Map-based Multiple Model Tracking of a Moving Object

Cody Kwok and Dieter Fox

Department of Computer Science & Engineering
University of Washington, Seattle, WA
{ctkwok,fox}@cs.washington.edu

Abstract. In this paper we propose an approach for tracking a moving target using Rao-Blackwellised particle filters. Such filters represent posteriors over the target location by a mixture of Kalman filters, where each filter is conditioned on the discrete states of a particle filter. The discrete states represent the non-linear parts of the state estimation problem. In the context of target tracking, these are the non-linear motion of the observing platform and the different motion models for the target. Using this representation, we show how to reason about physical interactions between the observing platform and the tracked object, as well as between the tracked object and the environment. The approach is implemented on a four-legged AIBO robot and tested in the context of ball tracking in the RoboCup domain.

1 Introduction

As mobile robots become more reliable in navigation tasks, the ability to interact with their environment becomes more and more important. Estimating and predicting the locations of objects in the robot's vicinity is the basis for interacting with them. For example, grasping an object requires accurate knowledge of the object's location relative to the robot; detecting and predicting the locations of people helps a robot to better interact with them. The problem of tracking moving objects has received considerable attention in the mobile robotics and the target tracking communities [1, 2, 10, 13, 6, 8]. The difficulty of the tracking problem depends on a number of factors, ranging from how accurately the robot can estimate its own motion, to the predictability of the object's motion, to the accuracy of the sensors being used.

This paper focuses on the problem of tracking and predicting the location of a ball with a four-legged AIBO robot in the RoboCup domain, which aims at playing soccer with teams of mobile robots. This domain poses highly challenging target tracking problems due to the dynamics of the soccer game, coupled with the interaction between the robots and the ball. We are faced with a difficult combination of issues:

– **Highly non-linear motion of the observer:** Due to slippage and the nature of legged motion, information about the robot's own motion is extremely noisy, blurring the distinction between motion of the ball and the robot's ego-motion. The rotation of these legged robots, in particular, introduces non-linearities in the ball tracking problem.

- **Physical interaction between target and environment:** The ball frequently bounces off the borders of the field or gets kicked by other robots. Such interactions result in highly non-linear motion of the ball.
- **Physical interaction between observer and target:** The observing robot grabs and kicks the ball. In such situations, the motion of the ball is tightly connected to the motion or action of the robot. This interaction is best modeled by a unified ball tracking framework rather than handled as a special case, as is done typically.
- **Inaccurate sensing and limited processing power**: The AIBO robot is equipped with a 176×144 CMOS camera placed in the robot's "snout". The low resolution provides inaccurate distance measurements for the ball. Furthermore, the robot's limited processing power (400MHz MIPS) poses computational constraints on the tracking problem and requires an efficient solution.

In this paper, we introduce an approach that addresses these challenges in a unified Bayesian framework. The technique uses Rao-Blackwellised particle filters (RBPF) [4] to jointly estimate the robot location, the ball location, its velocity, and its interaction with the environment. Our technique combines the efficiency of Kalman filters with the representational power of particle filters. The key idea of this approach is to sample the non-linear parts of the state estimation problem (robot motion and ball-environment interaction). Conditioning on these samples allows us to apply efficient Kalman filtering to track the ball. Experiments both in simulation and on an AIBO robot show that this approach is efficient and yields highly robust estimates of the ball's location and motion.

This paper is organized as follows. After discussing related work in the next section, we will introduce the application of Rao-Blackwellised particle filters to ball tracking in the RoboCup domain. Then we will discuss an efficient approximation to the RBPF technique. Experimental results are presented in Section 4, followed by conclusions.

2 Related Work

Tracking moving targets has received considerable attention in the robotics and target tracking communities. Kalman filters and variants thereof have been shown to be well suited for this task even when the target motion and the observations violate the linearity assumptions underlying these filters [2]. Kalman filters estimate posteriors over the state by their first and second moments only, which makes them extremely efficient and therefore a commonly used ball tracking algorithm in RoboCup [12].

In the context of maneuvering targets, multiple model Kalman filters have been shown to be superior to the vanilla, single model filter. Approaches such as the Interacting Multiple Model (IMM) and the Generalized Pseudo Bayesian (GPB) filter represent the target location using a bank of Kalman filters, each conditioned on different potential motion models for the target. An exponential explosion of the number of Gaussian hypotheses is avoided by merging hypotheses after each update of the filters [2]. This merging is done by collapsing all Gaussians belonging to the same discrete motion model into a single Gaussian estimate. While the resulting GPB and IMM approaches are efficient, the model merging step assumes that the state conditioned on each discrete motion model is unimodal. However, the uncertainties in our target tracking problem frequently produce multimodal distributions, even when conditioned on a single motion model. This is often caused by uncertainty in the relative location of field borders.

Particle filters provide a viable alternative to Kalman filter based approaches [5]. These filters represent posteriors by samples, which allows them to adequately represent and estimate non-linear, non-Gaussian processes. Recently, particle filters have been applied successfully to people tracking using mobile robots equipped with laser range-finders [13, 10]. While the sample-based representation gives particle filters their robustness, it comes at the cost of increased computational complexity, making them inefficient for higher-dimensional estimation problems.

As we will describe in Section 3.2, Rao-Blackwellised particle filters (RBPF) [4] combine the representational benefits of particle filters with the efficiency and accuracy of Kalman filters. This technique has been shown to outperform approaches such as the IMM filter on various target tracking problems [6, 8]. Compared to our method, existing applications of RBPFs consider less complex dynamic systems where only one part of the state space is non-linear. Our approach, in contrast, estimates a system where several components are highly non-linear (observer motion, target motion). Furthermore, our technique goes beyond existing methods by incorporating information about the environment into the estimation process. The use of map information for improved target tracking has also been proposed by [10]. However, their tracking application is far less demanding and relies on a vanilla particle filter to estimate the joint state space of the observer and the target. Our RBPF technique is more efficient since it relies on Kalman filters to estimate the target location.

3 Rao-Blackwellised Particle Filters for Multi Model Tracking with Physical Interaction

In this section, we will first describe the different interactions between the ball and the environment. Then we will show how RBPFs can be used to estimate posteriors over the robot and ball locations. Finally, we will present an approximation to this idea that is efficient enough to run onboard the AIBO robots at a rate of 20 frames per second.

3.1 Ball-Environment Interactions

Fig. 1(a) describes the different interactions between the ball and the environment:

- `None`: The ball is either not moving or in an unobstructed, straight motion. In this state, a linear Kalman filter can be used to accurately track its location and velocity.
- `Grabbed`: The ball is between the robot's legs or grabbed by them. The ball's position is thus tightly coupled with the location of the robot. This state is entered when the ball is in the correct position relative to the robot. It typically exits into the `Kicked` state.
- `Kicked`: The robot just kicked the ball. This interaction is only possible if the ball was grabbed. The direction and velocity of the following ball motion depend on the type of kick. There is also a small chance that the kick failed and the ball remained in the `Grabbed` state.
- `Bounced`: The ball bounced off one of the field borders or one of the robots on the field. In this case, the motion vector of the ball is assumed to be reflected by the object with a considerable amount of orientation noise and velocity reduction.

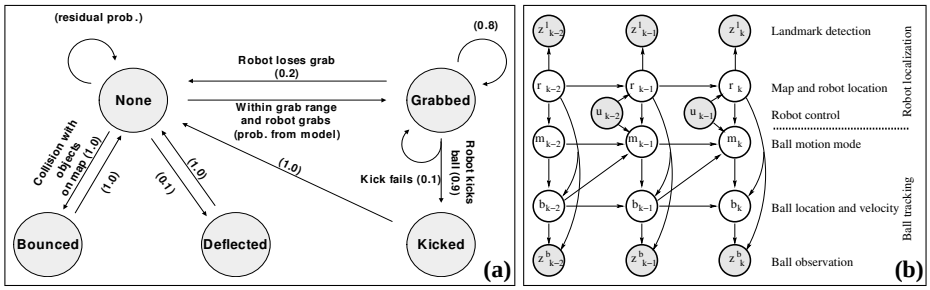

Fig. 1: (a) Finite state machine describing the transitions between different ball motion models. All transitions are probabilistic, the probabilities are enclosed in parentheses. (b) Graphical model for tracking a moving ball. The nodes in this graph represent the different parts of the dynamic system process at consecutive points in time, and the edges represent dependencies between the individual parts of the state space. Filled circles indicate observed nodes, where z_k^l are landmark and z_k^b ball observations.

– `Deflected`: The ball's trajectory has suddenly changed, most likely kicked by another robot. In this state, the velocity and motion direction of the ball are unknown and have to be initialized by integrating a few observations.

The transition probabilities between the states are parenthesized in the figure. Their values were tuned manually. From the `None` state, we assume that there is a 0.1 probability that the ball will be deflected at each update (*e.g.*, by an undetected robot). When the ball is close in front of the robot, the ball will enter the `Grabbed` state with probability defined by a two-dimensional linear probability function. When the ball collides with the borders or other robots, it will always reflect and move into the `Bounced` state. This transition is certain because each ball estimate is conditioned on a sampled robot position, which can be assumed to be the true robot position. This will be elaborated further in the next section. Finally, the `None` state transitions back to itself by default, hence it takes up the residual probability after computing the previously mentioned transitions. For the states `Kicked` and `Grabbed`, the transitions are associated with whether these actions succeed or not. Kicking the ball has a success rate of 0.9 and Grabbing 0.8. The `Bounced` and `Deflected` states are used to initiate changes in the Kalman filters. Once the changes are made, they transition immediately to the normal updates in the `None` state. Thus, most of the time, the ball is either in free motion (`None`) or grabbed by the robot (`Grabbed`). The other states are only valid for very short periods of time (our current system does not model that the ball can be grabbed by other robots).

While most of these interactions depend on the location of the ball relative to the robot, the ball's interactions with the environment (*e.g.* the field borders) strongly depend on the ball location on the field, *i.e.* in global coordinates. For instance, when the ball collides with a border, the predicted motion after collision depends on the global incident angle. In order to estimate global coordinates from relative observations, we need to associate relative ball positions with the robot's location and orientation on the field. Hence, the problem of tracking a ball requires the joint estimation of the ball location, the robot location, and the ball-environment interaction.

3.2 Rao-Blackwellised Posterior Estimation

Let $\langle m_k, b_k, r_k \rangle$ denote the state of the system at time k. Here, $m_k = \{\texttt{None}, \texttt{Grabbed}, \texttt{Kicked}, \texttt{Bounced}, \texttt{Deflected}\}$ are the different ball motion models resulting from the interactions between the ball and the environment. $b_k = \langle x_b, y_b, \dot{x_b}, \dot{y_b} \rangle$ denotes the ball location and velocity in global coordinates, and $r_k = \langle x_r, y_r, \theta_r \rangle$ is the robot location and orientation on the field. Furthermore, z_k are observations of the ball and landmarks, provided in relative bearing and distance.

A graphical model description of the ball tracking problem is given in Fig. 1(b). The graphical model describes how the joint posterior over $\langle b_k, m_k, r_k \rangle$ can be computed efficiently using independencies between parts of the state space. The nodes describe different random variables and the arrows indicate dependencies between these variables. The model shows that the robot location at time k, r_k, only depends on the previous location r_{k-1} and the robot motion control u_{k-1}. Landmark observations z_k^l only depend on the current robot location r_k. This is exactly as in standard robot localization [7, 9]. The location and velocity of the ball, b_k, typically depend on the previous ball state b_{k-1} and the current ball motion model m_k. The arc from m_k to b_k describes, for example, that the ball prediction depends on whether it was kicked or bounced off a field border. If $m_k = \texttt{Grabbed}$, then the ball location depends on the current robot location r_k, as indicated by the arrow from r_k to b_k. Relative ball observations, z_k^b, only depend on the current ball and robot positions. See Fig. 1(a) for the ball motion models.

Now that the dependencies between different parts of the state space are defined, we can address the problem of filtering, which aims at computing the posterior over $\langle b_k, m_k, r_k \rangle$ conditioned on all observations made so far. A full derivation of the Rao-Blackwellised particle filter (RBPF) algorithm is beyond the scope of this paper; see [4] for a thorough discussion of the generic RBPF. Just like regular particle filters, RBPFs represent posteriors by sets of weighted samples, or particles:

$$S_k = \{s_k^{(i)}, w_k^{(i)} \mid 1 \leq i \leq N\}$$

In our case, each particle $s_k^{(i)} = \langle b_k^{(i)}, m_k^{(i)}, r_k^{(i)} \rangle$, where $b_k^{(i)}$ are the mean and covariance of the ball location and velocity, $m_k^{(i)}$ is the ball motion model, and $r_k^{(i)}$ is the robot location [1]. The key idea of RBPFs is to condition the ball estimate $b_k^{(i)}$ on a particle's ball motion model $m_k^{(i)}$ and robot location $r_k^{(i)}$. This conditioning turns the ball location and velocity into a linear system that can be estimated efficiently using a Kalman filter. To see how this works, we factorize the posterior as follows:

$$\begin{aligned} p(b_k, m_k, r_k \mid z_{1:k}, u_{1:k-1}) = p(b_k \mid m_k, r_k, z_{1:k}, u_{1:k-1}) \cdot \\ p(m_k \mid r_k, z_{1:k}, u_{1:k-1})\, p(r_k \mid z_{1:k}, u_{1:k-1}) \end{aligned} \quad (1)$$

The task of filtering is to generate samples distributed according to (1) based on samples drawn from the previous posterior represented by the sample set S_{k-1}. To do so,

[1] An accurate derivation of the recursive RBPF algorithm would require reasoning about robot *trajectories* $r_{1:k}^{(i)}$ and ball motion model *histories* $m_{1:k}^{(i)}$. However, for the sake of brevity, we ignore this rather technical detail since it has no impact on the resulting algorithms.

the Rao-Blackwellisation technique first draws a sample $s_{k-1}^{(i)}$ from S_{k-1}. Then, conditioned on $s_{k-1}^{(i)}$, a new sample $s_k^{(i)}$ is generated stepwise by simulating (1) from right to left. It can be shown that this is achieved by the following steps (see [4] for more background). First, a new robot location, $r_k^{(i)}$, is sampled according to

$$r_k^{(i)} \sim p(r_k \mid r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}), \tag{2}$$

where $r_{k-1}^{(i)}$, $m_{k-1}^{(i)}$, and $b_{k-1}^{(i)}$ are provided by $s_{k-1}^{(i)}$. After this step, $s_k^{(i)} = \langle \_, \_, r_k^{(i)} \rangle$, where $\_$ denotes uninitialized value. Then, the new ball motion model, $m_k^{(i)}$, is sampled according to

$$m_k^{(i)} \sim p(m_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}), \tag{3}$$

where $r_k^{(i)}$ is the robot location generated in the first sampling step. This gives us $s_k^{(i)} = \langle \_, m_k^{(i)}, r_k^{(i)} \rangle$. Finally, the sample's ball location and velocity, $b_k^{(i)}$, are estimated:

$$b_k^{(i)} \sim p(b_k \mid r_k^{(i)}, m_k^{(i)}, b_{k-1}^{(i)}, z_k) \tag{4}$$

To efficiently estimate $b_k^{(i)}$, we make use of the fact that we already sampled the ball motion model and the robot location. By conditioning on $m_k^{(i)}$ and $r_k^{(i)}$, the ball estimation $b_k^{(i)}$ can be performed analytically using a Kalman filter; one for each particle.

Equations (2)–(4) follow directly from the three terms in (1) and the independences of the graphical model in Fig. 1(b). Furthermore, we make use of the fact that the previous sample $s_{k-1}^{(i)}$ is a sufficient statistic for all information prior to $k-1$. It remains to be shown how to generate samples distributed according to (2) and (3). Since it is not possible to sample exactly from these distributions, we apply a technique called importance sampling, where samples are weighted so as to compensate for the mismatch between the sampling distributions and the target distributions (2) and (3). The importance weight of a complete sample is then given by the product of the importance weights resulting from (2) and (3). We will now take a closer look at these distributions.

Robot Locations The target distribution (2) for the robot location can be transformed as follows:

$$\begin{aligned} &p(r_k \mid r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \\ &\quad \propto p(z_k \mid r_k, r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})\; p(r_k \mid r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) \quad (5) \\ &\quad = p(z_k \mid r_k, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})\; p(r_k \mid r_{k-1}^{(i)}, u_{k-1}) \quad (6) \end{aligned}$$

Here, (5) follows by Bayes rule, and (6) from the independencies represented in the graphical model in Fig. 1(b). To generate particles according to (6), we apply the standard update routine of particle filters for robot localization [5, 7]. More specifically, we first predict the next robot location $r_k^{(i)}$ using the previous location $r_{k-1}^{(i)}$ along with the most recent control information u_{k-1} and the robot motion model $p(r_k | r_{k-1}^{(i)}, u_{k-1})$ (right term in (6)). To represent a random sample from (6), this particle has to be (importance-) weighted by the left term, which is the likelihood of the measurement z_k. If z_k is a landmark detection, z_k^l, then this weight is given by

$$w_k^{(i)} \propto p(z_k^l \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) = p(z_k^l \mid r_k^{(i)}), \tag{7}$$

which corresponds exactly to the particle filter update for robot localization [9, 7]. If, however, z_k is a ball detection, z_k^b, then the weight must be computed as follows.

$$\begin{aligned} w_k^{(i)} &\propto p(z_k^b \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) \\ &= \sum_{M_k} p(z_k^b \mid M_k, r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})\; p(M_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) \quad (8) \\ &= \sum_{M_k} p(z_k^b \mid M_k, r_k^{(i)}, b_{k-1}^{(i)})\; p(M_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}), \quad (9) \end{aligned}$$

Here M_k ranges over all possible ball motion models, and (8) follows from the law of total probability. Thus, to compute the weight of a sampled robot location, we have to sum over all ball motion models. Each summand is given by the likelihood of the ball detection times the probability of the model. The value of the likelihood term follows from an update of the Kalman filter associated with the ball estimate $b_{k-1}^{(i)}$. This update is conditioned on the robot location $r_k^{(i)}$ and ball motion model M_k. The right term in (9) describes the probability of the ball motion model based on the previous model $m_{k-1}^{(i)}$, the current robot location $r_k^{(i)}$, the previous ball location and velocity $b_{k-1}^{(i)}$, and the robot control action u_{k-1}. This probability is based on reasoning about the different ball-environment interactions, as described in Section 3.1.

Ball Motion Models By applying Bayes rule and the independencies in the estimation problem, the distribution over ball motion models (3) can be transformed to

$$\begin{aligned} &p(m_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \\ &\qquad \propto p(z_k \mid m_k, r_k^{(i)}, b_{k-1}^{(i)})\; p(m_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}). \quad (10) \end{aligned}$$

To generate a sample $m_k^{(i)}$ from this distribution, we have to distinguish between landmark and ball detections. If z_k is a landmark detection, z_k^l, then we simply sample $m_k^{(i)}$ from the right term in (10). The corresponding importance weight is then given by the left term, the landmark observation likelihood $p(z_k^l \mid m_k^{(i)}, r_k^{(i)}, b_{k-1}^{(i)})$. This likelihood can be simplified to $p(z_k^l \mid r_k^{(i)})$, since the landmark observation only depends on the robot location. Thus, since the importance weight is the same for all $m_k^{(i)}$, it can be ignored and subsumed into a normalization constant.

If z_k is a ball detection, z_k^b, then (10) is identical to one of the summands in (9) (the one for which $m_k^{(i)} = M_k$). Since all ball motion models have to be generated and evaluated in order to get the importance weight (9) for $r_k^{(i)}$, we can re-use exactly these models to generate samples from (10). The importance weights of each sample $m_k^{(i)}$ is then identical to (10), since it is drawn uniformly among the models.

Ball Estimate To finalize the generation of a sample from the posterior (1), we need to determine the leftmost term of the factorization. As mentioned above, since we already sampled $r_k^{(i)}$ and $m_k^{(i)}$, the ball posterior of the particle can be computed analytically using a Kalman filter [2]. The Kalman filter prediction uses the previous ball state $b_{k-1}^{(i)}$ and the ball motion model $m_k^{(i)}$. The correction step is based on the robot location $r_k^{(i)}$

1.	**Inputs:**	
	$S_{k-1} = \{\langle b_{k-1}^{(i)}, m_{k-1}^{(i)}, r_{k-1}^{(i)}\rangle \mid i = 1, \ldots, N\}$	posterior at time $k-1$
	u_{k-1}	control measurement
	z_k	observation
2.	$S_k := \emptyset$	***// Initialize***
3.	**for** $i := 1, \ldots, N$ **do**	***// Generate samples***
4.	Sample $r_k^{(i)} \sim p(r_k \mid r_{k-1}^{(i)}, u_{k-1})$	*// Predict robot position, right term in (6)*
5.	**if** z_k is a landmark detection **do**	***// Integrate landmark observation***
6.	Sample $m_k^{(i)} \sim p(m_k \mid m_{k-1}^{(i)}, r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1})$	*// Predict motion model, see (10)*
7.	$b_k^{(i)} :=$ Kalman_prediction($r_k^{(i)}$, $m_k^{(i)}$, $b_{k-1}^{(i)}$)	*// Update ball estimate, see (4)*
8.	$S_k := S_k \cup \{\langle b_k^{(i)}, m_k^{(i)}, r_k^{(i)}\rangle\}$	*// Insert sample into sample set*
9.	$w_k^{(i)} := p(z_k \mid r_k^{(i)})$	*// Importance weight, see (7)*
10.	**else if** z_k is a ball detection **do**	***// Integrate ball observation***
11.	$w_k^{(i)} := 0$	
12.	**for each** model M_k^j **do**	*// Update Kalman filter for all ball models*
13.	$b_k^{(i,j)} :=$ Kalman_update($r_k^{(i)}$, M_k^j, $b_{k-1}^{(i)}$, z_k)	*// Update ball estimate, see (4)*
14.	$w_k^{(i,j)} := p(z_k \mid M_k^j, r_k^{(i)}, b_{k-1}^{(i)})\; p(M_k^j \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})$	*// See (10)*
15.	$w_k^{(i)} := w_k^{(i)} + w_k^{(i,j)}$	*// Sum up weights, see (9)*
16.	**for each** model M_k^j **do**	
17.	$w_k^{(i,j)} := w_k^{(i,j)} \times w_k^{(i)}$	*// Multiply weights from (9), (10)*
18.	$S_k := S_k \cup \{\langle b_k^{(i,j)}, M_k^j, r_k^{(i)}\rangle\}$	*// Insert $s_k^{(i,j)}$ into sample set*
19.	Multiply / discard samples in S_k based on normalized weights w_k such that $\lvert S\rvert = N$	
20.	**return** S_k	

Table 1: RBPF algorithm for joint robot-ball estimation.

along with the most recent ball observation z_k. The innovation of the correction step provides the observation likelihood needed in (9) and (10). Since landmark detections provide no information about the ball location, the Kalman correction step is not performed if z_k is a landmark detection.

RBPF Algorithm An algorithm for implementing our RBPF tracking is summarized in Table 1. The input to the algorithm is the previous sample set along with the most recent observation and control information. The algorithm generates N new robot locations (line 4). If the observation is a landmark detection, then the ball estimate belonging to each robot location is updated by sampling a new ball motion model (line 6) followed by a Kalman filter prediction step (line 7). The importance weight of each sample is given by the likelihood of the landmark detection, conditioned on the robot location (line 9). Thus, for landmark detections, each particle is updated exactly as in regular robot localization [7, 9], plus the sampling of a ball motion model followed by a Kalman prediction of the ball estimate. The weighted samples are resampled in line 19, thereby generating a sample set with uniformly weighted samples.

The update is slightly more complicated for ball observations. The importance weight of a generated robot location, $r_k^{(i)}$, is based on the summation given in (9). If there are M ball motion models, then the algorithm generates M models for each robot location, resulting in a total of $N \times M$ particles. Each particle performs a Kalman filter update using its robot location $r_k^{(i)}$ and ball motion model M_k^j (line 13). The importance weight, $w_k^{(i,j)}$, of each such particle follows from multiplication of the values given in (9) and (10). The first term is computed in line 14, and the summation is performed via lines 15 and 17. Finally, after generating $N \times M$ new particles, the number of samples is reduced to N by resampling in line 19.

3.3 Efficient Approximation

We implemented the RBPF algorithm and it worked very well on data collected by an AIBO robot. Unfortunately, the approach requires on the order of 300–500 particles, which is computationally too demanding for the AIBO robots, especially since each particle has a Kalman filter attached to it. The reason for this high number of samples is that for each robot location, we need to estimate multiple ball motion models with associated Kalman filters. This is necessary since the ball and robot estimates are coupled, with information flowing from the ball estimate to the robot location estimate, and *vice versa*. For example, the robot location of a particle influences the ball estimate, since it determines when the ball bounces into a border. On the other hand, ball estimates can influence the robot's localization, since a sample can be removed by resampling if its ball estimate is out-of-bounds. While this mutual influence is desired, it has the negative effect of increasing the number of required samples.

The requirement for large sample sets is dramatically reinforced by the following fact. In order to track the fast moving ball, its estimates are updated about 20 times per second. The robot location, on the other hand, is updated much less frequently, about twice per second. As a consequence, around 10 ball observations are integrated into the sample set for each landmark detection that is being integrated. The frequent ball updates result in importance weights that strongly outweigh the contribution of the relatively rare landmark detections. Resampling based on such weights results in the deletion of many valid robot location hypotheses even if the robot is not seeing any landmarks. This is contrary to the fact that the ball location provides much less information about the robot location than the other way around. To overcome this source of inefficiency, we ignore the information provided by ball detections on the robot's location. In the resulting approximation, information flows from robot locations to ball estimates, but not in the other direction. To do so, we partition the state space into robot and ball positions, which are then updated separately. While the robot location is estimated using a standard particle for robot localization [7, 9], the ball estimates are "re-attached" to updated robot locations whenever a landmark is observed.

The approximation algorithm is presented in Table 2. As can be seen, the set of samples S_k is now a pair $\langle R_k, B_k \rangle$, where R_k is the set of robot positions and B_k is the set of ball estimates. Each ball sample consists of the ball's position and velocity, the motion model it is conditioned on, and a robot position, $\rho_k^{(i)}$, used to estimate the motion model. Lines 3–9 update the ball estimates contained in B_k. Like the original algorithm, they implement the RBPF factorization (1) from right to left by first sampling

1.	**Inputs:**	
	$S_{k-1} = \langle R_{k-1}, B_{k-1} \rangle$	posterior at time $k-1$
	$R_{k-1} = \{r_{k-1}^{(i)} \mid i = 1, \ldots, N\}$	robot positions
	$B_{k-1} = \{\langle b_{k-1}^{(i)}, m_{k-1}^{(i)}, \rho_{k-1}^{(i)} \rangle \mid i = 1, \ldots, L\}$	ball estimates
	u_{k-1}	control measurement
	z_k	observation
2.	$R_k := \emptyset, B_k := \emptyset$	***// Initialize***
3.	**for** $i := 1, \ldots, L$ **do**	***// Update ball samples***
4.	Sample $\rho_k^{(i)} \sim p(\rho_k \mid \rho_{k-1}^{(i)}, u_{k-1})$	*// Predict robot position, see (6)*
5.	Sample $m_k^{(i)} \sim p(m_k \mid m_{k-1}^{(i)}, \rho_k^{(i)}, b_{k-1}^{(i)}, u_{k-1})$	*// Predict motion model, see (10)*
6.	$b_k^{(i)} :=$ Kalman_update($\rho_k^{(i)}$, $m_k^{(i)}$, $b_{k-1}^{(i)}$, z_k)	*// Update ball location, see (4)*
7.	$B_k := B_k \cup \{\langle b_k^{(i)}, m_k^{(i)}, \rho_k^{(i)} \rangle\}$	*// Insert sample into sample set*
8.	$w_k^{(i)} := p(z_k \mid m_k^{(i)}, \rho_k^{(i)}, b_{k-1}^{(i)})$	*// Compute importance weight, see (10)*
9.	Multiply / discard samples in B_k using weights w_k	***// Resample ball samples***
10.	**for** $i := 1, \ldots, N$ **do**	***// Predict robot positions in*** R_k
11.	Sample $r_k^{(i)} \sim p(r_k \mid r_{k-1}^{(i)}, u_{k-1})$ and insert into R_k	*// See (6)*
12.	**if** z_k is a landmark detection **do**	***// Integrate landmark observation***
13.	**for** $i := 1, \ldots, N$ **do**	
14.	$w_k^{(i)} := p(z_k \mid r_k^{(i)})$	*// Compute importance weight, see (7)*
15.	Multiply / discard samples in R_k	*// Resample robot samples*
16.	**for** $i := 1, \ldots, L$ **do**	*// Update mixture of ball-robot samples*
17.	Re-attach $\langle b_k^{(i)}, m_k^{(i)}, \rho_k^{(i)} \rangle$ to a robot position drawn randomly from R_k	
18.	**return** $S_k = \langle R_k, B_k \rangle$	

Table 2: Efficient approximation to the Rao-Blackwellised particle filter algorithm.

a new robot location $\rho_k^{(i)}$, followed by sampling a ball motion model $m_k^{(i)}$ conditioned on $\rho_k^{(i)}$, followed by a Kalman update of the ball estimate, conditioned on both $\rho_k^{(i)}$ and $m_k^{(i)}$ (lines 4–6). If z_k is a landmark observation, then the Kalman update consists of a prediction step only. The resulting importance weight of the sample is given by the likelihood of the observation, which is computed as part of the Kalman filter update (uniform likelihood for landmark detections). Note that these steps generate weighted samples distributed almost exactly according to (1). The only approximation is in ignoring the contribution from landmark detections and the weights according to (9). However, these weights model the impact of ball observations on robot locations, which is what we want to ignore anyhow. The weighted ball samples are resampled in line 9.

Lines 10–17 mostly update the set R_k, which represents the main estimate of the robot's location. In essence, R_k ignores ball detections and incorporates landmark observations in the same way as a regular particle filter for robot localization [7, 9]. Line 11 predicts the robot locations stored in R_k. If z_k is a landmark observation, then the importance weights of the samples are given by the observation likelihood (line 14). The robot samples are then resampled in line 15.

The key trick of the algorithm lies in line 17. Before we explain this step, let us briefly recap. The algorithm performs regular robot localization with R_k, whose updates ignore ball detections (lines 10–17). The ball sample sets B_k are updated based on ball detections only (lines 3–9). Ball-environment interactions are determined using robot locations $\rho_k^{(i)}$ of each ball sample, which are updated only according to the robot's motion (line 4). However, since these $\rho_k^{(i)}$ do not consider landmark detections, their locations become more and more inaccurate. Furthermore, due to resampling of B_k (line 9), the diversity of these robot locations decreases over time (this is the main reason for our approximation algorithm). Both problems are overcome by the step in line 17: Whenever the robot location sample set R_k is updated with a landmark detection, the ball estimates make use of this refined estimate by re-attaching the ball samples to the new robot locations. To do so, each ball sample $\langle b_k^{(i)}, m_k^{(i)}, \rho_k^{(i)} \rangle$ replaces its robot location $\rho_k^{(i)}$ by a location $r_k^{(i)}$, randomly drawn from R_k. The global ball location $b_k^{(i)}$ is then updated (shifted) to assume the same relative location w.r.t. $r_k^{(i)}$ as it had w.r.t $\rho_k^{(i)}$. This way, the ball estimates frequently (about once every second) inherit both the accuracy and the diversity of the robot localization samples. Such an idea of efficiently estimating a complex system by frequently decoupling and re-coupling parts of the state space has already been applied successfully in dynamic systems [3, 11].

In practice, the ball estimates generated by the algorithm are virtually identical to the original RBPF algorithm. The approximation also has only negligible impact on the robot location estimates, since the noisy ball observations provide only minor information about the robot's location. As a benefit, the decoupling of robot and ball samples significantly reduces the number of samples needed for successful tracking (using 50 robot and 20 ball particles, respectively).

3.4 Tracking and Finding the Ball

Since our approach estimates the ball location using multiple Kalman filters, it is not straightforward to determine the direction the robot should point its camera in order to track the ball. Typically, if the robot sees the ball, the ball estimates are tightly focused on one spot and the robot can track it by simply pointing the camera at the mean of the ball samples with the most likely motion model However, if the robot doesn't see the ball for a period of time, the distribution of ball samples can get highly uncertain and multi-modal. This can happen, for instance, after the ball is kicked out-of-sight by the robot or by other robots.

To efficiently find the ball in such situations, we use a grid-based representation to describe where the robot should be looking. The grid has two dimensions, the pan and the tilt of the camera. Each ball sample is mapped to these camera parameters using inverse kinematics, and is put into the corresponding grid cells. Each cell is weighted by the sum of the importance weights of the ball samples inside the cell. To find the ball, the robot moves its head to the camera position specified by the highest weighted cell. In order to represent all possible ball locations, the pan range of the grid covers $360°$. Cells with pan orientation exceeding the robot's physical pan range indicate that the robot has to rotate its body first.

An important aspect of our approach is that it enables the robot to make use of *negative information* when looking for the ball: Ball samples that are not detected even

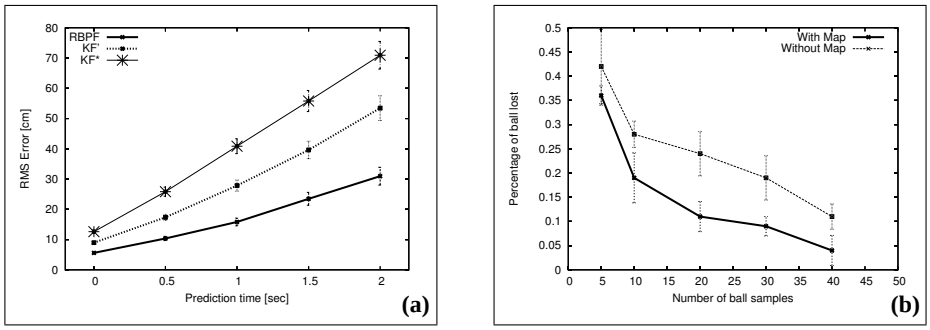

Fig. 2: (a) RMS error of the ball's position for different prediction times. (b) Percentage of time the robot loses track of the ball after a kick for different numbers of particles with and without map information.

though they are in the visible range of the camera get low importance weights (visibility considers occlusions by other robots). In the following resampling step, these ball samples are very unlikely to be drawn from the weighted sample set, thereby focusing the search on other areas. As a result, the robot scans the whole area of potential ball locations, pointing the camera at the most promising areas first. When none of the ball particles are detected, the ball is declared lost and the ball samples are re-initialized at the next ball detection.

4 Experiments

We evaluated the effectiveness of our tracking system in both simulated and real-world environments. We first illustrate the basic properties of our algorithm by comparing it with the traditional Kalman Filter. Then we evaluate how well the approach works on the real robot.

4.1 Simulation Experiments

In the RoboCup domain, robots often cannot directly observe the ball, due to reasons such as looking at landmarks for localization, or the ball being occluded by another robot. The goalkeeper robot in particular has to accurately predict the trajectory of the ball in order to block it. Hence, accurate *prediction over multiple camera frames* is of utmost importance. To systematically evaluate the prediction quality of our multiple model approach, we simulated a robot placed at a fixed location on the soccer field, while the ball is kicked randomly at different times. The simulator generates noisy observations of the ball. The observation noise is proportional to the distance from the robot and constant in the orientation, similar in magnitude to the real robot. Prediction quality is measured using the RMS error at the predicted locations.

In this experiment, we measure the prediction quality for a given amount of time in the future, which we call the *prediction time*. Map information is not used, and the ball is estimated with 20 particles (used to sample ball motion models at each iteration). The observation noise of the Kalman filters was set according to the simulated noise. To determine the appropriate Kalman prediction noise, we generated straight ball trajectories and used the prediction noise value that minimized the RMS error for these runs. This prediction noise was used by our multiple model approach when the motion

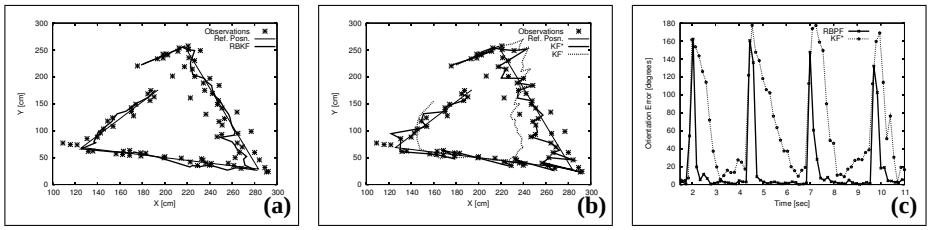

Fig. 3: Tracking of a ball trajectory with multiple kicks, the observer is located on the left. In (a) and (b), the observations are indicated by stars and the true target trajectory is shown as a thin line. (a) shows the estimated trajectory of our multiple-model approach. (b) shows the estimates using an extended Kalman filter for two different prediction noise settings. The dotted line represents the estimates when using prediction noise that assumes a linear ball trajectory, and the thick, solid line is estimated using inflated prediction noise. (c) Orientation errors over a time period including four kicks. Solid line RBPF, dashed line Kalman filter with inflated prediction noise.

model was `None` (see Section 3.1). The results for prediction times up to 2 seconds are shown in Fig. 2(a). In addition to our RBPF approach (thick, solid line), we compare it with single Kalman filters with different prediction noise models. The thin, solid line shows the RMS error when using a single Kalman filter with prediction noise of the straight line model (denoted KF^*). However, since the ball is not always in a straight line motion, the quality of the filter estimates can be improved by inflating the prediction noise. We tried several noise inflation values and the dotted line in Fig. 2(a) gives the results for the best such value (denoted KF'). Not surprisingly, our multiple model approach greatly improves the prediction quality.

The reason for this improved prediction performance is illustrated in Fig. 3. Our approach, shown in Fig. 3(a), is able to accurately track the ball location even after a kick, which is due to the fact that the particle filter accurately "guesses" the kick at the correct location. The Kalman filter with the straight line motion model quickly diverges, as shown by the dotted line in Fig. 3(b). The inflated prediction noise model (thick, solid line) keeps track of the ball, but the trajectory obviously overfits the observation noise. Further intuition can be gained from Fig. 3(c). It compares the orientation error of the estimated ball velocity using our approach versus the inflated Kalman filter KF^*(KF' shows a much worse performance; for clarity it is omitted from the graph). Clearly, our approach recovers from large errors due to kicks much faster, and it converges to a significantly lower error even during straight line motion.

4.2 Real-world Experiments

In this section we describe an experiment carried out on the real robot. It demonstrates that the use of map information brings significant improvements to the tracking performance. In the experiment, an AIBO robot and a ball are placed on the soccer field at random locations. The task of the robot is to track the ball and kick it as soon as it reaches it. The kick is a sideway head kick as shown in Fig. 4(c). The robot is not able to see the ball until it recovers from the kick motion. During the experiment, the robot stays localized by periodically scanning the markers on the field. Fig. 4 illustrates the evolution of the particles during a kick. As can be seen in Fig. 4(c) and (d), each ball particle $\langle b_k^{(i)}, m_k^{(i)}, \rho_k^{(i)} \rangle$ uses a different location for the border extracted from the robot

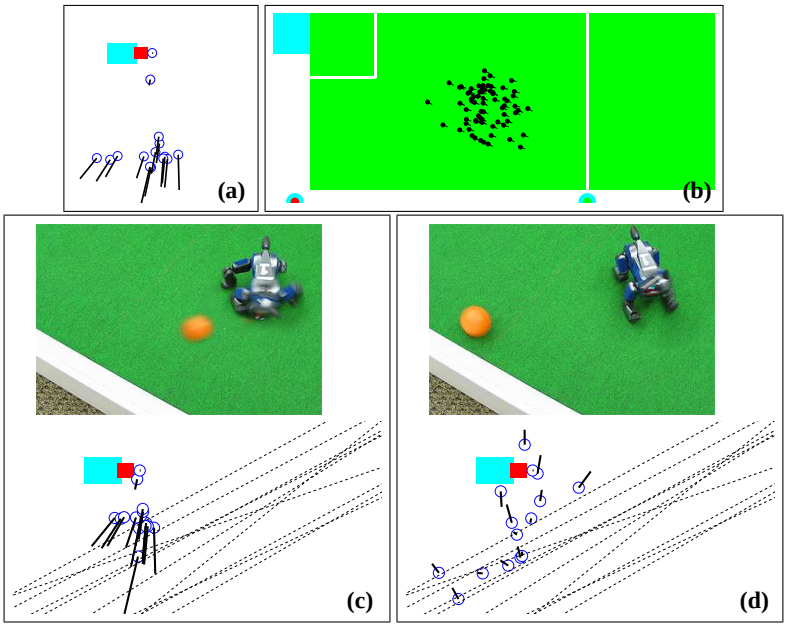

Fig. 4: (a) Robot-centric view of predicted ball samples. The robot kicked the ball to its right using its head, (small rectangle). If field borders are not considered, the ball samples travel in the kicked direction (ball motion is illustrated by the length and orientation of the small lines). (b) Particles $\rho_k^{(i)}$ representing robot's estimate of its position at the beginning of the kick command. In the approximation, these particles are frequently taken from the robot location sample set R_k. (c) The robot has kicked the ball toward the border. The robot particles shown in (b) are used to estimate the relative locations of borders. The sampled borders are shown as dashed lines. (d) Most ball samples transition into the `Bounced` state. Due to the uncertainty in relative border location, ball samples bounce off the border at different times, directions and velocities. The ball sample distribution predicts the true ball location much better than without considering the borders, as shown in (a). Ball samples can also bounce off the robot.

locations $\rho_k^{(i)}$ shown in (b). These borders determine whether and how the ball motion model transitions into the `Bounced` state.

The results of 100 kick experiments are summarized in Fig. 2(b). The solid line shows the rate of successfully tracking the ball after a kick. As can be seen, increasing the number of samples also increases the performance of the approach. The poor performance for small sample sizes indicates that the distribution of the ball is multi-modal, rendering the tracking task difficult for approaches such as the IMM [2]. Fig. 2(b) also demonstrates the importance of map information for tracking. The dashed line gives the results when not conditioning the ball tracking on the robot locations. Obviously, not considering the ball-environment interaction results in lower performance. On a final note, using our approach with 20 samples significantly reduces the time to find the ball, when compared to the commonly used random ball search strategy. When using the default search sequence, the robot takes on average 2.7 seconds to find the ball, whereas the robot can locate the ball in 1.5 seconds on average when using our approach described in Section 3.4.

5 Conclusion and Future work

In this paper we introduced a novel approach to tracking a moving target. The approach uses Rao-Blackwellised particle filters to sample the potential interactions between the observer and the target and between the target and the environment. By additionally sampling non-linear motion of the observer, the target and its motion can be estimated accurately using Kalman filters. Thus, our method combines the representational capabilities of particle filters with the efficiency and accuracy of Kalman filters. The approach goes beyond other applications of RBPFs in that it samples multiple parts of the state space and integrates environment information into the state transition model. In addition to the RBPF tracking algorithm, we introduce an efficient approximation that makes use of the information flow in the tracking problem. The approximation greatly reduces the number of particles needed for good performance and inherits the key benefits from the original algorithm. However, if enough processing power is available, we suggest to use the original algorithm.

The technique was implemented and evaluated using the task of tracking a ball with a legged AIBO robot in the RoboCup domain. This problem is extremely challenging since the legged motion of the robot is highly non-linear and the ball gets kicked and frequently bounces off obstacles in the environment. We demonstrate that our efficient implementation results in far better performance than vanilla Kalman filters. Other extensions of Kalman filters, such as the IMM and the GPB filters, are not well suited for this task, since the uncertainty in the ball location is often multi-modal (even when conditioned on a discrete state). The fact that our technique keeps track of multiple hypotheses for the ball's location allows the robot to efficiently search for the ball using negative information such as not seeing the ball. Finally, in real robot experiments, we show that taking the environment into account results in additional performance gains.

We strongly believe that estimating a joint over the ball and its interactions with the robots and the environment has also conceptual advantages. For example, our estimation system treats the question of whether or not the robot currently grabs the ball as part of the probabilistic estimation process. Thus, higher level robot control modules do not have to reason about such facts any more; the information is provided in a sound way by the comprehensive ball tracker. Furthermore, the ball tracker is "aware" of the robot's actions and automatically reasons about facts such as "if the robot grabs the ball, then the ball moves with the robot". Thus, such situations do not have to be treated as special cases; they are an integral part of the estimation process.

In the future we will extend the algorithm to integrate ball information observed by other robots, delivered via wireless communication. Such information can be transmitted efficiently by clustering the ball samples according to the different discrete ball motion states. The integration of transmitted ball estimates can then be done conditioned on the different discrete ball states. Another important area of future research is the integration of additional information provided by the vision system. Currently, we do not model the location of other robots on the field and the ball transits into the `deflected` model at random points in time. Furthermore, we estimate the relative location of the field borders using only the robot's location estimates. However, if the robot detects the ball and an object in the same camera image, then this image provides more accurate information about the relative location between the ball and an object.

Finally, we conjecture that further performance gains can be achieved using an unscented Kalman filter [14] to jointly track the position of the robot and the ball. Using the Rao-Blackwellisation described in this paper, the discrete state of the ball would still be sampled. However, each of these samples would be annotated with an unscented filter over both robot and ball locations (and velocity). By modeling more dimensions using efficient Kalman filters we expect to be able to track the robot / ball system with far less samples. Such an approach is especially appropriate for the more accurate sensors used in other RoboCup leagues.

6 Acknowledgments

This work has partly been supported by the NSF under grant number IIS-0093406.

References

1. Y. Bar-Shalom and X.-R. Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, 1995.
2. Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley, 2001.
3. X. Boyen and D. Koller. Exploiting the architecture of dynamic systems. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1999.
4. A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2000.
5. A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo in Practice*. Springer-Verlag, New York, 2001.
6. A. Doucet, N.J. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3), 2001.
7. D. Fox. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research (IJRR)*, 22(12), 2003.
8. F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P-J. Nordlund. Particle filters for positioning, navigation and tracking. *IEEE Transactions on Signal Processing*, 50(2), 2002.
9. J.S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.
10. M. Montemerlo, S. Thrun, and W. Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.
11. B. Ng, L. Peshkin, and A. Pfeffer. Factored particles for scalable monitoring. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.
12. T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 18(5), 2002.
13. D. Schulz, W. Burgard, and D. Fox. People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research*, 22(2), 2003.
14. E.A. Wan and R. van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proc. of Symposium on Adaptive Systems for Signal Processing, Communications, and Control*, 2000.