Automated Intruder Tracking using Particle Filtering and a Network of Binary Motion Sensors*

Jeremy Schiff
Dept. of EECS
University of California, Berkeley
Berkeley CA, 94720–1777, USA
jschiff@cs.berkeley.edu

Ken Goldberg
Dept. of IEOR and EECS
University of California, Berkeley
Berkeley CA, 94720–1777, USA
goldberg@berkeley.edu

***Abstract*— Our objective is to automatically track and capture photos of an intruder using a robotic pan-tilt-zoom camera. In this paper, we consider the problem of automated position estimation using a wireless network of inexpensive binary motion sensors. The challenge is to incorporate data from a network of noisy sensors that suffer from refractory periods during which they may be unresponsive. We propose an estimation method based on Particle Filtering, a numerical sequential Monte Carlo technique. We model sensors with conditional probability density functions and incorporate a probabilistic model of an intruder's state that utilizes velocity. We present simulation and experiments with passive infrared (PIR) motion sensors that suggest that our estimator is effective and degrades gracefully with increasing sensor refractory periods.**

Index Terms - Security, Sensor Networks, Particle Filter, Tracking, Sensor Fusion.

I. Introduction

Many new technologies for automated security, wireless networks, and sensing have emerged since the terrorist attacks of 9/11. Automation of security systems is an active area of research that also raises fundamental privacy concerns. In this paper we consider the problem of automatically tracking an intruder.

New robotic cameras can be servoed to observe high-resolution detailed images of activity over a wide field of view. For example, the Panasonic KX-HCM280 pan-tilt-zoom camera (commercially available for under $1000.00) can capture up to 500 Mpixels per steradian. We are exploring how these cameras can be automatically controlled for security applications using a new class of inexpensive binary sensors with built-in wireless communications.

Commercially available passive infrared (PIR) motion sensors (available for under $25.00 from x10.com) cover a field of view of 120 degrees. Wireless motion sensors like these are subject to two significant drawbacks: (1) they are binary and (2) after being triggered by motion in their field of view, they suffer from a refractory period of several seconds during which they are unresponsive. Since these and related sensors provide only coarse information about the presence or absence of an intruder, we propose a probabilistic tracking model based on Particle Filtering, where many sampled estimates ("particles") based on distributions of sensor response and intruder state are simultaneously estimated and fused to produce an aggregate point estimate of intruder position. Recent advances in computer processing make particle filtering feasible in real time. Using the estimate of the intruder's location the robotic camera actuates to document the intruder's movements for later analysis.

*This work was partially supported by NSF Awards 0535218, 0424422, and by UC Berkeley's Center for Information Technology Research in the Interest of Society (CITRIS).

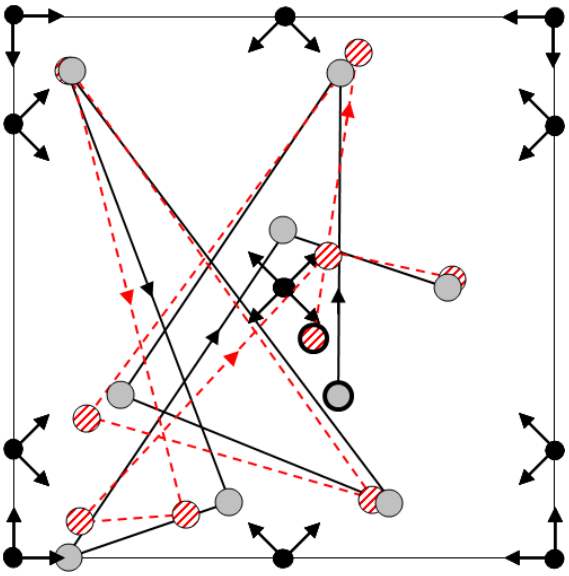

Fig. 1. True vs. estimated intruder path for one randomized simulator run. Position and orientation of a network of fourteen motion sensors is indicated with solid dots and arrows. The true intruder path is indicated with hashed circles and dashed lines. The estimated intruder path is indicated with grey circles and solid lines. The bold-edged circles indicates the intruder's estimated and true starting points. Error is the spatial distance between true and estimated intruder position. In our simulation experiments, we report the distribution of error values.

II. Related Work

There is an extensive literature on pursuit and evasion strategies for capturing an intruder. These range from locating a stationary intruder in an unknown room [1] to pursuit-evasion games, where a pursuing robot hunts a mobile evader that intelligently works to avoid capture. Examples and surveys can be found in [2], where the pursuer has a line of sight optical sensor, and [3], where the pursuer must avoid being seen by the evader. In our

problem, the environment is known and binary sensors are fixed and distributed through the environment in advance.

Many tracking systems make use of fixed video cameras that monitor an area with fixed resolution. Background subtraction techniques can be used to find moving objects in the video stream. Once these are identified, modeling and filtering is performed to extract body parts or differentiate between distinct objects in a scene. Examples of such systems can be found in [4], [5], [6].

Another approach uses combinations of different types of sensors. In these systems, these sensor types are often explicitly chosen to complement each other. This can be formulated as a multi-modal sensing problem as [7] describes. In this problem, the sensor types have different false positive and negative rates, as well as different sensing ranges. At each time step, only one mode of sensing can be used to locate an object. It is important to note that this work attempts to detect and localize a static object, rather than track a moving one.

Jeffery et al. present in [8] another multi-modal approach, where they use sound-detecting motes, PIR motion detectors, and RFID tags to track people. There is an additional constraint that the RFID tags were worn by the people being tracked. This leads us to yet another approach, refining a technology down to tracking a mobile tag attached to an object. For instance, [9] describes a scheme for tracking a tag which emits a near-infrared signal.

There are many ways to probabilistically estimate the state of a system, for instance the location of an object, via observations. Kalman Filtering [10] is one of the most fundamental. It uses three models: a prior distribution of the object's initial state, a transition model describing how an object transitions between states, and a observation model describing the probability of an observation given some object state. Kalman Filtering limits these functions to linear function of Gaussians. Other tools have been derived to relax these constraints, such as the Extended Kalman Filter [11], and the Particle Filter [12]. There have also been many specific variants of Particle Filters. A survey of Particle Filters is presented in [13].

A well studied problem in robotics is Simultaneous Localization and Mapping (SLAM). In this problem, a mobile robot explores an unknown area. It must use its sensors to build a complete map of its surroundings, and simultaneously deduce its position with reference to these surroundings. A book discussing many aspects of SLAM can be found in [12]. Methods for leveraging sensors to learn about the world apply, whether we are using the robot to map a room or using a room of sensors to find an intruder.

Robotic camera systems have also been researched extensively. There are other paradigms for camera control that have been researched other than autonomous systems. For instance in [14], [15], the problem of reconciling simultaneous requests on a single camera from multiple people. In this problem, requests for camera viewing locations are posed as rectangular regions over a panorama of the camera's range. They calculate the optimal frame to control the camera by maximizing the overlap of the frame requests. A related problem is the dynamic updating of the panorama from captured frames [16].

Another relevant research topic is the Art Gallery Problem, where the objective is to place cameras in an art gallery such that all points in the gallery can be viewed by at least one camera. Thorough surveys of problems related to the Art Gallery Problem can be found in [17], [18].

III. Problem Statement

In this section we formalize our problem. Our goal is to track (and document) the progress of an intruder moving through a known room using a sequence of binary sensor readings.

A. Inputs and Assumptions

Setup: polygonal room geometry, probabilistic sensor model, position and orientation of each sensor, probabilistic model of intruder movements, dimensions of intruder bounding prism. Input: time sequence of firing patterns from binary sensors. Output: estimate of intruder path.

Setup occurs once during initialization, while input is information that the system processes in real-time. Error is the spatial distance between estimated intruder position and actual intruder position. We report on the distribution of error values.

1) Room Geometry: The geometry of the room is described as a polygon represented by a set of vertices and edges. We denote all points in the room by the set R. We require that the room is star shaped so that a single robotic camera can be placed to have direct line of sight to any potential area the intruder may occupy. For now, we assume that there are no obstacles in the room.

2) Robotic Camera: Our system uses a single, robotic camera with a fixed base that focuses on specific areas within the observable space. The camera has three degrees of freedom: pan, tilt, and zoom.

3) Sensor Modeling: We have a set of $N = \{1 \dots n\}$ binary sensors, where each sensor i is modeled by two conditional probability density functions (CPDFs), each with a convex polygonal detection region R_i. Every point in the room must be covered by the region of at least one sensor:

$$R \subseteq \bigcup_{i=1}^{n} R_i$$

We use subset instead of equals because the union of the sensor region can exceed the room.

Each sensor i has a refractory period, whereby after triggering, it usually becomes unable to trigger again for another f_i seconds. The X10 PIR sensors we used for our experiments exhibit this problem.

We denote the generated sensor data by the set:

$$D = \left\{ (i,t) : \begin{array}{l} i \in N, t \in \mathbb{R}^+, \\ \text{sensor } i \text{ triggers at time } t \end{array} \right\}$$

We process the sensor data generated from the intruder every t_P seconds, and thus we discretize time into iterations that are t_P seconds apart. Formally, the time t_τ of the τ'th iteration is defined as :

$$\forall \tau \in \mathbb{N} : t_\tau = \tau \cdot t_P$$

We use a world-coordinate system to integrate data across sensors. We overlay a uniform three-dimensional grid which discretizes the world into cells $j \in \{1 \dots m\}$.

We define three indicator variables which are defined for each iteration. The event that an intruder fully occupied some world-space cell between the previous and current iteration:

$$O_{j,\tau} = \begin{cases} 1 & \text{if } \exists t : \text{ an intruder fully occupies cell } j \text{ at time } t, t_{\tau-1} < t \leq t_\tau \\ 0 & \text{otherwise} \end{cases}$$

The event that the sensor is triggered between the previous and current iteration:

$$S_{i,\tau} = \begin{cases} 1 & \text{if } \exists t : (i,t) \in D, t_{\tau-1} < t \leq t_\tau \\ 0 & \text{otherwise} \end{cases}$$

The event that the sensor experiences a refractory period between the previous and the current iteration:

$$B_{i,\tau} = \begin{cases} 1 & \text{if } \exists t : (i,t) \in D, t_{\tau-1} - f_i < t \leq t_\tau \\ 0 & \text{otherwise} \end{cases}$$

Each cell in the grid corresponds to entries in two CPDFs. The first describe the probability that a sensor is not experiencing a refractory period and triggers, provided a sensor stimulus fully occupies only that cell:

$$P(S_{i,\tau} | O_{j,\tau} = 1, B_{j,\tau} = 0).$$

The second is the probability that a sensor is not experiencing a refractory period and triggers, provided no sensor stimulus occupies any part of that cell:

$$P(S_{i,\tau} | O_{j,\tau} = 0, B_{j,\tau} = 0).$$

Also, we need to compute the probability that a sensor fires, given that it is undergoing a refractory period:

$$P(S_{i,\tau} | B_{j,\tau} = 1)$$

With these three distributions, we completely describe for any iteration and sensor, the probability that a sensor fires.

When modeling these probabilities, we make a number of significant assumptions such as the independence of an intruder's occupancy of world-space cells causing a sensor to trigger. We discuss these assumptions in detail in Appendix I.

4) Intruder Modeling: We assume that we know the times the intruder enters and exits the room. We model our intruder by position, speed, orientation, and size. At each timestep, we add a sample from a zero-mean Gaussian to the intruder's speed and bound it by both the intruder's maximum speed v_{MAX} as well as the geometric constraints of the room. The orientation is also modeled by adding a sample from a zero-mean Gaussian, however the variance changes over time and is inversely proportional to the current speed of the intruder. We discuss this in more detail in Section VII-B.1. To model size, we use a bounding rectangular prism, with given width, depth, and height. We model the path of the intruder as lines between samples, as is shown by the dashed line in Figure 1.

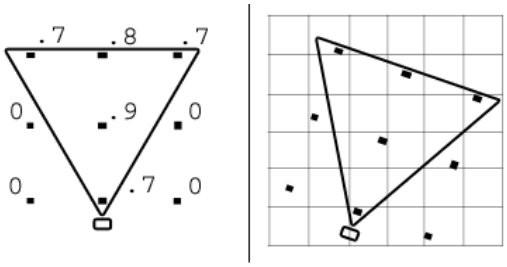

Fig. 2. The left image depicts a two dimensional slice of a sensor's characterization, with the corresponding values at each point depicted in a grid-overlay. The right image shows a slice where the points have been transformed into worldspace.

B. Outputs

The system uses the set of sensor firings to infer the intruder's path as illustrated in Figure 1. The objective is to minimize the error between the intruder's actual location x, and the system's estimation $\hat{x}$. It uses this information to generate a set of images represented by a unique tuple of pan, tilt, zoom, and time.

IV. Framework

Our system involves three phases. In the Characterization Phase, we build a probabilistic model for the sensor. In the Deployment Phase, we place the sensors at the surveillance location and transform the distributions accordingly. Finally in the Tracking Phase, the system uses these sensor instantiations, in conjunction with firing information, to estimate the presence and location of an intruder traveling around a room. We use this information to direct a robotic camera to take photos of the intruder.

V. Characterization Phase

In this phase, we represent the properties of the sensor according to our sensor model. We begin by finding the duration of the sensor's refractory period f_i by continuously stimulating the sensor, and determine the rate at which it provides data. This sampling rate is determined by how often we will process the sensor data t_P. By comparing the differences between times the sensor transmits data of the stimulus, we compute f_i. In order to compute the probability that the sensor triggers while undergoing a refractory period, $P(S_{i,\tau} = 1 | B_{i,\tau} = 1)$, we stimulate the sensor and then monitor how often it signals an intruder for each iteration during its refractory period.

To calculate the CPDFs for each sensor-type, we perform a characterization according to a single sensor's coordinate space and overlay a three dimensional uniform grid over this space. By repeatedly stimulating the sensor restricted

locally to the location of each index in the grid, we can calculate for each index the probability that a stimulus at only that location causes the sensor to fire. For index j, call this value η_j. We depict a two-dimensional slice of sampling locations, which results in a grid of values Figure 2 (left). Each sample tests the sensor's response patterns to learn the empirical sensitivity of the sensor and how this sensitivity changes with respect to the location of the stimulus. In a PIR sensor, the stimulus might be a small amount of movement centered at the sample point. It is important to ensure that the samples are not taken during the sensor's refractory period and that the stimuli not overlap with the stimulus at another point on the grid. We describe in Section VI how each sample point maps to an independent random variable, and avoiding stimulus overlap makes this independence assumption more reasonable.

VI. Deployment Phase

Once we have a sensor type's characterization, we use a sensor's position, orientation and type to transform it into the shared world coordinate frame. A 2-dimensional layer of an example is shown in Figure 2 (right). We approximate the conversion by placing the center of the sample rotated from the sensor i's space into the containing cell j in world space; we use this as the value for $P(S_{i,\tau} = 1|O_{j,\tau} = 1, B_{i,\tau} = 0)$. To sufficiently characterize a sensor such that all m world-space cells have readings, we would need to sample all permutations of occupancies of cells, namely

$$P(S_{i,\tau}|O_{1,\tau} = o_{1,\tau}, \ldots, O_{m,\tau} = o_{m,\tau}, B_{i,\tau} = 0)$$

Acquiring all 2^m assignments is not feasible, so we make the assumptions described in Appendix I. We determine the values of all world-space cells within the convex hull of the cells containing non-zero η_js using Inverse Distance Weighting [19] and all other cells have a value of zero.

Because we do not have samples for all permutations for occupancies of cells, we also cannot calculate $P(S_{i,\tau}|O_{j,\tau} = 0, B_{i,\tau} = 0)$. This again leads us to the Appendix I assumptions, which allow us to determine $P(S_{i,\tau}|O_{j,\tau} = 0, B_{i,\tau} = 0)$ by using the frequency of sensor false positives. Let this rate be h, the number of world-space cells where $P(S_{i,\tau}|O_{j,\tau} = 1, B_{i,\tau} = 0) > 0$ be ξ, and these cells be independent. Then

$$P(S_{i,\tau} = 1|O_{j,\tau} = 0, B_{i,\tau} = 0) = 1 - (1 - h)^{1/\xi}$$

Taking the convex hull of cells with non-zero values mapped to them yields the sensor shape's convexity requirement.

VII. Tracking Phase

A state is a set of information that describes the relevant properties of an object that evolves over time, such as position and velocity. The probabilistic model of how states transition over time is referred to as the transition model. Because our state estimation is tracking of the intruder, we refer to this as our intruder model. A probabilistic model of the observable evidence, given some unobservable state, is the observation or sensor model. If the system can be described by these two models, it can be represented as a Dynamic Bayesian Network (DBN) [20]. If the intruder model and sensor model are linear functions of Gaussians, where the new state is a linear function of the previous state plus Gaussian noise, then a Kalman Filter can be used to provide the optimal solution to the state estimation problem.

While our intruder model, which is presented in Section VII-B is a linear function of Gaussians, our sensor model is not. Because our sensor model is only restricted to a convex-shaped CPDF, there are two options: an Extended Kalman Filter or a Particle Filter. We can use a function to estimate our sensor model, for instance a spline, and then we can use an Extended Kalman Filter. Instead, we choose Particle Filtering because it can be used directly by sampling the CPDFs.

A. SIR Particle Filtering

While there are many versions of Particle Filters, we use the Sampling Importance Resampling (SIR) Filter as described in [13], [20]. It is a sampling-based method for performing state estimation of DBNs over discrete time. We represent our current state as a random variable X_τ with instantiation x_τ and our evidence of our hidden state E_τ with instantiation e_τ. We use three distributions to perform SIR Particle Filtering. We model the prior probability distribution of the intruder's state $P(X_0)$ as a way to initialize the system, the intruder model $P(X_\tau|X_{\tau-1})$, and the sensor model $P(E_\tau|X_\tau)$. The prior describes the distribution of the intruder's location at the beginning of inference. The intruder model describes, given the current state of the intruder, the distribution of the intruder's location at the next timestep. Lastly, the sensor model describes the distribution over sensor triggerings resulting from a specific intruder state. When using Particle Filtering, we maintain a vector of samples of states of the system, distributed according to the likelihood of our evidence. At each iteration, we advance each sample of the state according to the intruder model, assign a probability of each sample's likelihood according to the sensor model, and then resample from a normalized distribution across the samples.

B. Tracking with Particle Filtering

To use Particle Filtering, we need to define the state of our system at every iteration $x_{i,\tau}$ and the observed evidence e_τ. By using previous phases of the system, we derive the distributions for $P(X_0), P(X_\tau|X_{\tau-1})$, and $P(E_\tau|X_\tau)$ using Particle Filtering.

1) Intruder Model: The state of our system for each sample is represented by a 5 tuple of x-position, y-position, orientation, speed, and if the sensor is experiencing a refractory period.

While the rest of our system uses three-dimensions, the intruder model does not change its location along the z-axis. This is because the placement of the sensors and the

camera will not necessarily be in the same plane, but it is reasonable to assume the intruder will remain on the floor.

Because we are using Particle Filtering, the intruder model is a function that maps from the old state and yields some new state. The transition model we chose $P(X_\tau|X_{\tau-1})$ determines position by Euler Integration. The standard deviation of the Gaussian added to current speed is determined according to typical properties of our intruder. The Gaussian added to the orientation has a standard deviation that is inversely proportional to the current speed because the faster our intruder is traveling, the more likely it is that the intruder's path will not deviate significantly from the current one.

Define $\beta_{i,\tau}$ as deterministic variables, timestamped every time their respective $S_{i,\tau}$ is true. If there sensor is triggering, we set β_i, τ to be the blind time. Otherwise, the $\beta_{i,\tau}$ for sensor i subtracts off t_P from the previous timestep, and is restricted to be at least 0. Formally:

$$\beta_{i,\tau} = \begin{cases} f_i & \text{if } S_{i,\tau} = 1 \\ \max(0, \beta_{i,\tau-1} - t_P) & \text{else} \end{cases}$$

The random variable $B_{i,\tau} = 1$ iff $\beta_{i,\tau} = 0$.

We define $P(X_0)$ for our system as uniform at random for position, orientation and speed. This is sufficient because the system converges fast enough to the right distribution that no pre-seeding is necessary. Whenever the system begins, it sets $\beta_{i,0} = 0, \forall i \in N$ so that all sensors are defined as not undergoing a refractory period.

As the transitions discussed above have no knowledge of the room geometry, we need to impose the room's constraints on the transition model. To add these constraints, we use rejection sampling, first presented in [21]. We propose an update to our state according to our transition model. If the position is within the room and the speed is between 0 and v_{MAX}, then we accept this sample. If our proposal does not fit within these bounds, we continue to try new updates to the old state until we find one that satisfies these constraints.

2) Sensor Model: Denote the set of evidence among all sensors by:

$$e_t = \{S_{1,\tau} = s_{1,t}, \ldots, S_{N,\tau} = s_{N,\tau}\}$$

Because any sensor triggering is independent of all others given the location of the intruder:

$$\begin{aligned} P(E_\tau|X_\tau) &= P(S_{1,\tau}, \ldots, S_{N,\tau}|X_\tau) \\ &= \textstyle\prod_{i=1}^{N} P[S_{i,\tau}|X_\tau] \end{aligned}$$

In order to compute the probability that a specific sensor i fires at iteration τ, due to an intruder of state x we use two different distributions. If the sensor is undergoing a refractory period, we use:

$$\begin{aligned} &P(S_{i,\tau} = 1|X_\tau = x_\tau) \\ &\quad = P(S_{i,\tau} = 1|B_{i,\tau} = 1) \end{aligned}$$

Otherwise, we use:

$$\begin{aligned} &P(S_{i,\tau} = 1|X_\tau = x_\tau) \\ &\quad = 1 - \textstyle\prod_{k=1}^{m} \mathrm{L} \begin{pmatrix} P(S_{i,\tau} = 0|O_{k,\tau} = 0, B_{i,\tau} = 0), \\ P(S_{i,\tau} = 0|O_{k,\tau} = 1, B_{i,\tau} = 0), \\ \mathrm{V}(O_{k,\tau}, x_\tau) \end{pmatrix} \end{aligned}$$

We use L(a, b, c) to mean the linear interpolation of percentage c between a and b. V(a, b) is the volume percentage of cell a occupied by intruder with state b for a given bounding prism size.

C. Estimator

The Particle Filter gives us a mechanism to determine our confidence that an intruder is in a specific region. After the resampling step, the samples of the state space have been distributed according to the likelihood of the state being correct. The higher the density of particles in a region, the more likely that area is occupied by the intruder. Because the intruder's dimensions are approximated by a prism that bounds the size of the intruder, to determine the intruder's location at each timestep τ, we iterate through each world-space cell and count the number of samples of our state that reside within the bounding prism centered at the cell's position. We choose the cell that maximizes the number of samples as the estimated location of the intruder.

Once we estimate the location of the object in world space, we calculate the pan, tilt, and zoom necessary for the camera to take a photo. Because we bound the intruder with a rectangular prism, we pick the parameters that make the camera view the entire bounding prism, but no more. To determine pan and tilt, we find the center of the intruder bounding box, and then convert from cartesian to spherical coordinates, where the origin in spherical coordinates is the location of the camera. Next, we project the intruder bounding prism onto an image plane orthogonal to the viewing plane and determine the correct zoom to view just the projection, but no more. There is a direct mapping between the location of the bounding prism in world-space and a specific pan, tilt, and zoom for the camera that maps to this location.

VIII. Simulation Results

We implemented a simulator to evaluate our estimator. First, it generates random intruder paths and probabilistic sensor models to compute corresponding sets of sensor data. Second, it processes this data to generate estimates of the intruder's location. We compare the true positions with the estimated intruder positions over time and compute error based on Euclidean distance. The simulator runs on a 1.6 Gigahertz Pentium M with 768 Megabytes of RAM using 1000 state samples (particles).

We performed simulation experiments for three sensor types: (1) perfect optical beams, (2) perfect motion sensors, and (3) imperfect motion sensors. For each, we considered refractory periods of length 0, 4, 8, and 16 seconds. We also consider the effect of varying the number of binary sensors in the network.

In all simulations, we use a square room of size 10 x 10 x 10 cells of unit size. We defined our intruder bounding prism of 1 cell height and 2 cell width and length. We use the parameters: $t_P = 2$ seconds, $v_{MAX} = 4$ cells per second and velocity Gaussian standard deviation is 1.5 cells per second, and $P(S = 1|B = 1) = 10^{-6}$. We generated 10 3-minute trajectories evaluated at every 2

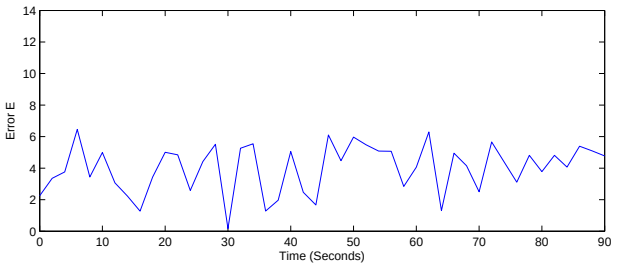

Fig. 3. Error in estimated intruder position over time for the baseline Naive Estimator that simply reports the intruder is stationary in the middle of the room.

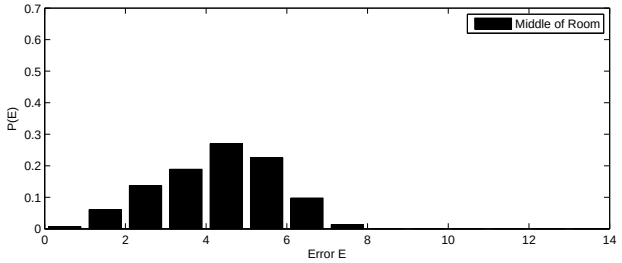

Fig. 4. Baseline Naive Estimator: Distribution of error values.

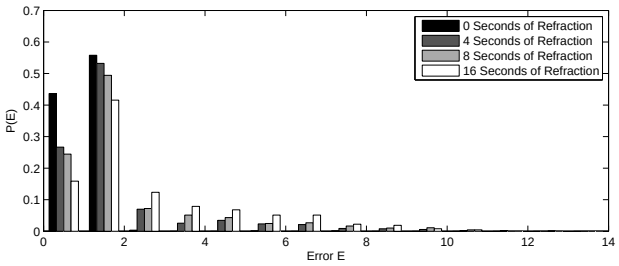

Fig. 5. Twenty-two Perfect Optical-Beam Sensors: Distribution of error values.

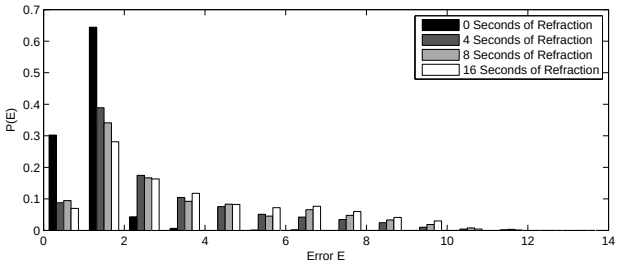

Fig. 6. Fourteen Perfect Motion Sensors: Distribution of error values.

seconds according to this model, which we then use across all tests. The locations of the intruder at each iteration x_τ is used to determine which sensors trigger. We send this set to the server, which then responds with its estimate $\hat{x}_\tau$. We determine our error E by the Euclidian distance between the estimation and the true state $|x_\tau - \hat{x}_\tau|$.

A. Baseline Naive Estimator

As a baseline for comparison, we consider the naive estimator that simply reports that an intruder is stationary in the center of the room. We compute the error values for this estimator on a simulated intruder path and compare them with those from our estimator.

First, we plot the point error values over time in Figure 3. Over the entire path, the Baseline Naive Estimator has an average error of 2.9 cells. The error is fairly well dispersed over the domain, with a slight trend of high error staying high, and low error staying low. This is because if the intruder is in a corner at one timestep, it is likely to remain near the corner in the next timestep. Next we present the distribution of error over all runs in Figure 4 and see that 55% of particles are less than 4 cells away.

B. Twenty-Two Perfect Optical Beam Sensors

Next we consider a set of binary optical beam sensors. These sensors have perfect detection, with no false positives or negatives. Their region of detection is 10 x 1 x 1 cells. We place these sensors at height (z value) 0, with one set of sensors going along the floor in the x direction for each cell index of y. We do the same in the y direction for each cell index in x, yielding a criss-crossing pattern of these sensors. Thus, if exactly one in the x set, and one in the y set trigger, we know within 1 world-space cell where the intruder is. We placed the intruder at height 0 and found the distributions illustrated in Figure 5.

This test illustrates a sanity check for a 0 second refractory period, as with perfect sensors, all estimations should be within two world-space cells. We also can see that we degrade gracefully as we increase our refractory period, and even with a 16 second refractory period, we still get 64% within two cells.

C. Fourteen Perfect Motion Sensors

We then performed a setup with 14 perfect motion sensors sensors with a characterization of η where for cells of height 0, $\eta_i = 1$ in the form of a isosceles triangle, with a resolution of a reading every two world-space cells. For all other heights, η_j = 0.

We present the sensor placement, orientation and an example run, with values shown for every 10 seconds for a perfect motion sensor with no refraction in Figure 1. This configuration generates a large number of intersections, allowing for higher quality localization. We present the distributions of error in Figure 6.

As we would expect due to the lower number of sensors, with more complicated form of intersection, this setup produces poorer quality localization of the intruder, even with no refractory period. It was interesting to note that there is little difference in localization between 4 and 8 second refractory periods. This is because there is sufficient overlap to compensate for the sensors experiencing refractory periods. However, the compensation is less robust in this simulation, as the change from a 0 to 4 second refractory period was much more significant.

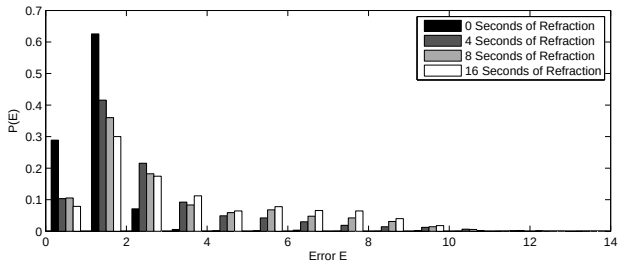

Fig. 7. Fourteen Imperfect Motion Sensors: Distribution of error values.

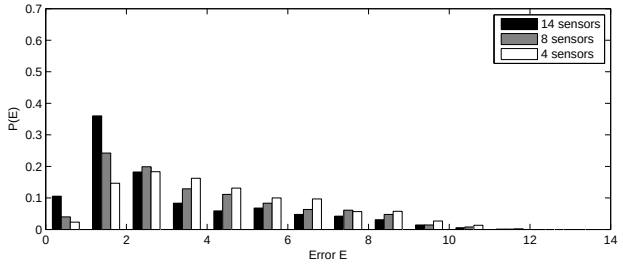

Fig. 8. Varying Number of Binary Motion Sensors: Distribution of error values for different number of binary motion sensors (each with 8 second refractory period).

D. Fourteen Imperfect Motion Sensors

Next, we modeled a sensor which becomes less sensitive as we stimulate it further away. We then ran the same tests with the new characterization that had false negatives which increased as a function of distance from the sensor, and a false negative rate $P(S_{i,\tau} = 1 | O_{1,\tau} = 0, \ldots, O_{m,\tau} = 0, B_{i,\tau} = 0) = 0.25$.

These distributions are very close to those observed with the perfect sensor over all refractory periods, though as expected the perfect sensors perform slightly better.

E. Varying Number of Sensors

By varying the number of sensors, it becomes evident that increasing the number of sensors, and thereby allowing more overlap for sensors to compensate one another, helps accommodate for the refractory period. While 4 Sensors does not do much better than our baselines, 8 sensors does better, and 14 does significantly better. We present this in Figure 8.

F. Errors over Time for Imperfect Motion Sensors

We display the error in estimated intruder position over time for the imperfect motion sensors with a 4 second refractory period in Figure 9. Note that there is high variance in the error values. High values result when all sensors are blind in the region where the intruder is. However, the estimator quickly compensates, and is able to recover.

IX. In-Lab Experiment

For the in-lab experiment, we ran tests on 8 X10 PIR sensors, which have an 8 second refraction time. The lab is 8 x 6 meters, and we use world-space cells of size .3 meters.

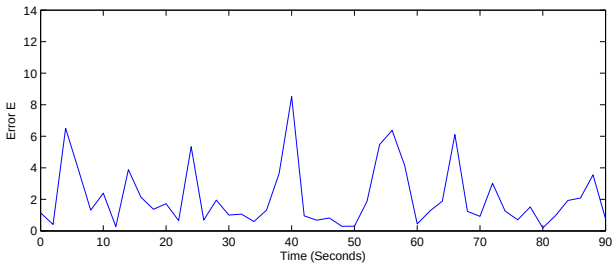

Fig. 9. Error in estimated intruder position over time for Fourteen Imperfect Motion Sensors (each with 4 second refractory period).

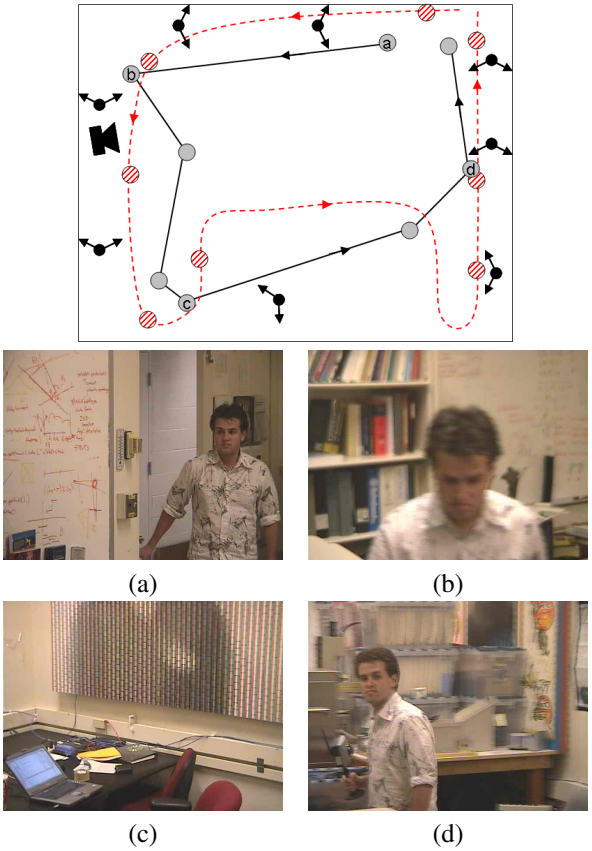

Fig. 10. Lab Experiment: A map of our lab (top), with intruder true path indicated by a dashed line as in Fig 1. Estimated path indicated by a solid line. Photos taken by a robotic camera (a-d) correspond to the numbered solid blue dots and their respective locations in the room.

Here, we present a sample of photos from our system. We used the parameters: $t_P = 2$ seconds, $v_{MAX} = 4$ feet per second and velocity Gaussian standard deviation is 1.5 feet per second. Again, we used 1000 samples of the distributions. In our test shown in Figure 10, we had an intruder walk around the room, and compare photos of the estimated path with the real path. This example shows that our system performs well in securing the lab.

X. Conclusion and Future Work

We have described a system that uses a network of binary motion sensors to track an intruder. We begin by defining probabilistic sensor models that include refractory periods.

We develop a new method for processing noisy sensor data based on Particle Filtering that incorporates a model of intruder velocity. We report experiments with this method using a new simulator and using physical experiments with X10 sensors and a robotic pan-tilt-zoom camera in our laboratory.

In the future, we will experiment with different sensor models and different spatial arrangements of sensors, and set up the camera system to run over extended duration in our lab. An interesting open problem optimal sensors placement, which can be considered a variant of the Art Gallery problem. We will also investigate methods that can simultaneously track multiple intruders. We are also interested in ways to decentralize the algorithm by moving processing onto a network of smart sensors. We would like to investigate how altering parameters, such as the number of samples, or data processing frequency affects performance. Lastly, we would like to use vision processing techniques to utilize information gathered from the camera to enhance our system.

For project updates, please visit our website: http://www.cs.berkeley.edu/∼jschiff/currentProjects.html.

XI. Acknowledgment

We thank Timothy Lee and Vladimir Giverts for assisting with system architecture and coding. We thank Gopal Kanakasabapathi, Dominic Antonelli and Alex Dimakis for their feedback and suggestions. We thank Ron Alterovitz, Lilia Gutnik, and Vijay Vasudevan for editorial input, and Prof. Deirdre Mulligan for advice on privacy issues. We thank Panasonic for its camera donation, and to our anonymous reviewers for their suggestions.

Appendix I

We assume that the triggering of all sensors is independent given the intruder's state:

$$P(S_{1,\tau}, \ldots, S_{n,\tau}|X_\tau) = \prod_{i=1}^{n} P(S_i|X_\tau)$$

We assume that the probability that an intruder's complete occupancy of a single cell causes a sensor to trigger equals the probability that the intruder only occupies that cell:

$$P(S_{i,\tau}|O_{j,\tau} = 1, B_{i,\tau} = 0) =$$
$$P(S_{i,\tau}|O_{1,\tau} = 0, \ldots, O_{j,\tau} = 1, \ldots, O_{m,\tau} = 0, B_{i,\tau} = 0)$$

We assume that the probability of the occupancy of any cell causing the sensor to trigger is independent of any other cells causing the sensor to trigger:

$$P(S_{i,\tau}|O_{1,\tau}, \ldots, O_{m,\tau}, B_{i,\tau} = 0) =$$
$$\prod_{j=1}^{m} P(S_{i,\tau}|O_{j,\tau}, B_{i,\tau} = 0)$$

We assume that during the sensor's refraction period, the distribution does not vary in the presence of any stimulus:

$$P(S_{i,\tau}|O_{1,\tau}, \ldots, O_{m,\tau}, B_{i,\tau} = 1) = P(S_{i,\tau}|B_{i,\tau} = 1)$$

We assume that the distributions of sensor's likelihood of firing given a sensor's position $P(S_{i,\tau}|O_{1,\tau}, \ldots, O_{m,\tau}, B_{i,\tau} = 0)$, smoothly changes between values over the detection region. When modeling a sensor type, we pick any sensor and view its parameters to be representative of all other sensors of the same type.

References

[1] S. P. Fekete, R. Klein, and A. Nüchter, "Online searching with an autonomous robot", in *Proc. of Sixth Workshop on Algorithmic Foundations of Robotics*, 2004, pp. 335–350.

[2] V. Isler, S. Kannan, and S. Khanna, "Locating and capturing an evader in a polygonal environment", in *Proc. of Sixth Workshop on Algorithmic Foundations of Robotics*, 2004, pp. 351–367.

[3] T. Bandyopadhyay, Y.P. Li, M.H. Ang Jr., and D. Hsu, "Stealth tracking of an unpredictable target among obstacles", in *Algorithmic Foundations of Robotics VI*, M. Erdmann et al., Eds., pp. 43–58. Springer-Verlag, 2004.

[4] S. Intille, "Tracking using a local closed-world assumption: Tracking in the football domain", 1994.

[5] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[6] A.S. Micilotta and R. Bowden, "View-based location and tracking of body parts for visual interaction.", in *Proc. of British Machine Vision Conference*, September 2004, vol. 2, pp. 849–858.

[7] D. Cochran, D. Sinno, and A. Clausen, "Source detection and localization using a multi-mode detector: A bayesian approach", in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Pheonix, March 1999.

[8] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom, "Virtual devices: An extensible architecture for bridging the physical-digital divide", Tech. Rep. UCB/CSD-05-1375, EECS Department, University of California, Berkeley, 2005.

[9] C. Shen, B. Wang, F. Vogt, S. Oldridge, and S. Fels, "Remoteeyes: A remote low-cost position sensing infrastructure for ubiquitous computing", in *Proc. of 1st International Workshop on Networked Sensing Systems*, Japan, June 2004, pp. 31–35.

[10] R.E. Kalman, "A new approach to linear filtering and prediction problems", *Transactions of the American Society of Mechanical Engineers, Journal of Basic Engineering*, pp. 35–46, March 1960.

[11] T. Lefebvre, H. Bruyninckx, and J. De Schutter, "Kalman Filters for nonlinear systems: a comparison of performance", *International Journal of Control*, vol. 77, no. 7, pp. 639–653, 2004.

[12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 1st edition, 2005.

[13] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking", *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[14] D. Song and K. Goldberg, "Sharecam part i: Interface, system architecture, and implementation of a collaboratively controlled robotic webcam", in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003.

[15] D. Song, K. Goldberg, and A. Pashkevich, "Sharecam part ii: Approximate and distributed algorithms for a collaboratively controlled robotic webcam", in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003.

[16] D. Song, Q. Hu, N. Qin, and K. Goldberg, "Automating inspection and documentation of remote building construction using a robotic camera", in *Proc. of IEEE/CASE International Conference on Automation Science and Engineering*, Edmonton, Canada, August 2005.

[17] T. Shermer, "Recent results in art galleries", Tech. Rep. 90-10, Simon Fraser University, School of Computing Science, October 1990.

[18] J. Urrutia, "Art gallery and illumination problems", in *Handbook on Computational Geometry*, J.R. Sack and J. Urrutia, Eds., pp. 973–1026. Elsevier Science, 2000.

[19] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data", in *Proceedings of the 1968 23rd ACM national conference*, New York, NY, USA, 1968, pp. 517–524, ACM Press.

[20] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education, 2nd edition, 1995.

[21] J. von Neumann, "Various techniques used in connection with random digits", *Applied Math Series*, vol. 12, pp. 36–38, 1951.