_ℓ. This can be explained via the following simple example: Suppose 2 cameras are close to each other. Once we know that one of these cameras cannot see the object, it is more likely that the other one also cannot

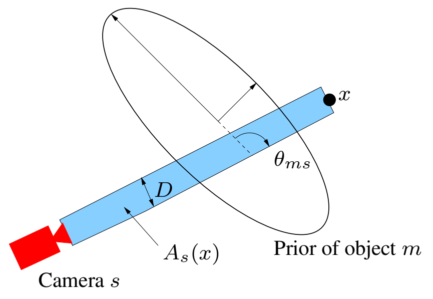


Figure 5: Computing $q_s^m(x)$.

see it. Hence, the 2 ηs are dependent given u_ℓ. Luckily, we can approximate the first term in 3 in a computationally feasible manner using recursion.

First, we ignore the static occluders and the limited FOV, and only consider the effect of the moving occluders. The effects of static occluders and limited FOV will be added in Subsection 4.3. Define the indicator functions $\eta_{s,m}$ for $s = 1, \dots, N$ and $m = 1, \dots, M$ such that $\eta_{s,m} = 1$ if object m does not occlude camera s, and 0, otherwise. Thus

$$\{\eta_s = 1\} = \bigcap_{m=1}^{M} \{\eta_{s,m} = 1\}.$$

The probability that object m occludes camera s given u is thus given by

$$\begin{aligned} \mathrm{P}\{\eta_{s,m} = 0|u\} &= \int f(x_m|u)\mathrm{P}\{\eta_{s,m} = 0|u, x_m\}\ dx_m \\ &\overset{(a)}{=} \int f(x_m)\mathrm{P}\{\eta_{s,m} = 0|x, x_m\}\ dx_m \\ &\triangleq q_s^m(x), \end{aligned}$$

where x is the position part of the state vector u and step (a) uses the facts that x_m is independent of u and $\eta_{s,m}$ is a deterministic function of x and x_m. To compute $q_s^m(x)$, refer to Figure 5. Without loss of generality, we assume that camera s is placed at the origin. We assume that the moving occluder diameter D is small compared to the occluder standard deviations. Object m occludes point x at camera s if its center is inside the rectangle $A_s(x)$. This means $\mathrm{P}\{\eta_{s,m} = 0|x, x_m\} = 1$ if $x_m \in A_s(x)$ and it is zero everywhere else:

$$\begin{aligned} q_s^m(x) &= \int_{A_s(x)} \frac{1}{2\pi\sqrt{|\Sigma_m|}} e^{-\frac{1}{2}(x_m-\mu_m)^T\Sigma_m^{-1}(x_m-\mu_m)}\ dx_m \\ &\overset{(b)}{\approx} \frac{1}{4}\left[\mathrm{erf}\left(\frac{\sqrt{\alpha}}{\|v_1'\|}\left(\frac{D}{2}-\varphi\right)\right) + \mathrm{erf}\left(\frac{\sqrt{\alpha}}{\|v_1'\|}\left(\frac{D}{2}+\varphi\right)\right)\right] \\ &\quad \left[\mathrm{erf}\left(\frac{\|x\|\|v_1\|^2 - \mu_m^T v_2}{\|v_1'\|}\right) + \mathrm{erf}\left(\frac{\mu_m^T v_2}{\|v_1'\|}\right)\right], \end{aligned} \tag{4}$$

where $v_1^T = [\cos(\theta_{ms})\ \sqrt{\alpha}\sin(\theta_{ms})]$, $v_1' = \sqrt{2}\sigma_m v_1$, $v_2^T = [\cos(\theta_{ms})\ \alpha\sin(\theta_{ms})]$, $\varphi = \mu_{my}\cos(\theta_{ms}) - \mu_{mx}\sin(\theta_{ms})$, and σ_m^2 and σ_m^2/α, $\alpha \geq 1$, are the eigenvalues of the covariance matrix Σ_m of the prior of occluder m. Step (b) follows by the assumption of small moving occluders.

To compute $p(\upeta|u)$, first consider the probability of all ηs of the cameras in subset S, given u, to be equal to 1,

$$\begin{aligned} \mathrm{P}\left(\bigcap_{s\in S}\{\eta_s = 1\}\middle| u\right) &= \mathrm{P}\left(\bigcap_{s\in S}\bigcap_{m=1}^{M}\{\eta_{s,m} = 1\}\middle| u\right) \\ &= \mathrm{P}\left(\bigcap_{m=1}^{M}\bigcap_{s\in S}\{\eta_{s,m} = 1\}\middle| u\right) \\ &\overset{(c)}{=} \prod_{m=1}^{M} \mathrm{P}\left(\bigcap_{s\in S}\{\eta_{s,m} = 1\}\middle| u\right) \\ &= \prod_{m=1}^{M}\left(1 - \mathrm{P}\left(\bigcup_{s\in S}\{\eta_{s,m} = 0\}\middle| u\right)\right) \\ &\overset{(d)}{\approx} \prod_{m=1}^{M}\left(1 - \sum_{s\in S}\mathrm{P}\{\eta_{s,m} = 0|u\}\right) \\ &= \prod_{m=1}^{M}\left(1 - \sum_{s\in S} q_s^m(x)\right) \\ &\triangleq p_S^{\mathrm{mv}}(x), \end{aligned} \tag{5}$$

where (c) follows by the assumption that the occluder positions are independent, and (d) follows from the assumption of small D and the reasonable assumption that the cameras in S are not too close so that the overlap between $A_s(x)$, $s \in S$, is negligible. Note that cameras that satisfy this condition can still be close enough, such that their FOVs overlap and ηs are dependent. The superscript "mv" signifies that "only moving occluders are taken into account."

Now we can compute $p^{\mathrm{mv}}(\upeta|u)$ using (5) and recursion as follows. Let $S = \{1, \dots, N\}$ (i.e., the set of all cameras). For any $i \in S$ such that $\eta_i = 0$, define

$$\begin{aligned} \upeta_a &= \{\eta_1, \dots, \eta_{i-1}, \eta_{i+1}, \dots, \eta_N\}. \\ \upeta_b &= \{\eta_1, \dots, \eta_{i-1}, 1, \eta_{i+1}, \dots, \eta_N\}. \end{aligned}$$

Then,

$$p^{\mathrm{mv}}(\upeta|u) = p^{\mathrm{mv}}(\upeta_a|u) - p^{\mathrm{mv}}(\upeta_b|u). \tag{6}$$

Both terms in the right-hand-side of (6) are one step closer to $p_S^{\mathrm{mv}}(u)$ (with different S), because one less element is zero in both $\upeta_a$ and $\upeta_b$ This means that any $p^{\mathrm{mv}}(\upeta|u)$ can be reduced recursively to terms consisting of $p_S^{\mathrm{mv}}(u)$, using (6). The bad news is, the computational load of this is exponential in the number of zeros in $\upeta$. However, this bottleneck is greatly alleviated by the limited FOV of the cameras as will be explained in the following subsection.

4.3 Adding Static Occluders and Limited FOV

Adding the effects of the static occluders and limited camera FOV to the procedure described above involves a geometric partitioning of the particles into bins. Each bin is assigned a set of cameras. After this partitioning, only the ηs of the assigned cameras are considered for the particles in that bin. This is explained using the example in Fig. 6. In this example, we have 2 cameras and a single static occluder. As denoted by the dashed line in the figure, we have 2 partitions. Let $\eta_1 = 0$ and $\eta_2 = \gamma_2 \in \{0, 1\}$. Let us consider a particle belonging to the upper partition, namely particle ℓ. If the object is at x_ℓ, the static occluder makes $\eta_1 = 0$, independent of where the moving occluders are. On the other hand, the static occluder and limited FOV do not occlude the second camera's view of particle ℓ. So, only Cam$_2$ is assigned to this partition, and the first term in the likelihood is given by

$$\mathrm{P}\left(\{\eta_1 = 0\} \cap \{\eta_2 = \gamma_2\}|u_\ell\right) = p^{\mathrm{mv}}(\eta_2|u_\ell).$$

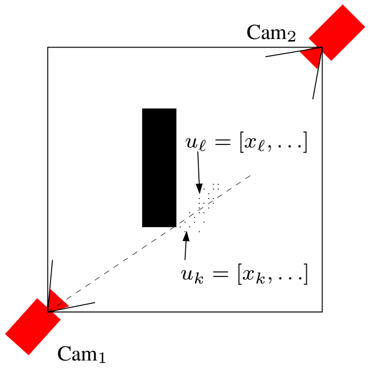


Figure 6: Geometric partitioning to add static occluders and limited FOV. If $\eta_1 = 1$, the object cannot be at x_ℓ. If $\eta_1 = 0$, only Cam$_2$ needs to be considered for computing $p(\eta|u_\ell)$. Both cameras need to be considered for computing $p(\eta|u_k)$.

Similarly,

$$\mathrm{P}\left(\{\eta_1 = 1\} \cap \{\eta_2 = \gamma_2\} | u_\ell\right) = 0$$
$$\mathrm{P}\left(\{\eta_1 = \gamma_1\} \cap \{\eta_2 = \gamma_2\} | u_k\right) = p^{\mathrm{mv}}(\eta_1, \eta_2 | u_k),$$

where the fist line follows because if the object is at x_ℓ, $\eta_1 = 0$, and the second line follows because the static occluder and limited FOV do not occlude particle k.

Note that the number of cameras assigned to a partition is not likely to be large, mainly due to the limited camera FOV. Since the number of zeros in η is at most the number of cameras assigned to a partition, the actual computational complexity of the recursion described in Subsection 4.2 is much lower than exponential. Also, because the camera placements, FOV and static occluder positions are known in advance, the room can be divided into regions beforehand, with each region assigned the cameras that can see it. The number of such regions grows at most quadratically in the number of cameras. During tracking, the particles can be easily divided into partitions depending on which pre-computed region each particle is.

We mentioned in Section 3 that the camera nodes can distinguish between the object and the occluders. This may be unrealistic in some practical settings. To address this problem, one can introduce another random variable that indicates the event of detecting and recognizing the object and include its probability in the likelihood. We have not implemented this modification in this paper, however.

4.4 Obtaining Occluder Priors

Our tracker assumes the availability of priors for the moving occluder positions. In this subsection we discuss how these priors may be obtained. In Section 5, we investigate the tradeoff between the accuracy of such priors and that of tracking.

Clearly, one could run a separate PF for each object, and then fit Gaussians to the resulting particle distributions. This requires solving the data association problem, which would require substantial local and centralized processing. Instead of solving the data association problem, trackers that represent the states of all objects in a joint state have been proposed (e.g. [20]). This approach, however, is computationally prohibitive as it requires employing an exponentially increasing number of particles in the size of the state.

Another approach to obtaining the priors is to use a hybrid sensor network combining, for example, acoustic sensors in addition to cameras. As these sensors use less energy than cameras, they could be used to generate the priors for the moving occluders. An example of this approach can be found in [23].

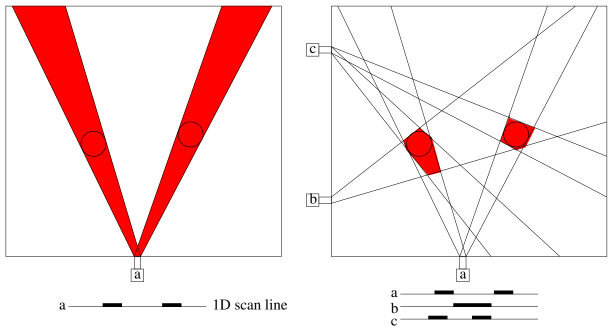


Figure 7: The visual hull is computed by back-projecting the scan lines to the room and intersecting the resulting cones.

Yet another approach to obtaining the occluder priors involves reasoning about occupancy using the "visual hull" (VH) as described in [24] (see Fig. 7). To compute the VH, the entire scan lines from the cameras are sent to the cluster head instead of only the centers of the object blobs in the scan lines as discussed in Section 3. This only marginally increases the communication cost. The cluster head then computes the VH by back-projecting the blobs in the scan lines to cones in the room. The cones from the multiple cameras are intersected to compute the total VH. Since the resulting polygons are larger than the occupied areas and "phantom" polygons that do not contain any objects may be present, VH provides an upper bound on occupancy. The computation of the VH is relatively light-weight, and does not require solving the data association problem. The VH can then be used to compute occluder priors by fitting ellipses to the polygons and using them as Gaussian priors. Alternatively, the priors can be assumed to be uniform distributions over these polygons. In this case the computation of $q_s^m(x)$ in (4) would need to be modified.

Although the VH approach to computing occluder priors is quite appealing for a WSN implementation, several problems remain to be addressed. These include dealing with the object's own blob and phantom removal [24], which is necessary because their existence can cause the killing of many good particles.

5. SIMULATION RESULTS

In a practical tracking setting one is given the room structure (including information about the static occluders), the range of the number of moving occluders and their motion model, and the required object tracking accuracy. Based on this information, one needs to decide on the number of cameras to use in the room and the amount of prior information about the moving occluder positions needed and how to best obtain this information. Making these decisions involve several tradeoffs, for example, between the occluder prior accuracy and the tracker performance, between the number of cameras used and the required occluder prior accuracy, and between the number of occluders present and the tracking performance. In this section we explore these tradeoffs using simulations.

In the simulations we assume a square room of size 100×100 units and a maximum of 8 cameras placed around its periphery (see Fig. 8). The black rectangle in the figure depicts a static occluder. Note, however, that in some of the simulations we assume no static occluders. All cameras look toward the center of the room. The

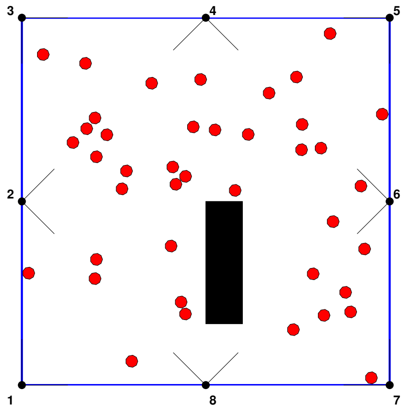


Figure 8: The setup used in simulations.

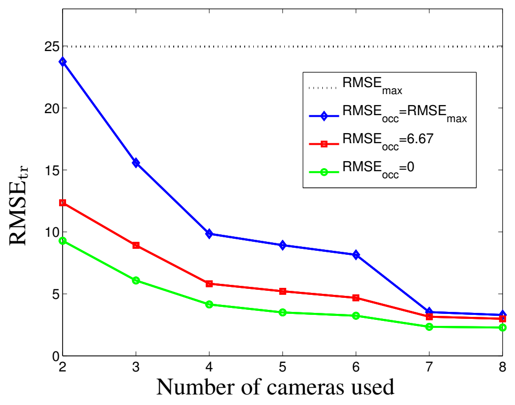


Figure 9: Average tracker RMSE versus the number of cameras for $M = 40$, and 1 static occluder. The dotted line is the worst case RMSE when no tracking is performed and the object is assumed to be at the center of the room.

camera FOV is assumed to be 90°. The standard deviation of the camera position error is $\sigma_{\rm pos} = 1$ unit, that of camera angle error is $\sigma_\theta = 0.01$ radians and read noise standard deviation is $\sigma_{\rm read} = 2$ pixels. The diameter of each moving occluder is assumed to be $D = 3.33$ units. We assume that the objects move according to random waypoints model. This is similar to the way we draw new particles from the importance density function as discussed in Subsection 4.1 with the following differences:

- The objects are only in regimes MTT or CT. There is no W regime.
- The objects choose their regimes deterministically, not randomly. If an object reaches its target or is heading toward the inside of a static occluder or outside the room boundaries, it transitions to the CT regime.
- Objects go around each other instead of colliding.

The average speed of the objects is set to 1 unit per time step. The standard deviation of the noise added to the motion each time step is 0.33 units. Fig. 8 also shows a snapshot of the objects for M=40 occluders. In the PF tracker we use 1000 particles. In each simulation, the object and the occluders move according to the random waypoints model for 4000 time steps.

To investigate tradeoffs involving moving occluder prior accuracy, we need a measure for the accuracy of the occluder prior. To develop such a measure, we assume that the priors are obtained using a KF run on virtual measurements of the moving occluder positions of the form

$$y_m(t) = x_m(t) + \upsilon(t),\ m = 1, 2, \ldots, M,$$

where $x_m(t)$ is the true occluder position, $\upsilon(t)$ is white Gaussian noise with covariance $\sigma_\upsilon^2 I$, and $y_m(t)$ is the measurement. We then use the average RMSE of the KF (RMSE$_{\rm occ}$) as a measure of the occluder prior accuracy. Lower RMSE$_{\rm occ}$ means higher accuracy sensors or more computation is used to obtain the priors, which result in more energy consumption in the network. At the extremes, RMSE$_{\rm occ}$ $= 0$ (when $\sigma_\upsilon = 0$) corresponds to complete knowledge of the moving occluder positions and RMSE$_{\rm occ}$ $= \rm RMSE_{max}$ (when $\sigma_\upsilon = \infty$) corresponds to no knowledge of the moving occluder positions. Note that the worst case RMSE$_{\rm max}$ is finite because when there are no measurements about the occluder positions, one can simply assume that they are located at the center of the room. This corresponds to RMSE$_{\rm max}$ $= 25.0$ units for the setup in Fig. 8.

To implement the tracker for these two extreme cases, we modify the $p(\eta|u)$ computation as follows. We assign 0 or 1 to $p(\eta|u)$ depending on the consistency of η with our knowledge about the occluders. For RMSE$_{\rm occ}$ $= 0$, i.e., when we have complete information about the moving occluder positions, the moving occluders are treated as static occluders. On the other hand, for RMSE$_{\rm occ}$ $=$ RMSE$_{\rm max}$, i.e., when there is no information about the moving occluder positions, we check the consistency with only the static occluder and the limited FOV information to assign zero probabilities to some particles. For the example in Fig. 6, we set P $(\{\eta_1 = 1\} \cap \{\eta_2 = \gamma_2\}|u_\ell) = 0$, because if camera i can see the object, the object cannot be at x_ℓ. Any other probability that is non-zero is set to 1. Note that for these 2 extreme cases, we no longer need the recursion discussed in Subsection 4.2 to compute the likelihood. Hence, the computational complexity is considerably lighter compared to using Gaussian priors.

First in Fig. 9 we plot the average RMSE of the tracker (RMSE$_{\rm tr}$) over 5 simulation runs for the two extreme cases of RMSE$_{\rm occ}$ $= 0$ and RMSE$_{\rm occ}$ $= \rm RMSE_{max}$ and for $\rm RMSE_{occ} = 6.67$ (obtained by setting $\sigma_\upsilon = 8$) versus the number of cameras (the cameras constitute a roughly evenly spaced subset of cameras in Fig. 8. For 2 cameras, orthogonal placement is used [2]). The dotted line represents the worst case RMSE, when there are no measurements and the object is assumed to be in the center of the room.

We then investigate the dependency of the tracker accuracy on the accuracy of the moving occluder priors. Fig. 10 plots the average RMSE for the tracker over 5 simulation runs versus RMSE$_{\rm occ}$ for $N = 4$ cameras. In order to include the effect of moving occluder priors only, we used no static occluders in these simulations. RMSE$_{\rm max}$ reduces to 21.3 units for this case. Note that there is around a factor of $2.35\times$ increase in RMSE$_{\rm tr}$ from the case of perfect occluder information (RMSE$_{\rm occ}$ $= 0$) to the case of no occluder information (RMSE$_{\rm occ}$ $= \rm RMSE_{max}$). Moreover, it is not realistic to assume that the occluder prior accuracy would be better than that of the tracker. With this consideration the improvement reduces to around $1.94\times$ (this is obtained by noting that RMSE$_{\rm tr}$ =RMSE$_{\rm occ}$ at around 3.72). These observations suggest that obtaining prior information may not be worthwhile in practice, unless it can be obtained cheaply and to a reasonable accuracy.

The tradeoff between RMSE$_{\rm occ}$ and the number of cameras needed to achieve average RMSE$_{\rm tr}$ $= 3$ is plotted in Fig. 11. As expected

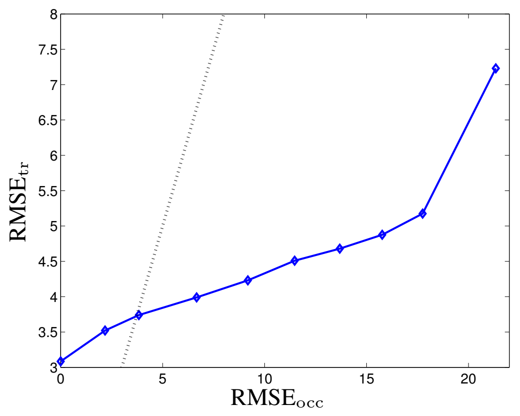


Figure 10: Dependency of the tracker average RMSE on the accuracy of the occluder prior for $N = 4$, $M = 40$ and no static occluders. The dotted line is for RMSE$_{\mathrm{tr}}$ =RMSE$_{\mathrm{occ}}$.

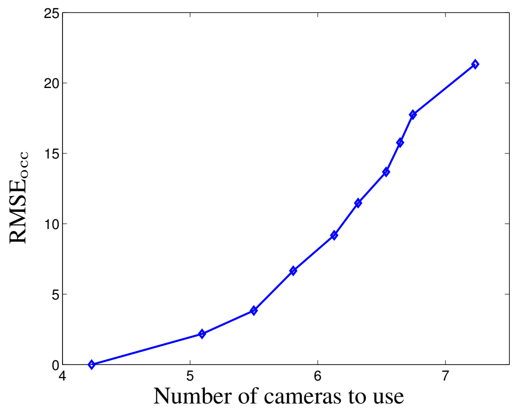


Figure 11: Tradeoff between the number of cameras and moving occluder prior accuracy for target tracker average RMSE=3 units for $M = 40$ and no static occluders.

there is a tradeoff between the number of cameras and the accuracy of the moving occluder priors as measured by RMSE$_{\mathrm{occ}}$. As more cameras are used, the accuracy of the prior information needed decreases. The plot suggests that if a large enough number of cameras is used, no prior information would be needed at all. Of course having more cameras means more communications and processing cost. So, in the design of a tracking system, one needs to compare the cost of deploying more cameras to that of obtaining better occluder priors.

Next we explore the question of how the needed moving occluder prior accuracy depends on the number of occluders present. To do so, in Fig. 12 we plot the RMSE$_{\mathrm{tr}}$ versus the number of moving occluders for the two extreme cases, RMSE$_{\mathrm{occ}}$=0 and RMSE$_{\mathrm{occ}}$ =RMSE$_{\max}$. Note that the difference between the RMSE$_{\mathrm{tr}}$ for the two cases is the potential improvement in the tracking performance achieved by having occluder prior information. When there are very few moving occluders, prior information does not help (because the object is not occluded most of the time). As the number of occluder increases prior information becomes more useful. But the difference in RMSE$_{\mathrm{tr}}$ between the two extreme cases decreases when too many occluders are present (because the object becomes occluded most of the time).

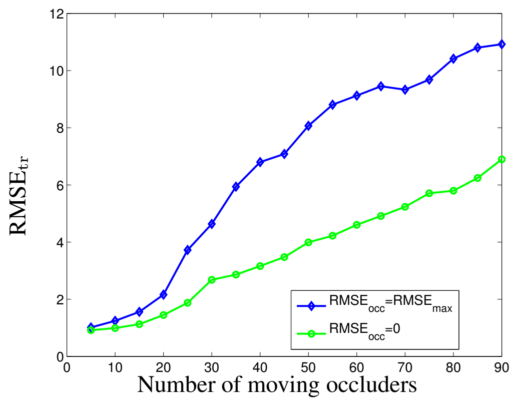


Figure 12: Tracker average RMSE versus the number of moving occluders for the two extreme cases RMSE$_{\mathrm{occ}}$=0 and RMSE$_{\mathrm{occ}}$=RMSE$_{\max}$. Here $N = 4$ and there are no static occluders.

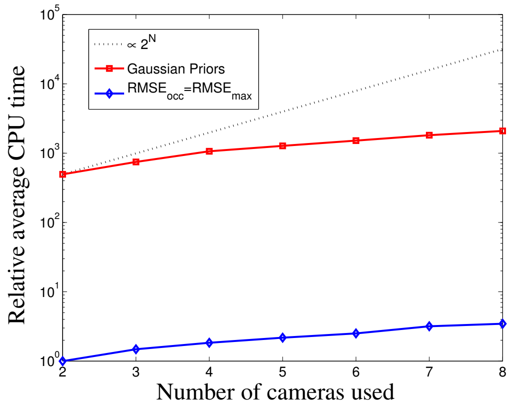
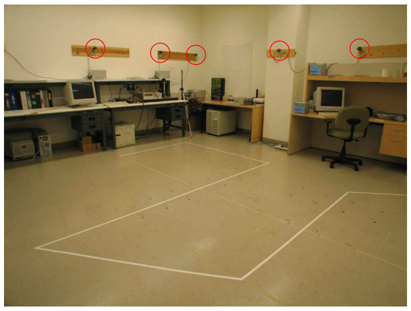


Figure 13: Average CPU time for computing the likelihoods relative to that for the case of 2 cameras and no occluder prior, i.e., RMSE$_{\mathrm{occ}}$=RMSE$_{\max}$. Here $M = 40$ and there is 1 static occluder.

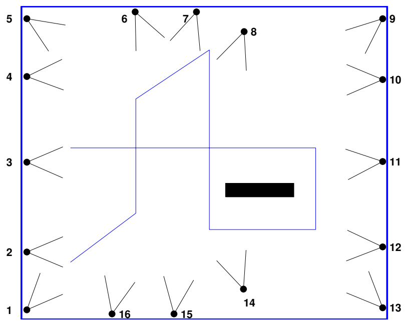
In Subsections 4.3 and 4.4, we mentioned that the complexity of computing the likelihood given u_ℓ is exponential in the number of cameras that cannot see the object and are assigned to the region x_ℓ belongs to. We proposed that the limited camera FOV significantly reduces this computational complexity. To verify this, in Fig. 13 we plot the average CPU time (per time step) used to compute the likelihood relative to that of RMSE$_{\mathrm{occ}}$=RMSE$_{\max}$ case for 2 cameras, versus the total number of cameras in the room. The simulations were performed on an 3GHz Intel Xeon Processor running MATLAB R14. Note that the rate of increase of the CPU time using priors is significantly lower than 2^N, where N is the number of cameras used, and it is close to the rate of increase of RMSE$_{\mathrm{occ}}$=RMSE$_{\max}$ case. In fact, the rate of increase for this particular example is close to linear in N.

6. EXPERIMENTAL RESULTS

We tested our tracking algorithm in an experimental setup consisting of 16 web cameras placed around a $22' \times 19'$ room. The horizontal FOV of the cameras used is $47°$. A picture of the lab is shown in Fig. 14(a) and the relative positions and orientations



(a)



(b)

Figure 14: Experimental setup. (a) View of lab (cameras are circled). (b) Relative locations of cameras and virtual static occluder. Solid line shows actual path of the object to track.

of the cameras in the room are provided in Fig. 14(b). Each pair of cameras is connected to a PC via IEEE 1394 (FireWire) interface and each can provide 8-bit 3-channel (RGB) raw video at 7.5 Frames/s. The data from each camera is processed independently as described in Section 3. The scan line data is then sent to a central PC (cluster head), where further processing is performed.

The object follows the pre-defined path (shown in Fig. 14) with no occlusions present and 200 time-steps of data is collected. The effect of static and moving occluders is simulated using 1 virtual static occluder and $M = 20$ virtual moving occluders: we threw away the measurements from the cameras that would have been occluded, had there been real occluders. We chose to simulate the occluders because it is otherwise impossible to obtain the perfect occluder positions ($\mathrm{RMSE}_{\mathrm{occ}} = 0$ case). The moving occluders walk according to the model explained in Section 5. D is chosen 12 inches for the moving occluders, and the camera noise parameters were assumed $\sigma_{\mathrm{pos}} = 6$ inches, $\sigma_{\mathrm{read}} = 2$ pixels and $\sigma_\theta = 0.005$ radians.

Figure 15 plots the average RMSE of the tracker over 40 simulation runs for the two extreme cases of $\mathrm{RMSE}_{\mathrm{occ}} = \mathrm{RMSE}_{\mathrm{max}} = 61.8$ inches and $\mathrm{RMSE}_{\mathrm{occ}} = 0$ and for $\mathrm{RMSE}_{\mathrm{occ}} = 14.2$ inches versus the number of cameras. There is a notable difference in the performance between the three cases throughout the entire plot, but the difference is more pronounced when the number of cameras is small, agreeing with the tradeoffs discussed in Section 5.

7. CONCLUSION

We described a sensor network approach for tracking a single object in a structured environment using multiple cameras. Instead of tracking all objects in the environment, which is computationally very costly, we track only the target object and treat others as occluders. The tracker is provided with complete information about the static occluders and some prior information about the moving occluders. One of the main contributions of this paper is developing a systematic way to incorporate this information into the tracker formulation. Using simulations we explored tradeoffs involving the occluder prior accuracy, the number of cameras used, the number of occluders present, and the accuracy of tracking with some interesting implications.

Several areas need to be explored further, including (i) running simulations and experiments over real world environments to validate our preliminary findings, (ii) developing a theoretical framework for investigating the aforementioned tradeoffs, (iii) exploiting the independence between the η_is for cameras that are far apart to further reduce the computational complexity of computing the likelihoods, and (iv) developing a cheap method for obtaining reasonable accuracy occluder priors (perhaps based on VH).

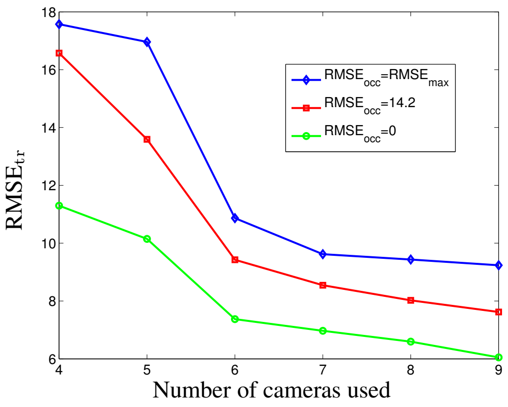


Figure 15: Experimental results. Average tracker RMSE versus the number of cameras for $M = 20$, and 1 static occluder.

Acknowledgments

The authors would like to thank Dr. Jack Wenstrand for helpful discussions. The authors wish to acknowledge the support of DARPA Microsystems Technology Office Award No. N66001-02-1-8940, NSF grants CNS-0435111, CNS-0626151, ARO grant W911NF-06-1-0275, and the support of DoD Multidisciplinary University Research Initiative (MURI) program administered by ONR under Grant N00014-00-1-0637.

8. REFERENCES

[1] R. Holman and T. Ozkan-Haller, "Applying video sensor networks to nearshore environment monitoring," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 14–21, 2003.

[2] A. O. Ercan, D. B.-R. Yang, A. El Gamal, and L. J. Guibas, "Optimal placement and selection of camera network nodes for target localization," in *Proceedings of DCOSS*, June 2006.

[3] A. O. Ercan, A. El Gamal, and L. J. Guibas, "Camera network node selection for target localization in the presence of occlusions," in *Distributed Smart Cameras*, October 2006.

[4] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter, Particle Filters for Tracking Applications*. Artech House, 2004.

[5] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor

networks," *Proceedings of IEEE*, vol. 91, no. 8, pp. 1163–1171, August 2003.

[6] F. Zhao, J. Liu, J. Liu, L. J. Guibas, and J. Reich, "Collaborative signal and information processing: an information-directed approach," *Proceedings of IEEE*, vol. 91, no. 8, pp. 1199–1209, August 2003.

[7] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *Proceedings of SENSYS*, November 2003.

[8] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration and tracking in an ad-hoc sensor network," in *Proceedings of IPSN*, April 2006.

[9] N. Shrivastava, R. Mudumbai, and U. Madhow, "Target tracking with binary proximity sensors: Fundamental limits, minimal descriptions, and algorithms," in *Proceedings of SENSYS*, November 2006.

[10] P. V. Pahalawatta, D. Depalov, T. N. Pappas, and A. K. Katsaggelos, "Detection, classification, and collaborative tracking of multiple targets using video sensors," in *Proceedings of IPSN*, April 2003, pp. 529–544.

[11] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proceedings of IPSN*, April 2006.

[12] P. F. Gabriel, J. G. Verly, J. H. Piater, and A. Genon, "The state of the art in multiple object tracking under occlusion in video sequences," in *Proceedings of ACIVS*, September 2003.

[13] A. Yilmaz, X. Li, and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1531–1536, November 2004.

[14] Q. Cai and J. K. Aggarwal, "Tracking human motion in structured environments using a distributed-camera system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 1241–1247, 1999.

[15] S. Khan, O. Javed, Z. Rasheed, and M. Shah, "Human tracking in multiple cameras," in *Proceedings of ICCV*, July 2001.

[16] W. Zajdel, A. T. Cemgil, and B. J. A. Krose, "Online multicamera tracking with a switching state-space model," in *Proceedings of ICPR*, August 2004.

[17] A. Utsumi, H. Mori, J. Ohya, and M. Yachida, "Multiple-view-based tracking of multiple humans," in *Proceedings of the ICPR*, 1998.

[18] K. Otsuka and N. Mukawa, "Multiview occlusion analysis for tracking densely populated objects based on 2-d visual angles," in *Proceedings of CVPR*, 2004.

[19] S. L. Dockstander and A. M. Tekalp, "Multiple camera tracking of interacting and occluded human motion," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1441–1455, October 2001.

[20] A. Doucet, B.-N. Vo, C. Andrieu, and M. Davy, "Particle filtering for multi-target tracking and sensor management," in *Proceedings of ISIF*, 2002, pp. 474–481.

[21] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University, Technical Report CMU-CS-91-132, April 1991.

[22] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York, NY: John Wiley & Sons Inc., 2001.

[23] X. Sheng and Y.-H. Hu, "Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 53, no. 1, pp. 44–53, 2005.

[24] D. B.-R. Yang, H. Gonzales-Banos, and L. J. Guibas, "Counting people in crowds with a real-time network of image sensors," in *Proceedings of ICCV*, October 2003.

APPENDIX

A. DERIVATION OF THE CAMERA MEASUREMENT NOISE VARIANCE

Without loss of generality, assume that the camera is at the origin, and the object is at $x = [x_1 \ x_2]^T$. Then $h_i(x)$ and $d_i(x)$ in Fig. 3 are given by

$$h_i(x) = \sin(\theta_i)x_1 - \cos(\theta_i)x_2$$
$$d_i(x) = -\cos(\theta_i)x_1 - \sin(\theta_i)x_2.$$

We take partial derivatives of z_i given in (1) with respect to θ_i, x_1 and x_2 to obtain

$$\frac{\partial z_i}{\partial \theta_i} = -f_i\left(1 + \frac{h_i^2(x)}{d_i^2(x)}\right)$$
$$\frac{\partial z_i}{\partial x_1} = f_i\frac{d_i(x)\sin(\theta_i) + h_i(x)\cos(\theta_i)}{d_i^2(x)}$$
$$\frac{\partial z_i}{\partial x_2} = f_i\frac{-d_i(x)\cos(\theta_i) + h_i(x)\sin(\theta_i)}{d_i^2(x)}.$$

An error in the camera position translates into errors in x_1 and x_2. Assuming the errors in the two directions to be independent, zero mean and have the same standard deviation σ_{pos}, and the error in the angle to be zero mean with standard deviation σ_θ and independent of the position errors, we obtain

$$\begin{aligned}\sigma_{v_i}^2 &= \sigma_{z_i|x}^2 \\ &= \left(\frac{\partial z_i}{\partial \theta_i}\right)^2\sigma_{\theta_i}^2 + \left(\frac{\partial z_i}{\partial x_1}\right)^2\sigma_{x_1}^2 + \left(\frac{\partial z_i}{\partial x_2}\right)^2\sigma_{x_2}^2 + \sigma_{\text{read}}^2 \\ &= f_i^2\left(1 + \frac{h_i^2(x)}{d_i^2(x)}\right)^2\sigma_\theta^2 + f_i^2\frac{h_i^2(x) + d_i^2(x)}{d_i^4(x)}\sigma_{\text{pos}}^2 + \sigma_{\text{read}}^2.\end{aligned}$$