Multi-Model Tracking Using Team Actuation Models

Yang Gu
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: guyang@cs.cmu.edu

Manuela Veloso
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: mmv@cs.cmu.edu

***Abstract*— Robots need to track object. Object tracking efficiency completely depends on the accuracy of the motion model and of the sensory information. Interestingly, when multiple teammembers can actuate the object being tracked, the motion can become highly discontinuous and nonlinear. We have previously developed a successful tracking approach that switches among target motion models as a function of one robot's actions. In this paper, we report on a tracking approach that can use a dynamic multiple motion model based on a team coordination plan. We present the multi-model probabilistic tracking algorithms in detail and present empirical results both in simulation and in a human-robot Segway soccer team. The team coordination plan allows the robot to much more effectively track mobile targets.**

I. Introduction

There have been plenty of investigations into the problem of tracking moving objects e.g. [1]. There has been a similar interest in tracking objects from various robot platforms within the robotics community, e.g. [2]. When tracking is performed by a robot executing specific tasks acting over the object(target) being tracked, such as a Segway RMP soccer robot kicking and receiving a ball, the motion model of the object becomes tightly dependent on the robot's actions [3]. The robot's tactic provides valuable information in terms of the object behavior. A tactic-based motion modelling and tracking in such scenarios has been introduced in [4].

However, for the environments in which the Segway RMP soccer robot operates in, there are multiple objects, besides the ball, e.g. the teammember and the opponents, which need to be tracked properly. All the players on the field can also actuate over the ball, namely grab and kick the ball according to the rules which makes the motion model of the ball even more complex. When the robot is playing a game as a member of a human-robot team, the team coordination knowledge provides further information that can be incorporated into the motion modelling and tracking process. In this paper, we present an extension to the tactic-based tracking scheme introduced in [4] to solve a plan-dependent multi-target tracking problem.

The paper is organized as follows. We first give a brief description of the Segway RMP soccer robot. Next we show the play-based motion modelling for multiple objects and we incorporate the team coordination knowledge into the motion modelling. We then describe the multi-model tracking algorithm for multiple objects, leading to our experimental results, related work, conclusions and future work.

II. Segway RMP Soccer Robot

The Segway platform is unique due to its combination of wheel actuators and dynamic balancing. Segway RMP, or Robot Mobility Platform, provides an extensible control platform for robotics research [5].

In our previous work, we have developed a Segway RMP robot base capable of playing Segway soccer. We briefly describe the two major components of the control architecture, the sensor and the robot cognition, which are highly related to our multi-model motion tracking.

A. Vision Sensor

The goal of vision is to provide as many valid estimates of objects as possible. Tracking then fuses this information to track the most interesting objects (the ball and the teammember, in this paper) of relevance to the robot. We do not discuss the localization of the robot in the sense that a lot of soccer tasks (known as tactics and plays in later sections) can be done by the Segway RMP robot independently of knowing where it is in the world. Also we use global reference in this paper (global position and velocity) which means it is relative to the reference point where the robot starts to do dead reckoning.

B. Robot Cognition

A control architecture, called Skills-Tactics-Plays, was proposed in [6] to achieve the goals of responsive, adversarial team control. The key component of STP is the division between single robot behavior and team behavior.

A play, P, is a fixed team plan which consists of a set of applicability conditions, termination conditions, and N roles, one for each team member. Each role defines a sequence of tactics $T_1, T_2, \cdots$ and associated parameters to be performed by that role in the ordered sequence. Assignment of roles to team members is performed dynamically at run time. Upon role assignment, each robot i is assigned its tactic T_i to execute from the current step of the sequence for that role.

A tactic, T, encapsulates a single robot behavior. Each robot i executes its own tactic as created by the current play P.

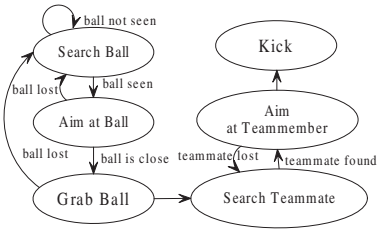

Fig. 1. Skill state machines (SSMs) for an example tactic: *CatchKickToTeammember*.

A tactic T_i determines the skill state machine SSM_i to be executed by the robot i.

A skill, S, is a focused control policy for performing some complex action. Each skill is a member of one, or more, skill state machines $SSM_1, SSM_2, \cdots$. Each skill S determines what skill it transitions to S' based upon the world state, the time skill S has been executing for, and the executing tactic for that robot.

We construct the robot cognition using a similar architecture. Plays, tactics, and skills, form a hierarchy for team control. Plays control the team behavior through tactics, while tactics encapsulate individual robot behavior and instantiate actions through sequences of skills. Skills implement the focused control policy for actually generating useful actions. Figure 1 shows the $SSMs$ and transitions for an example tactic: *CatchKickToTeammember*, which contains six skills. Each node in the figure is a skill and the edges show the transition between skills.

Segway soccer is a team sport, and therefore the building of our game strategy required not only execution of single robot behavior , but also coordination with the teammember, the human player. The current coordination is simple and basically based upon two fixed plays for offensive and defensive situation respectively. Our offensive play is shown as follows, in which the termination condition is either play aborted or the situation changed (a turn-over of ball possession announced by the referee). There are two roles in this play, one passes the ball to the other who positions down field and waits for receiving a pass.

```
PLAY Naive Offense
APPLICABLE offense
DONE aborted !offense
ROLE 1
  pass 2
  none
ROLE 2
  position_down_field
  receive_pass
  none
```

Our current coordination is purely observation based. Each player assigns role from his own eyeshot without communication. For example, should the robot think the teammember is closer to the ball, the robot would choose to position and receive the ball (ROLE 2) from its teammember (ROLE 1). Furthermore, the robot knows which side gains possession of the ball from the referee announcement (whistle), therefore it tells offensive from defensive situation clearly and thus it has deterministic idea of which play the team is using. The robot assumes that its teammember is performing the same game play as itself. The robot can infer what tactic the teammember is executing from the team play. For instance, after receiving the ball from the teammember, as a passer, the robot would assume the teammember go forward to a tactically advantageous position to receive a pass. The predefined play for team coordination provides useful information for motion modelling, which will be further discussed in section III.

III. Play-Based Motion Modelling

In this section, we take a multi-target tracking problem as a detailed example to show the extension of the tactic-based motion modelling method in general when the team coordination knowledge (play) is incorporated. First we give an introduction of the environment and objects under the Segway soccer setup. Second, we describe detailed motion models for both the ball and the teammember. Third, we extend the tactic-based motion modelling to the play level when both the ball and the teammember are included into the tracking. We show how we model the play-dependent interactions between the teammember, the robot and the ball and set up a base for giving the multi-model tracking algorithm in the next section.

A. Tracking Scenario

In a Segway soccer game, there are multiple moving objects on the field. e.g, the ball, the human teammember and the two opponents. Each team is identified by their distinct color. The ball is in orange [7]. We construct two single target trackers in the system, for the ball and the teammember respectively. We use two separate trackers instead of one multi-target tracker for both of them because we can differentiate the ball with the teammember thanks to the color-based vision system.

The general parameterized state-space system for the kth target $\mathbf{x}_{k,t}$ at time t is given by:

$$\mathbf{x}_{k,t} = f_k^m(\mathbf{x}_{k,t-1}, \mathbf{u}_{k,t-1}^m, \mathbf{v}_{k,t-1}^m) \quad (1)$$

$$\mathbf{z}_{k,t} = h_k^m(\mathbf{x}_{k,t}, \mathbf{n}_{k,t}^m) \quad (2)$$

where f_k^m and h_k^m are the parameterized state transition and measurement functions for the mth model of the kth target; $\mathbf{x}, \mathbf{u}, \mathbf{z}$ are the state, input and measurement vectors; $\mathbf{v}, \mathbf{n}$ are the process and measurement noise vectors of known statistics; m is the model index that can take any one of N_k values, where N_k is the number of models of the kth target being tracked (ball/teammember);

B. Ball Motion Modelling

In our Segway RMP soccer robot environment, we define five models to model the ball motion (for the rest of this paper, for simplicity, we use $\mathbf{x}_t$ to represent the ball state, and use $\mathbf{x}'_t$ to represent the teammember state).

- *Free-Ball*. The ball is not moving at all or moving straight with a constant speed decay d which depends on the environment surface.

$$\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{v}^1_{t-1} \quad (3)$$
$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{n}^1_t \quad (4)$$

where $\mathbf{x}_t = (x_t, y_t, \dot{x}_t, \dot{y}_t)^T$, $\mathbf{z}_t = (x_t, y_t)^T$; x_t, y_t are the ball's x, y position in the global coordinate at time t; and $\dot{x}_t, \dot{y}_t$ are the ball's velocity in x and y direction in the global coordinate. The superscript "^{1}" indicates the model index. $\mathbf{F}_t$ and $\mathbf{H}_t$ are known matrices as follows:

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & d \end{bmatrix}, \mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

where Δt is the time interval between vision frames.
- *Robot-Grab-Ball*. The ball is grabbed by the robot's catcher. In the case of robot grabbing ball, no vision is needed to track the ball, because we assume the ball moves with the robot. Therefore the ball has the same velocity as the robot (but plus the noise) and its global position at time t is just the robot's global position plus their relative position, which is assumed to be a constant, plus the noise.
- *Human-Grab-Ball*. The ball is held by the teammember. we can infer the ball position similarly if we know the teammember position well.
- *Robot-Kick-Ball*. The ball is kicked by the robot therefore its velocity is equal to a predefined initial speed plus the noise. The ball is supposed to move toward either the human teammember or the goal.
- *Human-Kick-Ball*. The ball is kicked by the teammember and it is supposed to be either a pass to the robot or a shoot at the goal .

C. Teammember Motion Modelling

We define four models to model the human teammember's motion.

- *Random Walk*. The teammember is wondering in the field. So the state at the new time is the state at the current time with some additive zero-mean (assumed Gaussian) noise.
- *Holding Ball*. The teammember is holding the ball without moving and waiting for the robot to receive the ball. Should the robot know the ball position well, it can infer the teammember position by the ball position in a similar way as *Robot-Grab-Ball* for ball motion modelling.
- *Accelerating*. The teammember dashes and obtains a velocity in a short time.
- *Positioning*. The teammember is going to a predefined tactical position with a constant speed. This case happens mostly after the teammember passing the ball to the robot and moving down the field toward opponent's goal.

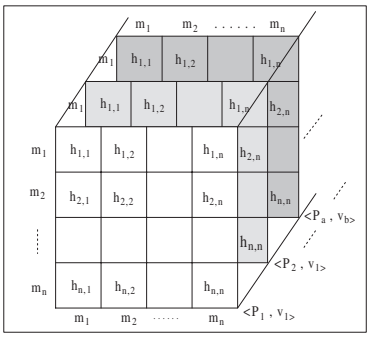

Fig. 2. Play-Based motion modelling, where $m_1, m_2, \cdots, m_n$ are n models, P_a is the team play, v_b is the additional information. $h_{i,j}$ is the transition probability from model m_i to model m_j given m_i, and $\langle P_a, v_b \rangle$. Each layer in the graph is conditioned on a particular combination of the play executed and the additional information obtained.

D. Play Based Model Transitions

Given the knowledge of the team coordination plan (the play P_{t-1} at time $t-1$), the robot can infer what tactic the teammember is executing (T'_{t-1}), which provides valuable information about the motion model of the teammember (m'_t). Both the robot and the teammember act over the ball in a Segway soccer game. The motion model of the ball (m_t) is therefore affected by what tactic the robot (T_{t-1}) and the teammember (T'_{t-1}) are executing.

From the previous subsection, we know that the model index m determines the present model being used. For our teammember tracking example, $m'_t = i, i = 1, \cdots, 4$. In our approach, it is assumed that the teammember motion model index, m'_t, conditioned on the previous tactic executed T'_{t-1} by the teammember, and other useful information v'_t (such as ball state), is governed by an underlying Markov process, such that, the conditioning parameter can branch at the next time-step with probability.

$$p(m'_t = i | m'_{t-1} = j, T'_{t-1}, v'_t) = h'_{i,j} \quad (5)$$

where $i, j = 1, \cdots, N_{m'}$. Since T'_{t-1} can be determined by P_{t-1}, we get

$$h'_{i,j} = p(m'_t = i | m'_{t-1} = j, P_{t-1}, v'_t) \quad (6)$$

Since we can draw $p(m'_t = i | m'_{t-1} = j)$ in an $N_{m'} \times N_{m'}$ table, we can create a table for Equation 6 with a third axis which is defined by the tuple $\langle P_a, v_b \rangle$ as shown in Figure 2. Here the play P_a, is the primary factor that determines whether m_i transits to m_j and what the transition probability is, while the information v_b determines the prior condition of the transition. Each layer in the graph is conditioned on a particular combination of the tactic executed and the additional information obtained.

For our ball tracking example, $m_t = i, i = 1, \cdots, 5$. Similarly,

$$h_{i,j} = p(m_t = i | m_{t-1} = j, T_{t-1}, T'_{t-1}, v_t) \quad (7)$$

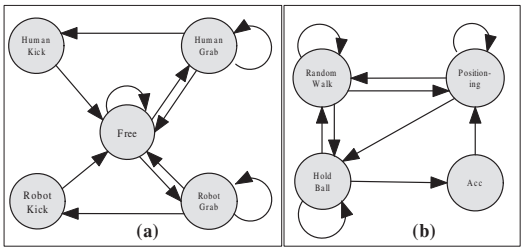

Fig. 3. Object motion modelling based on the play: *Naive Offense*. Each node is a model. Models transit to one another according to the predefined probabilities (not shown in the figure). (a) Ball motion model. (b) Human teammember motion model.

where $i, j = 1, \cdots, N_m$. Since T_{t-1}, T'_{t-1} can be determined by P_{t-1}, we get

$$h_{i,j} = p(m_t = i | m_{t-1} = j, P_{t-1}, v_t) \tag{8}$$

Suppose the current team play is the *Naive Offense* in Section II-B, we can obtain the corresponding motion model transitions for both the ball and the teammember using the play-based method (Figure 3).

IV. Multi-Model Motion Tracking

Following the play-based motion model given in the previous section, we can use dynamic Bayesian networks (DBNs) to represent the whole system for teammember and ball tracking in a natural and compact way as shown in Figure 4 and Figure 5 respectively. In this graph, the system state is represented by variables (play P, tactic T, infrared sensor measurement s, ball state $\mathbf{x}$, ball motion model index m, vision sensor measurement of ball $\mathbf{z}$, teammember state $\mathbf{x}'$, teammember motion model index m', vision sensor measurement of teammember $\mathbf{z}'$), where each variable takes on values in some space. The variables change over time in discrete intervals, so that e.g. $\mathbf{x}_t$ is the ball state at time t.

Furthermore, the edges indicate dependencies between the variables. For instance, in Figure 5 the ball motion model index m_t depends on $m_{t-1}, T_{t-1}, T'_{t-1}, s_t$ and $\mathbf{x}_{t-1}$, hence there are edges coming from the latter five variables to m_t.

For the rest of this section, we give the ball-tracking algorithm following Figure 5. The teammember-tracking algorithm can be obtained similarly following Figure 4.

We use the sequential Monte Carlo method to track the motion model m and the object state $\mathbf{x}$. Particle filtering is a general purpose Monte Carlo scheme for tracking in a dynamic system. It maintains the belief state at time t as a set of particles $p_t^{(1)}, p_t^{(2)}, \cdots, p_t^{(N_s)}$, where each $p_t^{(i)}$ is a full instantiation of the tracked variables $\{p_t^{(i)}, w_t^{(i)}\}$, $w_t^{(i)}$ is the weight of particle $p_t^{(i)}$ and N_s is the number of particles. In our case, $p_t^{(i)} = \langle \mathbf{x}_t^{(i)}, m_t^{(i)} \rangle$.

The equations below follow from the ball-tracking DBN.

$$m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, T_{t-1}, T'_{t-1}) \tag{9}$$

$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)}) \tag{10}$$

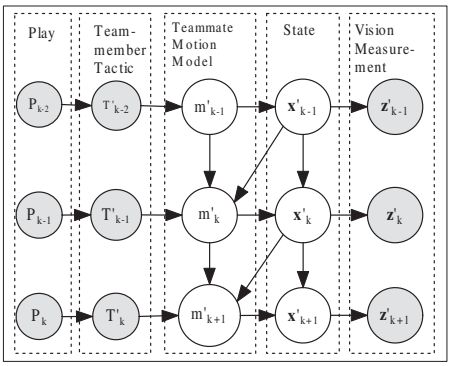

Fig. 4. A dynamic Bayesian network for teammember tracking with a Segway RMP robot. Filled circles represent deterministic variables which are observable or are known as the tactic or the play that the robot is executing.

Note that T_{t-1} and T'_{t-1} are inferred deterministically from P_{t-1} instead of sampling. Also note that in Equation 10, the ball state is conditioned on the ball motion model $m_t^{(i)}$ sampled from Equation 9.

Then we use the Sample Importance Resampling (SIR) algorithm to update the state estimates. The sampling algorithm is as follows:

$$[\{\mathbf{x}_t^{(i)}, m_t^{(i)}, w_t^{(i)}\}_{i=1}^{N_s}] = \textbf{\textit{SIR}}[\{\mathbf{x}_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_s}, \mathbf{z}_t, s_t, T_{t-1}, T'_{t-1}]$$

01 **for** $i = 1 : N_s$
02 draw $m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, T_{t-1}, T'_{t-1})$.
03 draw $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)})$.
04 set $w_t^{(i)} = p(\mathbf{z}_t | \mathbf{x}_t^{(i)})$
05 **end for**
06 Calculate total weight: $w = \sum [\{w_t^i\}_{i=1}^{N_s}]$
07 **for** $i = 1 : N_s$
08 Normalize: $w_t^i = w_t^i / w$
09 **end for**
10 ***Resample***.

The inputs of the algorithm are samples drawn from the previous posterior $\langle \mathbf{x}_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)} \rangle$, the present vision and infrared sensory measurement $\mathbf{z}_t, s_t$, the robot's tactic T_{t-1}, and the teammember's tactic T'_{t-1}. The outputs are the updated weighted samples $\langle \mathbf{x}_t^{(i)}, m_t^{(i)}, w_t^{(i)} \rangle$. In the sampling algorithm, first, a new ball motion model index, $m_t^{(i)}$, is sampled according to Equation 9 at line 02. Then given the model index, and previous ball state, a new ball state is sampled according to Equation 10 at line 03. The importance weight of each sample is given by the likelihood of the vision measurement given the predicted new ball state at line 04. Finally, each weight is normalized and the samples are resampled. Then we can estimate the ball state based on the mean of all the $\mathbf{x}_t^{(i)}$. Similarly the state of the teammember $\mathbf{x}'_t$ can be obtained from the teammember tracker.

V. Experiment

From previous work we knew the initial speed and accuracy of the ball velocity after a kick motion. We profiled the system and measurement noise as well. In this section, we evaluate the effectiveness of our tracking system in both simulated and real-world tests.

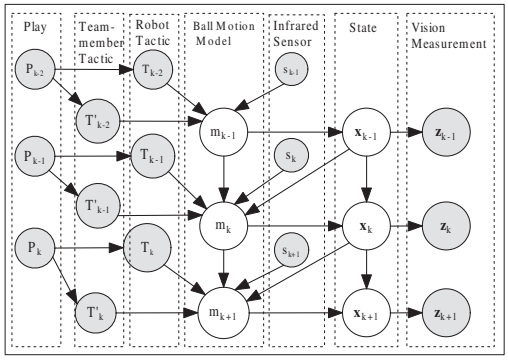

Fig. 5. A dynamic Bayesian network for ball tracking with a Segway RMP robot.

TABLE I

THE AVERAGE RMS ERROR OF POSITION ESTIMATION AND VELOCITY ESTIMATION FROM HUMAN TRACKERS AND BALL TRACKERS.

Motion Model	Single Model	Multi-Model
Human Position Est RMS (m)	0.0030	0.0014
Human Velocity Est RMS (m/s)	0.42	0.025
Ball Position Est RMS (m)	0.0028	0.0017
Ball Velocity Est RMS (m/s)	0.4218	0.0597

A. Simulation Experiments

Because it is difficult to know the ground truth of the object's position and velocity in the real robot test, we do the simulation experiments to evaluate the precision of tracking.

Experiments are done following the *Naive Offense* play, in which the robot acts as the receiver and the human teammember acts as the passer. Noises are simulated according to the model we profiled in previous work. In the beginning, the teammember holds the ball. After a fixed amount of time, the ball is kicked towards the robot, and the teammember moves forward to a predefined location.

We implement both a single model tracker and a play-based multi-model tracker for the ball and the teammember. We simulate the experiment for 50 runs, and then compare the performance of the two trackers with different implementations. The average RMS error of position estimation and velocity estimation are shown in Table I. The results show that the play-based multi-model scheme performs much better than the single model especially in terms of velocity estimation. Because with the play-based motion model, when the ball is being kicked, most particles evolving using the transition model determined by the play will change its motion model $m_t^{(i)}$ from *Free-Ball* to *Human-Kick-Ball*, and a velocity will be added to the ball accordingly.

Figure 6 and Figure 7 show the ball velocity estimation and the teammember velocity estimation during a short term for a given simulation test. In both figures, The left graph shows the x-component of the velocity (v_x) estimation through single model tracking and through play-based multi-model tracking. The right graph shows the y-component of the velocity (v_y)

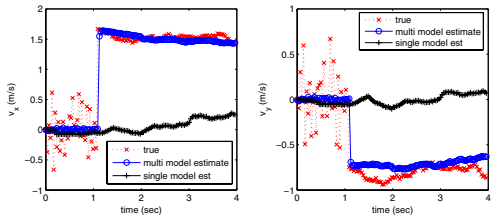

Fig. 6. Ball velocity estimation.

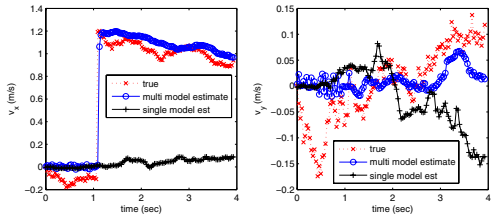

Fig. 7. Human teammember velocity estimation.

estimation. The dotted line with x-mark represents the true value, the solid line with circle represents the the velocity estimation through play-based multi-model tracking, the solid line with cross represents the the velocity estimation through single model tracking. We note that the velocity estimation with multi-model trackers approximate the true velocity in terms v_x and v_y much more consistently than with single model trackers, since the former tracker switches model effectively responding to the actual change happened on the tracked object.

B. Team Cooperation Test

In the real-world test, we do experiments on the Segway RMP soccer robot executing the offensive play and coordinating with the human teammember. The test setup is demonstrated in Figure 8, in which the digits along the lines show the sequence of the whole strategy, the filled circle at position B represents the robot, the unfilled circle at position E represent an opponent player, and the shaded circle represent the human teammember.

When each run begins, the human teammember is at position A . With this team cooperation plan (play), the robot chooses the tactic *CatchKickToTeammember* to execute, in which the robot starts with the skill *Search-Ball*. When the robot finds the ball, the teammember passes the ball directly to the robot and chooses a positioning point to go to either at C or D. The robot grabs the ball after the ball is in the catchable area and is detected by the infrared sensor (skill *Grab-Ball*). Next the robot searches for the teammember holding the ball with its catcher (skill *Search-Teammember*). After the robot finds the teammember, the robot kicks the ball to its teammember and the teammember shoots at the goal(skill *KickToTeammember*, completing the whole offensive play. Each run ends in one of the following conditions.

- succeed if the human receives the ball from the robot or the human does not receiver the ball but the pass can be

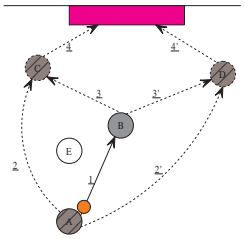

Fig. 8. A demonstration of a naive team cooperation plan in offensive scenario. The digits along the lines show the sequence of the whole plan. The filled circle at position B represents the robot. The unfilled circle at position E represent an opponent player. The shaded circle represent the human teammember.

TABLE II

THE AVERAGE TIME TAKEN OVER ALL THE SUCCESSFUL RUNS.

Motion Model	Single Model	Multi-Model
Mean Time (sec)	33.4	22.6

considered as a "good" one.

- fail if the robot is in searching for the ball or the teammember for more than 30 seconds.
- fail if the ball is out of the field before the robot catches it.

In the experiment over 15 runs, the robot with single model trackers fails 5 of the total. While the robot with play-based multi-model trackers fails 2 of the total. We also keep track of the mean time taken in all the successful runs. We list the result in Table II. Using play-based multi-model tracking saves 32.3% time in terms of completing the whole play over single model tracking. During the experiment, we note that when using the single model tracking, most time were spent on searching the teammember. Incorporating the team cooperation knowledge known as play into the teammember motion modelling greatly improves the accuracy of the teammember motion model and therefore avoids taking time in searching a lost target from scratch.

VI. Related Work

Tracking moving objects using a Kalman filter is the optional solution if the system follows a single model, f and h in Equation 1 and 2 are known linear functions and the noise $\mathbf{v}$ and $\mathbf{n}$ are Gaussians [8]. Multiple model Kalman filters such as Interacting Multiple Model (IMM) are known to be superior to the single Kalman filter when the tracked object is maneuvering [9]. For nonlinear systems or systems with non-Gaussian noises, a further approximation is introduced, but the posterior densities are therefore only locally accurate and do not reflect the actual system densities.

Since the particle filter is not restricted to Gaussian densities, a multi-model particle filter is introduced. However, this approach assumes that the model index, m, is governed by a Markov process such that the conditioning parameter can branch at the next time-step with probability $p(m_t = i|m_{t-1} = j) = h_{i,j}$ where $i, j = 1, \cdots, N_m$. But the uncertainties in our object tracking problem do not have such a property due to the interactions between the robot and the tracked object. In this motivation, a tactic-based motion modelling method is proposed in [4]. Based on that approach, we further introduce the play-based motion modelling method when team coordination knowledge is available. In [3], an approach were proposed for tracking a moving target using Rao-Blackwellised particle filter. They use a fixed transition table between different models. Our transition model is dependent on the play that the robot is executing and the additional information that matters. This play-based motion modelling can be flexibly integrated into our existing skills-tactics-plays architecture.

VII. Conclusions and Future Work

Motivated by the interactions between a team and the tracked object, we contribute a method to achieve efficient tracking through using a play-based motion model and combined vision and infrared sensory information. This method gives the robot a more exact task-specific motion model when executing different tactics over the tracked object (e.g. the ball) or collaborating with the tracked object (e.g. the teammember). Then we represent the system in a compact dynamic Bayesian network and use particle filter to keep track of the motion model and object state through sampling. The empirical results from the simulated and the real experiments show the efficiency of the multi-model tracking over single model tracking.

Future work will include modelling the multi-target motion when each object has multiple hypothesis, which is caused by incorrect measurements originating from the clutter. We would like to see how the information from the tactic and the play can help to eliminate false alarms and achieve efficient resampling under the framework of the particle filter.

References

[1] A. Doucet, N. D. Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.

[2] D. Schulz, W. Burgrad, and D. Fox, "People tracking with mobile robots using sample-based joint probabilistic data association filters," *International Journal of Robotics Research*, vol. 22(2), 2003.

[3] C. Kwok and D. Fox, "Map-based multiple model tracking of a moving object," *Proceedings of eight RoboCup International Symposium*, July 2004.

[4] Y. Gu, "Tactic-based motion modelling and multi-sensor tracking," *Proceedings of Twentieth National Conference on Artificial Intelligence*, 2005.

[5] J. Searock, B. Browning, and M. Veloso, "Turning Segways into Soccer Robots," in *Proceedings of IROS'04*, Sendai, Japan, September 2004.

[6] B. Browning, J. Bruce, M. Bowling, and M. Veloso, "STP: Skills, tactics and plays for multi-robot control in adversarial environments," *IEEE Journal of Control and Systems Engineering*, vol. 219, pp. 33–52, 2005.

[7] M. Veloso, B. Browning, P. Rybski, and J. Searock, "Segwayrmp robot football league rules," http://www.cs.cmu.edu/ robosoccer/segway/, Tech. Rep., 2005.

[8] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002. [Online]. Available: citeseer.ist.psu.edu/article/arulampalam01tutorial.html

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