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 community [1, 2, 9, 11, 5, 6]. 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.

D. Nardi et al. (Eds.): RoboCup 2004, LNAI 3276, pp. 18–33, 2005.

- **Physical interaction between target and environment:** The ball frequently bounces off the borders of the field or gets kicked by other robots. Such interaction results 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 all these challenges in a unified Bayesian framework. The technique uses Rao-Blackwellised particle filters (RBPF) [3] 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 the AIBO platforms 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 basics of Rao-Blackwellised particle filters and their application to ball tracking in the RoboCup domain. 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 [10]. 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 represents the target locations 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 the Gaussian estimates after each update of the filters [2]. While these approaches are efficient, the model merging step assumes that the state conditioned on each discrete motion model is unimodal. However, our target tracking problem depends heavily on the uncertainty of the observer position in addition to the motion model.

These two factors can interact to produce multimodal distributions conditioned on each model, and an example will be provided in section 3.1.

Particle filters provide a viable alternative to Kalman filter based approaches [4]. These filters represent posteriors by samples, which allows them to optimally estimate non-linear, non-Gaussian processes. Recently, particle filters have been applied successfully to people tracking using a mobile robot equipped with a laser range-finder [11, 9]. While the sample-based representation gives particle filters their robustness, it comes at the cost of increased computational complexity, making them inefficient for complex estimation problems.

As we will describe in Section 3.2, Rao-Blackwellised particle filters (RBPF) [3] 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 [5, 6]. 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 [9]. However, their tracking application is less demanding and relies on a vanilla particle filter to estimate the joint state space of the observer and the target. Our RBPFs are far more efficient since our approach rely 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 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 remains in the `Grabbed` state.

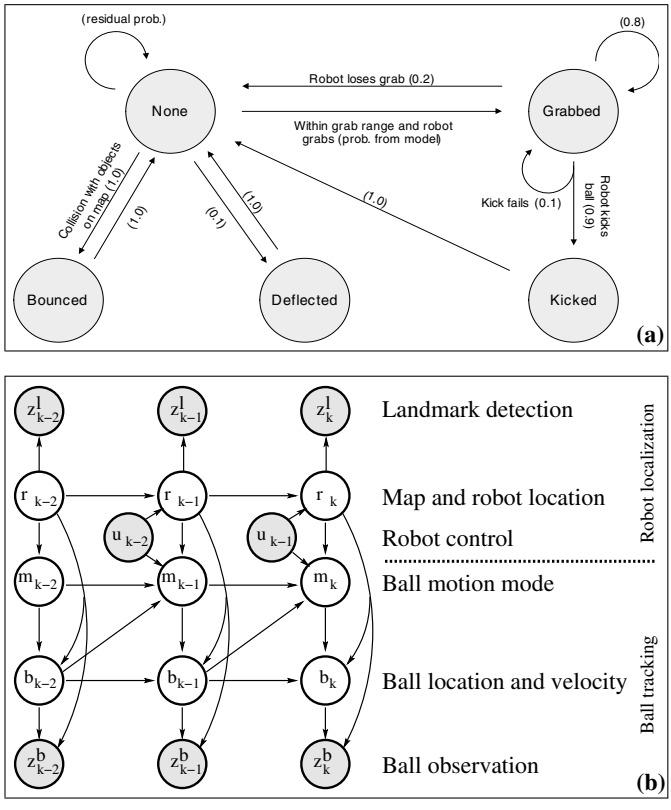

Fig. 1. (a) Finite state machine describing the transitions between different ball motion models. All transitions are probabilistic, the probabilities are enclosed in parentheses. (b) Graphical model for tracking a moving ball. The nodes in this graph represent the different parts of the dynamic system 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

- `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.
- `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 states are parenthesized in the figure. From the `None` state, we assume that there is a 0.1 probability the ball will be deflected at each update. When the ball is somewhere 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

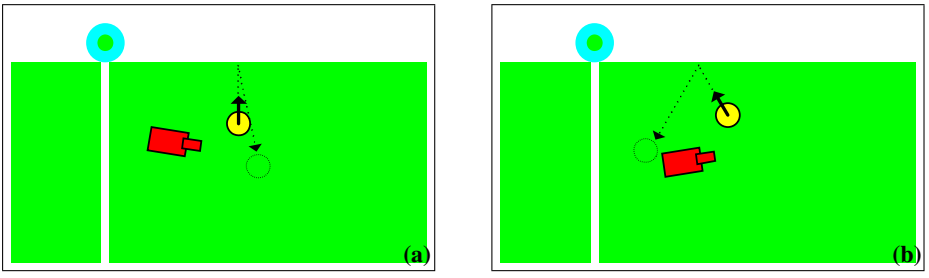

Fig. 2. Effect of robot position on ball-environment interation. The two figures represent two different estimates of the robot's position, differing by less than 20 degrees and 20cm. They result in very different predicted ball positions after collision with the border, with (a) in front of the robot and (b) behind

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 transition 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 0.9 success rate and Grabbing 0.8. Finally, 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.

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. 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. Figure 2 shows an example of their interdependence. The robot is tracking a ball travelling towards the border. It is uncertain about its own location, and Figures 2(a) and (b) are both possible locations. In the figures, the robot has the same estimates of the relative position and velocity of the ball. However, their slight difference in positions leads to very different predicted ball positions after collision with the border, with (a) in front and (b) behind the robot. If we ignore the uncertainty in robot position, this interaction cannot be modeled correctly. In the next section we will see how RBPF can be used to perform this joint estimation.

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 types of interaction between the robot 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, just like in standard robot localization, the robot location 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. 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, the change in ball prediction if the ball 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 position. Transitions of the ball motion model m_k are probabilistic versions of those described in Fig. 1(a).

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 RBPF algorithm is beyond the scope of this paper; see [3] for a thorough discussion of the basic RBPF and its properties. 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 $s_k^{(i)} = \langle b_k^{(i)}, m_{1:k}^{(i)}, r_{1:k}^{(i)} \rangle$, where $b_k^{(i)}$ are the mean and covariance of the ball location and velocity and $m_{1:k}^{(i)}$ and $r_{1:k}^{(i)}$ are the histories of ball motion models and robot locations, respectively. The key idea of RBPFs is to condition the ball estimate $b_k^{(i)}$ on a particle's history of ball motion models $m_{1:k}^{(i)}$ and robot locations $r_{1:k}^{(i)}$. This conditioning turns the ball location and velocity into a linear system that can be estimated efficiently using a Kalman filter.

To see how RBPFs recursively update posterior estimates, we factorize the posterior as follows:

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

The task is to generate samples distributed according to (1) based on samples drawn from the posterior at time $k-1$, represented by the previous sample set S_{k-1}. We generate the different components of each particle $s_k^{(i)}$ stepwise by simulating (1) from right to left. In the first step, a sample $s_{k-1}^{(i)} = \langle b_{k-1}^{(i)}, m_{1:k-1}^{(i)}, r_{1:k-1}^{(i)} \rangle$ is drawn from S_{k-1}. Through conditioning on this sample, we first expand the robot trajectory to $r_{1:k}^{(i)}$, then the ball motion models to $m_{1:k}^{(i)}$ conditioned on $r_{1:k}^{(i)}$, followed by an update of $b_k^{(i)}$ conditioned on both the robot trajectory and the motion model history. Let us start with expanding the robot trajectory, which requires to draw a new robot position $r_k^{(i)}$ according to

$$r_k^{(i)} \quad \sim \quad p(r_k | s_{k-1}^{(i)}, z_{1:k}, u_{1:k-1}). \quad (2)$$

The trajectory $r_{1:k}^{(i)}$ resulting from appending $r_k^{(i)}$ to $r_{1:k-1}^{(i)}$ is then distributed according to the rightmost term in (1). The distribution for $r_k^{(i)}$ can be transformed as follows:

$$
\begin{aligned}
& p(r_k \mid s_{1:k-1}^{(i)}, z_{1:k}, u_{1:k-1}) && (3)\\
& \quad = p(r_k | s_{k-1}^{(i)}, z_k, u_{k-1}) && (4)\\
& \quad \propto p(z_k | r_k, s_{k-1}^{(i)}, u_{k-1})\; p(r_k | s_{k-1}^{(i)}, u_{k-1}) && (5)\\
& \quad = p(z_k | r_k, r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})\; p(r_k | r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) && (6)\\
& \quad = p(z_k | r_k, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})\; p(r_k | r_{k-1}^{(i)}, u_{k-1}) && (7)
\end{aligned}
$$

Here, (4) follows from the (Markov) property that r_k is independent of older information given the previous state. (5) follows by Bayes rule, and (7) from the independencies represented in the graphical model given in Fig. 1(b). To generate particles according to (7) we apply the standard particle filter update routine [4]. More specifically, we first pick a sample $s_{k-1}^{(i)}$ from S_{k-1}, then we predict the next robot location $r_k^{(i)}$ using the particle's $r_{k-1}^{(i)}$ along with the most recent control information u_{k-1} and the robot motion model $p(r_k | r_{k-1}^{(i)}, u_{k-1})$ (rightmost term in (7)). This gives the extended trajectory $r_{1:k}^{(i)}$. The importance weight of this trajectory is given by the likelihood of the most recent measurement: $w_k^{(i)} \propto p(z_k | r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})$. If z_k is a landmark detection, then this likelihood is given by $p(z_k | r_k^{(i)})$, which corresponds exactly to the particle filter update for robot localization. If, however, z_k is a ball detection z_k^b, then total probability gives us

$$
\begin{aligned}
& p(\; z_k \; | r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) && (8)\\
& \quad = \sum_{M_k} p(z_k | r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}, M_k) p(M_k | r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) && (9)\\
& \quad = \sum_{M_k} p(z_k | r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1}, M_k) p(M_k | r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) && (10)
\end{aligned}
$$

where M_k ranges over all possible ball motion models. The second term in (10) can be computed from the transition model, but $p(z_k | r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1}, M_k)$ is the likelihood obtained from a Kalman update, which we need to perform for each M_k (see below). In the next section, we will describe how we avoid this complex operation.

At the end of these steps, the robot trajectory $r_{1:k}^{(i)}$ of the particle is distributed according to the rightmost term in (1). We can now use this trajectory to generate the ball motion model part $m_{1:k}^{(i)}$ of the particle using the second to last term in (1). Since we already have $m_{1:k-1}^{(i)}$, we only need to sample

$$
\begin{aligned}
m_k^{(i)} \;&\sim\; p(m_k | m_{1:k-1}^{(i)}, r_{1:k}^{(i)}, b_{k-1}^{(i)}, z_{1:k}, u_{1:k-1}) && (11)\\
&\propto\; p(z_k | m_k, r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1})\; p(m_k | m_{k-1}^{(i)}, r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1}). && (12)
\end{aligned}
$$

(12) follows from (11) by reasoning very similar to the one used to derive (7). The rightmost term in (12) describes the probability of the ball motion mode at time k given the

previous mode, robot location, ball location and velocity, and the most recent control. As described above, this mode transition is crucial to model the ball's interaction with the environment. To generate motion model samples using (12), we predict the mode transition using reasoning about the different ball-environment interactions (see Fig. 1(a)). The importance weight of the sample $s_k^{(i)}$ has then to be multiplied by the observation likelihood $p(z_k|m_k^{(i)}, r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1})$, which is given by the innovation of a Kalman update conditioned on $m_k^{(i)}, r_k^{(i)}, b_{k-1}^{(i)}$, and u_{k-1}.

To finalize the computation of the posterior (1), we need to determine the leftmost term of the factorization. As mentioned above, since we sampled the non-linear parts $r_{1:k}^{(i)}$ and $m_{1:k}^{(i)}$ of the state space, the posterior

$$b_k^{(i)} \sim p(b_k|m_{1:k}^{(i)}, r_{1:k}^{(i)}, z_{1:k}, u_{1:k-1}) \tag{13}$$

can be computed analytically using a regular Kalman filter. The Kalman filter prediction uses the motion model $m_k^{(i)}$ along with the most recent control u_{k-1}. The correction step is then based on the robot location r_k along with the most recent observation z_k. The Kalman correction step is not performed if z_k is a landmark detection.

To summarize, we generate particles at time k by first drawing a particle $s_{k-1}^{(i)} = \langle b_{k-1}^{(i)}, m_{1:k-1}^{(i)}, r_{1:k-1}^{(i)} \rangle$ from the previous sample set. In the first step, we expand this particle's robot trajectory by generating a new robot location using (7), which gives us $r_{1:k}^{(i)}$. Conditioning on $r_{1:k}^{(i)}$ allows us to expand the history of ball motion models by predicting the next motion model using (12). Finally, $r_{1:k}^{(i)}$ and $m_{1:k}^{(i)}$ render the ball location and velocity a linear system and we can estimate $b_k^{(i)}$ using regular Kalman filter updating. The importance weight of the new particle $s_k^{(i)} = \langle b_k^{(i)}, m_{1:k}^{(i)}, r_{1:k}^{(i)} \rangle$ is set proportional to

$$w_k^{(i)} \propto p(z_k|r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1}) p(z_k|m_k, r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1}). \tag{14}$$

3.3 Efficient Implementation

We implemented the RBPF algorithm described above and it worked very well on data collected by an AIBO robot. By computing the joint estimate over the robot location and the ball, the approach can handle highly non-linear robot motion, predict when the ball bounces into field borders, and eliminates inconsistent estimates (*e.g.*, when the ball is outside the field). 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 main reason for this high number of samples is that for each robot location, we need to estimate multiple ball motion models. Furthermore, in RBPF the ball and robot estimates are coupled, with the weights of each sample depending on the contributions from the ball and the robot position. One consequence is that ball estimates can influence the robot's localization, since a sample can be removed by resampling if its ball estimate is not very accurate. This influence can be useful in some situations, such as invalidating a sample when the ball estimate is out-of-bounds, since we can infer that its robot position must also be

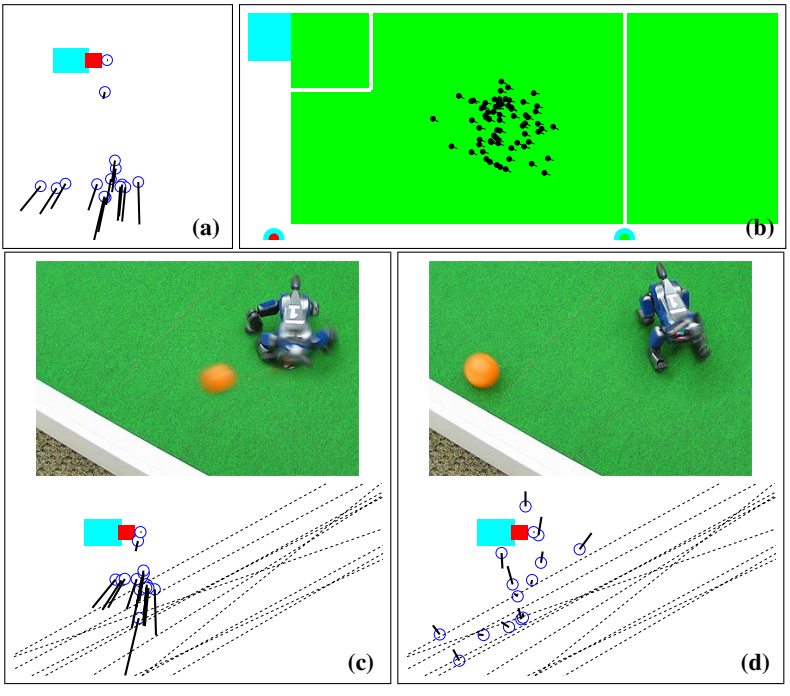

Fig. 3. (a) Robot-centric view of predicted ball samples. The robot kicked the ball to its right using its head, indicated by the small rectangle. If field borders are not considered, the ball samples travel in the kicked direction (ball motion is illustrated by the length and orientation of the small lines). (b) Particles representing robot's estimate of its position at the beginning of the kick command. (c) The robot has kicked the ball towards the border. The ball samples are attached to the robot particles shown in (b) in order 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, with different directions and velocities. The ball sample distribution predicts the true ball location much better than without considering the borders (compare to (a)). Note that ball samples can also bounce off the robot

erroneous. However, this artifact is undesirable in general; while the global position of the ball is conditioned on the robot position, the ball does not really provide significant information for localization. This problem is further accentuated by the fact that while ball estimates need to be updated about 20 times per second, the robot location on the field needs to be updated about only once per second. The frequent ball updates result in importance weights that outweigh the contribution of (rather rare) landmark detections. This results in the deletion of many robot location hypotheses even if the robot was not seeing any landmarks.

Our efficient approximation to the full Rao-Blackwellised approach described in the previous section is based on the observation that ball detections do not provide significant information about the robot's location on the field. The key idea is we partitioned the

state space into robot and ball positions, which are updated separately. We recombine them into Rao-blackwellised particles when needed. The set of samples S_k is now the pair $\langle R_k, B_k \rangle$, where R_k is the set of robot position samples, and B_k is the set of ball samples, each sample consists of the ball's position and the model it is conditioned on. At the beginning of each update, instead of sampling $s_{k-1}^{(i)} = \langle b_{k-1}^{(i)}, m_{1:k-1}^{(i)}, r_{1:k-1}^{(i)} \rangle$ from S_{k-1}, we sample robot positions $r_{1:k-1}^{(j)}$ from R_{k-1} and ball model-position pairs $b_{k-1}^{(i)}, m_{1:k-1}^{(i)}$ from B_{k-1}. With the decoupled state space, we can use M samples to estimate the robot's position and N samples to estimate the ball's location. Usually, the distribution of ball motion model is less complex than the robot position, so $N < M$. As before, we extend the robot trajectory to $r_k^{(i)}$, but this time we can drop the dependency of the robot position on the ball since $r_k^{(i)}$ is not coupled with a ball position. Thus we have the following approximation of equation (2)

$$r_k^{(j)} \sim p(r_k | s_{k-1}^{(j)}, z_{1:k}, u_{1:k-1}) \tag{15}$$

$$\approx p(r_k | r_{1:k-1}^{(j)}, z_{1:k}, u_{1:k-1}). \tag{16}$$

Thus the estimation of the robot position becomes regular particle filter-based localization [4], with ball observations having no influence on r_k. The distribution of $r_k^{(j)}$ is simply

$$p(r_k | r_{1:k-1}^{(j)}, z_{1:k}, u_{1:k-1}) \propto p(z_{1:k} | r_k, u_{1:k-1}) p(r_k | r_{1:k-1}^{(j)}, u_{1:k-1}) \tag{17}$$

which has the exact form of a robot localization update. We obtain $r_k^{(j)}$ by sampling from the robot motion model $p(r_k | r_{1:k-1}^{(j)}, u_{1:k-1})$ using the most recent control information u_{k-1}, as usual. This gives us an updated robot sample set R_k.

We now turn our attention to the ball estimates. Since they are no longer coupled with robot positions, they are estimated in a pseudo-global coordinate system. Each ball sample $b_k^{(i)}$ has an associated observer position $\rho_{1:k}^{(i)}$ which is initialized to the origin $\rho_1 = (0, 0, 0)$. At each iteration, $\rho_k^{(i)}$ is computed from $\rho_{1:k-1}^{(i)}$ by sampling from the robot motion model and u_{k-1}. When we compute the ball's interaction with the environment, we need the ball estimates in global coordinates. We obtain this by sampling a robot position $r_k^{(j)}$ from R_k for each ball sample, and applying the offset $b_k^{(i)} - \rho_k^{(i)}$ to $r_k^{(j)}$. With this scheme, we approximate (12) as follows:

$$p(z_k | m_k, r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1}) \approx p(z_k | m_k, \rho_k^{(i)}, b_{k-1}^{(i)}, u_{k-1}) \tag{18}$$

$$p(m_k | m_{k-1}^{(i)}, r_k^{(i)}, b_{k-1}^{(i)}, u_{k-1}) \approx p(m_k | m_{k-1}^{(i)}, r_k^{(j)}, b_{k-1}^{(i)}, u_{k-1}) \tag{19}$$

In (18) we compute the likelihood of the ball observation based on $\rho_k^{(i)}$ rather than the joint robot position $r_k^{(i)}$. This approximation is fairly faithful to the original RBPF since the trajectory represented by $\rho_{1:k}^{(i)}$ is generated from $u_{1:k-1}$, similar to $r_{1:k}^{(i)}$. In (19) we predict the motion model of the ball $b_k^{(i)}$ using the global position obtained from the sampled robot position $r_k^{(j)}$ instead of the paired $r_k^{(i)}$ in RBPF. While the variance of this approximation is higher, the expected distribution resulting from interacting with the

environment is the same as RBPF's. Note that we can set $j = i$ to avoid extra sampling without loss of generality. In this case, $r_k^{(i)}$ is interpreted as the i-th sample in R_k, not the coupled robot position in RBPF.

Finally, the posterior for the ball positions can be computed by a Kalman filter update using $\rho_k^{(i)}$ instead of $r_k^{(i)}$ in (13)

$$b_k^{(i)} \sim p(b_k | m_{1:k}^{(i)}, \rho_k^{(i)}, z_{1:k}, u_{1:k-1}). \tag{20}$$

The complete approximation algorithm is shown in Table 1. At each iteration, a different relative offset is generated for each ball by sampling from the robot motion model using u_k. Then the ball samples are translated back to global coordinates by attaching their relative ball estimates to the most recent particles of the robot localization. These particles are selected by sampling N robot positions from the M in the set. Thus, the ball and its motion model are estimated exactly as described before, with sampling from the highly non-linear robot motion and the ball motion models. Furthermore, since ball estimates are in global coordinates, the ball-environment interaction can be predicted as before. The only difference is that information about the ball does not contribute to the estimated robot location. However, our approximation drastically reduces the number of robot and ball samples needed for good onboard results (we use 50 robot and 20 ball particles, respectively). The key idea of this algorithm is summarized in Fig. 3. As can be seen in Fig. 3(c) and (d), each ball particle $\langle b_k^{(i)}, m_k^{(i)} \rangle$ uses a different location for the border extracted from the robot location particles $r_k^{(j)}$ shown in (b). These borders determine whether the ball motion model transitions into the `Bounced` state.

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 mode. 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 though they are in the visible range of the camera get low importance weights (visibility

Table 1. The efficient implementation of the Rao-blackwellised particle filter algorithm for ball tracking

---

1. **Inputs:**
   $S_{k-1} = \langle R_{k-1}, B_{k-1} \rangle$ representing belief $Bel(s_{k-1})$, where
   $R_{k-1} = \{\langle r_{k-1}^{(j)}, w_{k-1}^{(j)} \rangle \mid j = 1, \ldots, N\}$ represents robot positions,
   $B_{k-1} = \{\langle b_{k-1}^{(i)}, m_{k-1}^{(i)}, \rho_{k-1}^{(i)}, \omega_{k-1}^{(i)} \rangle \mid i = 1, \ldots, M\}$ represents ball positions
   control measurement u_{k-1},
   observation z_k
2. $R_k := \emptyset, B_k := \emptyset$ *// Initialize*
3. **for** $j := 1, \ldots, N$ **do** *// Generate N robot samples*
4. Sample an index l from the discrete distribution given by the weights in R_{k-1} *// Resampling*
5. Sample $r_k^{(j)}$ from $p(r_k \mid r_{k-1}, u_{k-1})$ conditioned on $r_{k-1}^{(l)}$ and u_{k-1}
6. $w_k^{(j)} := p(z_k | r_k^{(j)})$ *// Compute likelihood*
7. $R_k := R_k \cup \{\langle r_k^{(j)}, w_k^{(j)} \rangle\}$ *// Insert sample into sample set*
8. **end do**
9. Normalize the weights in R_k
10. **for** $i := 1, \ldots, M$ **do** *// Update M ball samples*
11. Sample an index l from the discrete distribution given by the weights ω_{k-1} in B_{k-1}
12. Sample $m_k^{(i)}$ from $p(m_k | m_{k-1}^{(l)}, r_k^{(i)}, b_{k-1}^{(l)}, u_{k-1})$
13. Sample $\rho_k^{(i)}$ from $p(r_k \mid r_{k-1}, u_{k-1})$ conditioned on $\rho_{k-1}^{(l)}$ and u_{k-1}
14. $b_k^{(i)} :=$ Kalman update using $b_{k-1}^{(l)}$, $m_k^{(i)}$, z_k and u_{k-1}
15. $\omega_k^{(i)} := p(z_k | m_k, \rho_k^{(i)}, b_{k-1}^{(i)}, u_{k-1})$ *// Compute importance weight*
16. $B_k := B_k \cup \{\langle b_k^{(i)}, m_k^{(i)}, \rho_k^{(i)}, \omega_k^{(i)} \rangle\}$ *// Insert sample into sample set*
17. **end do**
18. Normalize the weights in B_k
19. **return** $S_k = \langle R_k, B_k \rangle$

---

considers occlusions by other robots). In the next update step of the Rao-Blackwellised particle filter, these ball samples are very unlikely to be drawn from the weighted sample set, thereby focusing the search to 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.

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 several reasons such as looking at landmarks for localization, or the ball is 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 information available 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 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 model was `none`. The results for prediction times up to 2 seconds are shown in Fig. 4(a). In addition to our RBPF approach (thick, solid line), we compare it with 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. 4(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. 5. Our approach, shown in Fig. 5(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. 5(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. 5(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.

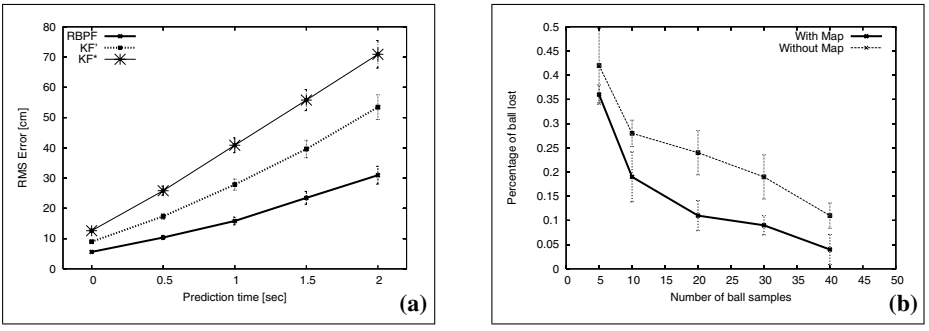

Fig. 4. (a) RMS error of the ball's position for different prediction times. (b) Percentage of time the robot loses track of the ball after a kick for different numbers of particles with and without map information

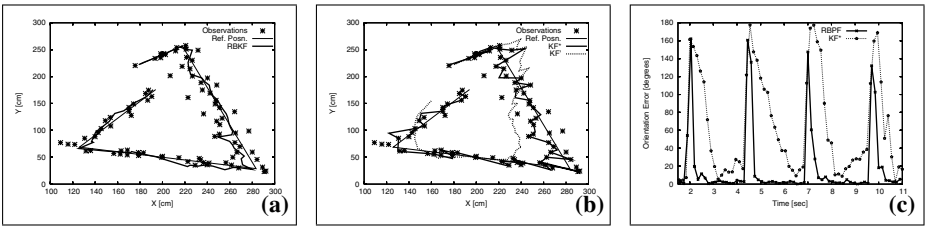

Fig. 5. Tracking of a ball trajectory with multiple kicks, the observer is located on the left. In (a) and (b), the observations are indicated by stars and the true target trajectory is shown as a thin line. (a) shows the estimated trajectory of our RBPF multiple-model approach. (b) shows the estimates 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 represents RBPF, dashed line a Kalman filter with inflated prediction noise

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. 3(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 scanning the markers on the field periodically.

The solid line in Fig. 4(b) 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. 4(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 moving targets. 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, estimating the target and its motion can be performed efficiently using Kalman filters. Thus, our method combines the representational complexity 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.

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 frequently bounces off obstacles in the environment. We demonstrate that our efficient implementation results in far better performance than vanilla Kalman filters. Furthermore, we show that taking the environment into account results in additional performance gains. We belief that our approach has applications to tracking problems beyond the RoboCup domain. It can be applied whenever the observer robot performs highly non-linear motion and the environment provides information about the motion of the object being tracked.

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 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 [12] 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 attached 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.

See `http://www.cs.washington.edu/balltracking` for further information about our approach. [8, 7]

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. 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*, 2000.
4. A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo in Practice*. Springer-Verlag, New York, 2001.
5. 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.
6. 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.
7. C.T. Kwok, D. Fox, and M. Meilă. Adaptive real-time particle filters for robot localization. In *Proceedings of the 2003 IEEE International Conference on Robotics Automation (ICRA '03), Taipei, Taiwan*, September 2003.
8. C.T. Kwok, D. Fox, and M. Meilă. Real-time particle filters. *IEEE Special Issue on Sequential State Estimation*, March 2004.
9. 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*, 2002.
10. 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.
11. 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.
12. E.A. Wan and R. van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proc. of Symposium 2000 on Adaptive Systems for Signal Processing, Communications, and Control*, 2000.