Effective Multi-Model Motion Tracking using Action Models

Yang Gu and Manuela Veloso

The International Journal of Robotics Research 2009; 28; 3
DOI: 10.1177/0278364908097587

Yang Gu
Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, USA
{guyang,velosa}@cs.cmu.edu

Effective Multi-Model Motion Tracking using Action Models

Abstract

We consider tasks where robots act on the target that is visually tracked, such as kicking a ball or pushing an object. We introduce a principled approach to incorporate models of the robot-object interaction into the tracking algorithm to effectively improve the performance of the tracker. We first present the integration of a single robot behavioral model with multiple actions into our dynamic Bayesian probabilistic tracking algorithm. We then extend to multiple motion tracking models corresponding to known multi-robot coordination plans or from multi-robot communication. We evaluate our resulting informed-tracking approach empirically in simulation and using a setup Segway robot soccer task. The input of the multiple single and multi-robot behavioral models allows a robot to visually track mobile targets with dynamic trajectories more effectively.

KEY WORDS—multi-model, motion tracking, human-robot-team, action models

1. Introduction

Object tracking is widely used in robot applications (e.g. Schulz et al. 2003). As mobile robots become more common in everyday life, interactions between a robot and a target being tracked increase, leading to trajectories of the objects that are not captured by a simple single motion model. Robots act on targets changing their location and velocity (Kwok and Fox 2004). In this article, we address this specific problem of visual tracking when single and multiple robots act on the object being tracked. The motivating example for the general algorithmic approach we contribute is a Segway RMP soccer robot interacting with a ball and teammates, such as grabbing and kicking (Browning et al. 2005b).

The International Journal of Robotics Research
Vol. 28, No. 1, January 2009, pp. 3–19
DOI: 10.1177/0278364908097587

Figures 1, 2, 11–18 appear in color online: http://ijr.sagepub.com

Object tracking efficiency directly depends on the accuracy of the motion model and of the sensory information. The motion model of the object becomes particularly complex under action, as well as highly dependent on the specific robot's actions. In addition to the actions of a single robot, we also consider multiple team member robots performing actions on the object being tracked, creating a further discontinuous and nonlinear object motion. Our approach builds upon two main facts: (i) an individual robot knows its own actions; and (ii) robots in a team collaborate according to predefined coordination plans or dynamic communication (Gu 2005; Gu and Veloso 2006b). We claim that knowledge of the single robot control strategy and the multi-robot coordination is a valuable source of information for tracking (Gu and Veloso 2006a,c,d).

In order to concretely motivate how robot action models, in particular team actions, contribute to improved tracking performance, we provide a brief illustrative example. Figures 1 and 2 show a sequence of views from a Segway robot's pan-tilt camera (Browning et al. 2004). Each picture corresponds to one frame identified by the frame id shown at the lower right corner. The frame rate is approximately 30 frames per second and the camera has a limited field of view (FOV). When the object is in the FOV of the camera, tracking is fine. When the object leaves the FOV of the camera, the prediction of the tracker is used to keep the camera pointed to the object: the prediction from the tracker is translated into a command to position the camera in order for the object to be consistently located in the camera's FOV.

Figure 1 shows the first example in which a robot is using our developed tracking algorithm (frames 431–477). Figure 2 shows the second example in which a robot is tracking without our algorithm (frames 486–710). Initially, the ball is static and the robot finds the ball at frame 431 and at frame 486 in the two examples, respectively. We describe the examples for their sequence of relevant frames as follows.

- From frames 431–477: The robot integrates the team action model into tracking. At frame 431, the ball is kicked towards the robot. Although the ball moves fast, the robot quickly finds the ball at frame 445 based on the

Fig. 1. Motivation example: from frame 431 to 477.

Fig. 2. Motivation example: from frame 486 to 710.

knowledge that the team member passes the ball to it. The ball is temporarily not seen and it takes the robot only about 15 frames (0.5 s) to find the ball and then the robot keeps tracking the ball.

- From frames 486–710: A new trial starts without our incorporation of the team action model. The ball is kicked towards the robot, but it now takes approximately 120 frames (4 s) for the robot to find the ball. After frame 608, the ball is in the FOV of the camera and the robot tracks it correctly.

The superiority of our tracking algorithm, as depicted in the example, is due to the inclusion of the team action models into the tracker. The robot knows that its team member will pass the ball towards it. When the ball is temporarily not seen, the robot calculates the estimated ball position with the motion model under the team member action, trying to find the ball at the estimated position which is right beneath its kicker. Furthermore, when the team member announces the action as soon as it kicks the ball (not shown in the example, but also introduced in our work), the robot selects a correct motion model and tracks the ball even more effectively. Instead, when the control strategy integration is disabled, the robot has great difficulty locating the ball in a short time because the robot's action (e.g., at frame 486) makes the motion of the ball highly discontinuous and nonlinear. The robot has to track for the ball

without an appropriate motion model, leading to its long plain search.

The example supports our work in exploiting different kinds of non-standard (prior and dynamic) information into the tracker to produce better estimates of the object state. The techniques that we describe are applicable to any domain that includes one or a team of agents cooperating on a specific task and acting on the target objects for their tracking. Agents act using behaviors and cooperate using predefined team plans. Information about action on the objects can also be sent as a communication message to a specific team member to enable the update of the motion model of the tracked object. We use a probabilistic temporal Bayesian modeling framework to represent and recognize the multiple motion models.

The contributions of our work, as reported in this article, include the following.

- We incorporate single robot and team action models into a Dynamic Bayesian Network (DBN)-based temporal representation for tracking.
- We introduce several multi-model tracking algorithms based on: the robot's own actions; predefined team plays; and communicated team actions.
- We evaluate our new tracking algorithms in robot platforms, a human-robot team and in simulation experiments.
- We present an empirical comparison between several tracking algorithms. We examine the performance of each algorithm according to a variety of metrics.

The article is organized as follows. We introduce the Segway RMP soccer robot and two of its main components related to this article. We identify the challenges of this domain and give a problem statement. We briefly describe the relations between the robot cognition and the object motion models. We show how we use the team action models in a DBN-based representation. We describe the multi-model tracking algorithm, leading to our experimental results. Finally, we present related work and a conclusion.

2. Segway RMP Soccer Robot

The Segway platform is unique due to its combination of wheel actuators and dynamic balancing. The Segway RMP (Robot Mobility Platform) provides an extensible control platform for robotics research. It endows the robot with the novel characteristics of a fast platform and can travel long ranges. It is able to carry significant payloads, can navigate in relatively tight spaces for its size and provides the opportunity to mount sensors at a height comparable to human eye level.

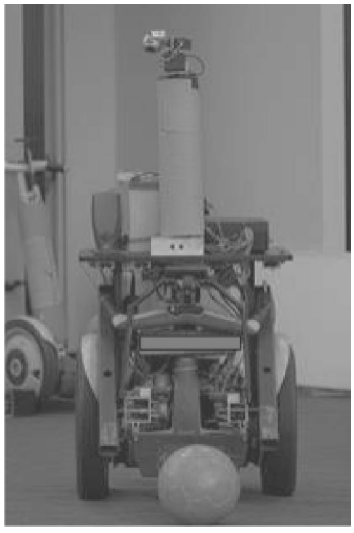

Fig. 3. The Segway RMP soccer robot equipped with a kicker, a catcher, infrared sensors and a camera mounted on a custom pan-tilt unit (Searock et al. 2004).

In our previous work, we have developed a Segway RMP robot base capable of playing Segway soccer (Searock et al. 2004) (Figure 3). We briefly describe the two major components of the control architecture, namely the sensors and the robot cognition, which are highly related to our motion tracking algorithm.

2.1. Vision Sensor and Infrared Sensors

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 of relevance to the robot. We use one pan-tilt camera as the vision sensor (Browning et al. 2004). We do not discuss the localization of the robot in the sense that many soccer tasks can be performed by the Segway RMP robot without localization knowledge.

We have equipped each robot with infrared sensors (IR) to reliably detect an object located in the catchable area of the robot. Its measurement is a binary value indicating whether or not an object is in the area. In most cases, the catchable area is the blind area of the pan-tilt camera. Therefore, the infrared sensor is particularly useful when the robot is grabbing the ball (Searock et al. 2004).

2.2. Robot Cognition

The Skills-Tactics-Plays (STP) control architecture (Browning et al. 2005a) achieves the goals of responsive team control. One of the key components of STP is the division between

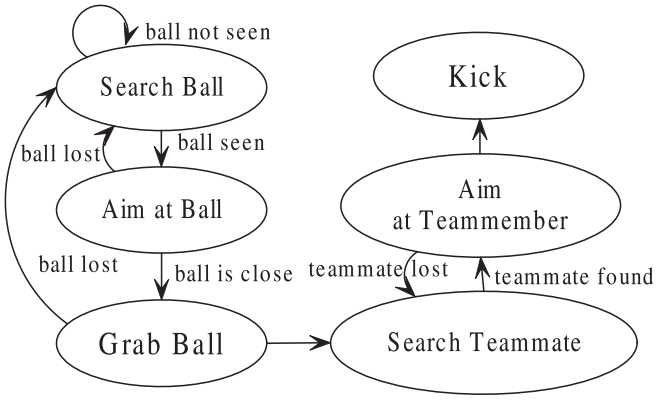

Fig. 4. Skill state machines (SSMs) for an example tactic: `CatchKickToTeammate`.

Table 1. An example play.

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

single robot behavior (skills and tactics) and team behavior (plays).

We construct the robot cognition using an STP-based architecture. Plays, tactics and skills form a hierarchy for team control. Plays control the team behavior through combination of 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 4 shows the Skill State Machines (SSMs) and transitions for an example tactic: `CatchKickToTeammate`. (Note that the granularity of the skills used is domain-dependent. The goal is for skills to capture the low-level robot hardware specific behaviors.) The tactic describes a sequence of six individual skills to search the ball, aim and grab the ball, search and aim at the teammate and finally kich the ball to the teammate. Each node in the figure is a skill. The edges show the transitions between skills based on perceptual information.

Robot soccer is a team sport, therefore the building of our team strategy requires not only execution of single robot behaviors, but also coordination with the team member. Table 1 shows an example play. The name of the play is `Naive Offense`. The play is only applicable in an offensive situation. The termination condition is either play aborted or the situation changed and our team is not in offensive, e.g. a turn-over of ball possession. There are two roles in this play, one passes the ball to the other who positions downfield and waits to receive a pass. Once the positioning is done (e.g. the other is closer to the opponent goal), the passing is performed.

The role assignment is done at run time in a distributed way based on each robot's own observation. Each robot selects a role based on its own data without communication. For example, should the robot think the team member is closer to the ball, the robot (ROLE 2) would choose to position and receive the ball from its team member (ROLE 1). Furthermore, the robot has deterministic knowledge of which play the team is using. The robot makes an assumption that its team member is performing the same game play as itself. The robot infers what tactic the team member is executing from the team play. For example, after receiving the ball from the team member as a passer, the robot would assume the team member will go forward to a tactically advantageous position to receive a pass.

3. Problem Statement and Basic Approach

In a Segway robot soccer game, there are multiple moving objects on the field e.g. the ball, the human teammate and the two opponents. Each team is identified by their distinct color. The ball is orange (Veloso et al. 2005). In our test, we are not limited to using only human teammates. We also consider multiple robot team members. The challenges of tracking in such a dynamic, adversarial and noisy environment include the following.

- **Inaccurate sensing**: Each robot uses a color pan-tilt camera as its primary sensor to perceive the world. Each robot is equipped with infrared sensors (IR) to detect the objects located in the catchable area of the robot.
- **Limited visual scope:** The camera has only a limited FOV. It is difficult to consistently keep an object in the FOV of the camera.
- **Complex motion of the sensor:** The sensors (camera and IR) are not located on a static position. Instead, the sensors are mounted on a mobile robot (the Segway RMP with its balancing motion is particularly challenging). The moving trajectory of the robot does not follow any particular route. Robot dash and stop, changing their velocity.
- **Interaction between robots and objects:** Robots manipulate the ball with their kickers and catchers. The actuation on the tracked object makes the motion of the object highly non-continuous.

The problem of tracking can be abstracted as follows:

- n moving robots (including 1 human);
- all the robots act on the object;
- robots act using behaviors;
- robots cooperate using team plans;
- robots communicate;
- one moving object (the ball);
- M motion models which are functions of available information; and
- deciding which motion model to take depends on the availability of the different sources of information including the robot's own actions, team plays and communicated information.

The general parameterized state-space system for describing the object state at time t is given by:

$$\mathbf{x}_t = f^m(\mathbf{x}_{t-1}, \mathbf{u}^m_{t-1}, \mathbf{v}^m_{t-1}) \quad (1)$$

$$\mathbf{z}_t = h^m(\mathbf{x}_t, \mathbf{w}^m_t) \quad (2)$$

where f^m and h^m are the parameterized state transition and measurement functions for the mth motion model of the object; $\mathbf{x}$, $\mathbf{u}$, $\mathbf{z}$ are the state, input and measurement vectors; $\mathbf{v}$, $\mathbf{w}$ are the process and measurement noise vectors of known statistics; m is the model index that can take any one of M values and M is the number of motion models of the object being tracked. If the model index m is governed by a discrete-state Markov chain with transitional probabilities

$$h_{i,j} = P\{m_t = i | m_{t-1} = j\}, (i, j \in S),$$

where $S = \{1, 2, \ldots, M\}$ and the transitional probability matrix $\mathcal{H} = [h_{i,j}]$ is an $M \times M$ matrix. We can represent such a system using a jump Markov model (Ristic et al. 2004). In this article, the decision on which model to use and how to transition from one model to another ($h_{i,j}$) are based on the team action models i.e. it is an extension of the ordinary jump Markov model.

4. Object Tracking using Tactic-Based Models

In this section, we take the ball-tracking problem as a detailed example to show how to use a single robot's own actions (tactics).

4.1. From Tactics to Motion Models

Tactics are the topmost level of our single robot control. Each tactic encapsulates a single robot behavior. Each tactic is parameterized allowing for more general tactics to be created to be applicable to a wider range of world states. As described above, each tactic instantiates actions through the skill layer (Browning et al. 2005a).

In our Segway RMP soccer robot environment, we define three models to model the ball motion according to all the possible robot tactics and corresponding actions on the ball.

- *Free* or *Free-Ball*: The ball is not moving at all or is moving straight with a constant speed decay d that depends on the environment surface. There are no external actions on the ball. The state-space model of the *Free-Ball* motion can be represented as:

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

$$\mathbf{z}_t = \mathbf{H}^1_t \mathbf{x}_t + \mathbf{w}^1_t \quad (4)$$

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

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

  where Δt is the time interval between vision frames.

- *Grabbed* or *Robot-Grab-Ball*: The ball is grabbed by the robot's catcher. In this case, 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 (plus noise) and its global position at time t is the robot's global position plus fixed relative position plus noise. These two noise components form the noise vector $\mathbf{v^2}$. We use the same measurement model as Equation (4).

- *Kicked* or *Robot-Kick-Ball*: The ball is kicked. Its velocity is equal to a predefined initial speed plus the noise, and its position is equal to its previous position plus the noise. These two noise components form the noise vector $\mathbf{v^3}$. We use the same measurement model as Equation (4).

The model index m determines the present motion model being used. For our ball tracking example, $m = 1, 2, 3$, which

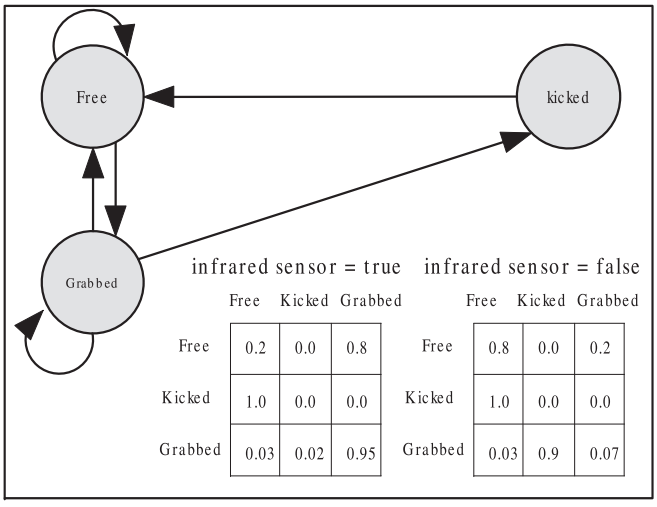

Fig. 5. Ball motion model for tactic `CatchKickto-Teammate`. Each node is a single motion model. The tables list the transition probability between any two motion models.

represents the motion model *Free*, *Grabbed* and *Kicked*, respectively. In our approach, it is assumed that the model index m_t, conditioned on the previous tactic executed $\mathcal{T}_{t-1}$ and other useful information $\mathcal{I}_t$ (such as ball state $\mathbf{x}_{t-1}$, infrared measurement s_t or the combination of two or more variables), transits from j to i with probability

$$h_{i,j} = \mathrm{P}(m_t = i | m_{t-1} = j, \mathcal{T}_{t-1}, \mathcal{I}_t) \tag{5}$$

where $i, j = 1, \ldots, M$.

With this tactic-based modeling method, we can obtain the corresponding motion models for the tactics shown in Figure 5. We use the infrared binary measurement as the branching parameter. The two tables list the transition probabilities between any two models conditioned on 'the infrared sensor can/cannot sense the ball', respectively. For example,

- $\mathrm{P}(Free|Free, IR = False) = 0.8$. The infrared sensor reading is false and the previous ball model is *Free*. The ball remains in its *Free* model with a relatively high probability.
- $\mathrm{P}(Grabbed|Free, IR = True) = 0.8$. The infrared sensor reading is true and the previous ball model is *Free*. It is very likely that the ball is just grabbed by the robot given the infrared sensor reading.
- $\mathrm{P}(Free|Grabbed, IR = True) = 0.03$. The infrared sensor reading is true and the previous ball model is *Grabbed*. The sensor reading indicates that the ball is grabbed. However, it is possible that the robot loses the control of the ball due to fault operation with a very small probability.

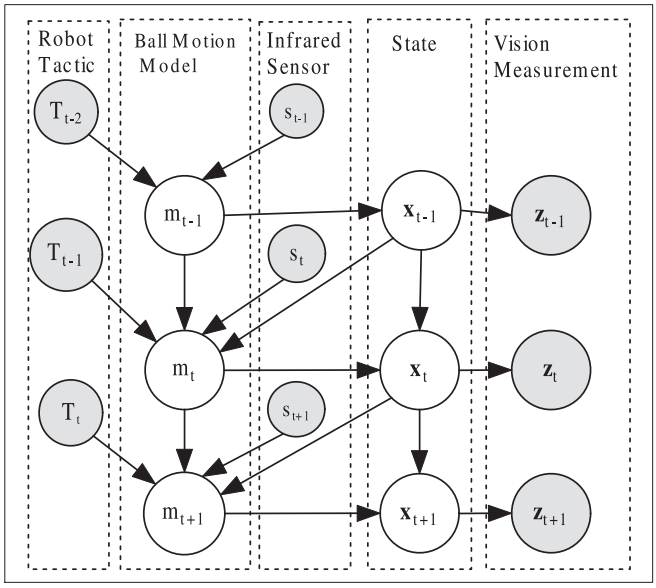

Fig. 6. TBMM: A dynamic Bayesian network for ball-tracking with a Segway RMP robot using its own tactic. Filled circles represent deterministic variables which are observable or are known as the tactic that the robot is executing.

In this way, we can build motion models for any existing tactics we have designed.

4.2. Tactic-Based Object Tracking

We use a dynamic Bayesian network (DBN) to represent the Tactic-Based Motion Model (TBMM) in a natural and compact way as shown in Figure 6.

In this graph, the system state is represented by variables:

- tactic $\mathcal{T}$;
- infrared sensor measurement s;
- ball state $\mathbf{x}$;
- ball motion model index m; and
- vision sensor measurement $\mathbf{z}$.

Each variable takes on values in some space. The variables change over time in discrete intervals, so that $\mathbf{x}_t$ is the ball state at time t. Furthermore, the edges indicate dependencies between the variables. For instance, the ball motion model index m_t depends on m_{t-1}, $\mathcal{T}_{t-1}$, s_t and $\mathbf{x}_{t-1}$, hence there are edges coming from the latter four variables to m_t.

We use the sequential Monte Carlo method to track the motion model m and the ball state $\mathbf{x}$. Particle filtering is a general purpose Monte Carlo scheme for tracking in a dynamic

Table 2. TBPF algorithm.

$[\{\mathbf{x}_t^{(i)}, m_t^{(i)}, w_t^{(i)}\}_{i=1}^{N_s}] = \text{TBPF}[\{\mathbf{x}_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_s}, \mathbf{z}_t, s_t, \mathcal{T}_{t-1}]$

1 **for** $i \leftarrow 1$ **to** N_s
2 **do** draw $m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_{t-1})$
3 draw $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)})$
4 $w_t^{(i)} \leftarrow p(\mathbf{z}_t | \mathbf{x}_t^{(i)})$
5 Calculate total weight: $w \leftarrow \sum[\{w_t^{(i)}\}_{i=1}^{N_s}]$
6 **for** $i \leftarrow 1$ **to** N_s
7 **do** Normalize: $w_t^{(i)} \leftarrow w_t^{(i)}/w$
8 Resample

system. It maintains the belief state at time t as a set of particles $p_t^{(1)}, p_t^{(2)}, \ldots, p_t^{(N_s)}$, 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, $p_t^{(i)} = \langle x_t^{(i)}, m_t^{(i)} \rangle$.

The equations below follow from the DBN:

$$m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_{t-1}) \quad (6)$$

$$\mathbf{x}_t^{(i)} \sim p(x_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)}). \quad (7)$$

Note that in Equation (7), the ball state is conditioned on the ball motion model $m_t^{(i)}$ sampled from Equation (6). Table 2 shows our Tactic-Based Particle Filtering (TBPF) algorithm used by the tracker to update the state estimates.

The inputs to the TBPF 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 and the tactic $\mathcal{T}_{t-1}$. The outputs are the updated weighted samples $\langle \mathbf{x}_t^{(i)}, m_t^{(i)}, w_t^{(i)} \rangle$. In the sampling algorithm, a new ball motion model index $m_t^{(i)}$ is first sampled according to Equation (6) at Line 2. Then given the model index, and the previous ball state, a new ball state is sampled according to Equation 7 at Line 3. The importance weight of each sample is calculated by the likelihood of the vision measurement given the predicted new ball state at Line 4. Finally, each weight is normalized and the samples are resampled. We then estimate the ball state based on the mean of all the $\mathbf{x}_t^{(i)}$.

5. Incorporating Team Action Models into Object Motion Models

In this section, we show how to incorporate team action models with two different types of available information: team plays and communicated information.

5.1. From Plays to Motion Models

Plays form the highest level in the control hierarchy providing strategic level control of the entire team. A tactic, as determined by the executing play, is created with the parameters defined for the play. That tactic then continues to execute until the play transitions to the next tactic in the sequence (Browning et al. 2005a).

In our Segway RMP soccer robot environment, we define two more models to model the ball motion.

- *Human-Grab-Ball*: The ball is held by the teammate. The robot can infer the ball position if the robot knows the teammate position well.
- *Human-Kick-Ball*: The ball is kicked by the teammate and it is supposed to be either a pass to the robot or a shoot at the goal.

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

- *Random Walk*: The teammate is wandering in the field. The state at the new time is the state at the current time with some additive zero-mean (assumed Gaussian) noise.
- *Holding Ball*: The teammate 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 teammate position using the ball position in a similar way as *Grabbed*.
- *Accelerating/Stopping*: The teammember dashes and obtains a velocity or stops in a short time.
- *Positioning*: The teammate is going to a predefined tactical position with a constant speed. This case happens mostly after the teammate passing the ball to the robot and moving down the field toward the opponent's goal.

Given the knowledge of the team coordination plan (the play $\mathcal{P}_{t-1}$ at time $t-1$), the robot can infer what tactic the teammate is executing ($\mathcal{T}'_{t-1}$), which provides valuable information about the motion model of the teammate (m'_t). Both the robot and the teammate act on the ball in a Segway soccer game. The motion model of the ball (m_t) is therefore affected by what tactic the robot ($\mathcal{T}_{t-1}$) and the teammate ($\mathcal{T}'_{t-1}$) are executing. The transition probability of the teammate model is as follows:

$$h'_{i,j} = \text{P}(m'_t = i | m'_{t-1} = j, \mathcal{T}'_{t-1}, \mathcal{I}'_t), \quad (8)$$

where $i, j = 1, \ldots, M'$ and M' is the number of teammate motion models. Since T'_{t-1} can be determined by $\mathcal{P}_{t-1}$, we get

$$h'_{i,j} = \text{P}(m'_t = i | m'_{t-1} = j, \mathcal{P}_{t-1}, \mathcal{I}'_t). \quad (9)$$

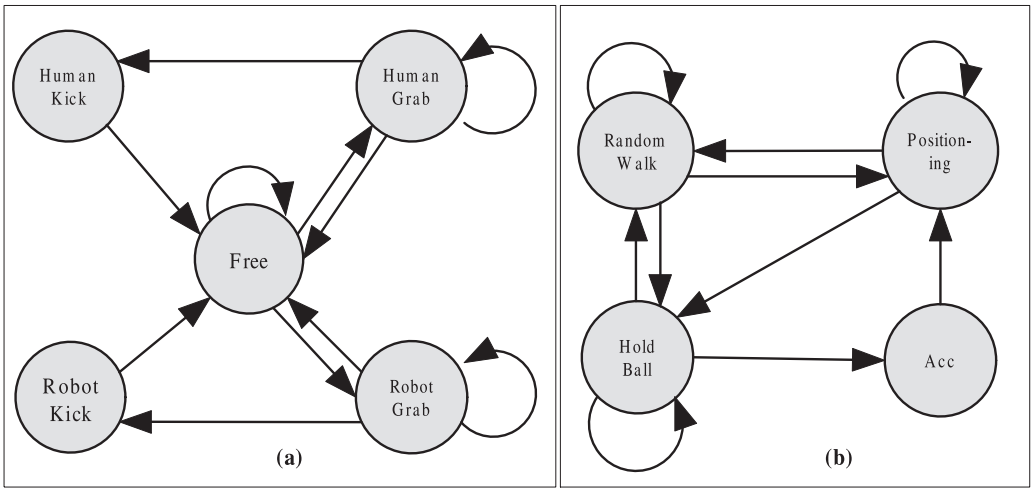

Fig. 7. Object motion modeling based on the play: `Naive Offense`. Each node is a model. Models transit to one another according to the predefined probabilities (not shown in the figure): (a) ball motion model; (b) human teammate motion model.

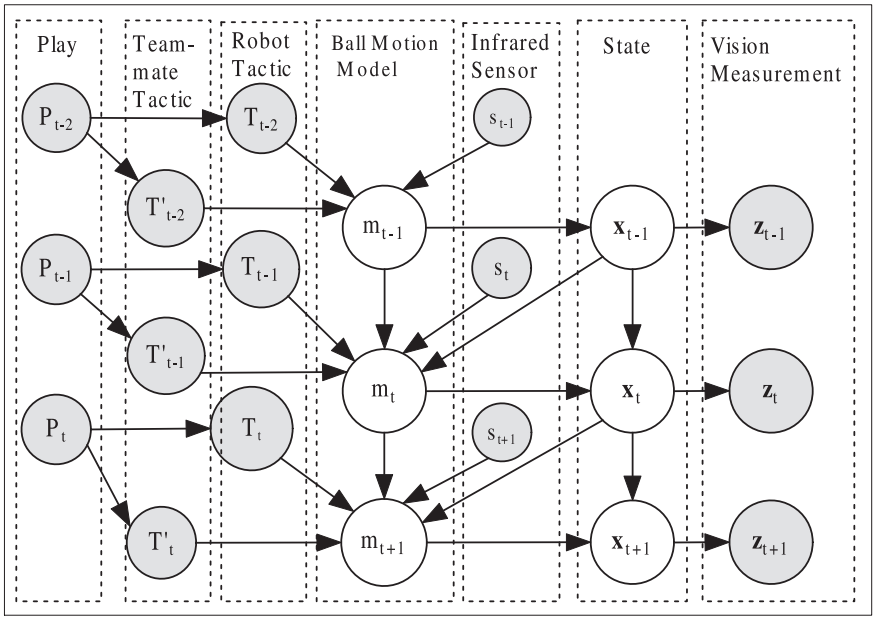

Fig. 8. PBMM: A dynamic Bayesian network for ball tracking with a Segway RMP robot using team play.

Similarly, the transition probability of the ball model is :

$$h_{i,j} = \mathrm{P}(m_t = i | m_{t-1} = j, \mathcal{T}_{t-1}, \mathcal{T}'_{t-1}, \mathcal{I}_t), \qquad (10)$$

where $i, j = 1, \ldots, M$. Since $\mathcal{T}_{t-1}, \mathcal{T}'_{t-1}$ can be determined by $\mathcal{P}_{t-1}$, we get

$$h_{i,j} = \mathrm{P}(m_t = i | m_{t-1} = j, \mathcal{P}_{t-1}, \mathcal{I}_t). \qquad (11)$$

If the current team play is the `Naive Offense` in Section 2.2, we can obtain the corresponding motion model transitions for both the ball and the teammate using the play-based method (Figure 7).

Figure 8 shows the updated DBN which includes the team play knowledge, representing a Play-Based Motion Model (PBMM). We introduce the Play-Based Particle Filtering algorithm (PBPF) which is similar to TBPF in Table 2, with the modification of the model index sampling step as follows:

$$m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_{t-1}, \mathcal{T}'_{t-1}). \qquad (12)$$

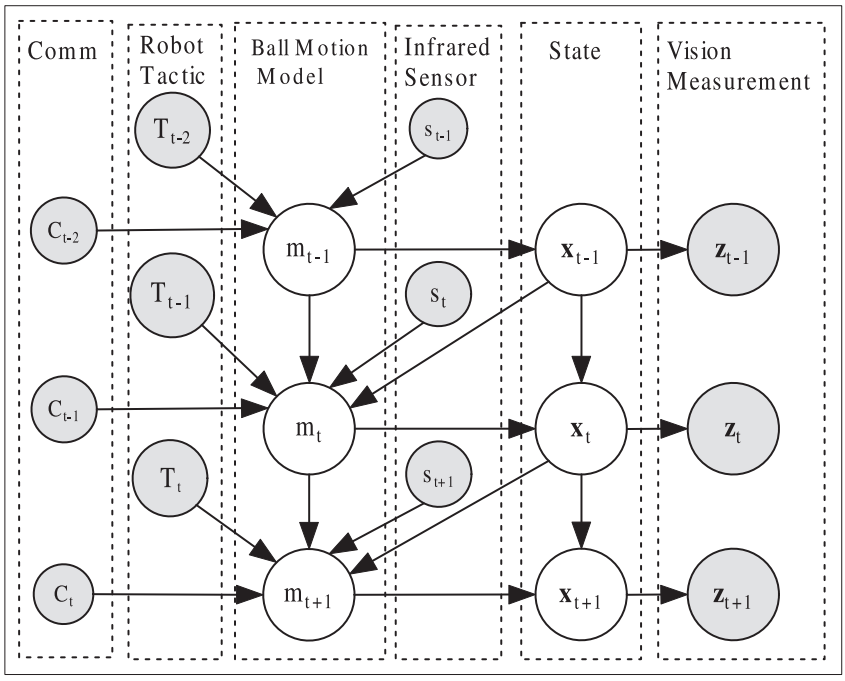

Fig. 9. CBMM: A dynamic Bayesian network for ball tracking with a Segway RMP robot using communication.

Note that $\mathcal{T}_{t-1}$ (the tactic of the robot) and $\mathcal{T}'_{t-1}$ (the tactic of the human teammate) are inferred deterministically from $\mathcal{P}_{t-1}$ based on predefined play instead of sampling.

5.2. Teammate Actuation Information

We use the same models as in Section 5.1. We have a different strategy on how to infer which model to use and how to transit from one model to another. Briefly speaking, when a team play is used, we assume the robot infers the teammate's tactic from the present team play, and assign transitional probabilities based on the assumed teammate's tactic. In this section, we obtain teammate's action from explicitly communicated information.

Current communication between robots is through peer-to-peer User Datagram Protocol (UDP) sockets. Each announcement is repeated for several time-steps to avoid possible data loss in transmission. We define three types of communication messages in terms of different action models. Each type of message has a unique message ID.

- HOLD: After grabbing the ball, the robot announces HOLD indicating the ball is under its kicker.
- SHOOT: The robot announces SHOOT when it kicks the ball to the opponent's goal.
- PASS: The robot announces PASS when it decides to pass the ball to its teammate.

With the communicated information, robots do not need to infer teammate's tactic from the team play any more (Gu and Veloso 2006c). Instead, every action on the ball is announced to keep the ball motion updated among team members.

Given the communication information ($\mathcal{C}_{t-1}$ = {NONE, HOLD, SHOOT, PASS}), the robot can infer which motion model the ball should take. The motion model of the ball (m_t) is therefore affected by what tactic the robot ($\mathcal{T}_{t-1}$) is executing and what action the teammate is performing. The transition probability of the ball model is :

$$h_{i,j} = p(m_t = i | m_{t-1} = j, \mathcal{T}_{t-1}, \mathcal{C}_{t-1}, \mathcal{I}_t). \quad (13)$$

Note that $\mathcal{C}_{t-1}$ = NONE indicates no message is received at time $t-1$, and $\mathcal{C}_{t-1} \neq$ NONE indicates a messaged referring to a teammate action is receive. The updated DBN to include communicated information, representing a Communication-Based Motion Model (CBMM), is shown in Figure 9.

We introduce the Communication-Based Particle Filtering algorithm (CBPF) which is similar to TBPF in Table 2, with the modification of the model index sampling step as follows:

$$m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_{t-1}, \mathcal{C}_{t-1}). \quad (14)$$

Note that the motion model is not dependent on the teammate's plays or tactics, since the communication message contains all the information presented by the teammate's tactics.

6. Experimental Results

This section investigates the performance of the tracking filters described in the above sections. We first describe how we acquire the parameters for the ball motion models and noise models. We test the proposed tracking filters by simulation and using a setup Segway robot soccer task.

6.1. Ball Motion and Measurement Noise Profiling

We assume we know the initial speed and accuracy of the ball velocity after a kick motion. Here, our goal is to estimate the ball speed decay d. We put the ball on the top of a ramp and let it roll off the ramp with initial speed $v_0 = \sqrt{2gh}$ without taking the friction on the surface of the ramp into account, where g is gravitational acceleration and h is the height of the ramp. We record the distance the ball travelled L from the position the ball rolls off the ramp to the position it stops. Obviously, the ball speed decay can be approximated as $d = 1 - v_0 \Delta t / L$ where $\Delta t \sim 0.033$ s. Following the test results, we use $d = 0.99$ for the cement surface. From the test, we note that the faster the ball's speed is, the smaller the system noise, hence the ball's trajectory tends to a straight line. We therefore model the system noise to be inversely proportional to the ball speed when the motion model is *Free-Ball*.

In order to profile the measurement noise, we put the ball on a series of known positions, read the measurement from vision sensor and then determine the error in that measurement. From the results, we know that the nearer the ball is, the smaller the observation noise. We therefore choose to approximate the error distribution as zero-mean and varying-variance Gaussians dependent on the distance from the robot to the ball.

6.2. Single Robot Results

6.2.1. Simulation Experiments

Because it is difficult to acquire ground-truthing data to verify the ball's position and velocity in the real robot test, we perform simulation experiments to evaluate the precision of our proposed tracking.

Experiments are done following the `ChaseBall` tactic (Figure 10). Noise is simulated according to the model we experimentally acquired. When the ball is near the robot and its speed is slower than $0.2\,\mathrm{m\,s^{-1}}$, it is kicked again with initial velocity $(0.4, 0)\,\mathrm{m\,s^{-1}}$. We then compare the performance of the tracker with a single model *Free-Ball* and the tracker with TBMM in Figure 5(a). After 50 runs, the results show that in terms of the average RMS error of position estimation, the single model tracker is 0.0055 m while the TBPF is 0.0027 m. In terms of the average RMS error of velocity estimation, the single model tracker is $0.05\,\mathrm{m\,s^{-1}}$ while the TBPF is $0.0052\,\mathrm{m\,s^{-1}}$. The TBPF performs much better than the single model tracking especially in terms of velocity estimation. Most particles in the TBPF (when the ball is near the robot and has a slow speed) evolving via the transition model, determined by the tactic `ChaseBall`, change their motion models $m_t^{(i)}$ from *Free-Ball* to *Kicked-Ball* and a velocity is added to the ball accordingly.

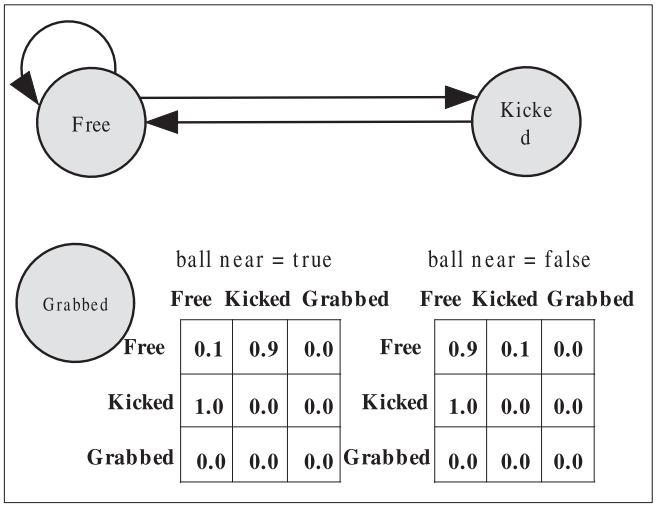

Fig. 10. Ball motion model for tactic `Chase-ball`.

In Figure 11, we compare the speed estimation of each tracker. The dark cross represents the true value of the ball speed and the gray circle is the estimated value. We note that the speed estimation in Figure 11(b) tracks the true speed very well. Each time the speed jumps (the ball is kicked), the estimation follows it perfectly. However, in Figure 11(a) following the single motion model *Free-Ball*, the speed estimation keeps decaying and cannot track the dynamic character of the ball speed.

6.2.2. Test on the Real Robot

In the real-world test, we do experiments on the Segway RMP soccer robot executing the tactic `Grab-and-Kick`. In all runs, the robot starts with the skill `Search`. When it finds the ball, the ball is kicked directly to the robot. The robot then grabs the ball after the ball is in the catchable area and is detected by the infrared sensor. Each run ends with the skill `Kick`. If the robot can still see the ball 2 s later after the kick, we count this run as successful. If the robot begins executing the skill `Search` a second time, we count that run as fail. That is to say, we only permit one search at the beginning of each run; after that, the robot should consistently keep track of the ball. Note that in the `Grab-and-Kick` tactic, the robot is commanded to search the ball if the ball is not visible in $t = 0.5$ s. In the experiments over 15 runs, the tracker

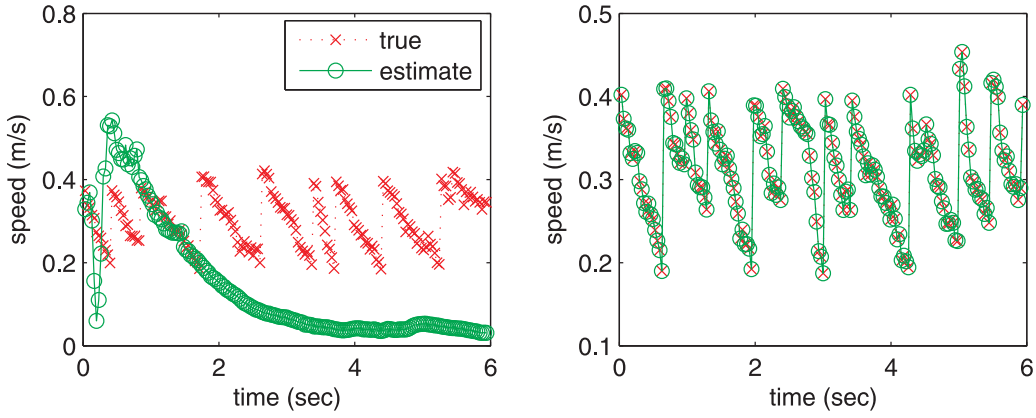

Fig. 11. `Chase-Ball` speed estimation: (a) single model tracking and (b) TBPF. In both graphs, the dark crosses represent the true value of the ball speed and the circles are the speed estimation.

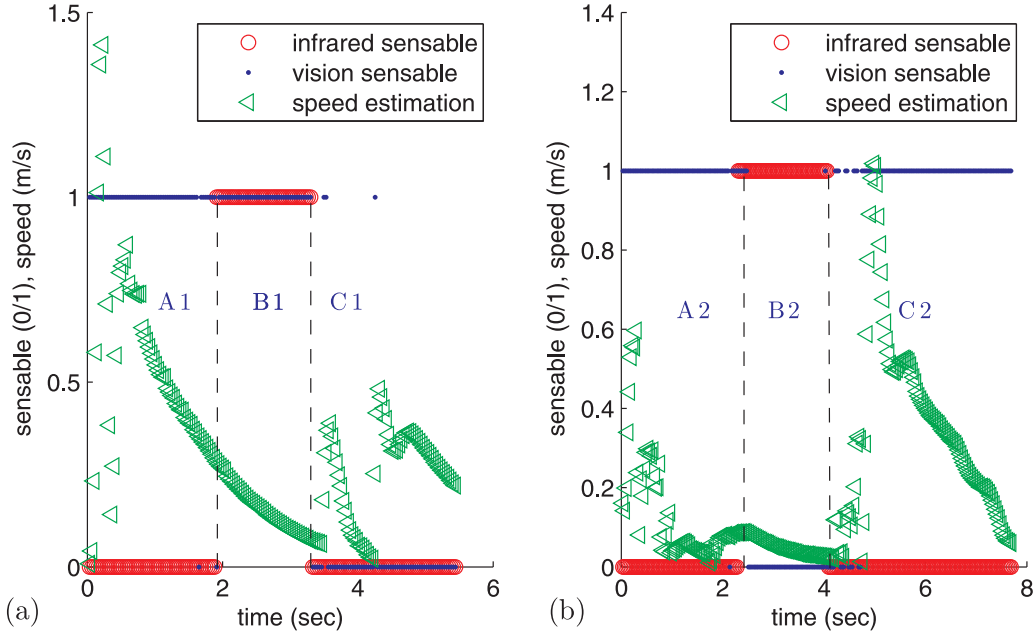

Fig. 12. `Grab-and-Kick` speed estimation using (a) a single model and (b) TBMM.

with a single model tracks only 6.7% of the total while the `Grab-and-Kick` based TBPF successfully tracks 80% of the total.

Figure 12 shows how the TBMM tracker beats the single model tracker. In Figure 12(a) and (b), graphs in the first row show the speed estimation results; graphs in the second row show the corresponding IR sensor readings (ON/OFF, or 1/0), indicating whether the ball is detectable by the IR sensor; graphs in the last row show the binary information of whether the ball is in the FOV of the camera (Y/N) from the single model tracker and the TBPF tracker respectively.

Since the ball is moving towards the robot in each run and is then kicked away by the robot, the IR sensor always outputs 0 before the robot grabs the ball and after the robot kicks the ball. It outputs 1 when the robot is grabbing the ball and aiming at the object. The most interesting thing happens at the time after the robot kicks the ball and the IR sensor outputs 0 again. In Figure 12(a), the ball is visible in a few frames and is finally lost due to the underestimation of the ball speed. In Figure 12(b), the ball is visible consistently due to the correct estimation of the ball speed as soon as the IR sensor outputs 0. This change of IR sensor measurement triggers the mo-

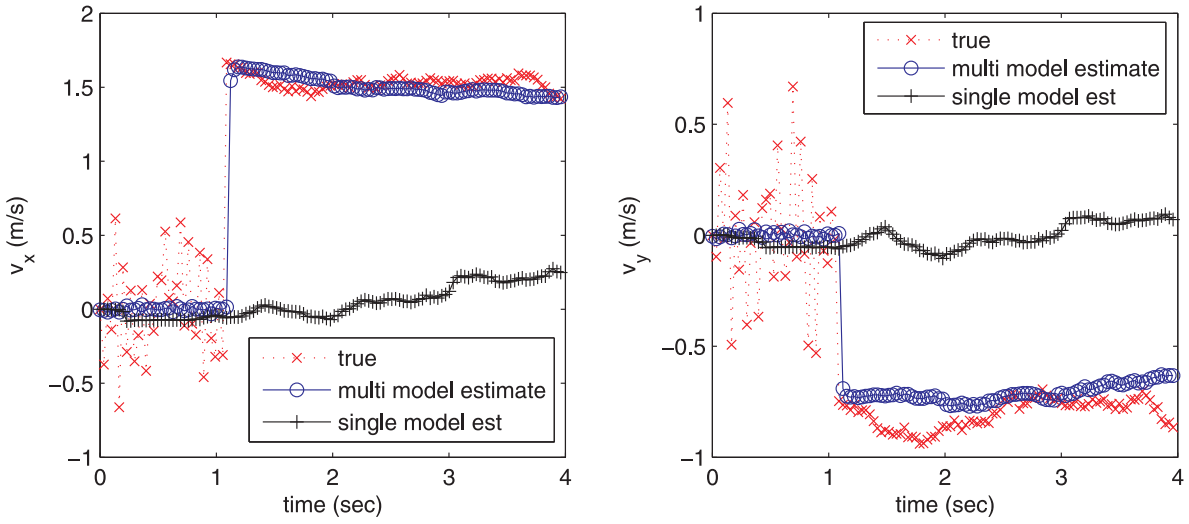

Fig. 13. Ball velocity estimation.

tion model of most particles transiting from *Grabbed-Ball* to *Kicked-Ball* then to *Free-Ball*, which models exactly what is going on in the real world.

6.3. Team Results

6.3.1. Simulation Experiments

Experiments are carried out following the `Naive Offense` play, in which the robot acts as the receiver and the human teammate acts as the passer. Noise is simulated according to the model we experimentally acquired. At the beginning, the teammate holds the ball. After a fixed amount of time, the ball is kicked towards the robot and the teammate moves forward to a predefined location.

We implement both a single model tracker and a PBMM tracker for the ball and the teammate. 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 3. The results show that the PBMM scheme performs much better than the single model, especially in terms of velocity estimation. This is because when the ball is being kicked in the PBMM, most particles evolving using the transition model determined by the play change their motion model $m_t^{(i)}$ from *Free-Ball* to *Human-Kick-Ball*. A velocity will be added to the ball accordingly.

Figures 13 and 14 show the ball velocity estimation and the teammate 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 play-based multi-model tracking. The right-hand graph shows the y component of the velocity (v_y) estimation. The dotted line with crosses represents the true value, the solid line with circles represents the velocity estimation through play-based multi-model tracking and the solid line with crosses represents the velocity estimation through single model tracking. We note that the velocity estimation with PBMM keeps track of the true velocity in terms of v_x and v_y much more consistently than with single model trackers.

Table 3. The average RMS error of position estimation and velocity estimation from human trackers and ball trackers.

Motion Model	Single Model	PBMM
Human Position Est RMS (m)	0.0030	0.0014
Human Velocity Est RMS ($\mathrm{m\,s^{-1}}$)	0.42	0.025
Ball Position Est RMS (m)	0.0028	0.0017
Ball Velocity Est RMS ($\mathrm{m\,s^{-1}}$)	0.4218	0.0597

6.3.2. Team Cooperation Test

In the real-world test, we conduct experiments on the Segway RMP soccer robot executing the offensive play and coordinating with the human teammate on a Segway. The test setup is demonstrated in Figure 15, 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 represents an opponent player and the shaded circles represent the human teammate.

When each run begins, the human teammate is at position A. With this team cooperation plan (play), the robot chooses

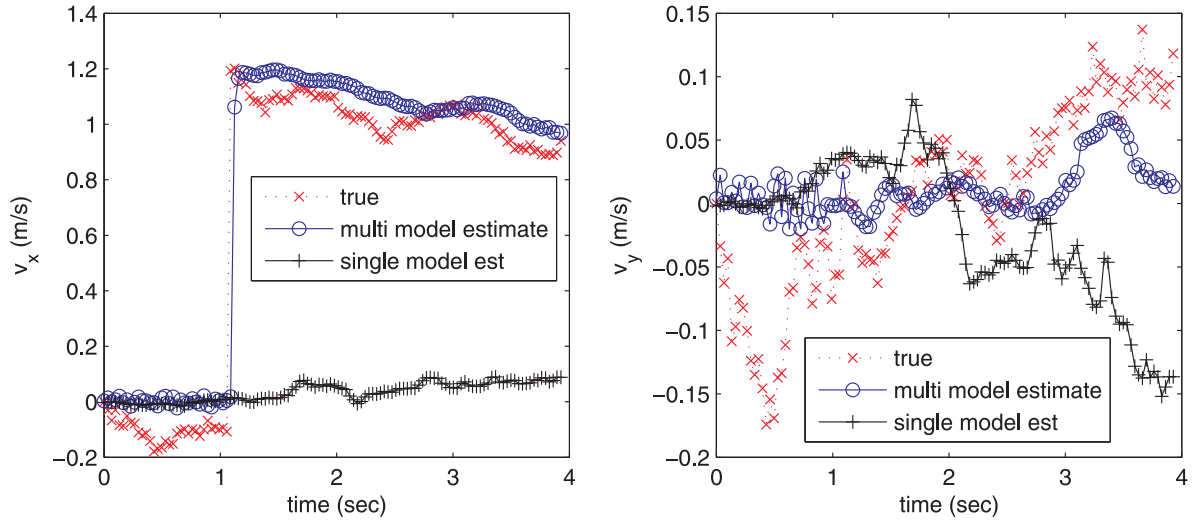

Fig. 14. Human teammate velocity estimation.

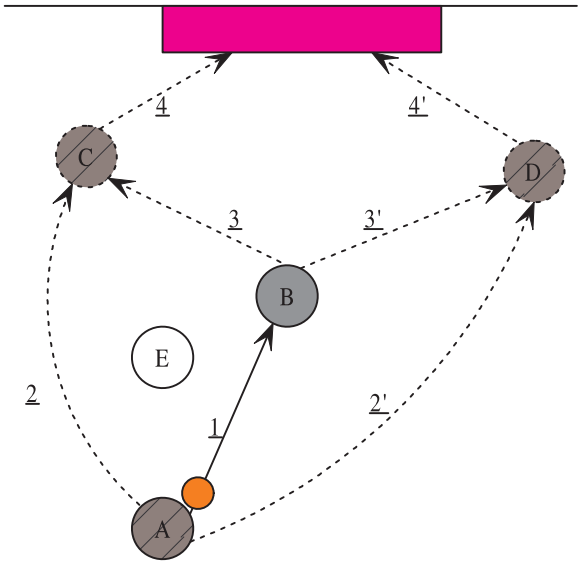

Fig. 15. A demonstration of a naive team cooperation plan in offensive scenario. The digits along the lines show the sequence of the whole plan. The filled circle at position B represents the robot. The unfilled circle at position E represent an opponent player. The shaded circle represents the human teammate.

the tactic `CatchKickToTeammate` to execute, in which the robot starts with the skill `Search`. When the robot finds the ball, the teammate passes the ball directly to the robot and chooses a positioning point either at C or D to go to. 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 teammate holding the ball with its catcher (skill `Search-Teammate`). After the robot finds the teammate, the robot kicks the ball to its teammate and the teammate shoots at the goal (skill `KickToTeammate`), completing the whole offensive play. Each run ends in one of the following conditions:

- **Success** if the human receives the ball from the robot or the human does not receive the ball but the pass is directed to an area close (less than 0.5 m) to the human.
- **Failed** if the robot is searching for the ball or the teammate for more than 30 seconds. The size of game field in Segway soccer is much larger than that of any RoboCup league. It takes a Segway robot approximately 15 s to scan the whole field using its pan-tilt camera. We pick the threshold value to be 30 s to allow the robot scanning no more than twice.
- **Failed** if the ball is outside the field boundary before the robot catches it.

In the experiment over 15 runs, the robot with the single model trackers fails 33% of the total, while the robot with PBMM trackers fails 13% of the total. We also keep track of the mean time taken in each successful run, listed in Table 4. Using PBPF 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 was spent on searching for the teammate. Incorporating the team cooperation knowledge known as play into the teammate motion modeling greatly improves the accuracy of the teammate motion model and therefore avoids spending time in searching for a lost object.

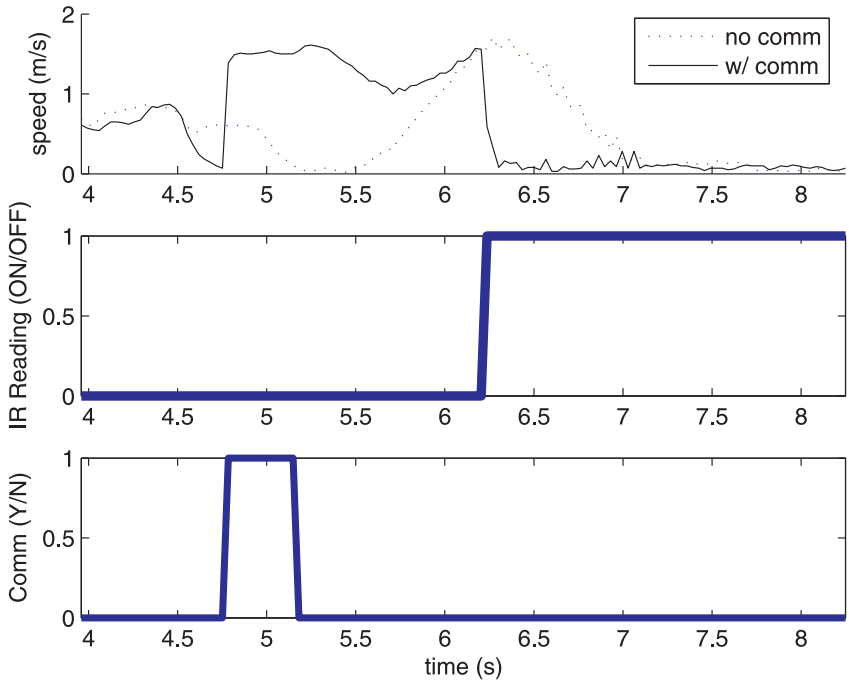

Fig. 16. Ball speed estimation results from the multi-model tracker with and without the communicated information.

Table 4. The average time taken over all the successful runs.

Motion model	Single model	PBMM
Mean time (s)	33.4	22.6

6.4. Communication Results

We perform experiments on the Segway RMP soccer robot executing the `Catch` tactic to receive the ball from the robot teammate. The teammate robot is executing the `GrabKickToTeammate` tactic. When the kick motion is finished, the teammate announces the PASS message through peer-to-peer communication. We compare the performance of multi-model tracker with and without the communicated information.

Figure 16 plots the ball speed estimation results from each tracker. As is shown in the figure, the estimation without communication is about 0.5 s longer than the true value, while the estimation with communication (CBPF) is well predicted because the announcement is received right after the action is made. To illustrate the superior tracking of CBPF, Figures 17 and 18 plot the corresponding model weighting from each tracker. We can clearly see that the model transition in Figure 18 exactly describes the real world testing scenario in which the motion model transits from *Free-Ball* to *Teammate-Kick-Ball*, to *Free-Ball* and finally to *Robot-Grab-Ball*.

7. Related Work

There are several areas of previous work related to this research. We discuss them along the three main aspects of our approach: (i) probabilistic state estimation; (ii) cooperatively tracking by multiple robots; and (iii) improvement of tracking performance through integration of prior knowledge or dynamic information.

Tracking moving objects using a Kalman filter is the optional solution if the system follows a linear model and the noise is assumed Gaussian (Kalman 1960). 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 (Bar-Shalom et al. 2001). For nonlinear systems or systems with non-Gaussian noise, further approximations such as an extended Kalman filter are 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 $h_{i,j} = \mathrm{P}(m_t = i | m_{t-1} = j)$ where $i, j = 1, \ldots, M$.

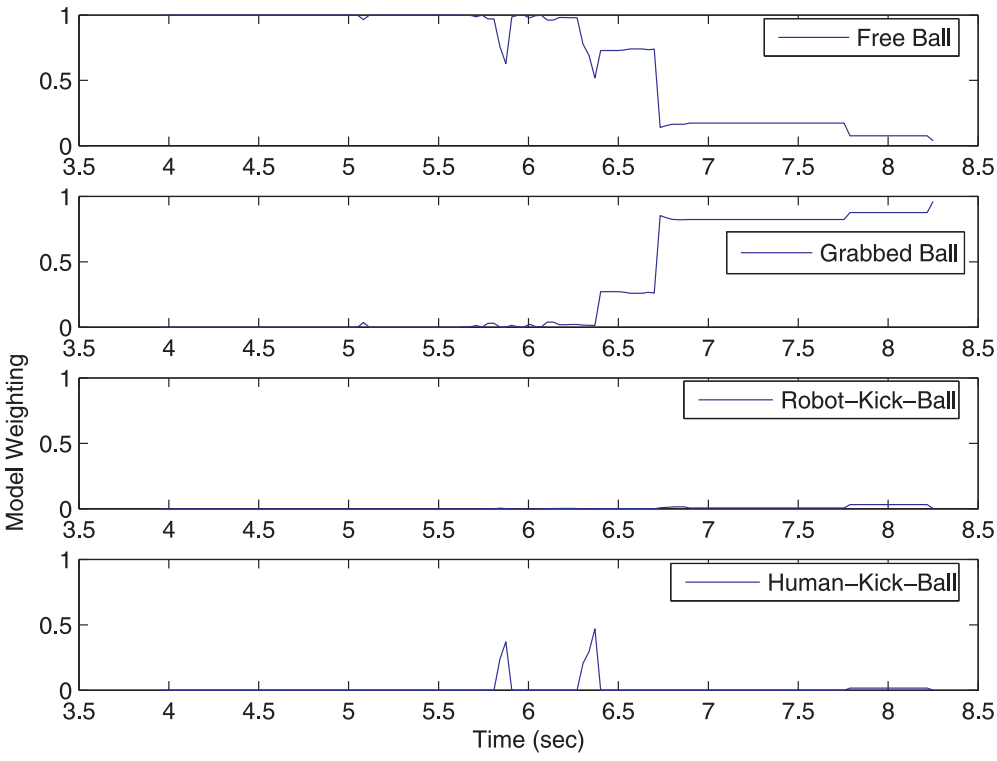

Fig. 17. Model weightings when communication is disabled.

However, the uncertainties in our object tracking problem do not have such a property due to the interactions between the robot and the tracked object. To solve this problem, we have proposed a tactic-based motion modeling method (Gu 2005). Based on that approach, we further introduce the play-based motion modeling method when team coordination knowledge is available.

There have been many investigations into the problem of mapping with robot teams (e.g. Dissanayake et al. 2000; Spletzer et al. 2001). Robots observe each other and the environment. They use the shared observation to increase the total information available to each robot for localization or tracking mobile objects. Approaches that use behaviors to deliberately reduce the uncertainty in sensor readings enable multiple robots to cooperatively track multiple objects (Stroupe and Balch 2003). However, because of the positional uncertainty, global positions of objects reported by teammates can very easily be erroneous. Therefore, sharing global information about the position of tracked objects is very difficult. One approach is to explicitly maintain separate estimates of self and teammate information, which has been proven to be an effective solution to deal with this uncertainty (Roth et al. 2003). Our approach does not communicate information in terms of global position. Instead, the action that can substantially change the motion characteristics of the tracked object is communicated, which can greatly avoid the problem of erroneous information.

There are several approaches incorporating some kind of prior knowledge related to the general problem of tracking under no action. For example, hard constraints on object position, speed or acceleration have been considered in tracking problems to improve tracking performance (e.g. Wang et al. 2002). This kind of information is simple and easily represented as a truncated density. The only thing we need to do is to sample from a truncated density using rejection sampling techniques.

Another example is the situation where a number of objects are moving in formation and there is a strong dependency between the individual sensor measurements, which provides valuable information on object behavior. This problem can actually be modeled as independent individual object motions superimposed on a common group effect. A model of this type was introduced (Salmond and Gordan 1999) in which the motion of the group and disposition of the measurement sources relative to the group are modeled as two separate components.

In a terrain-aided tracking problem using the ground moving target indicator (GMTI), one may have some prior information of the terrain, road maps and visibility conditions (Arulampalam et al. 2002). The algorithm is referred to as the variable structure multiple-model particle filter, since it adaptively selects a subsets of modes that are active at a particular time.

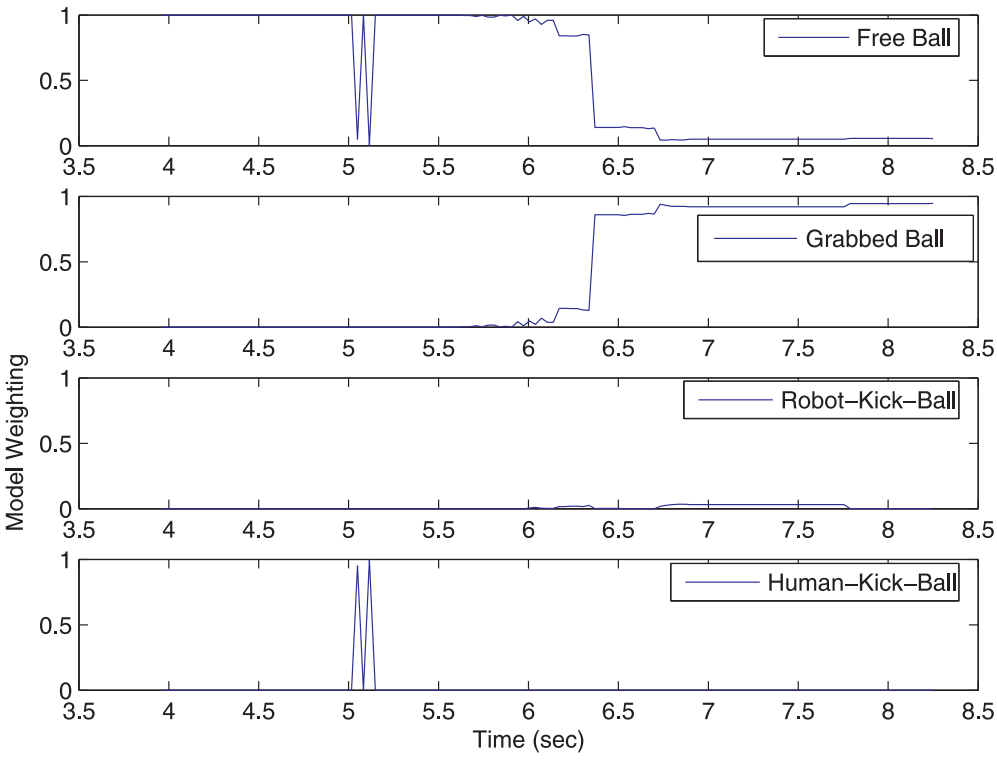

Fig. 18. Model weightings when communication is enabled.

This approach outperformed the normal tracking method without integrating the prior information due to the better dynamics models, which capture the motion dynamics with terrain information in an intricate but accurate manner.

All of the above approaches deal with the problem of tracking under no action. A tracking approach concerned with our problem of action on tracked objects has been presented Kwok and Fox (2004). A joint state estimation have been used successfully for tracking a dynamic object with a mobile robot, where the actions of the robot change the process characteristics of the tracked object. Our approach extends the above approach by using a dynamic transition table dependent on the play that the robot is executing and the additional information that matters. This play-based motion modeling can be flexibly integrated into our existing skills-tactics-plays architecture.

8. Concluding Remarks and Future Work

Motivated by the interactions between a team and the tracked object, we contribute a method to achieve efficient tracking through the robot's own actions, team plays or communicated information to decide a motion model and combine vision and infrared sensory information. The team-driven motion modeling 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 teammate). We incorporate the skills-tactics-plays architecture into a DBN-based temporal representation for tracking and use a particle filter to keep track of the motion model and object state through sampling. The empirical results from simulated and real experiments in multiple robot platforms, including teams of one robot and one human, show the efficiency of the team-driven multi-model tracking over single model tracking.

Future work includes modeling 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 tactics and the plays can help to eliminate false alarms and achieve efficient resampling under the framework of the particle filter.

If the teammate is a human, not a robot, the certainty that the teammate is executing the expected play or tactic could be reduced. That is, the human teammate could fail to execute the desired play or tactic. Future work will take such uncertainty into account. A better human team member modeling (including intercepting the moving ball, marking a player and covering the goal) will also help. It would also be interesting to know how the performance of the presented method is affected by the presence of tactics of the team member that are not exactly determined in the team coordination plan.

Another interesting direction might be learning the parameters of the DBN with known structure and partial observability. In the current DBN, all the conditional probability distributions (CPDs) are manually set by experience. The goal of learning in this case is to find the values of the parameters of each CPD that maximize the likelihood of the training data.

Acknowledgements

We would like to thank the members of the CMBalance Segway soccer team for their help with developing the infrastructure for the Segway robots. In particular, we thank Brett Browning for the vision and initial control systems and Brenna Argall for the game skills.

This research was partly sponsored by DARPA under grant numbers NBCH1040007 and NBCHC030029, and Boeing under contract number CMU-BA-GTA-1. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the US government or any other entity.

References

Arulampalam, M., Gordon, N., Orton, M. and Ristic, B. (2002). A variable structure multiple model particle filter for GMTI tracking. *Proceedings of 5th International Conference on Information Fusion*, 8–11 July 2002, Annapolis, MD, USA.

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

Browning, B., Xu, L. and Veloso, M. M. (2004). Skill acquisition and use for a dynamically-balancing soccer robot. *Nineteenth National Conference on Artificial Intelligence, Conference Proceedings*, San Francisco, CA, USA, 599–604.

Browning, B., Bruce, J., Bowling, M., and Veloso, M. (2005a). Stp: Skills, tactics and plays for multi-robot control in adversarial environments. *IEEE Journal of Control and Systems Engineering* **219**: 33–52, Engineering Award 2005.

Browning, B., Searock, J., Rybski, P. E. and Veloso, M. (2005b). Turning segways into soccer robots. *Industrial Robot* **32**(2): 149–156.

Dissanayake, G., Durant-Whyte, H. and Bailey, T. (2000). A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *ICRA 2000*.

Gu, Y. (2005). Tactic-based motion modelling and multi-sensor tracking. *Proceedings of Twentieth National Conference on Artificial Intelligence*, Pittsburgh, PA, USA, pp. 1274–1279.

Gu, Y. and Veloso, M. (2006a). Effective multi-model motion tracking with communication. In *Proceedings of 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China.

Gu, Y. and Veloso, M. (2006b). Effective team-driven multi-model motion tracking. In *Proceedings of the 2006 Conference on Human-Robot Interaction*, Saltlake City, UT.

Gu, Y. and Veloso, M. (2006c). Multi-model tracking using team actuation models. In *Proceedings of the 2006 International Conference on Robotics and Automation*, Orlando, FL.

Gu, Y. and Veloso, M. (2006d). Multi-target multi-model motion tracking under multiple teammember actuators. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Hakodate, Japan.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of ASME, Journal of Basic Engineering* **82**: 35–45.

Kwok, C. and Fox, D. (2004). Map-based multiple model tracking of a moving object. *Proceedings of Eight RoboCup International Symposium*, Lisbon, Portugal.

Ristic, B., Arulampalam, S. and Gordon, N. (2004). *Beyond the Kalman Filter, Particle Filters for Tracking Applications*. Boston, MA, Artech House Publishers.

Roth, M., Vail, D. and Veloso, M. (2003). A world model for multi-robot teams with communication. In *IROS-2003*.

Salmond, D. and Gordan, N. (1999). Group and extended object tracking. *Proceedings of SPIE* 3809, IEE Colloquium on Target Tracking: Algorithms and Applications (1999/090), London, UK, p. 16.

Schulz, D., Burgrad, W. and Fox, D. (2003). People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research* **22**(2): 99–116.

Searock, J., Browning, B. and Veloso, M. (2004). Turning segways into soccer robots. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan.

Spletzer, J., Das, A., Fierro, R., Taylor, C., Kumar, V. and Ostrowski, J. (2001). Cooperative localization and control for multi-robot manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, USA.

Stroupe, A. and Balch, T. (2003). Value-based observation with robot teams (VBORT) using probabilistic techniques. In *Proceedings of the 11th International Conference on Advanced Robotics*, University of Coimbra, Portugal.

Veloso, M., Browning, B., Rybski, P. and Searock, J. (2005). Segway RMP robot football league rules. Technical report, http://www.cs.cmu.edu/~robosoccer/segway/.

Wang, L.-S., Chiang, Y.-T. and Chang, F.-R. (2002). Filtering method for nonlinear systems with constraints. *IEEE Proceedings of Control Theory and Applications*, 525–531.