Effective Motion Tracking Using Known and Learned Actuation Models

Yang Gu

CMU-CS-08-137

June 6, 2008

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Manuela Veloso, Chair
Brett Browning
Geoffrey Gordon
Martial Hebert
Tucker Balch (Georgia Institute of Technology)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the 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 author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Object Tracking, Robot-Human Team, Motion Model Learning

Abstract

Robots need to track objects. We consider tasks where robots actuate on the target that is visually tracked. Object tracking efficiency completely depends on the accuracy of the motion model and of the sensory information. The motion model of the target becomes particularly complex in the presence of multiple agents acting on a mobile target. We assume that the tracked object is actuated by a team of agents, composing of robots and possibly humans. Robots know their own actions, and team members are collaborating according to coordination plans and communicated information. The thesis shows that using a previously known or learned action model of the single robot or team members improves the efficiency of tracking.

First, we introduce and implement a novel team-driven motion tracking approach. Team-driven motion tracking is a tracking paradigm defined as a set of principles for the inclusion of a hierarchical, prior knowledge and construction of a motion model. We illustrate a possible set of behavior levels within the Segway soccer domain that correspond to the abstract motion modeling decomposition.

Second, 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 present the integration of a single robot behavioral model in terms of skills and tactics with multiple actions into our dynamic Bayesian probabilistic tracking algorithm.

Third, we 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 soccer task. The input of the multiple single and multi-robot behavioral sources allow a robot to much more effectively visually track mobile targets with dynamic trajectories.

Fourth, we present a parameter learning algorithm to learn actuation models. We describe the parametric system model and the parameters we need to learn in the actuation model.

As in the KLD-sampling algorithm applied to tracking, we adapt the number of modeling particles and learn the unknown parameters. We successfully decrease the computation time of learning and the state estimation process by using significantly fewer particles on average. We show the effectiveness of learning using simulated experiments. The tracker that uses the learned actuation model achieves improved tracking performance.

These contributions demonstrate that it is possible to effectively improve an agent's object tracking ability using tactics, plays, communication and learned action models in the presence of multiple agents acting on a mobile object. The introduced tracking algorithms are proven effective in a number of simulated experiments and setup Segway robot soccer tasks. The team-driven motion tracking framework is demonstrated empirically across a wide range of settings of increasing complexity.

Acknowledgements

I would like to thank many people for their support and guidance during my years as a graduate student here at Carnegie Mellon.

First and foremost, I gratefully acknowledge the support and encouragement of my advisor, Manuela Veloso. She has been instrumental in directing me from my initial explorations to the final thesis corrections. She has helped me to get through the difficulty and encourage me to achieve to the best of my ability. Without Manuela, this dissertation would not have happened.

I also thank my committee member, Brett Browning for his valuable advice and support when I have been working with him on the Segway soccer robot project. I thank my committee members, Brett Browning, Geoffrey Gordon, Martial Hebert and Tucker Balch, for their valuable comments regarding my thesis research. They have served to greatly improve the content and presentation of this work.

Besides Manuela, I also thank other members of the CMBalance Segway soccer team, Brett Browning for developing the architecture and vision system, Brenna Argall for developing the skill and game play, Jeremy Searock, Kristina Rohlin and Mike Licitra for developing and supporting the robot hardware. I also thank other members of the Coral research group who shared their ideas with me during the meetings and discussions. Many people gave me helpful comments during my thesis proposal and thesis practice talk. I would like to thank Maayan Roth, Douglas Vail, Colin McMillen, Elisabeth Crawford, Sonia Chernova, Daniel Borrajo for their great comments that helped me to prepare the thesis talk and document.

Table of Contents

1 Introduction **19**

1.1 Motivation 20

1.2 Objectives 22

1.3 Approach 23

1.4 Contributions 25

1.5 Guide to the Thesis 26

1.6 Summary 27

2 Team-Driven Motion Tracking **29**

2.1 Principles 30

2.2 Multi-Robot Instantiation Motivated by Segway Soccer 32

2.2.1 Implemented Components 32

2.2.2 Future Components 34

2.3 Summary 35

3 Background **37**

3.1 Nonlinear Filtering Problem 37

3.2 Bayesian Filters 38

3.3 Kalman Filter 39

3.4 Sequential Importance Sampling 41

3.5 Multiple Switching Dynamic Models 43

3.6 Multiple Model Particle Filter 44

3.7 Dynamic Bayesian Network 45

3.8 Summary 46

4 Tracking Using Own Actuation Model **49**

4.1 Tactics 50

4.2 Tactic-Based Model Transitions 51

4.2.1 Multi-Model System 51

4.2.2 Motion Modeling Based on the Tactic 54

4.3 Tactic-Based Object Tracking Algorithm 56

4.4 Results 58

4.4.1 Ball Motion and Measurement Noise Profiling 58

4.4.2 Metrics 59

4.4.3 Simulation Experiments 61

4.4.4 Segway Soccer and Segway RMP Robot 66
4.4.5 Test on the Real Robot 68
4.5 Summary 70

5 Tracking Using Team Actuation Model **75**

5.1 Plays 76
5.2 Play-Based Model Transitions 78
5.2.1 Tracking Scenario 79
5.2.2 Play-Based Motion Model 80
5.3 Communication 82
5.3.1 Types of Communicated Message 82
5.3.2 Communication-Based Motion Model 82
5.4 Team-Based Tracking Algorithms 83
5.4.1 DBN Representation 83
5.4.2 Play-Based Particle Filtering (PBPF) Algorithm 83
5.4.3 Communication-Based Particle Filtering (CBPF) Algorithm 85
5.5 Results 86
5.5.1 Metrics 86
5.5.2 Simulation Experiments 88
5.5.3 Team Cooperation Test 91

5.5.4 Team Communication Test 93
5.6 Summary 93

6 Learning of Actuation Models **103**
6.1 Parametric System Model 104
6.2 Learning of Actuation Model 105
6.2.1 Partition of Parameter Space 105
6.2.2 Likelihood Function 107
6.2.3 Augment the State Vector of Particles 108
6.2.4 Adapting the Number of Particles 109
6.3 Results 110
6.4 Summary 112

7 Categorization of Previous Work **119**
7.1 Multi-Model Motion Tracking 119
7.2 Cooperatively Tracking 120
7.3 Motion Prediction 120
7.4 Tracking Using Prior Knowledge or Dynamic Information 122
7.5 Summary 122

8 Conclusions and Future Work **125**
8.1 Conclusions 126

8.2 Directions for Future Work 127

8.3 Concluding Remarks 128

A Notations **129**

B Skill and Tactic Descriptions **131**

List of Figures

1.1 Motivation example: from frame 431 to 477. This experiment uses our tracking approach. 21

1.2 Motivation example: from frame 486 to 710. This experiment does not use our tracking approach and results in a longer interval. 21

2.1 Top-level structure of our proposed approach 30

3.1 A DBN for a state-space model. 46

3.2 A DBN for multiple switching dynamic models. 47

4.1 Skill state machine for tactics. Each node is a skill and the edges show the transition between skills. Transitions between skills are based on the perceived state. (a) Skill state machine for tactic `chase_ball`. (b) Skill state machine for tactic `grab_and_kick`. 50

4.2 Tactic-based motion modeling, where $m_1, m_2, \cdots, m_n$ are n single models, $\mathcal{T}_a$ is the tactic, v_b is the additional information, and $n = N_m$, the number of models. $\pi_{i,j}$ is the transition probability from model m_i to model m_j given m_i, and $\langle \mathcal{T}_a, v_b \rangle$. Each layer in the graph is conditioned on a particular combination of the tactic executed and the additional information obtained. 55

4.3 Ball motion model. Each node is a single model. The tables list the transition probability between any two single models. (a) Ball motion model based on tactic `chase_ball`. (b) Ball motion model based on tactic `grab_and_kick`. . 56

4.4 A dynamic bayesian network for tactic-based object tracking. Filled circles represent deterministic variables which are observable or are known as the tactic that the robot is executing. 57

4.5 Test setup for estimating the ball speed decay d. The ball rolls off the ramp (with height h) with speed v_0 and it stops after it travels a distance of L. . . 59

4.6 A typical ball trajectory when a robot is executing `grab_and_kick` tactic. . . 62

4.7 A snapshot of the particle cloud and the model probabilities at $t = 1$ and $t = 3$. 64

4.8 A snapshot of the particle cloud and the model probabilities at $t = 19$ and $t = 24$. 65

4.9 A snapshot of the particle cloud and the model probabilities at $t = 80$ and $t = 86$. 66

4.10 RMS position error versus time 67

4.11 IMM tracking result: model probabilities 68

4.12 TBPF tracking result: model probabilities 69

4.13 RMS position error versus the maximum kicking speed 70

4.14 The percent of time in the view of the camera versus the maximum kicking speed 71

4.15 The Segway RMP soccer robot equipped with a kicker, a catcher, infrared sensors, and a camera mounted on a custom pan-tilt unit. 72

4.16 `grab_and_kick` speed estimation (a) Single model tracking. (b) Tactic-based multi-model tracking (TBPF). 73

5.1 An illustration of the sequence of behavior execution of play 1. 77

5.2 An illustration of the sequence of behavior execution of play 2. 78

5.3 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 teammember motion model. 81

5.4 A dynamic Bayesian network for team member tracking. Filled circles represent deterministic variables which are observable or are known as the tactic or the play. 84

5.5 A dynamic Bayesian network for play-based object tracking. 85

5.6 A DBN for team-driven object tracking when communication is enabled. . . 87

5.7 The snapshots of the particle cloud and the model probabilities at $t = 4$, 23, 24, and 27. 94

5.8 The snapshots of the particle cloud and the model probabilities at $t = 75$, 77, 79, and 86. 95

5.9 RMS position error versus the number of particles 96

5.10 Play 1: RMS position error and velocity error versus time. 97

5.11 Play 2: RMS position error and velocity error versus time. 98

5.12 Play 2: RMS position error versus the amount of noise. 99

5.13 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. . . 99

5.14 Ball speed estimation results from the multi-model tracker with and without including the communicated information. 100

5.15 Model weightings when communication is disabled. 101

5.16 Model weightings when communication is enabled. 101

6.1 The illustration of the unknown parameter: passing angle. 106

6.2 Learning result: t = 3 115

6.3 Learning result: $\mu = 92, \phi = 6, t = 400$ 115

6.4 Learning result: $\mu = 105, \phi = -10, t = 250$. 116

6.5 Learning result: $\mu = 85, \phi = 4, t = 250$. 116

6.6 Actuation model learning result 117

List of Tables

2.1 Components using different behavior levels 32

4.1 List of skills used in this chapter along with brief descriptions. 51

4.2 The model index and the corresponding ball motion model. 54

4.3 TBPF algorithm. 58

4.4 The nominal filter parameters used in the simulation. 63

4.5 Performance comparison for three trackers 65

4.6 Performance comparison of IMM and TBPF for predicting the ball motion model 65

5.1 A simple example play: Play 1. 76

5.2 Play 2 that involving sequencing of behaviors. 77

5.3 List of tactics used in our example play along with brief descriptions. 78

5.4 The model index and the corresponding ball motion model. 80

5.5 PBPF algrithm. 86

5.6 CBPF algrithm. 88

5.7 The nominal filter parameters used in the simulation. 89

5.8 Play 1: RMS error comparison for three trackers 90

5.9 Play 2: RMS error comparison for three trackers 90

5.10 Play 1: Performance comparison for three trackers 91

5.11 Play 2: Performance comparison for three trackers 91

5.12 The average time taken over all the successful runs. 92

6.1 Partition of the parameter space. 106

6.2 Update the probability of each bin. 108

6.3 Actuation model learning algorithm. 113

6.4 The nominal filter parameters used in the simulation. 114

6.5 The preset parameters for Figures 1.3-1.5 114

6.6 RMS error comparison for three trackers 114

Chapter 1

Introduction

Target tracking is one of the important elements of surveillance, guidance, or obstacle avoidance systems. The role of tracking is to determine the number, position, and movement of targets through recursive target state estimation [32]. The target state typically consists of kinematic components (e.g., position, orientation, velocity) and attributes (e.g., target signal-to-noise ratio, spectral characteristics). Noise-corrupted measurements are collected by a single or multiple sensors. The tracking system forms and maintains a track for each target from a sequence of measurements that have been associated with the target over time.

Target tracking is widely used in multi-robot applications. We identify two kinds of tracking problems in which:

1. The tracker is static or does not actuate on the tracked object and
2. The tracker, or its teammate agent, actuates on the object.

This thesis focuses on the latter problem in which tracking is performed by a robot executing specific tasks where the object being tracked is actuated by the robot tracker or by its teammates. An example of such tasks is a Segway RMP soccer robot grabbing and kicking a ball. Object tracking efficiency completely depends on the accuracy of the motion model and of the sensory information. The motion model of the target becomes particulary complex in our case and highly dependent on the robot's actions. In addition, this thesis also considers the challenging environment of multiple team members actuating the object being tracked. In this case, the motion can become highly discontinuous and nonlinear.

We assume that robots know their own actions, and robots in a team are collaborating ac-

cording to coordination plans. The thesis shows how the knowledge in terms of the single robot control strategy and the multi-robot coordination plan is a valuable source of information for tracking.

1.1 Motivation

In order to concretely motivate how robot action models, in particular team actions, contribute to improve tracking performance, we provide a brief illustrative example. Figure 1.1 and 1.2 show a sequence of views from a Segway robot's pan-tilt camera [9]. 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 1.1 shows the first example in which a robot is using our developed tracking algorithm (frames 431 - 477). Figure 1.2 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 frame 431 to frame 477: The robot integrates the team action model into tracking. At frame 431, the ball is kicked towards the robot. Though the ball moves fast, the robot quickly finds the ball at frame 445 based on the 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 second) to find the ball again and then the robot keeps tracking the ball well.
- From frame 486 to frame 710: A new trial starts without the incorporation of the team action model. The ball is kicked towards the robot, but it now takes approximately 120 frames (4 seconds) 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

Figure 1.1: Motivation example: from frame 431 to 477. This experiment uses our tracking approach.

Figure 1.2: Motivation example: from frame 486 to 710. This experiment does not use our tracking approach and results in a longer interval.

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 efficiently.

Instead, when the control strategy integration is disabled, the robot performs an uninformed search locating the ball in a long time because the action 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 nonstandard (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 of their tracking.

1.2 Objectives

Agents act using behaviors as tactics and cooperate using predefined team plays. 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. Furthermore an actuation model could be learned from observation.

The thesis seeks to answer the question,

> Can an agent effectively improve its object *tracking* ability using *tactics*, *plays*, *communication* and *learned* action models in the presence of multiple agents *acting* on a mobile object?

By **tracking**, we mean the data-processing algorithm whose role is to process sensor measurements based on the object motion models in order to form and maintain a sequence of object state estimation up to the current time.

Tactics correspond to the individual robot behaviors. Each robot executes tactics independently of the other robots. **Plays** correspond to the team strategy which is responsible for assigning tactics to each individual robot in order to achieve a joint policy in a team. **Communication** is the intentional exchange of information between team members. Communication improves the performance of a multi-agent system. Though tactics and plays are the primary components of the Skills-Tactics-Plays control architecture [7] that is used in a robot soccer task, the techniques that we present are applicable to any domain where a team of agents are cooperating on a specific task and executing actions using behavior models. Actuation on the targets is sent as communicated message to a specific team member to update and synchronize the motion model of the tracked target.

Action model **learning** is the process of an agent acquiring a set of parameters through observation that define other agents' action effects on the ball through the agent's interactions with the environment. Learning is particularly useful when constructing unknown action models.

By **acting**, we mean a robot changes the motion of the object. In a multi-agent system, all agents might act on the object which makes the motion of the object more complex.

1.3 Approach

We identify the challenges of tracking in such a dynamic, adversarial and noisy environment as follows:

- **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 limited FOV. It is difficult to consistently keep an object in the FOV of the camera. Actually, as long as robots are unable to cover the entire environment with their own sensors, the approach presented in this thesis reduces the tracking error significantly.
- **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 [1]. The moving trajectory of the robot does not follow any particular route. Robots dash and stop, changing their velocity immediately.

[1]The Segway RMP with its balancing motion, is particularly challenging.

- **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 noncontinuous.

- **Observable actions of other players:** Observation of others' actuation on the ball is hard. It is a challege to learn an action model.

We are interested in tracking in a multi-robot actuating environment. In a soccer task, there are many robot behaviors related to actuation on the tracked object. Soccer tasks also give multiple actuating agents. Therefore we use robot soccer as our test platform. The ball is considered the object to be tracked.

Our approach builds upon three primary facts:

- An individual robot knows its own actions. If it is the case that a robot is manipulating the ball, the robot can predict the ball's motion using its own actions.

- Robots in a team collaborate according to pre-defined coordination plans or dynamic communication. If it is the case that a teammate is manipulating the ball, the robot that is not controlling the ball can predict the ball's motion by reasoning about the possible teammate actions. If the communication between teammates is allowed, the robot that is not controlling the ball can explicitly know what the action is from their teammates.

- Robots can learn the opponent's action effects on the ball. At a low-level, robots can learn to know the opponent's kicking power and direction. At a high-level, robots can learn to know the opponent's kicking strategies, e.g., in a certain situation whether to shoot or to pass. If it is the case that an opponent robot is manipulating the ball, the robot in our team can predict the ball's motion using the learned action models during the game.

Based on the above facts, we claim that the knowledge about the single robot control strategy and the multi-robot coordination and communication are valuable sources of information for tracking. Therefore, we address the challenges by including the information into a temporal representation. Variations of actuation effects on the ball are represented as different motion models of the ball. Variations of information sources are used to infer what the most likely motion model the ball is following and what the most likely motion model it is going to transit to.

1.4 Contributions

In a Segway 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 in orange [40]. The problem of tracking can be abstracted as follows:

- n moving "robots".
- All the "robots" act on the object.
- "Robots" act using behaviors.
- "Robots" cooperate using team plans.
- "Robots" communicate.
- Multiple moving objects (the ball and other "robots").
- The object has N_m motion models which are functions of available information. The resulting motion model is a switching state-space model.
- Deciding which motion model to take is dependent on the availability of the different sources of information including robot's own actions, team plays and communicated information.

The contributions of our work are:

- We identify the relationship between a hierarchical robot control architecture and team actuation models. The individual robot and team behaviors are transferred into probabilistic representations of multiple object motion models. Depending on the available information, we hierarchically build Tactic-Based Motion Model (TBMM), Play-Based Motion Model (PBMM) and Communication-Based Motion Model (CBMM). TBMM is the basic object motion model when the robot actuates on the object. PBMM is an extension of TBMM when multiple team members act on the object and robots have access to predefined team plays. CBMM is an extension of PBMM when robots communicate their actions on the object.
- We incorporate a single robot and a team actuation 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 (Tatic-Based Particle Filtering),
  - Predefined team plays (Play-Based Particle Filtering),
  - Communicated team actions (Communication-Based Particle Filtering),
  - Learned action models.

  The layered structure of the object motion model makes the tracker scalable to different amount of available information.
- We present an empirical comparison between several classical tracking algorithms including Kalman Filter (KF), Interacting Multiple Model (IMM) and our proposed tracking algorithms, i.e., TBPF, PBPF and CBPF. We examine the performance of each algorithm according to a variety of metrics. We show that our proposed tracking algorithms significantly improves the tracking performance.
- We evaluate the new tracking algorithms in simulated experiments, robot platforms, and a human-robot team. Robot tests are performed through different setup Segway robot soccer tasks.
- We present a parameter learning algorithm to learn actuation models. We evaluate the presented algorithm in simulated experiments and compare their performance according to a variety of metrics. We use the learned actuation model to improve the tracking performance.

1.5 Guide to the Thesis

The thesis is organized as follows:

Chapter 2 We present the high-level architecture of our approach. We introduce three principles for our team-driven motion tracking and give the corresponding instantiation in our Segway soccer platform.

Chapter 3 We give a brief tutorial of the background knowledge and the framework we use.

Chapter 4 We include the low-level individual skills to help construct the motion model. When tracking is performed by a robot executing specific tasks acting over the object being tracked, such as a Segway RMP soccer robot grabbing and kicking a ball, the motion model

of the object becomes complex, and dependent on the robot's actions. We build motion models as a function of the robot's tactic.

Chapter 5 We include the team plan to build the motion model in a multi-agent setup. 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. We also include the communicated messages between team members.

Chapter 6 We learn the actuation models of the team member and the opponent player.

Chapter 7 We give a brief overview of the related work.

Chapter 8 We summarize our major contributions and present future directions.

1.6 Summary

In this chapter:

- We give the motivation of our work using an illustrative example.
- We present the problem abstraction. We identify the challenges of the problem and describe how to address the challenges using our approach.
- We identify our contributions.
- We give a brief guide to the remaining part of the thesis.

Chapter 2

Team-Driven Motion Tracking

In this chapter we overview the complete thesis approach and system framework. We discuss the principles followed in the thesis. We present the components implemented in the thesis and discuss future components to be developed.

Figure 2.1 shows the top-level structure of our complete approach for team-driven motion tracking. We integrate multiple information sources to affect the motion model used by our tracker. Concretely, Figure 2.1 shows the five sources used in this thesis, namely tactic, play, communication, learned actuation model, and sensor observation. Different information sources are represented in various formats.

- A tactic refers to an individual robot behavior (Chapter 4), while a play controls the team behavior (Chapter 5). We use the information contained in tactics and plays to identify the robot behavior. We further use the robot behavior to identify the object motion model in order to construct the transition probability matrix of various object motion models.
- Communication stands for messages sent between team members to share information or trigger team actions (Chapter 5). A communicated message is represented with message identification numbers. Actuation on the targets is sent as a communicated message to a specific team member to update and synchronize the motion model of the tracked target.
- Learned model refers to the parameters of the actuation model. Learned model is obtained by learning through observation (Chapter 6).
- Sensors provide signals that can be observed.

Motion prediction depends on the motion model and its role is to predict object's future motion. State estimation is usually implemented using a tracking filter to carry out recursive object state estimation. A track is a sequence of object state estimates up to the current time.

The output of the system is fed into the motion model to learn model-specific parameters in the learning stage (Chapter 6). The output of the system is also used to help upper level decision making. As a future direction, sensors will receive the feedback from the output of the system to control the sensor position (e.g., pointing, Chapter 8.2). For example, a robot with cameras will use this feedback to know where its cameras should point to in order to keep the object of interest in its focus of view.

The introduction and implementation of the team-driven motion tracking approach is one of the main contributions of this thesis. Team-driven motion tracking is a tracking paradigm defined as a set of principles for the inclusion of a hierarchical, prior knowledge and construction of a motion model. This chapter lays out the principles of team-driven motion tracking (Section 2.1) and gives an overview of the implementation which is detailed in Chapter 4-Chapter 6 (Section 2.2). Section 2.3 concludes this chapter.

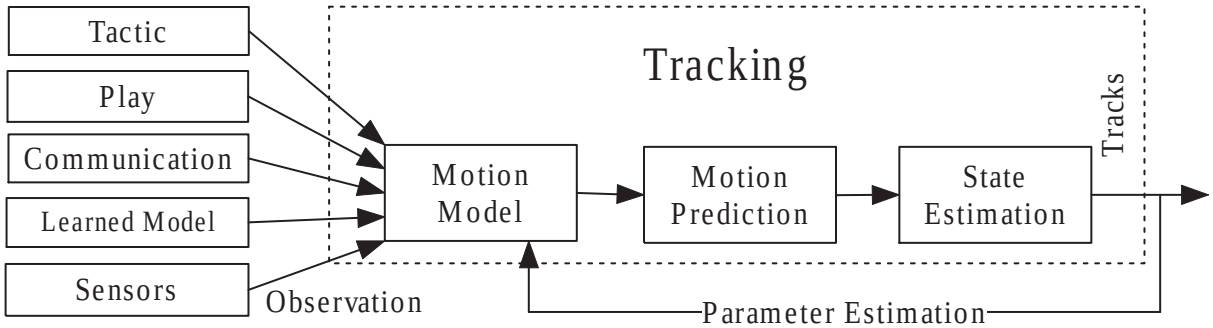

Figure 2.1: Top-level structure of our proposed approach

2.1 Principles

Team-driven motion tracking is defined by three principles. In this section, we identify, motivate and specify these three principles.

Principle 1: Motivated by robot soccer, team-driven motion tracking is designed for a mobile object whose motion is too difficult to model.

Instead, the team-driven motion tracking approach consists of breaking a motion modeling problem down into several behavioral layers and includes the useful behavioral information at each level one at a time. This approach uses a bottom-up incremental approach to integrate hierarchical levels of prior information. Starting with low-level behaviors, the process of inclusion of new information continues until reaching high-level strategic behaviors that deal with the full domain complexity.

Principle 2: The appropriate behavior granularity and the aspects of the behaviors to be learned are decided as a function of the specific domain. The task of motion model decomposition is not automated. Instead, the hierarchical levels are defined by the prior information available to a specific domain.

In our Segway soccer domain, the ball's motion model problem is decomposed into an incremental problem of inclusion of a robot's own tactic, multi-robot team plans, communicated messages and opponent actuation models through learning. In this approach, inclusion of information begins with individual behaviors, which facilitate multi-agent collaborative behaviors, and eventually lead to adversarial behaviors.

Principle 3: An agent's cognition architecture (behavior model) is used to infer its possible action and corresponding results towards the objects.

Our robot control architecture, Skills-Tactics-Plays (STP), consists of *Skills* for low-level control policies, *Tactics* for high-level single robot behavior, and *Plays* for team coordination [7]. Though the STP control architecture [7] is used in a robot soccer task, the techniques that we present are applicable to any domain where a team of agents are cooperating on a specific task and executing actions using behavior models. Actuation on the targets is sent as a communicated message to a specific team member to update and synchronize the motion model of the tracked target.

In summary, team-driven motion tracking as introduced in this thesis is a tracking paradigm designed to allow agents to include more domain-specific information to construct better motion models. Team-driven motion tracking allows for a bottom-up definition of motion modeling capabilities at different levels in a complex domain. Extra motion modeling capabilities are identified when agents have internal behaviors or agents communicate. The principles of the team-driven motion tracking techniques are summarized as follows.

1. A direct construction of a motion model is too complex.
2. A bottom-up, hierarchical modeling decomposition is given.
3. An agent cognition architecture provides additional information for motion model con-

Component	Strategic Level	Behavior type
tactic	robot-ball	individual
play	multi-robot-ball	multi-agent
communication	multi-communication-robot-ball	team with communication
parameter estimation	team-to-opponent	adversarial

Table 2.1: Components using different behavior levels

struction.

2.2 Multi-Robot Instantiation Motivated by Segway Soccer

Table 2.2 illustrates a possible set of behavior levels within the Segway soccer domain that correspond to the abstract motion modeling decomposition represented in Section 2.1. Tactics, e.g., `pass-ball`, are individual behaviors which enable the robot to act and control the object effectively. Plays are multi-agent behaviors which define the team coordination behaviors. Communication is also a multi-agent behavior in which case one agent talks to another by sending messages. Parameter estimation enables the agent to learn by observation.

We identify an important level of motion modeling that must be constructed before moving on to higher-level strategies. We build a more complex motion model upon it to create higher-level team behaviors. Section 2.2.1 gives an overview of our team-driven motion tracking implementation in the Segway soccer domain and Section refarch:future discusses how the implementation could be extended further.

2.2.1 Implemented Components

Our implementation consists of three included team actuation information and a learning component, each of which is described more fully along with extensive empirical tests later in the thesis.

1. Tracking using own actuation model (using individual behavior)

First, we include the low-level individual skills to help construct the motion model. When tracking is performed by a robot executing specific tasks acting over the object being tracked, such as a Segway RMP soccer robot grabbing and kicking a ball, the motion model of the object becomes complex, and dependent on the robot's actions. We build motion models as a function of the robot's tactic.

2. Tracking using team actuation model (using collaborative team behavior)

   Second, we include the team plan to build the motion model in a multi-agent setup. Robots in a team collaborate according to pre-defined coordination plans or dynamic communication. If it is the case that a teammate is manipulating the ball, the robot that is not controlling the ball can predict the ball's motion by reasoning about the possible teammate actions. The team coordination knowledge provides further information that can be incorporated into the motion modelling and tracking process.

3. Tracking using communicated message (using communication agent)

   Third, we include the communicated messages between team members. Communication improves the performance of such a multi-agent system. When communication is enabled, a shared world model that stores the state of the team can be constructed. We present our solution to integrate the communication information into our team-driven multi-model motion tracking.

4. Learning of actuation models (learning behavior)

   Fourth, we learn the actuation models of the team member and the opponent player as well. Any model consists of one or multiple parameters. Usually the model parameters are set by a human expert, based upon the experience with the environment and the robot. We present a novel method of automating the procedure of acquiring this probabilistic motion model. We present a parametric system model. We apply particle filtering and extend the use of KLD-sampling to learning parameters in this model. The result is a fast algorithm which runs on-line and achieves accurate-enough learning of parameters. Furthermore, this method provides a refined motion model based on the current one, resulting in more accurate tracking.

In general, the four included pieces of actuation information above illustrate the principles of the team-driven motion tracking paradigm as laid out in Section 2.1:

- The decomposition of the motion modeling problem into smaller subtasks enables the construction of a more complex motion model.
- The hierarchical task decomposition is constructed in a bottom-up, domain-dependent manner.

- The domain-dependent agent architecture is used to exploit available behavioral information to build more complex motion models.

2.2.2 Future Components

This section carries the Segway soccer example of our team-driven motion tracking beyond its implementation in this thesis.

When there are limited tracking resources, i.e., there are more objects to be tracked than tracking resources. E.g., in Segway soccer, the robot needs to track several objects (including the ball, team members and opponents) using one pan-tilt camera. When multiple objects need to be tracked using limited trackers, the question of assigning different trackers to different objects is then crucial. This is a sensor scheduling or sensor management problem.

- Instead of continuously scanning over the full field of view to locate the target, the tracker will restrict its scanning area, on the basis of a real-time analysis of the current robot strategy, team plan and object trajectory in its memory, to a very narrow window precisely the size of the useful targets.

- Instead of always focusing on one target, the tracker will shift its focus elegantly between several interesting targets.

- Instead of using a manually tuned or fixed time on each target, the tracker will learn to know how much time to spend on each target in order to ensure consistent tracking of each target.

The thesis contributions provide practical algorithms for tracking to be applied in domains with ever increasing robot-object interactions. First we will need to identify the known behavior model of the specific application. We will apply the team-driven motion tracking framework and follow the introduced principles to decompose the motion modeling problem and take use of the behavior knowledge. Even in some application domains without robot-object interaction, e.g., Unmanned Aerial Vehicle (UAV), by including the cooperative tactics and communicated information, we will still reduce the tracking error by using the communication-based motion model.

2.3 Summary

In this chapter:

- We present the top-level structure of our proposed team-driven motion tracking. We explain the functions of main components.
- We identify and motivate three principles for team-driven motion tracking.
- We illustrate a possible set of behavior levels within the Segway soccer domain that correspond to the abstract motion modeling decomposition. We identify the bottom-up construction of motion models.
- We give the brief description of our implemented components in our team-driven motion tracking paradigm.

Chapter 3

Background

The fundamental building block of a tracking system is a tracking filter. The Kalman-Bucy Filter is the best known filter, a simple and elegant algorithm formulated more than 40 years ago [22]. It is an optimal recursive Bayesian estimator for a restricted class of linear Gaussian problems. Recently there has been a surge of interest in nonlinear and non-Gaussian filtering, in which sequential Monte Carlo estimation is one of the most popular tools, also known as particle filters [11]. This section presents the background of this thesis and is organized as follows. Section 3.1 defines the problem of nonlinear filtering and section 3.2 gives the conceptual solution. Section 3.3 gives a brief description of Kalman filtering and section 3.4 shows the sampling and filtering methods, leading to particle filtering. Then section 3.5 and 3.6 discuss multi-model related problems and solutions. Section 3.7 describes the dynamic Bayesian network and how to represent the state-space model and multiple switching dynamic models with a DBN.

3.1 Nonlinear Filtering Problem

Let $\mathbf{x}_t$ denote a target state vector. The general parameterized system process at time t is given by [4]:

$$\mathbf{x}_t = \mathbf{f}_{t-1}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}) \tag{3.1}$$

$$\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t, \mathbf{w}_t) \tag{3.2}$$

where $\mathbf{f}_{t-1}$ and $\mathbf{h}_t$ are the parameterized state transition and measurement functions; $\mathbf{x}, \mathbf{z}$

are the state, and measurement vectors; $\mathbf{v}$ is the process noise vector, which caters for any mismodeling effects or unforeseen disturbances in the target motion model; $\mathbf{w}$ is the measurement noise vector. The noise vectors $\mathbf{v}$ and $\mathbf{w}$ are assumed to be white, with known probability density functions and mutually independent. The initial target state is assumed to have a known pdf $p(\mathbf{x}_0)$ and also to be independent of the noise vectors.

We try to find filtered estimates of $\mathbf{x}_t$ based on the sequence of all available measurements $\mathbf{z}_{1:t} \stackrel{\triangle}{=} \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_t\}$ up to time t.

3.2 Bayesian Filters

From the Bayesian view, the problem is to recursively construct some degree of belief in the state $\mathbf{x}_t$ at time t, given the data $\mathbf{z}_{1:t}$ up to time t. That is, we seek estimates of $\mathbf{x}_t$ by constructing the posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$.

Recursive filters essentially consist of two steps:

- **prediction step**: $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) \rightarrow p(x_t|\mathbf{z}_{1:t-1})$
  (usually deforms/translates/spreads state pdf due to noise)
  Using the Chapman-Kolmogorov equation, [1]

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1} \tag{3.3}$$

  This is the prior of the state $\mathbf{x}_t$ at time t without knowledge of the measurement $\mathbf{z}_t$, i.e., the probability given only previous measurements.

- **update step**: $p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{z}_t) \rightarrow p(\mathbf{x}_t|\mathbf{z}_{1:t})$
  (combines likelihood of current measurement with predicted state; usually concentrates

[1] Chapman-Kolmogorov equation: $f(x_n|x_s) = \int_{-\infty}^{\infty} f(x_n|x_r) f(x_r|x_s) dx_r$. This equation gives the transitional densities of a Markov sequence. Here, $n > r > s$ are any integers.

on state pdf)[2]

$$
\begin{aligned}
& p(\mathbf{x}_t|\mathbf{z}_{1:t}) && && (3.4)\\
= \; & \frac{p(\mathbf{z}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{z}_{1:t})} && \texttt{(Bayes' Theorem)} && (3.5)\\
= \; & \frac{p(\mathbf{z}_t, \mathbf{z}_{1:t-1}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{z}_t, \mathbf{z}_{1:t-1})} && && (3.6)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_t)p(\mathbf{z}_{1:t-1}|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})p(\mathbf{z}_{1:t-1})} && \texttt{(Conditional probability)} && (3.7)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})p(\mathbf{z}_{1:t-1})} \cdot p(\mathbf{z}_{1:t-1}|\mathbf{x}_t) && \texttt{(Reorganize)} && (3.8)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})p(\mathbf{z}_{1:t-1})} \cdot \frac{p(\mathbf{x}_t|\mathbf{z}_{1:t-1})p(\mathbf{z}_{1:t-1})}{p(\mathbf{x}_t)} && \texttt{(Bayes' Theorem)} && (3.9)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} && \texttt{(Cancel out identical terms)} && (3.10)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} && \texttt{(Markov property)} && (3.11)
\end{aligned}
$$

Now we have

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \tag{3.12}$$

that is

$$posterior = \frac{likelihood \cdot prior}{evidence} \tag{3.13}$$

where *prior* is given by prediction equation, *likelihood* is given by observation model, and *evidence* is the normalizing constant in the denominator,

$$p(\mathbf{z}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})d\mathbf{x}_t \tag{3.14}$$

3.3 Kalman Filter

The Kalman filter assumes that the posterior density at every time step is Gaussian and hence completely characterized by its mean and covariance [42]. Suppose Equation (3.1)

[2]Useful equations: $p(a, b|c) = p(a|b, c) \cdot p(b|c)$ and $p(a, b) = p(a|b) \cdot p(b)$

and (3.2) can rewritten as:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}_{t-1}\mathbf{x}_{t-1} + \mathbf{v}_{t-1} && (3.15)\\ \mathbf{z}_t &= \mathbf{H}_t\mathbf{x}_t + \mathbf{w}_t && (3.16)\\ & && (3.17) \end{aligned}$$

where $\mathbf{F}_{t-1}$ and $\mathbf{H}_t$ are known matrices defining linear functions. $\mathbf{v}_{t-1}$ and $\mathbf{w}_t$ are mutually independent zero-mean white Gaussian, with covariances $\mathbf{Q}_{t-1}$ and $\mathbf{R}_t$ respectively.

The Kalman filter algorithm can be viewed as the following recursive relationship:

$$\begin{aligned} p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) &= \mathcal{N}(\mathbf{x}_{t-1}; \hat{\mathbf{x}}_{t-1|t-1}, \mathbf{P}_{t-1|t-1}) && (3.18)\\ p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) &= \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|t-1}, \mathbf{P}_{t|t-1}) && (3.19)\\ p(\mathbf{x}_t|\mathbf{z}_{1:t}) &= \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t}) && (3.20) \end{aligned}$$

where $\mathcal{N}(\mathbf{x}; \xi, \mathbf{P})$ is a Gaussian density with argument $\mathbf{x}$, mean ξ, and covariance $\mathbf{P}$.

The mean and covariance of the Kalman filter are given by:

$$\begin{aligned} \hat{\mathbf{x}}_{t|t-1} &= \mathbf{F}_{t-1}\hat{\mathbf{x}}_{t-1|t-1} && (3.21)\\ \mathbf{P}_{t|t-1} &= \mathbf{Q}_{t-1} + \mathbf{F}_{t-1}\mathbf{P}_{t-1|t-1}\mathbf{F}_{t-1}^T && (3.22)\\ \hat{\mathbf{x}}_{t|t} &= \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}) && (3.23)\\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T && (3.24) \end{aligned}$$

where

$$\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t \tag{3.25}$$

is the covariance of the innovation term $\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}$, and the Kalman gain is

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{S}_t^{-1} \tag{3.26}$$

The Kalman filter recursively computes the mean and covariance of the Gaussian posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. This is the optimal solution to the tracking problem provided the following assumptions hold:

- $\mathbf{v}_{t-1}$ and $\mathbf{w}_t$ are drawn from Gaussian densities of known statistics
- $\mathbf{f}_{t-1}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1})$ is a known linear function of $\mathbf{x}_{t-1}$ and $\mathbf{v}_{t-1}$
- $\mathbf{h}_t(\mathbf{x}_t, \mathbf{w}_t)$ is a known linear function of $\mathbf{x}_t$ and $\mathbf{w}_t$

3.4 Sequential Importance Sampling

Particle filters are suboptimal filters. They perform sequential Monte Carlo (SMC) estimation based on a point mass representation of probability densities. By applying importance sampling to perform nonlinear filtering specified by the conceptual solution in section 3.2, we obtain the sequential importance sampling algorithms.

Sequential Importance Sampling (SIS) is the basic framework for most particle filter algorithms [2, 11, 26]. Let $\{\mathbf{x}_{0:t}^{(i)}\}$ be a set of support points (samples, particles), $i = 1, \cdots, N_s$, and $w_t^{(i)}$ be associated weights, which are normalized to $\sum_i w_t^{(i)} = 1$. Then we obtain

$$p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) \approx \sum_{i=1}^{N_s} w_t^{(i)} \delta(\mathbf{x}_{0:t} - \mathbf{x}_{0:t}^{(i)}) \tag{3.27}$$

Usually we cannot draw samples $\mathbf{x}_k^{(i)}$ from $p(\cdot)$ directly. Assume we sample directly from a different importance function $q(\cdot)$. Our approximation is still correct if

$$w_t^{(i)} \propto \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{z}_{1:t})} \tag{3.28}$$

If the importance function is chosen to factorize such that

$$q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) q(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) \tag{3.29}$$

then one can obtain new particles $\mathbf{x}_{0:t}^{(i)} \sim q(\mathbf{x}_{0:t}|\mathbf{z}_{1:t})$ by augmenting each of the existing old particles $\mathbf{x}_{0:k-1}^{(i)} \sim q(\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1})$ with the new state $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t})$ (later we will see also $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$).

Now we want to do the weight update.

$$
\begin{aligned}
& p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t}) && && (3.30)\\
= \; & p(\mathbf{x}_{0:t}|\mathbf{z}_t, \mathbf{z}_{1:t-1}) && && (3.31)\\
= \; & \frac{p(\mathbf{x}_{0:t}, \mathbf{z}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} && \texttt{(Bayes' Theorem)} && (3.32)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t-1})p(\mathbf{x}_{0:t}|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} && \texttt{(Conditional probability)} && (3.33)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t-1})p(\mathbf{x}_t, \mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} && && (3.34)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \times p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) && && (3.35)\\
= \; & \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \times p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) && \texttt{(Markov property)} && (3.36)\\
\propto \; & p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{0:t-1}|\mathbf{z}_{1:t-1}) && && (3.37)
\end{aligned}
$$

By substituting both numerator (3.29) and denominator (3.30) into (3.28), the weight update equation can then be shown to be

$$
\begin{aligned}
w_k^{(i)} \; \propto \; & \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})p(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{z}_{1:t-1})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})q(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{z}_{1:t-1})} && (3.38)\\
= \; & \frac{p(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{z}_{1:t-1})}{q(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{z}_{1:t-1})} \times \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} && (3.39)\\
= \; & w_{t-1}^{(i)} \times \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} && (3.40)\\
= \; & w_{t-1}^{(i)} \times \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \mathbf{z}_t)} && (3.41)
\end{aligned}
$$

The last step assumes $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$. So that the posterior density $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ can be approximated as

$$
p(\mathbf{x}_t|\mathbf{z}_{1:t}) \approx \sum_{i=1}^{N_s} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}) \tag{3.42}
$$

The SIS algorithm thus consists of recursive propagation of the weights and support points as each measurement is received sequentially.

SIS

1 **for** $i \leftarrow 1$ **to** N_s
2 **do** Draw $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{z}_t)$
3 Assign the particle a weight, $w_t^{(i)}$, according to (3.41)

How to choose importance density? It is often convenient to use the prior:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{z}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}) \tag{3.43}$$

Substitution of (3.43) into (3.41) then yields

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) \tag{3.44}$$

Sampling Importance Resampling Filter (SIR) proposed in [15] uses the dynamic prior as the importance density and resamples in each udpate step.

SIR

1 **for** $i \leftarrow 1$ **to** N_s
2 **do** Draw $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$
3 $w_t^{(i)} \leftarrow w_{t-1}^{(i)} \times p(z_t|x_t^{(i)})$
4 **for** $i \leftarrow 1$ **to** N_s
5 **do** Normalize: $w_t^{(i)} \leftarrow w_t^{(i)} / \sum_j w_t^{(j)}$
6 Resample

This is the most common choice of importance density since it is intuitive and simple to implement. We use SIR in Chapter 5 intensively. We choose a combination of dynamic prior and a distribution that is proportional to the perceptual likelihood $p(\mathbf{z}_t|\mathbf{x}_t)$ in Chapter 6.

3.5 Multiple Switching Dynamic Models

There are many engineering applications that deal with nonlinear dynamic systems characterized by a few possible models (or regimes) of operation [6]. These problems are referred to as hybrid-state estimation problems involving both continuous-valued target state and a discrete-valued regime variable [12].

A discrete-time hybrid-state system is described by the following dynamics and measurement equations:

$$\mathbf{x}_t = \mathbf{f}_{t-1}(\mathbf{x}_{t-1}, m_t, \mathbf{v}_{t-1}) \tag{3.45}$$

$$\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t, m_t, \mathbf{w}_t) \tag{3.46}$$

where m_t is the model (region) variable which determines which dynamic model is in effect during the sampling period $(t-1, t]$. The model variable is commonly modeled by a time-homogeneous N_m-state first-order Markov chain with transition probabilities

$$\pi_{i,j} \triangleq \mathrm{P}\{m_t = i | m_{t-1} = j\} \qquad (i, j \in S) \tag{3.47}$$

where $S \triangleq \{1, 2, \cdots, N_m\}$. The transition probability matrix $\mathbf{\Pi} = [\pi_{i,j}]$ is thus an $N_m \times N_m$ matrix with elements satisfying

$$\pi_{i,j} \geq 0 \ \text{ and } \sum_{j=1}^{N_m} \pi_{i,j} = 1, \tag{3.48}$$

for each $i, j \in S$. The initial model probabilities are denoted as

$$\mu_i \triangleq \mathrm{P}\{m_1 = i\} \tag{3.49}$$

for $i \in S$, such that $\mu_i \geq 0$ and $\sum_{j=1}^{N_m} \mu_i = 1$.

3.6 Multiple Model Particle Filter

The multi-model particle filter is used to perform nonlinear filtering with switching dynamic models. Some problems belong to a class of hybrid state estimation problems where the state vector consists of both a continuous-valued part and a discrete-valued part, i.e., $\mathbf{X}_t = (\mathbf{x}_t^T, m_t)^T$, $m_t \in S = \{1, 2, \cdots, N_m\}$ [27].

The first step in the Multiple Model Particle Filtering algorithm is to generate a random set $\{m_t^{(i)}\}_{i=1}^{N_s}$ based on $\{m_{t-1}^{(i)}\}_{i=1}^{N_s}$ and the transition probability matrix $\Pi = [\pi_{i,j}]$, where $i, j \in S$.

The next step performs a model conditioned SIS filtering. The optimal model conditioned importance density is given by

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)}, m_t^{(i)}, \mathbf{z}_t)_{opt} = p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)}, m_t^{(i)}, \mathbf{z}_t). \tag{3.50}$$

The most popular choice appears to be the transitional prior [2]:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, m_t^{(i)}, \mathbf{z}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, m_t^{(i)}). \tag{3.51}$$

There are a lot of other densities that can be used which result in various names of the particle filter, e.g., Auxiliary Particle Filter, the Regularised Particle Filter [2]. The choice is an important design step in the design of a particle filter.

3.7 Dynamic Bayesian Network

The data in the tracking scenario comes in the form of temporal (time-series) data, which is generated sequentially by the state process. A Dynamic[3]. Bayesian Network (DBN) is a natural representation language for modeling time-series data. A DBN allows us to represent complex systems in a compact and natural way [29].

A DBN is a directed graphical model [23]. Each node in the graph corresponds to a random variable. Each edge in the graph denotes the dependency of a variable on its parents. We use filled circles to denote observable variables. The graphical structure of a DBN encodes a set of conditional independence assumptions: each node is conditionally independent of its non-descendants given its parents.

As described in Section 3.1, in a state-space model, we have the state transition model, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, the measurement model, $p(\mathbf{z}_t|\mathbf{x}_t)$, and the initial state pdf, $p(\mathbf{x}_0)$. In a DBN, $\mathbf{x}_t, \mathbf{z}_t$ represent sets of system variables. A DBN for a state-space model is illustrated in Figure 3.1. In this graph,

- $\mathbf{z}$ is an observable variable.
- $\mathbf{x}_t$ is dependent on $\mathbf{x}_{t-1}$.
- $\mathbf{z}_t$ is dependent on $\mathbf{x}_t$.

As described in Section 3.5, in multiple switching dynamic models, we add a model variable m to describe the switching between different kinds of dynamics models. The dynamics

[3]The term "dynamic" means we are modeling a dynamic system. It does not mean the graph structure changes over time.

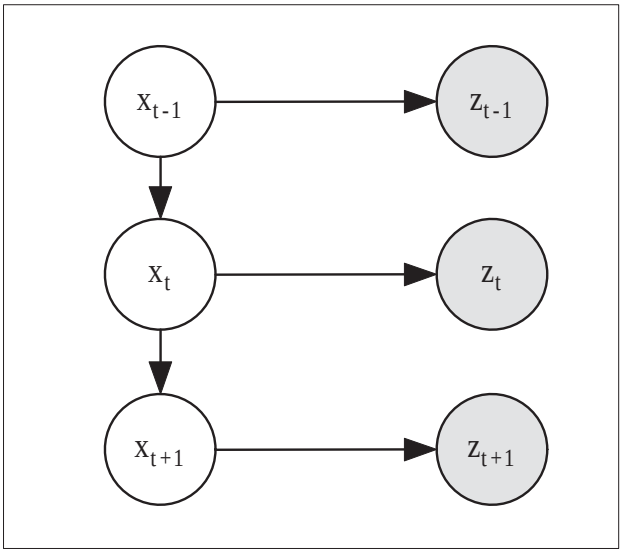

Figure 3.1: A DBN for a state-space model.

of the models themselves are governed by a discrete-state Markov chain. Since this model contains both discrete and continuous variables, it is sometimes called a hybrid DBN [29]. A DBN for multiple switching dynamic models is illustrated in Figure 3.2. In this graph,

- $\mathbf{z}$ is an observable variable.
- $\mathbf{x}_t$ is dependent on $\mathbf{x}_{t-1}$ and m_t.
- m_t is only dependent on m_{t-1}.
- $\mathbf{z}_t$ is dependent on $\mathbf{x}_t$.

3.8 Summary

In this chapter, we focus on reviewing the relevant background for the thesis work.

- We describe the nonlinear filtering problem. We introduce the system model and the measurement model.

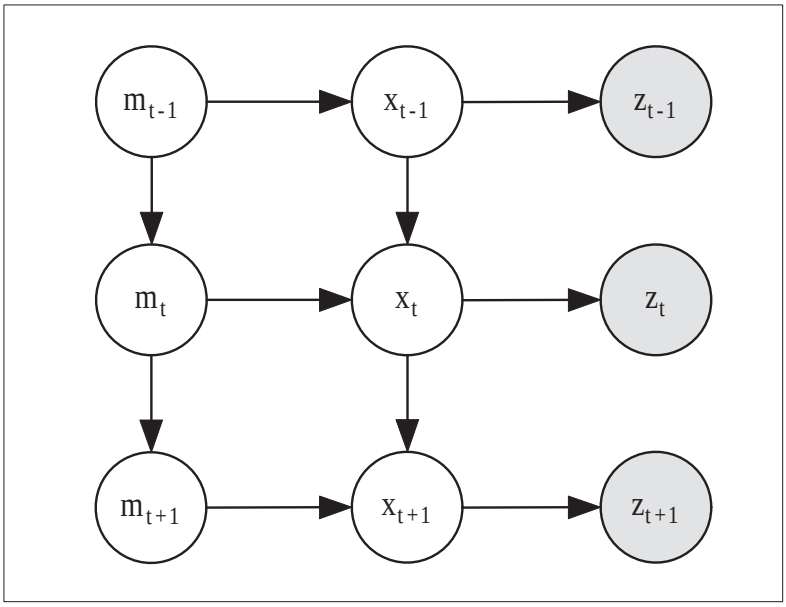

Figure 3.2: A DBN for multiple switching dynamic models.

- We describe the general recursive Bayesian filter. We show the prediction step and update step in a recursive Bayesian filter.
- We describe briefly the Kalman filter and its applicability.
- We describe the sequential importance sampling framework and the particle filter algorithm.
- We present the multiple switching dynamic models. We present the multiple model particle filter which is used to perform nonlinear filtering with switching dynamic models.
- We describe the dynamic Bayesian network and how to represent the state-space model and multiple switching dynamic models with a DBN.

Chapter 4

Tracking Using Own Actuation Model

Mobile robots need to track objects e.g., [35]. When tracking is performed by a robot executing specific tasks acting on the object being tracked, such as a Segway RMP soccer robot grabbing and kicking a ball, the motion model of the object becomes complex, and dependent on the robot's actions [25]. In this chapter we show how this thesis contributes the use of multiple motion models as a function of the robot's tactic within a particle-filter based tracker.

Over the years, a lot of different sensors such as vision sensors, infrared and ultrasound sensors have been used in the robotics community. For the environments the Segway RMP operates in, there are few sensors that can compete with color vision for low cost, compact size, high information volume and throughput, relatively low latency,[1] and promising usage for object recognition. Thus, we choose vision as the primary sensor. Recently, we have equipped each robot with an infrared sensor to reliably detect objects close to it. We introduce how this additional information can be combined with vision sensor to achieve more effective tracking.

[1]Latency of the digital camera refers to the amount of time between the the moment the shutter is pressed and when the camera is ready to take the next shot. So it is the time to process a photo and store it on the internal memory card.

4.1 Tactics

Our control architecture, called Skills-Tactics-Plays, consists of *Skills* for low-level control policies, *Tactics* for high-level single robot behavior, and *Plays* for team coordination [7]. We construct the robot cognition using this architecture and in this chapter we focus on skills and tactics that form the components for single robot intelligence. Skills can be connected as a finite-state-machine for a given tactic. Thus, a tactic can perform a range of complex actions by triggering the appropriate sequence of skill execution.

Figure 4.1 (a) shows the skill state machine for a simple `chase_ball` tactic, which contains two skills, `search` and `go_near_ball`. The tactic starts from `search`, and when the ball is visible then it transfers to the skill `go_near_ball`. If the ball is lost, the state machine transfers back to `search`. Figure 4.1 (b) shows the skill state machine for a more complex `grab_and_kick` tactic. This tactic executes a sequence of skills namely `search`, `go_near_ball`, `grab_ball`, `aim`, and the final `kick` skill. Transitions between skills are based on the perceived state.

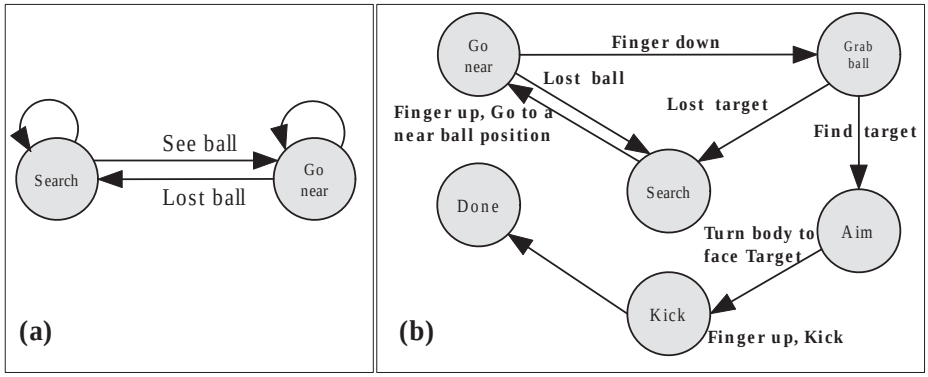

Figure 4.1: Skill state machine for tactics. Each node is a skill and the edges show the transition between skills. Transitions between skills are based on the perceived state. (a) Skill state machine for tactic `chase_ball`. (b) Skill state machine for tactic `grab_and_kick`.

Table 4.1 enumerates the skills used in this chapter along with brief descriptions. In this chapter we use these two tactics, namely `chase_ball` and `grab_and_kick`.

In order to use the tactics as information for the tracker, we need to map tactics into motion models of the object being tracked.

Skill	Description
`search`	turn body and camera until the ball is seen
`go_near_ball`	move to a position near the ball and look at the ball
`grab_ball`	put down the catcher(finger)
`aim`	look for the target and turn body to look at it
`kick`	put up the catcher and use the kicker to kick the ball out

Table 4.1: List of skills used in this chapter along with brief descriptions.

4.2 Tactic-Based Model Transitions

In this section, we take the ball-tracking problem as a detailed example to show the tactic-based motion modeling method in general.

4.2.1 Multi-Model System

As described in Section 3.5, the general parameterized state-space system is given by:[2]

$$\mathbf{x}_t = f_{t-1}(\mathbf{x}_{t-1}, m_t, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}) \tag{4.1}$$

$$\mathbf{z}_t = h_t(\mathbf{x}_t, m_t, \mathbf{w}_t) \tag{4.2}$$

where f_t and h_t are the parameterized state transition and measurement functions at time t; $\mathbf{x}_t, \mathbf{u}_t, \mathbf{z}_t$ are the state, input and measurement vectors; $\mathbf{v}_t, \mathbf{w}_t$ are the process and sensor noise vectors of known statistics. The model variable m_t can take any one of N_m values, where N_m is the number of models used in our system. For details of the multi-model system, please refer to Section 3.5.

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

- *Free-Ball*. The ball is not moving at all[3] or moving straight with a constant speed decay d which depends on the environment surface and sampling period Δt. We obtain the value of d by experiments on different surfaces (See detail in Section 4.4.1).

$$\mathbf{x}_t = \mathbf{F}_t^{Free}\mathbf{x}_{t-1} + \mathbf{v}_{t-1}^{Free} \tag{4.3}$$

$$\mathbf{z}_t = \mathbf{H}_t^{Free}\mathbf{x}_t + \mathbf{w}_t^{Free} \tag{4.4}$$

[2]We add an input vector $\mathbf{u}$ to the state transition function.

[3]We do not have *Stopped-Ball* model because it is merely a special case of *Free-Ball* when the ball speed is zero.

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 in global coordinates at time t; $\dot{x}_t, \dot{y}_t$ are the ball's velocity in x and y direction in global coordinates at time t; z_{x_t}, z_{y_t} are the ball's measurement in the x and y direction in global coordinates at time t. The superscript "*Free*" indicates the model index. $\mathbf{F}_t^{Free}$ and $\mathbf{H}_t^{Free}$ are known matrices as follows:

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

where Δt is the time interval between two consecutive vision frames, d is the speed decay term depending on surface.

- *Grabbed-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 their relative position, which is assumed to be a constant, plus noise. These two noise variables (position noise and velocity noise) form the noise vector $\mathbf{v}^{Grabbed}$. We use the same measurement model as Equation 4.4.

- *Kicked-Ball.* The ball is kicked therefore its velocity is equal to a predefined initial speed plus noise. And its position is equal to its previous position plus noise. These two noise variables form the noise vector $\mathbf{v}^{Kicked}$. We use the same measurement model as Equation 4.4.

- *Bounced-Ball.* This model is only used in simulation [4] to describe the motion of the ball when it is bounced off one of the field borders. The motion of the ball is assumed to be reflected by the field border with a considerable amount of velocity reduction. In our simulation, the same parameter speed decay d is used to describe the amount of velocity reduction.

$$\mathbf{x}_t = \mathbf{F}_t^{Bounced}\mathbf{x}_{t-1} + \mathbf{u}_{t-1}^{Bounced} + \mathbf{v}_{t-1}^{Bounced} \tag{4.5}$$

  Depending on which field border the ball is bouncing off, we have four set of corresponding $\mathbf{F}_t^{Bounced}(i)$ and $\mathbf{u}_t^{Bounced}(i), i = 1, 2, 3, 4$. We use the same measurement model as Equation 4.4. [5]

[4] We add walls to the border of the field in the simulated experiment. In the robot tests, there are no walls at the field border.

[5] This model is only used in simulation. We do not consider the effect of the spinning of the object in order to simplify the measurement model. To use such model in the real robot tests, we probably need lots of noise to model the effect of unmeasured spin.

$$\mathbf{F}_t^{Bounced}(1) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -d & 0 \\ 0 & 0 & 0 & d \end{bmatrix}, \mathbf{u}_t^{Bounced}(1) = \begin{bmatrix} -\texttt{FIELD\_WIDTH}/2 \\ 0 \\ 0 \\ 0 \end{bmatrix};$$

$$\mathbf{F}_t^{Bounced}(2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -d & 0 \\ 0 & 0 & 0 & d \end{bmatrix}, \mathbf{u}_t^{Bounced}(2) = \begin{bmatrix} \texttt{FIELD\_WIDTH}/2 \\ 0 \\ 0 \\ 0 \end{bmatrix};$$

$$\mathbf{F}_t^{Bounced}(3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & -d \end{bmatrix}, \mathbf{u}_t^{Bounced}(3) = \begin{bmatrix} 0 \\ -\texttt{FIELD\_HEIGHT}/2 \\ 0 \\ 0 \end{bmatrix};$$

$$\mathbf{F}_t^{Bounced}(4) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & -d \end{bmatrix}, \mathbf{u}_t^{Bounced}(4) = \begin{bmatrix} 0 \\ \texttt{FIELD\_HEIGHT}/2 \\ 0 \\ 0 \end{bmatrix};$$

where `FIELD_HEIGHT` is the height of the total field, and `FIELD_WIDTH` is the width of the total field.

Table 4.2 concludes the index of the model variable m and the corresponding ball motion model in this chapter.

m	Ball motion model
1	*Free-Ball*
2	*Grabbed-Ball*
3	*Kicked-Ball*
4	*Bounced-Ball*

Table 4.2: The model index and the corresponding ball motion model.

4.2.2 Motion Modeling Based on the Tactic

From the previous section, we know that the model index m determines the present single model being used. For our ball tracking example, $m = 1, 2, 3, 4$ for the single model *Free-Ball*, *Grabbed-Ball*, *Kicked-Ball* and *Bounced-Ball* 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 v_t (such as ball state $\mathbf{x}_{t-1}$, infrared measurement s_t or the combination of two or more variables), is governed by an underlying Markov process, such that, the conditioning parameter can branch at the next time-step with probability

$$\pi_{i,j} = \mathrm{P}(m_t = i | m_{t-1} = j, \mathcal{T}_{t-1}, v_t) \tag{4.6}$$

where $i, j = 1, \cdots, N_m$. Since we can draw $\mathrm{P}(m_t = i | m_{t-1} = j)$ in an $N_m \times N_m$ table, we can create a table for Equation 4.6 with a third axis which is defined by the tuple $\langle \mathcal{T}_a, v_b \rangle$ as shown in Figure 4.2. Here the tactic $\mathcal{T}_a$ is the primary factor that determines whether m_i transits to m_j and what the probability is of the transition, while the information v_b determines the prior condition of the transition. This is why we call it tactic-based. Each layer in the graph is conditioned on a particular combination of the tactic executed and the additional information obtained.

With this tactic-based modeling method, we can obtain the corresponding motion models for the tactics shown in Figures 4.3. In Figure 4.3 (a), there are only two possible single models: *Free-Ball* and *Kicked-Ball*. We take the information "ball is near" as a branching parameter, which can be obtained by reasoning, using ball state information and robot's self state information. Because it is a binary parameter, we can use two tables to define all the transition probabilities. Similarly, in Figure 4.3 (b), we use the infrared binary measurement as the branching parameter. The two tables list the transition probabilities between any two single models conditioned on "the infrared sensor can/cannot sense the ball" respectively. In this way, we can build motion models for any existing tactics we have designed. Note that Figure 4.3 describes the motion model of the object, while Figure 4.1 describes the behavior model (tactic) of the robot. We construct the object motion model using the knowledge of the robot behavior model.

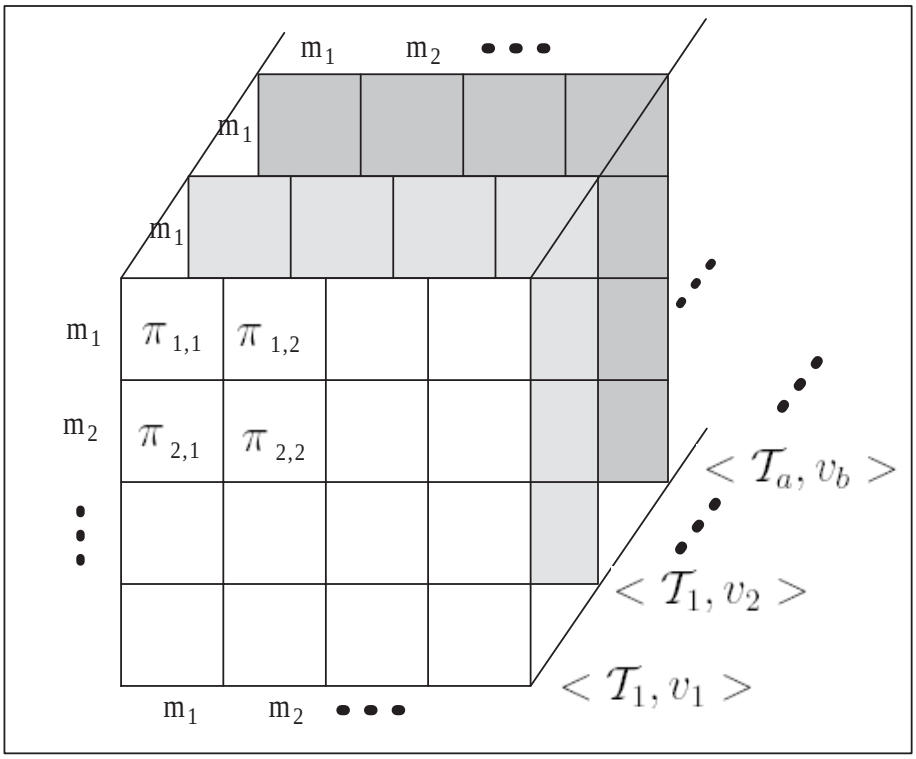

Figure 4.2: Tactic-based motion modeling, where $m_1, m_2, \cdots, m_n$ are n single models, $\mathcal{T}_a$ is the tactic, v_b is the additional information, and $n = N_m$, the number of models. $\pi_{i,j}$ is the transition probability from model m_i to model m_j given m_i, and $\langle \mathcal{T}_a, v_b \rangle$. Each layer in the graph is conditioned on a particular combination of the tactic executed and the additional information obtained.

In general, to build the Tactic-Based Motion Model (TBMM), we need first to use the information contained in tactics to identify the robot behavior. We use the robot behavior to identify the object motion models that are affected by the robot's actions. We construct the transition probability matrix of various object motion models based on the identified robot behavior and their actions on the object.

In other domains, we need to identify the known behavior model of the specific application. We apply the team-driven motion tracking framework and follow the introduced principles to decompose the motion modeling problem. The granularity of decomposing is domain-dependent. We take use of the individual agent's behavior knowledge to construct the object's motion model.

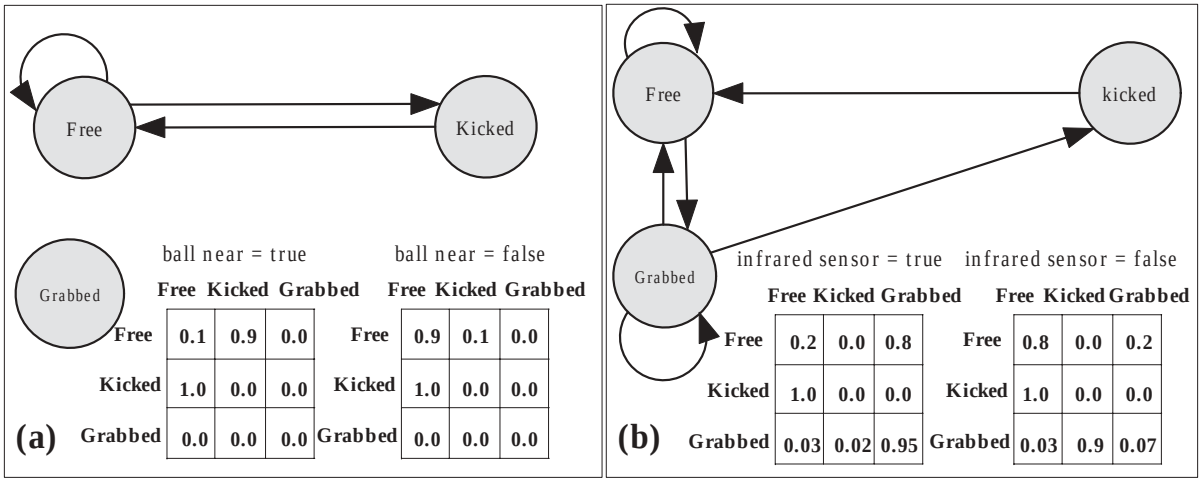

Figure 4.3: Ball motion model. Each node is a single model. The tables list the transition probability between any two single models. (a) Ball motion model based on tactic `chase_ball`. (b) Ball motion model based on tactic `grab_and_kick`.

4.3 Tactic-Based Object Tracking Algorithm

Following the tactic-based motion model given in the previous section, we can use a dynamic Bayesian network (DBN) to represent the whole system in a natural and compact way as shown in Figure 4.4. In this graph, the system state is represented by variables (tactic $\mathcal{T}$, infrared sensor measurement s, object state $\mathbf{x}$, object motion model index m, vision sensor measurement $\mathbf{z}$), where each variable takes on values in some space. The variables change over time in discrete intervals, so that $\mathbf{x}_t$ is the object state at time t. Furthermore, the edges indicate dependencies between the variables. For instance, the object motion model index m_t depends on $m_{t-1}, \mathcal{T}_t, s_t$ and $\mathbf{x}_{t-1}$, hence there are edges coming from the latter four variables to m_t. Note that we use an approximation here. We assume the measurement of the infrared sensor is always the true value, so it does not depend on the object state. Under this assumption, there is no edge from $\mathbf{x}_{t-1}$ to s_t, which greatly simplifies the DBN and the sampling algorithm as well.

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 DBN.

$$m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_t) \quad (4.7)$$

$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)}) \quad (4.8)$$

Note that in Equation 4.8, the object state is conditioned on the object motion model $m_t^{(i)}$ sampled from Equation 4.7.

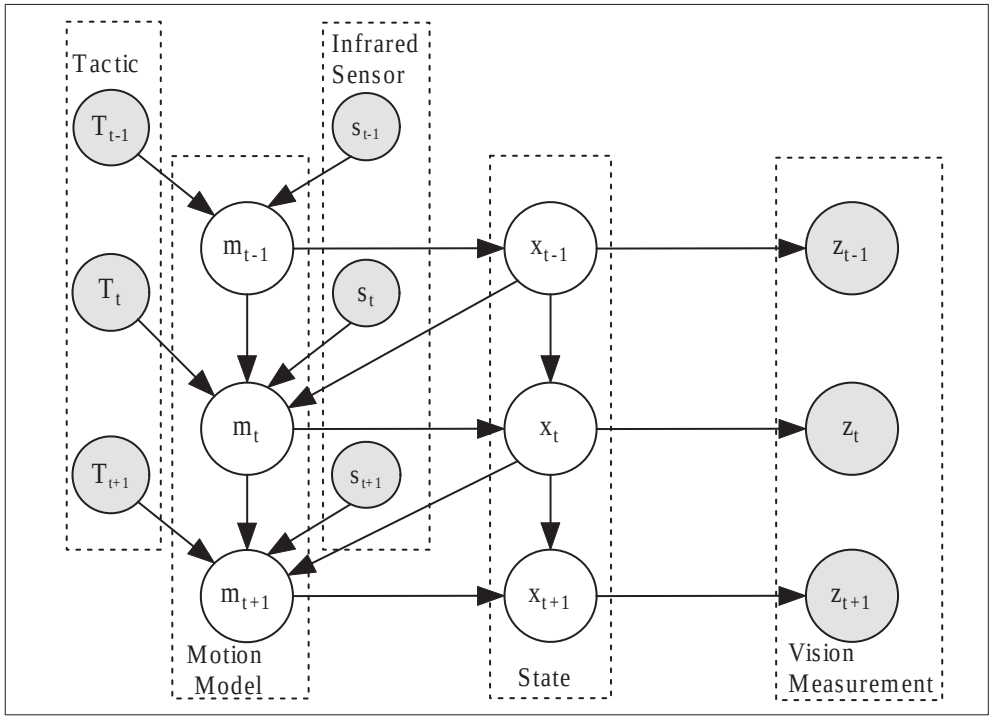

Figure 4.4: A dynamic bayesian network for tactic-based object tracking. Filled circles represent deterministic variables which are observable or are known as the tactic that the robot is executing.

Then we use the Tactic-Based Particle Filtering (TBPF) algorithm to update the state estimates. Table 4.3 lists our TBPF algorithm in detail.

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 $\mathbf{z}_t, s_t$, and the tactic $\mathcal{T}_t$. 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 object motion model index, $m_t^{(i)}$, is sampled according to Equation 4.7 at Line 2. Then given the model index, and previous object state, a new object state is sampled according

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

1 **for** $i \leftarrow 1$ **to** N_s
2 **do** draw $m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_t)$
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

Table 4.3: TBPF algorithm.

to Equation 4.8 at Line 3. The importance weight of each sample is given 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. Then we can estimate the object state based on the mean of all the $\mathbf{x}_t^{(i)}$.

4.4 Results

In this section, we first design experiments to estimate the ball speed decay in Δt (time interval between vision frames) on different surfaces. We profile the system and measurement noise. We give four performance metrics to present a performance comparison of three tracking algorithms. Finally we evaluate the effectiveness of our tracking system in both simulated and real-world tests.

4.4.1 Ball Motion and Measurement Noise Profiling

From previous work 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

$$v_0 = \sqrt{2gh}$$

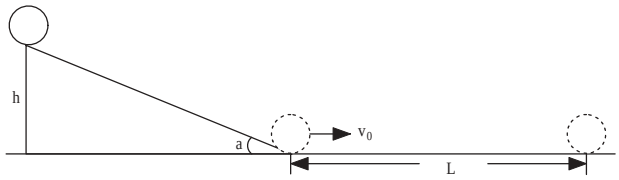

Figure 4.5: Test setup for estimating the ball speed decay d. The ball rolls off the ramp (with height h) with speed v_0 and it stops after it travels a distance of L.

without taking the friction on the surface of the ramp into account, where g is the gravity 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

$$d = 1 - \frac{v_0 \Delta t}{L}$$

where $\Delta t \approx 0.033$ sec. Following the test result, 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 more the ball's trajectory forms a straight line. We therefore model the system noise to be inverse 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, the smaller the observation noise. Therefore we choose to approximate the error distribution as different Gaussians based on the distance from the robot to the ball.

4.4.2 Metrics

We present a performance comparison of three tracking algorithms:

- A simple Kalman filter (KF) using a single model (*Free-Ball*).
- IMM using multiple motion models. Three motion models: *Free-Ball*, *Grabbed-Ball* and *Kicked-Ball* are considered. The transitional probability matrix Π_{IMM} used by IMM is the same as the one used by TBPF when infrared sensor reading is 'OFF' and the ball is not close to any field border (no bouncing possibility).

- TBPF using multiple motion models and multi-sensor observations. All four motion models mentioned in Section 4.2.1 are included. The transitional probability matrices are determined by robot's tactic and additional state information like the ball's predicted global position.

The comparison is based on a set of Monte-Carlo (MC) simulations and real robot test. Each MC simulation run generates different system and measurement noise according to the profiled noise model. We give a description of the three performance metrics that will be used in the analysis of this chapter and in the following chapters as well:

- The root-mean square(RMS) position error;
- Percent of time in the view of the camera;
- Model prediction correct rate.

Let $(\hat{x}_t^i, \hat{y}_t^i)$ and (x_t^i, y_t^i) denote the estimated and true object positions at time t at the i^{th} MC run. Let M be the total number of independent MC runs. The RMS position error at time t can be defined as

$$RMS_t = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\hat{x}_t^i - x_t^i)^2 + (\hat{y}_t^i - y_t^i)^2} \tag{4.9}$$

In the simulation, we use a simulated pan tilt camera mounted on the center roof, 5 meters over the field. The camera motion is modeled as two idealized rotations around the origin, followed by a perspective camera transformation. At time t of each MC run, we command the camera to point to the estimated object position $(\hat{x}_t^i, \hat{y}_t^i)$. We then calculate the effective field of view (FOV) of the camera at time t. We count how many times the true object position (x_t^i, y_t^i) falls inside FOV using the current tracking algorithm during the entire MC run. The percent of time that the object is within FOV can be computed as

$$\eta_{FOV} = \frac{1}{M}\sum_{i=1}^{M}\frac{\#\ of\ times\ the\ object\ is\ in\ FOV}{\#total\ time\ steps} = \frac{1}{M}\sum_{i=1}^{M}\frac{\#of\ times\ the\ object\ is\ in\ FOV}{T}\times 100\% \tag{4.10}$$

where T is the total time steps in each MC run.

Let $\hat{m}_t^i$ and m_t^i denote the estimated and true object motion model at time k at the i^{th} MC run. The model prediction correct rate is defined as

$$e = \frac{1}{M}\sum_{i=1}^{M}\frac{\#correct\ model\ prediction}{\#total\ time\ steps} = \frac{1}{M}\sum_{i=1}^{M}\frac{\hat{m}_t^i == m_t^i}{T} \times 100\% \tag{4.11}$$

Another metric is the non-Free-Ball model prediction correct rate.

$$e' = \frac{1}{M}\sum_{i=1,\hat{m}_t^i \neq 1}^{M}\frac{\hat{m}_t^i == m_t^i}{T} \times 100\% \tag{4.12}$$

4.4.3 Simulation Experiments

Because it is difficult to know the ground truth of the ball'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 `grab_and_kick` tactic (Figure 4.1 (b)). Noises are simulated according to the model profiled in the previous section. A typical scenario to be investigated, shown in Figure 4.6, involved tracking of a moving ball from the viewpoint of a robot during the first 100 time steps ($T = 100$). The solid line shows the ground truth of the ball's moving trajectory. The ball, which is initially static at the position (0,0), represented with an x-mark in the figure, is kicked by the robot at time $t = 1$. The ball's trajectory is illustrated with the solid line segments. At time $t = 20$, the ball bounces off the upper field border (blue circle in the figure). At time $t = 24, 59, 73$, the ball bounces and changes its velocity orientation. At time $t = 90$, the ball is grabbed by the robot (black circle in the figure). The ball is kicked again at time $t = 94$.

The transitional probability matrix Π_{IMM} used in the simulation is as follows.

$$\Pi_{IMM} = \begin{bmatrix} 0.75 & 0.20 & 0.05 \\ 0.15 & 0.05 & 0.80 \\ 0.80 & 0.10 & 0.10 \end{bmatrix} \tag{4.13}$$

Here are the transition probability matrices $\Pi_{s_t=0,close_t=0}$, $\Pi_{s_t=1,close_t=0}$ used by TBPF in the simulation, where s_t is the infrared sensor reading at time t, $close_t$ is a binary indicator.

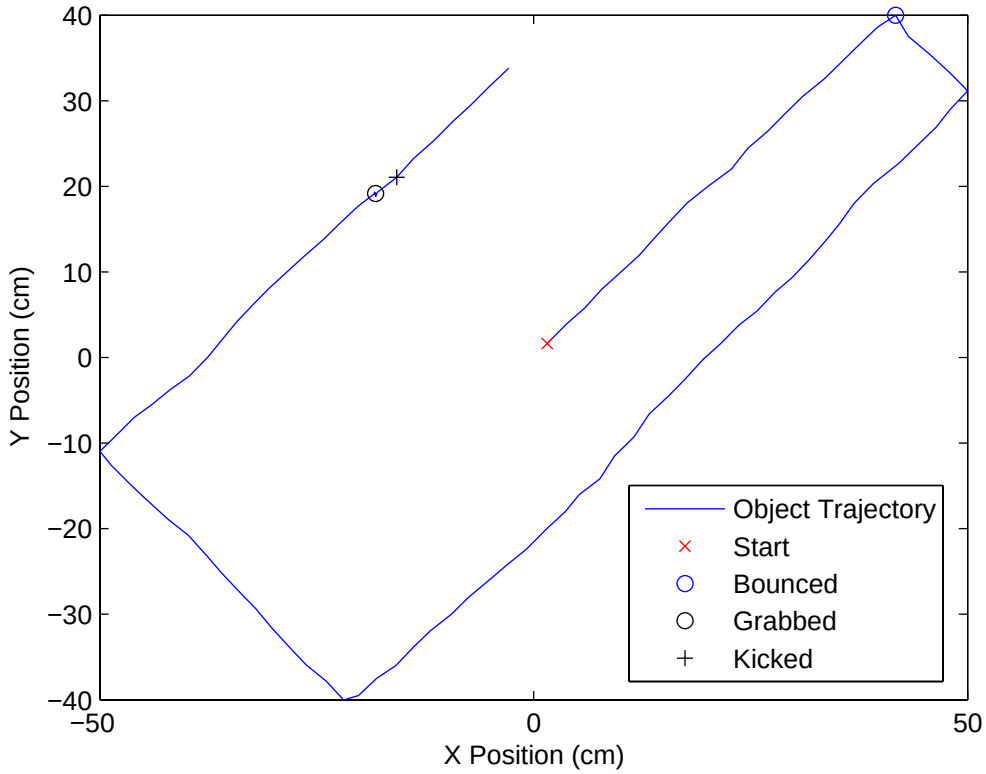

Figure 4.6: A typical ball trajectory when a robot is executing `grab_and_kick` tactic.

When the distance from the predicted ball position at time t to any field border line is less than 5 cm, $close_t$ is set to be true.

$$\Pi_{s_t=0,close_t=0} = \begin{bmatrix} 0.75 & 0.20 & 0.05 & 0 \\ 0.15 & 0.05 & 0.80 & 0 \\ 0.80 & 0.10 & 0.10 & 0 \\ 1.0 & 0 & 0 & 0 \end{bmatrix}, \Pi_{s_t=1,close_t=0} = \begin{bmatrix} 0.10 & 0.85 & 0.05 & 0 \\ 0.05 & 0.90 & 0.05 & 0 \\ 0.85 & 0.05 & 0.10 & 0 \\ 1.0 & 0 & 0 & 0 \end{bmatrix} \quad (4.14)$$

$$p(m_t = 4 | m_{t-1} = i, close_t = 1) = 0.9, \quad i = 1, 2, 3 \quad (4.15)$$

$$p(m_t = 1 | m_{t-1} = 4) = 1 \quad (4.16)$$

Unless otherwise mentioned, the nominal filter parameters used in the simulation are listed

in Table 4.4.

Parameter	Value	Description
Δt	0.033	The time interval between two consecutive vision frames
d	0.999	Speed decay
T	200	Total time steps in one Monte Carlo run
M	50	The number of MC runs
N_s	1000	The number of particles
N_{thr}	$N_s/3$	Particle resample threshold
w_0	[0.9 0.1 0 0]	Initial model probabilities

Table 4.4: The nominal filter parameters used in the simulation.

The particle filter uses $N_s = 1000$ particles with resampling only when the effective sample size $\hat{N}_{eff} = 1/\sum_{i=1}^{N_s}(w_t^i)^2 < N_{thr} = N_s/3$. The initial model probability w_0 is set to be [0.9 0.1 0 0]. That is,

$$p(m_1 = 1) = 0.9, \tag{4.17}$$

$$p(m_1 = 2) = 0.1. \tag{4.18}$$

The kicking speed is 100 cm/s and the direction is always north-east. The process noise v and the sensor noise w are set to be identical in the four models.

$v \sim \mathcal{N}(0, Q)$, the process noise covariance matrix

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \tag{4.19}$$

$w \sim \mathcal{N}(0, R)$, the sensor noise covariance matrix

$$R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \tag{4.20}$$

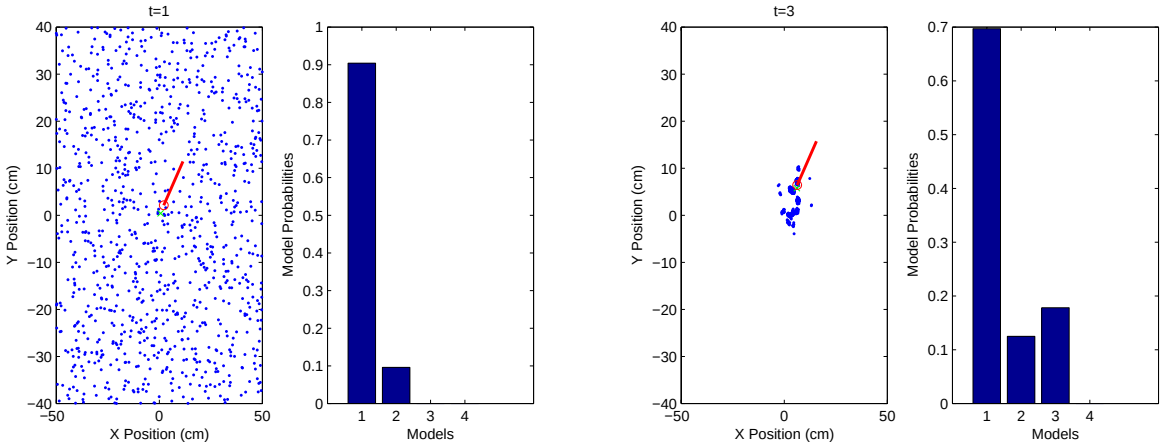

Figure 4.7: A snapshot of the particle cloud and the model probabilities at $t = 1$ and $t = 3$.

Figures 4.7, 4.8, and 4.9 display the output of the TBPF at time $t = 1$, 3, 19, 24, 80, and 86 in one MC run, respectively. The left-hand side in this set of figures always shows the distribution of particles $(x_t^{(i)}, y_t^{(i)})$ at time t, with their weighted mean value $(\hat{x}_t, \hat{y}_t)$ represented by a green x-mark. The true position of the ball at time t (x_t, y_t) is represented by a red circle. The true velocity of the ball at time t $(\dot{x}_t, \dot{y}_t)$ is represented by a red line. The right-hand side presents the probabilities of models. For example, at $t = 1$ (Figure 4.7), the ball is at its initial position. The particles are initialized to be uniformly distributed over the entire field. The initial model probability $w_0 = [0.9\ 0.1\ 0\ 0]$. At time $t = 3$, most of the particles are distributed close to the true ball position. The probability of model 1 (*Free-Ball*) is approximately 0.7, which is the dominant model. Observe from Figure 4.8, the ball is bouncing off the field border at time $t = 19$ and $t = 24$. The model 4 (*Bounced-Ball*) is becoming competitive with *Free-Ball* at time 19 and it is dominant at time $t = 24$. Figure 4.9 corresponds to $t = 80$ and $k = 86$ when the ball is being grabbed and finally kicked out. The change of model probabilities is consistent with the model variations.

The first trial examines the RMS position error versus time for the KF, IMM, and TBPF, and compares it with the computed CRLB. The performance results are shown in Figure 4.10 where we see that the TBPF achieves the best performance with the RMS error. While the KF and IMM show a tendency to diverge which results in subsequent poor track estimates. A summary of these results are also tabulated in Table 4.5.

The second trial examines the performance of IMM and TBPF for predicting the ball motion model. We use two performance metrics e and e' introduced in Section 4.4.2. A summary of the performance results are shown in Table 4.6 which shows the IMM's inability to

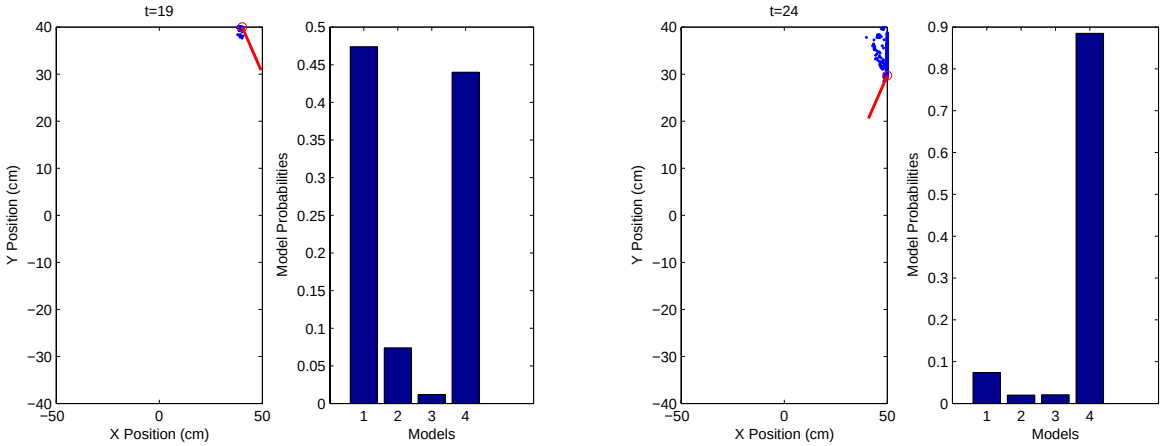

Figure 4.8: A snapshot of the particle cloud and the model probabilities at $t = 19$ and $t = 24$.

Algorithm	RMS Mean (cm)	RMS Std (cm)
KF	6.5701	0.6674
IMM	3.1277	0.6840
TBPF	0.9280	0.2108

Table 4.5: Performance comparison for three trackers

detect any motion model change. This is clear from the model probability curves shown in Figure 4.11 and 4.12. Alrough there is slight bump in the model probability for the correct change of model, the IMM algorithm is unable to establish the occurrence of actuation on the ball. The overall result is a track that is showing tendency to diverge from the true track. For the same set of parameters, the TBPF shows excellent performance as can be seen from Figure 4.6. Here we note the model probability curves for the TBPF shows that unlike the result of IMM, the TBPF model probabilities indicate higher probability of occurrences of actuation on the ball. The overall result is a much better tracker performance for the same set of measurements.

Algorithm	e (%)	std of e (%)	e' (%)	std of e' (%)
IMM	89.0	4.57	0	0
TBPF	95.3	1.57	78.16	8.06

Table 4.6: Performance comparison of IMM and TBPF for predicting the ball motion model

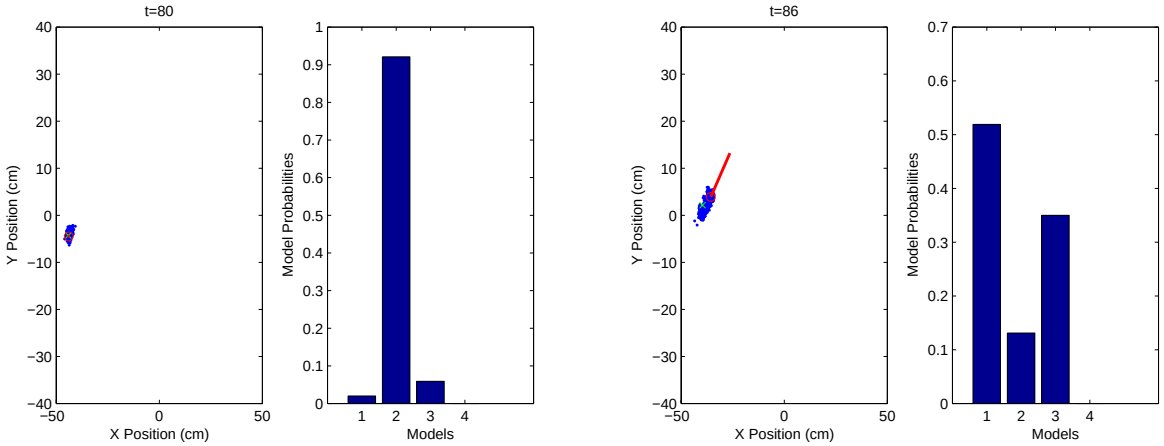

Figure 4.9: A snapshot of the particle cloud and the model probabilities at $t = 80$ and $t = 86$.

The third trial examines the performance of the considered filters as a function of the maximum kicking speed. Figure 4.13 and 4.14 show the RMS error and η_{FOV} as a function of the maximum kicking speed. As we expected, for the KF and IMM, the performance is acceptable when the maximum kicking speed is less than 60 cm/s. When the kicking speed gets bigger, the ability to detect the change of model becomes more important to the tracker, which results in the subsequent poor performance of KF and IMM in terms of RMS error and η_{FOV}. While the TBPF performs significantly better than the above two filters. We do not find noticeable performance decrease even if the kicking speed is as big as 140 cm/s.

4.4.4 Segway Soccer and Segway RMP Robot

The Segway platform is unique due to its combination of wheel actuators and dynamic balancing [8]. The Segway RMP, or Robot Mobility Platform, provides an extensible control platform for robotics research [18]. It imbues the robot with the novel characteristics of a fast platform and the ability to travel long ranges, carry significant payloads, navigate in relatively tight spaces for its size, and provides the opportunity to mount sensors at a height comparable to human eye level [36].

In the thesis work, we test our tracking algorithm on the Segway RMP robot [8] (Figure 4.15). We briefly describe the two major components of the control architecture, the sensor and the robot cognition, which are highly related to our motion tracking algorithm.

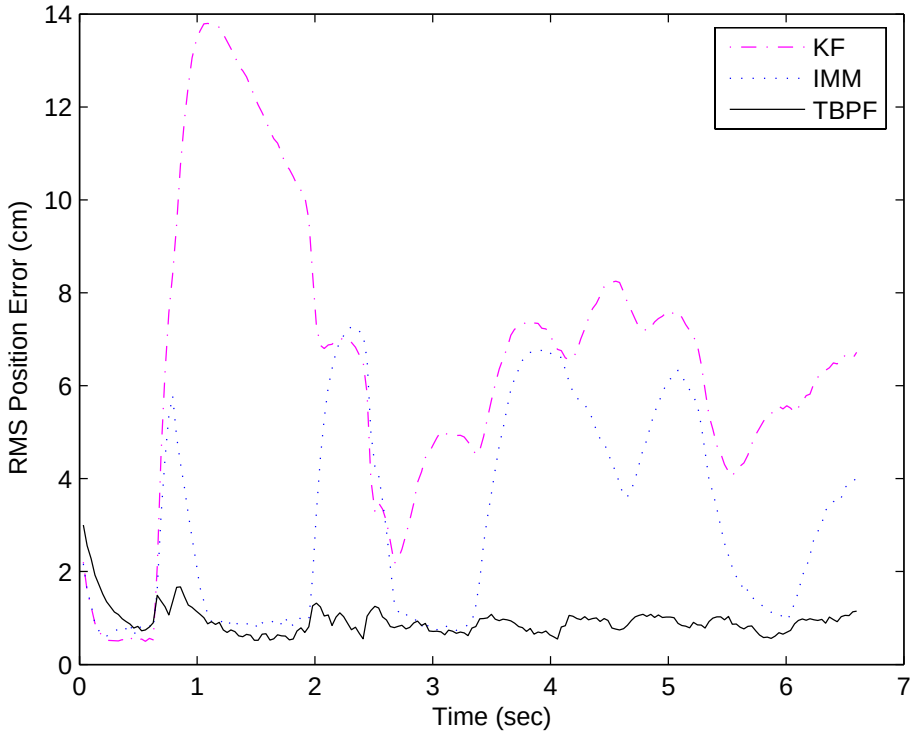

Figure 4.10: RMS position error versus time

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. We do not discuss the localization of the robot in the sense that a lot of soccer tasks can be done by the Segway RMP robot without localization knowledge. Also we use global reference for position and velocity in this thesis which means it is relative to the reference point where the robot starts to do dead reckoning.

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

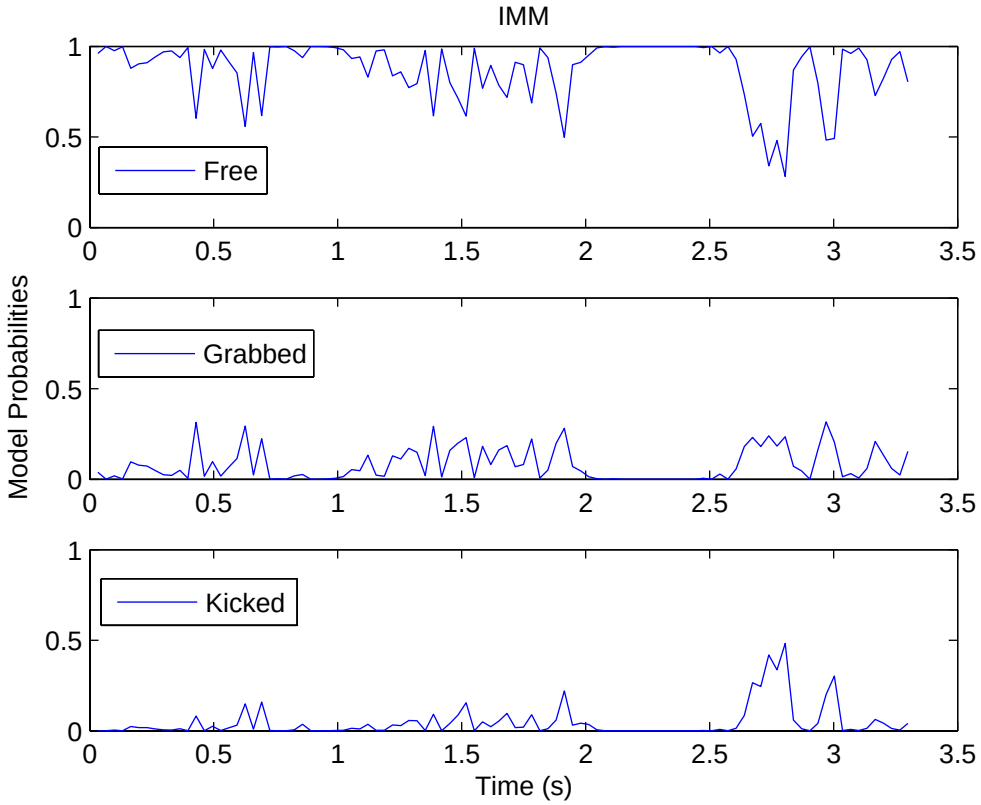

Figure 4.11: IMM tracking result: model probabilities

4.4.5 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 will be kicked directly to the robot. Then the robot 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`. And two seconds later after the kick if the robot can still see the ball, we count this run as successful. Anytime the robot begins executing the skill `search` a second time, we count that run as fail. That is to say, we only permit one searching which is 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 0.5 sec. In the experiments over 45 runs, the tracker with a single model fails 94.5% of the total runs. While the `grab_and_kick` based TBPF tracker only lost the ball 13.2% of the total runs.

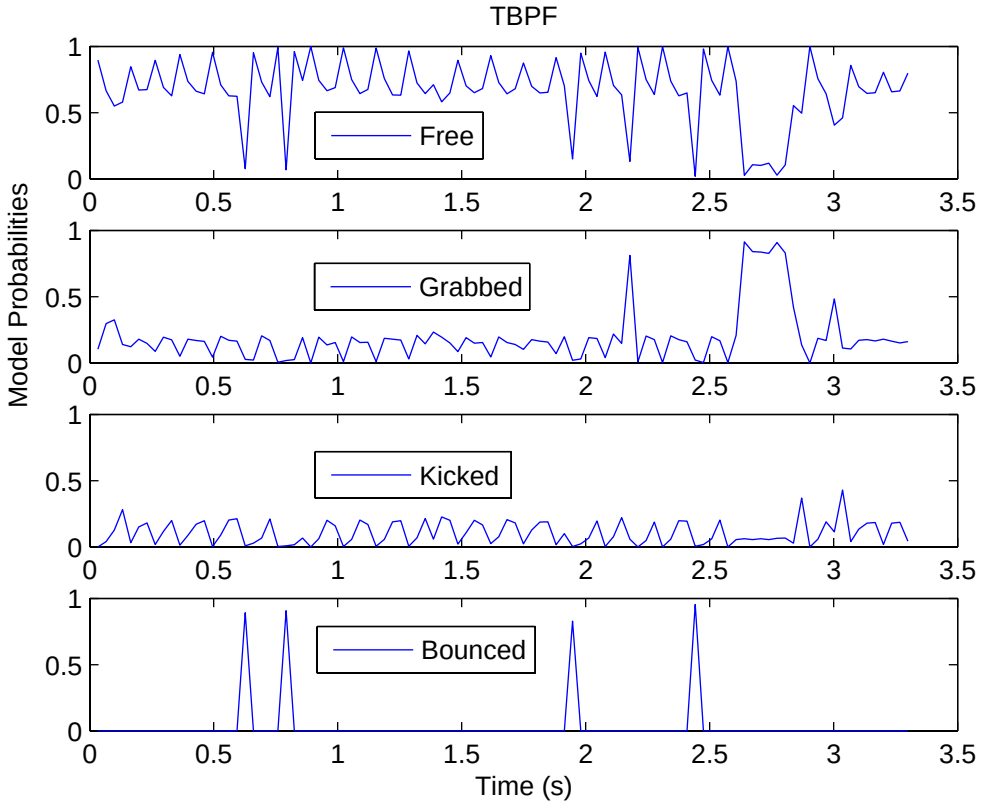

Figure 4.12: TBPF tracking result: model probabilities

Figure 4.16 shows how the TBMM tracker beats the single model tracker. In Figure 4.16 (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.

Because in each run, the ball is moving towards the robot, then it is 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 4.16 (a), the ball is visible in some few frames and is finally lost due to the underestimation of the ball speed. In Figure 4.16 (b), the ball is visible consistently thanks to the correct estimation of the ball speed as soon as the IR sensor outputs 0. This change of IR sensor measurement triggers the motion model of most particles transiting from

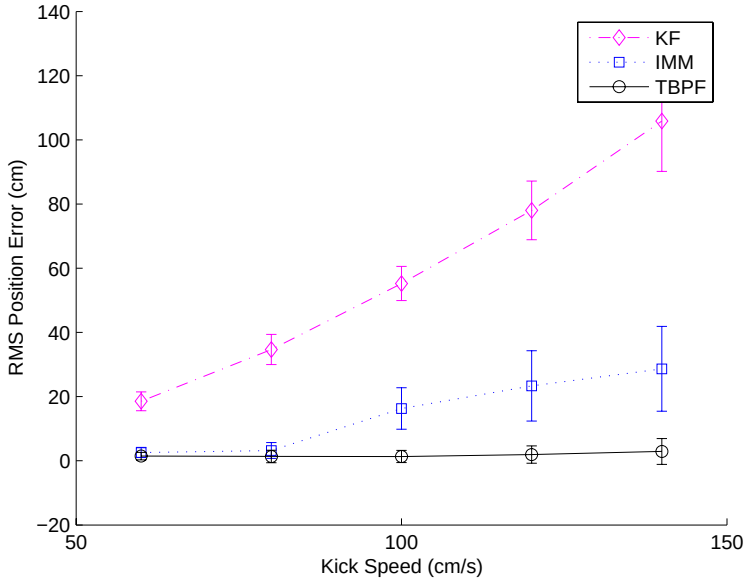

Figure 4.13: RMS position error versus the maximum kicking speed

Grabbed-Ball to *Kicked-Ball* then to *Free-Ball*, which models exactly what is going on in the real world.

4.5 Summary

In this chapter:

- We contribute a method to achieve efficient tracking through using a tactic-based motion model. The tactic-based motion modeling method gives the robot a more exact task-specific motion model when executing different tactics over the tracked object.
- We present a method to combine the vision and infrared sensory information. The infrared sensor provides useful information of the tracked object when the object moves into the blind area of the vision sensor.
- We represent the system in a compact dynamic bayesian network and propose a TBPF algorithm to keep track of the motion model and object state through sampling.

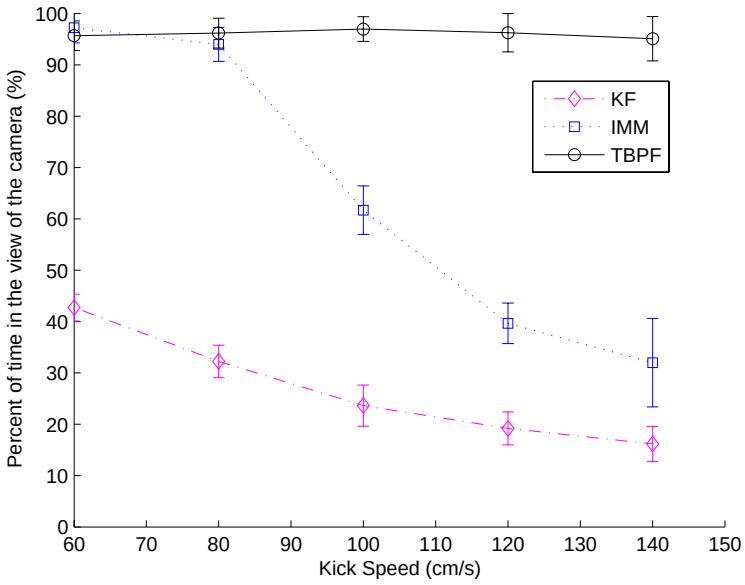

Figure 4.14: The percent of time in the view of the camera versus the maximum kicking speed

- We give three performance metrics to compare tracking performance of different trackers. We show the efficiency of TBPF over single model tracking and IMM through the empirical results from the simulated and the real experiments.

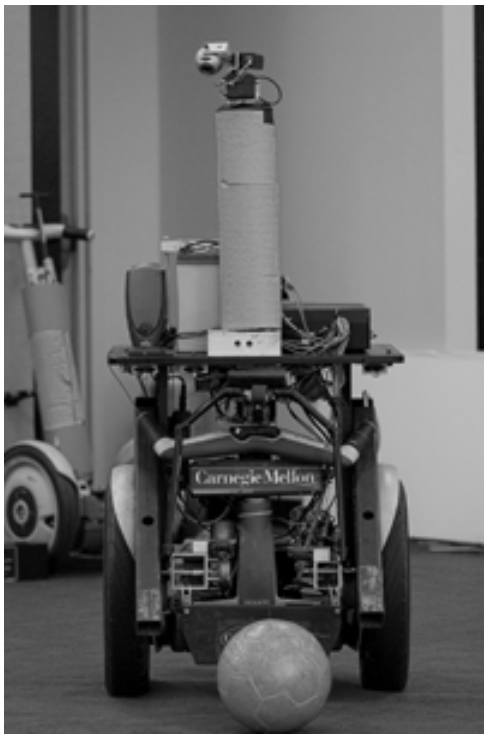

Figure 4.15: The Segway RMP soccer robot equipped with a kicker, a catcher, infrared sensors, and a camera mounted on a custom pan-tilt unit.

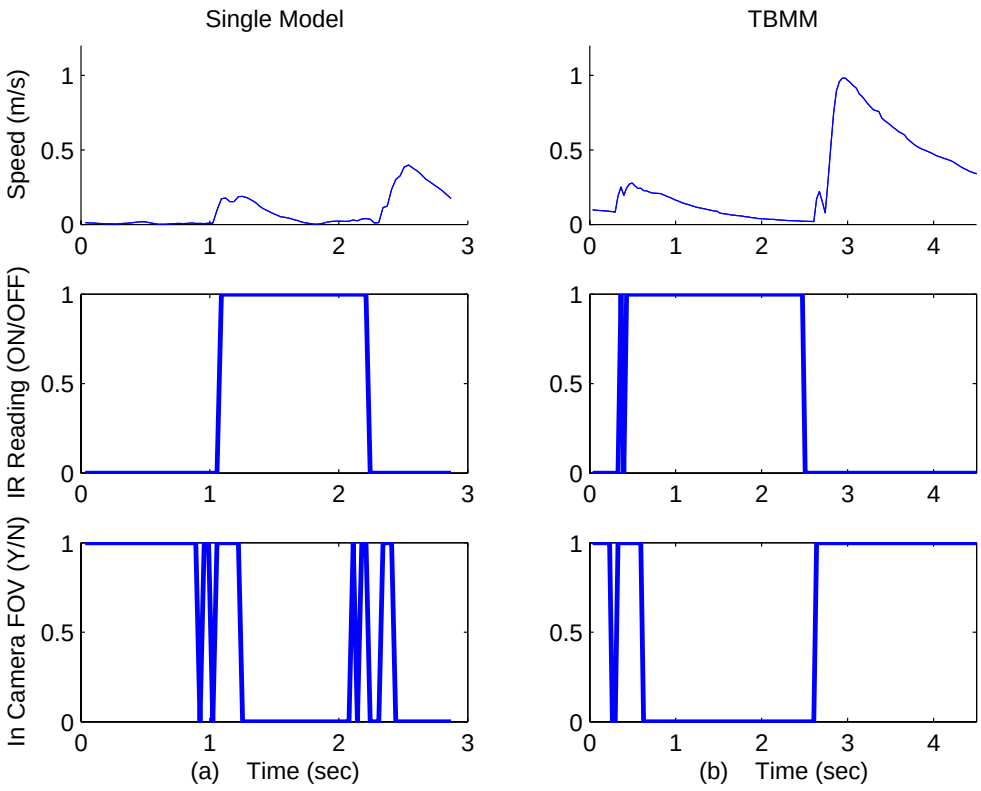

Figure 4.16: `grab_and_kick` speed estimation (a) Single model tracking. (b) Tactic-based multi-model tracking (TBPF).

Chapter 5

Tracking Using Team Actuation Model

Robots in a team collaborate according to pre-defined coordination plans or dynamic communication. For example, if it is the case that a teammate is manipulating the ball, the robot that is not controlling the ball can predict the ball's motion by reasoning about the possible teammate actions. 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. While we may not know of the actuation of the opponents, known team coordination plans provide valuable information that can be incorporated into the motion modeling and tracking process.

When team plans may not be available, we develop our solution to integrate the communication information into a team-driven multi-model motion tracking. If communication between teammates is allowed, the robot that is not controlling the ball can explicitly know what the action is from their teammates. Actuation on the targets is sent as communication messages to a specified team member to update and synchronize the motion model of the tracked target.

The organization of this chapter is as follows. We describe the concept of play as a team plan in our robot cognition architecture. We extend the tactic-based tracking scheme introduced in the previous chapter to solve a plan-dependent tracking problem. We then show our team-based mechanism to incorporate team communication into tracking. We evaluate our approach using simulated tests and Segway robots.

5.1 Plays

A play, P, is a fixed team plan consisting of a set of applicability conditions, termination conditions, and N roles, one for each team member [7]. 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 from the current step of the sequence for that role.

Two different plays are used in the test. The first play is defined in Table 5.1.

```
PLAY Play 1
APPLICABLE offense
DONE aborted !offense
ROLE 1
  pass 2
  none
ROLE 2
  position_for_pass
  receive_pass
  shoot
  none
```

Table 5.1: A simple example play: Play 1.

Each play has two roles as we focus on two teammates. A role consists of a list of tactics for the player to perform in sequence. In the example, play 1 in Table 5.1, there are two roles. The first role has only one tactic to execute. The second role has three sequenced tactics (behaviors). In this case the robot will position first. After the first tactic finishes, the robot filling that role will switch to the `receive_pass` tactic and try to intercept the ball. After the second tactic finishes, the robot will switch to `shoot` tactic and try to kick the ball toward the goal. The complete sequence of behaviors is illustrated in Figure 5.1.

The second play is defined in Table 5.2. In this play, both roles have sequenced behaviors, which require coordination between the two players. In this play, one player is assigned to pass the ball to another player. Once the pass behavior is completed, all the roles transition to their next behavior. The passing player will switch to a `position_for_pass` behavior,

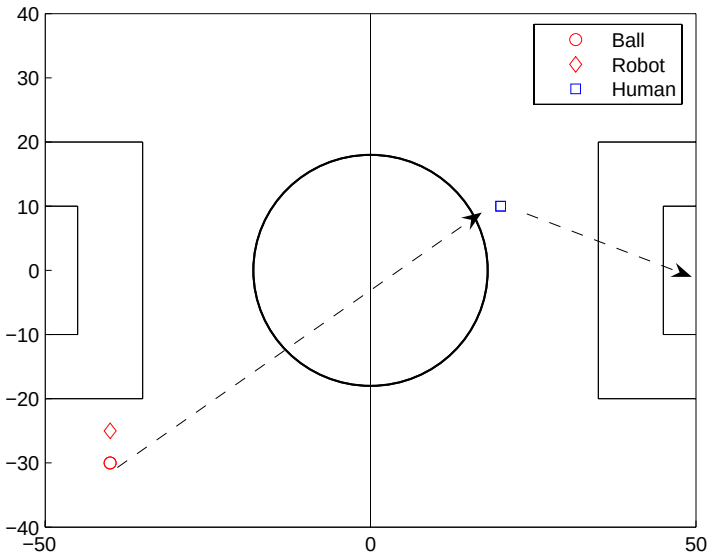

Figure 5.1: An illustration of the sequence of behavior execution of play 1.

```
PLAY Play 2
APPLICABLE offense
DONE aborted !offense
ROLE 1
   pass 2
   position_for_pass
   receive_pass
   pass 2
   none
ROLE 2
   position_for_pass
   receive_pass
   pass 1
   position_for_pass
   receive_pass
   shoot
   none
```

Table 5.2: Play 2 that involving sequencing of behaviors.

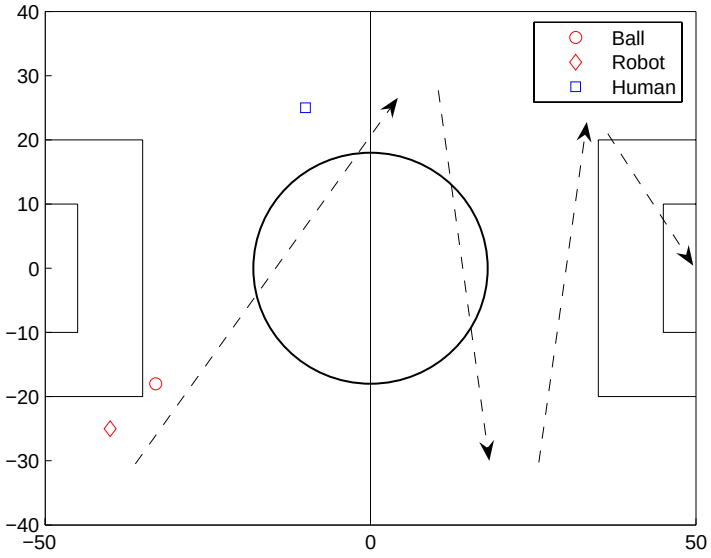

Figure 5.2: An illustration of the sequence of behavior execution of play 2.

and the target of the pass will switch to a behavior to receive the pass, after which it will switch to a `pass` behavior. The complete sequence of behaviors is illustrated in Figure 5.2. The figure shows the initial position of the ball, the robot and the human player. The figure also shows the moving trajectory of the ball as the dotted lines with arrows.

Table 5.3 concludes the tactics used in our example play along with a brief description.

Tactic	Description
`pass`	kick the ball toward a teammate.
`shoot`	kick the ball to go into the goal.
`position_for_pass`	move to a position on the field to anticipate a pass.
`receive_pass`	move to a position to receive a pass.

Table 5.3: List of tactics used in our example play along with brief descriptions.

5.2 Play-Based Model Transitions

In this section, we take an object tracking problem as a detailed example to show the extension of the tactic-based motion modeling 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 modeling 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.

5.2.1 Tracking Scenario

Many tracking scenarios involve multiple moving targets. In a Segway soccer test, we need to track the ball and the team member. Each team is identified by their distinct color. The ball is orange [40]. We use two separate trackers instead of one because we can differentiate the ball with the team member thanks to the color-based vision system.

Ball Motion Modeling

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

- *Free-Ball*, *Grabbed-Ball*, and *Bounced-Ball*. They are the identical models we use in Chapter 4.
- *Human-Grab-Ball*. The ball is held by the teammember. we can infer the ball position if we know the teammember position well.
- *Kicked-Ball*. The ball is kicked by the robot or the teammember. Based or whether it is a pass to the teammember or a shot at the goal, we further introduced two motion models *Passed-Ball* and *Shot-Ball*.

Table 5.4 concludes the model index and the corresponding ball motion model in this chapter.

Teammember Motion Modeling

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

m	Ball motion model
1	*Free-Ball*
2	*Grabbed-Ball*
3	*Human-Grabbed-Ball*
4	*Passed-Ball*
5	*Shot-Ball*
6	*Bounced-Ball*

Table 5.4: The model index and the corresponding ball motion model.

- *Random Walk*. The teammember is wandering 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 modeling.

- *Accelerating/Stopping*. The teammember dashes and obtains a velocity or stops 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.

5.2.2 Play-Based Motion Model

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 teammember is executing ($\mathcal{T}'_{t-1}$), which provides valuable information about the motion model of the teammember (m'_t). Both the robot and the teammember 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 teammember ($\mathcal{T}'_{t-1}$) 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 $\mathcal{T}'_{t-1}$ 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.

$$\pi'_{i,j} = \mathrm{P}(m'_t = i | m'_{t-1} = j, \mathcal{T}'_{t-1}, v'_t) \tag{5.1}$$

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

$$\pi'_{i,j} = \mathrm{P}(m'_t = i | m'_{t-1} = j, \mathcal{P}_{t-1}, v'_t) \tag{5.2}$$

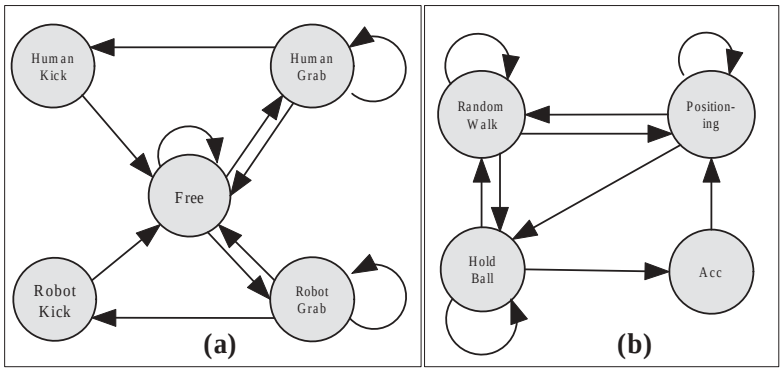

Figure 5.3: Object motion modeling based on the play: `Naive Offense`. Each node is a model. Models transit to one another according to the predefined probabilities (not shown in the figure). (a) Ball motion model. (b) Human teammember motion model.

For our ball tracking example, $m_t = i, i = 1, \cdots, 6$. Similarly,

$$\pi_{i,j} = \mathrm{P}(m_t = i | m_{t-1} = j, \mathcal{T}_{t-1}, \mathcal{T}'_{t-1}, v_t) \tag{5.3}$$

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

$$\pi_{i,j} = \mathrm{P}(m_t = i | m_{t-1} = j, \mathcal{P}_{t-1}, v_t) \tag{5.4}$$

Suppose the current team play is the `Play 1` in Section 5.1, we can obtain the corresponding motion model transitions for both the ball and the teammember using the play-based method (Figure 5.3).

In general, to build the Play-Based Motion Model (PBMM), we need first to use the information contained in plays to identify the related team behavior and individual robot behavior. We use the robot behavior to identify the object motion models that are affected by the robot's actions. We construct the transition probability matrix of various object motion models based on the identified robot behavior and their actions on the object.

5.3 Communication

In the previous section, we introduce an extension to the tactic-based tracking to solve the plan-dependent object tracking problem. The dynamic multiple motion models are based on the predefined team coordination plan. Communication between robots is not used and every robot coordinates based on observation only. In some cases, when the tactics of the team member are not exactly determined, the motion model of the target is unprecise so that the tracking performance will be affected seriously.

Communication improves the performance of a multi-agent system [19]. When communication is enabled, a shared world model that stores the state of the team can be constructed. The focus of this chapter is to present our solution to integrate the communication information into our team-driven multi-model motion tracking. The techniques that we describe are applicable to any domain where a team of agents are cooperating on a specific task and any of them can actuate on the targets in the field. Actuation on the targets is sent as communication messages to a specified team member to update and synchronize the motion model of the tracked target.

5.3.1 Types of Communicated Message

Each communicated message will be repeated for the several following time steps to avoid possible data loss in transmission. We define three types of communication messages in terms of different actuation model. Each type of message has a unique message ID.

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

5.3.2 Communication-Based Motion Model

Given the communication information ($\mathcal{C}_{t-1} = \{\text{NONE}, \text{HOLD}, \text{SHOOT}, \text{PASS}\}$), 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 actuation the teammate is doing.

We know that the model index m determines the present motion model being used. For our ball tracking example, $m_t = i, i = 1, \cdots, 6$. In our approach, it is assumed that the team member motion model index, m_t, conditioned on the previous tactic executed $\mathcal{T}_{t-1}$ by the robot, the communication message $\mathcal{C}_{t-1}$ which indicates teammate actuation, and other useful information v_t (such as ball state and infrared sensor reading), is governed by an underlying Markov process, such that, the conditioning parameter can branch at the next time-step with probability.

$$\pi_{i,j} = p(m_t = i | m_{t-1} = j, \mathcal{T}_{t-1}, \mathcal{C}_{t-1}, v_t) \tag{5.5}$$

Note that $\mathcal{C}_{t-1} = \text{NONE}$ indicates no message is received in the current time step. While $\mathcal{C}_{t-1} \neq \text{NONE}$ indicates a message is received concerned with a teammate actuation.

5.4 Team-Based Tracking Algorithms

In this section, we first use dynamic Bayesian networks to represent the whole system. We give the detailed Play-Based Particle Filtering (PBPF) algorithm of tracking in a multi-model multi-hypothesis scheme. We further include the communicated information to extend PBPF into Communication-Based Particle Filtering (CBPF).

5.4.1 DBN Representation

Following the play-based motion model, we can use dynamic Bayesian networks (DBNs) to represent the whole system for team member and ball tracking in a natural and compact way as shown in Figure 5.4 and Figure 5.5 respectively. For the rest of this section, we give the ball-tracking algorithm following Figure 5.5. The team-member-tracking algorithm can be obtained similarly following Figure 5.4.

5.4.2 Play-Based Particle Filtering (PBPF) Algorithm

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 [11]. It maintains the belief state at time t as a set of particles $p_t^{(1)}, p_t^{(2)}, \cdots, p_t^{(N_s)}$,

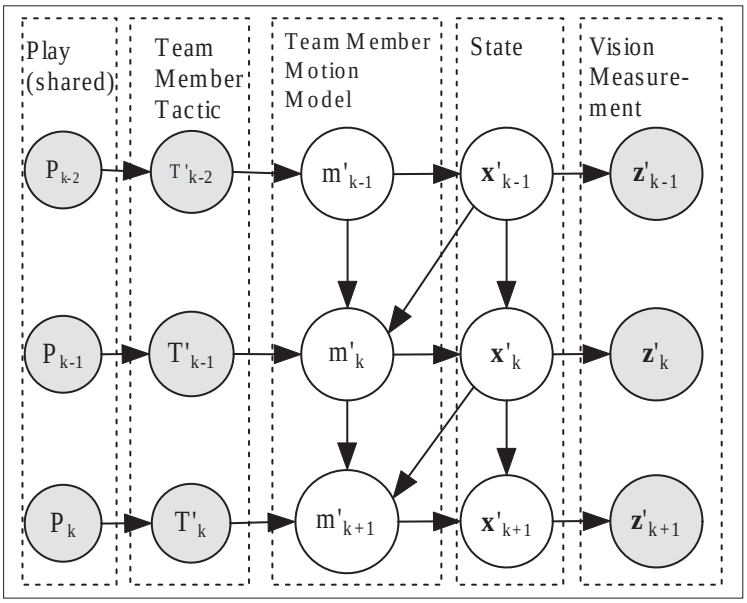

Figure 5.4: A dynamic Bayesian network for team member tracking. Filled circles represent deterministic variables which are observable or are known as the tactic or the play.

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, $p_t^{(i)} = \langle \mathbf{x}_t^{(i)}, m_t^{(i)} \rangle$.

We sample the object motion model following the object-tracking DBN as below:

$$m_t^{(i)} \quad \sim \quad p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_{t-1}, \mathcal{T}'_{t-1}) \tag{5.6}$$

Note that $\mathcal{T}_{t-1}$ and $\mathcal{T}'_{t-1}$ are inferred deterministically from $\mathcal{P}_{t-1}$ instead of sampling. Conditioned on the object motion model $m_t^{(i)}$, we then use the importance function introduced in [30] to sample the object state $\mathbf{x}_t^{(i)}$:

$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)}). \tag{5.7}$$

Table 5.5 lists our PBPF algorithm in detail.

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 $\mathbf{z}_t, s_t$, and the tactic $\mathcal{T}_t$. 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 object motion model index, $m_t^{(i)}$, is sampled according to Equation 5.6 at Line 2. Then given the model index, and previous object state, a new object state is sampled according

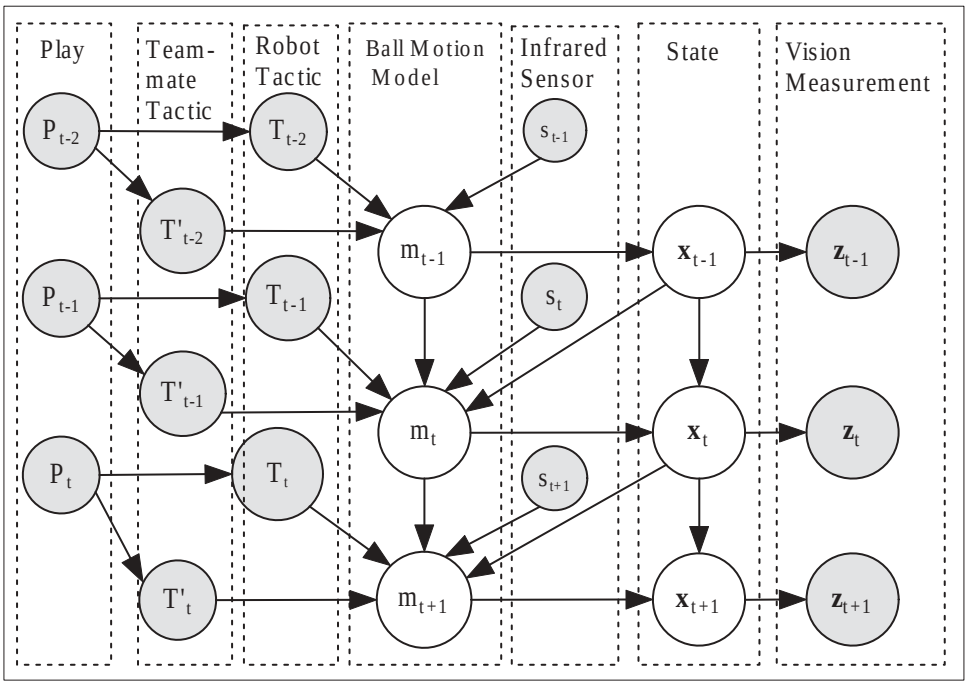

Figure 5.5: A dynamic Bayesian network for play-based object tracking.

to Equation 5.7 at Line 3. The importance weight of each sample is given 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. Then we can estimate the object state based on the mean of all the $\mathbf{x}_t^{(i)}$.

5.4.3 Communication-Based Particle Filtering (CBPF) Algorithm

With the communication information, robots do not need to infer teammate's tactic from the team play any more if received message is not NONE [17]. Instead, every actuation on the ball is announced to keep the ball motion updated among team members. After including communicated information, the DBN is updated as in Figure 5.6. Note that the motion model is not dependent on teammate's play or tactics when the received message is not NONE, since the communication message contains all the information presented by teammate's tactics.

Table 5.6 lists our CBPF algorithm in detail.

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

1 **for** $i \leftarrow 1$ **to** N_s
2 **do** draw $m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_t, \mathcal{T}_t')$
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

Table 5.5: PBPF algrithm.

Compared to the PBPF algorithm in Table 5.5, the modifications is made when we do receive a message from the teammates. Otherwise, it runs the same sampling procedures as in PBPF.

5.5 Results

In this section, we introduce the performance metrics we use to compare the tracking performance of three trackers. We evaluate the effectiveness of our tracking system in both simulated and real-world tests. We evaluate the target detection performance between our method and IMM.

5.5.1 Metrics

We present a performance comparison of four tracking algorithms:

- A simple Kalman filter (KF) using a single model (*Free-Ball*).
- TBPF using multiple motion models and multi-sensor observations introduced in Chapter 4. The new motion models that describe the human team member's actions on the object introduced in this chapter are not included. The transitional probability ma-

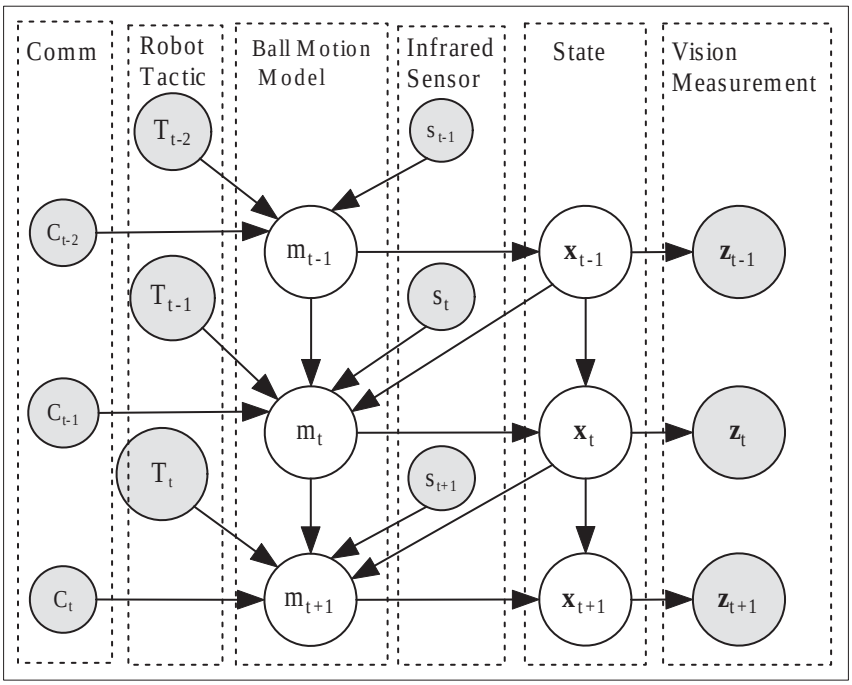

Figure 5.6: A DBN for team-driven object tracking when communication is enabled.

trices are determined by robot's tactic and additional state information like the ball's predicted global position.

- PBPF using all the motion models described in this chapter. The transitional probability matrices are determined by robot's tactic, the team member's tactic inferred from the play, and additional state information.
- CBPF using all the motion models described in this chapter. Besides the factors mentioned in PBPF, the transitional probability matrices are determined by the communicated information as well.

The comparison is based on a set of Monte-Carlo (MC) simulations and real robot test. Besides the performance metrics defined in Chapter 4, we give two more metrics.

- The root-mean square(RMS) velocity error;
- The execution time of a certain play. This metric shows how soon a play is finished. As we know, in a competition game, the less time we use to finish a play, the more chance we have to win.

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

1 **for** $i \leftarrow 1$ **to** N_s
2 **do if** $\mathcal{C}_{t-1} = \text{NONE}$
3 **then** draw $m^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_{t-1}, \mathcal{T}_{t-1}')$
4 **else** draw $m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, s_t, \mathcal{T}_{t-1}, \mathcal{C}_{t-1})$
5 draw $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)})$
6 $w_t^{(i)} \leftarrow p(\mathbf{z}_t | \mathbf{x}_t^{(i)})$
7 Calculate total weight: $w \leftarrow \sum[\{w_t^{(i)}\}_{i=1}^{N_s}]$
8 **for** $i \leftarrow 1$ **to** N_s
9 **do** Normalize: $w_t^{(i)} \leftarrow w_t^{(i)}/w$
10 Resample

Table 5.6: CBPF algrithm.

5.5.2 Simulation Experiments

Simulated experiments are done following the play 1 and play 2 in Table 5.1 and 5.2 respectively. In each trial, the ball's initial position is at (-40, -30) and its initial speed is set to zero. The robot's initial position is at (-35, -20). The human player's initial position is at (20, 10).

Unless otherwise mentioned, the nominal filter parameters used in the simulation are listed in Table 6.4. The kicking speed is 100 cm/s and the direction depends on whether it is a pass or shoot. The process noise v and the sensor noise w are set to be identical in all models. We decrease the speed decay d to consider more frictions and make the ball's movement slow down. As a result, simulated agents will be able to intercept the moving ball.

$v \sim \mathcal{N}(0, Q)$, the process noise covariance matrix

$$Q = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix} \tag{5.8}$$

$w \sim \mathcal{N}(0, R)$, the sensor noise covariance matrix

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.9}$$

Parameter	Value	Description
Δt	0.033	The time interval between two consecutive vision frames
d	0.96	Speed decay
T	200	Total time steps in one Monte Carlo run
M	50	The number of MC runs
N_s	500	The number of particles
N_{thr}	$N_s/3$	Particle resample threshold
w_0	[0.9 0.1 0 0]	Initial model probabilities

Table 5.7: The nominal filter parameters used in the simulation.

Figures 5.7 and 5.8 display the output of the PBPF using play 1 at time $t = 4$, 23, 24, 27, 75, 77, 79, and 86 in one MC run, respectively. The left-hand side in this set of figures always shows the distribution of particles $(x_t^{(i)}, y_t^{(i)})$ at time t, with their weighted mean value $(\hat{x}_t, \hat{y}_t)$ represented by a green x-mark. The true position of the ball at time t (x_t, y_t) is represented by a red circle. The true velocity of the ball at time t $(\dot{x}_t, \dot{y}_t)$ is represented by the red line. The estimated ball velocity $(\hat{\dot{x}}_t, \hat{\dot{y}}_t)$ is represented by the cyan line.

The right-hand side presents the probabilities of models. For example, at $t = 4$, the ball is still at its initial position. The robot is coming towards the ball. The particles are distributed surround the position of the ball. The dominant model is *Free-Ball*. At time $t = 23$, the IR sensor outputs 1 which indicates the ball is being grabbed by the robot. Most of the particles are distributed close to the true ball position. The probability of model 2 (*Grabbed-Ball*) is approximately 0.85, which is the dominant model at this time. Observe from the 3rd subgraph in Figure 5.7, the ball is being kicked out. Note that the ball velocity prediction (cyan line) is very close to the true ball velocity (red line). The dominant motion model changes to (*Passed-Ball*). At time $t = 27$, robot already finishes its behavior and it assumes the human player is executing the tactic `receive_pass`. The first subgraph in Figure 5.8 corresponds to $t = 76$ and the human player is holding the ball. The motion model prediction shows the current dominant model is model 3 (*Human-Grabbed-Ball*). At time $t = 79$, model 5 (*Shot-Ball*) is becoming dominant. In the last subgraph in Figure 5.8, the ball is bouncing off the field border. The change of model probabilities is consistent with the model variations.

The first trial examines the error performance of the PBPF using play 1 for varying values

of the number of particles. Figure 5.9 show the RMS position error as a function of the number of particles for $50 \leq N_s \leq 1000$. As expected, we see an improvement in performance as the number of particles is increased. However, note that as N_s is increased beyond 500, there is insignificant improvement in performance. Thus we us $N_s = 500$ in our following simulations.

The second trial examines the RMS position error and velocity error versus time for the KF, TBPF, and PBPF. The performance result following play 1 and play 2 are shown in Figure 5.10 and 5.11 respectively. A summary of these results are also tabulated in Table 5.8 and 5.9. We see that the PBPF achieves the best performance with the minimum RMS error compared to TBPF and KF. While the KF and TBPF show a tendency to diverge which results in subsequent poor track estimates. Note that TBPF performs much better than KF when the ball is manipulated by the robot. Because in this case, robot's own tactic provides enough information to predict the ball's motion model. While the ball is controlled by the human teammate, TBPF has little performance advantage over KF since it does not consider the teammate's actions on the ball. PBPF extends the idea of TBPF and includes the inferred teammate's tactic to predict the ball's motion model.

Algorithm	RMS Position Mean (cm)	RMS Position Std (cm)	RMS Velocity Mean (cm/s)	RMS Velocity Std (cm/s)
KF	2.1638	0.0692	38.3330	0.9038
TBPF	1.7748	0.2158	23.5554	2.8932
PBPF	0.5862	0.3599	4.1461	4.9243

Table 5.8: Play 1: RMS error comparison for three trackers

Algorithm	RMS Position Mean (cm)	RMS Position Std (cm)	RMS Velocity Mean (cm/s)	RMS Velocity Std (cm/s)
KF	2.1339	0.0572	37.1350	0.7692
TBPF	2.0683	0.2099	25.3736	4.7967
PBPF	0.5031	0.0680	4.0677	1.6480
CBPF	0.4951	0.1598	3.75	2.8338

Table 5.9: Play 2: RMS error comparison for three trackers

The third trial examines the metric η_{FOV} and the execution time for the KF, TBPF, and PBPF using play 1 and play 2. A summary of these results are tabulated in Table 5.10 and 5.11. The results show that the robot using PBPF almost keeps the ball in field of view all the time and finishes the execution of both plays in the shortest time. As we

expected, TBPF performs better than KF since approximately half of the time the robot is manipulating the ball. TBPF begins to show its inability to keep the ball in its camera's visible area after the robot passes the ball to the human player. Due to the inaccurate estimation of ball position and velocity, though the robot and the simulated human player use the identical way to intercept the ball, the player that use KF or TBPF takes more time to catch the ball.

Algorithm	η_{FOV} mean	η_{FOV} std	mean time used	std time used
KF	70.3	2.48	102.68	10.89
TBPF	84.22	4.33	96.9	10.12
PBPF	98.8	3.11	87.46	5.27

Table 5.10: Play 1: Performance comparison for three trackers

Algorithm	η_{FOV} mean	η_{FOV} std	mean time used	std time used
KF	67.12	2.47	195.2	19.67
TBPF	79.88	7.07	176.9	18.77
PBPF	99.86	0.38	162.9	9.82
CBPF	99.86	0.48	163	9.18

Table 5.11: Play 2: Performance comparison for three trackers

The fourth trial examines the performance improvement after integrating team communicated message using play 2. Figure 5.12 shows the RMS error for varying amount of sensor noise. Note that the default sensor noise $w \sim \mathcal{N}(0, R)$. In this test we set the covariance of sensor noise to be $2R, 5R, 8R, 11R$. From the result, we find that for a large amount of sensor noise, the CBPF shows the least degradation in performance.

5.5.3 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 5.13, 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 `receive_pass` to execute, in which the robot starts

with the skill `search`. 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 `kickto_teammember`, 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 is directed to a close area (the distance to the human is less than 0.5 m) to the human.

- **Fail** if the robot is in 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 seconds to scan the whole field using its pan-tilt camera. We pick a threshold value to be 30 seconds so as to allow the robot scanning no more than twice.

- **Fail** if the ball is outside the field boundary before the robot catches it.

In the experiment over 25 runs, the robot with single model trackers fails 8 of the total. While the robot with play-based multi-model trackers fails 3 of the total. We also keep track of the mean time taken in all the successful runs. We list the result in Table 5.12. 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 modeling greatly improves the accuracy of the teammember motion model and therefore avoids taking time in searching a lost target from scratch.

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

Table 5.12: The average time taken over all the successful runs.

5.5.4 Team Communication Test

In the real-world test, we do experiments on the Segway RMP soccer robot executing the `receive_pass` tactic to receive ball from robot teammate. The teammate robot is executing the `pass` tactic. When the kick motion is done, the teammate announces the "PASS" message through peer-to-peer communication. We implement multi-model tracker with and without including communication information to compare the performance.

Figure 5.14 plots the ball speed estimation results from each tracker in one of the experiment. As is shown in the figure, the estimation without communication lags from the true value about 0.5 s, while the estimation with communication is well predicted because the announcement is received right after the actuation is made. To illustrate the superior tracking of the tracking with communication, Figure 5.15 and 5.16 plots the corresponding model weighting from each tracker. We can clearly see that the model transition in Figure 5.16 exactly describes the real world testing scenario in which motion model transits from *Free-Ball* to *Passed-Ball*, then *Free-Ball* and at last *Grabbed-Ball*.

5.6 Summary

In this chapter:

- We extend the tactic-based tracking to include team-level coordination plan. This extension enables the tracker to use more exact motion models when team members are acting on the tracked object.
- We give a Play-Based Particle Filtering algorithm to keep track of the motion model and object state through sampling.
- We include communicated messages between teammates to further improve the motion model of the object. We give a Communication-Based Particle Filtering algorithm that extends PBPF when communication is enabled.
- We use a few performance metrics proposed in previous chapter. We give two new performance metrics to compare tracking performance of different trackers.
- We show the efficiency of the PBPF and CBPF over single model tracking and TBPF through the empirical results from the simulated and the real experiments.

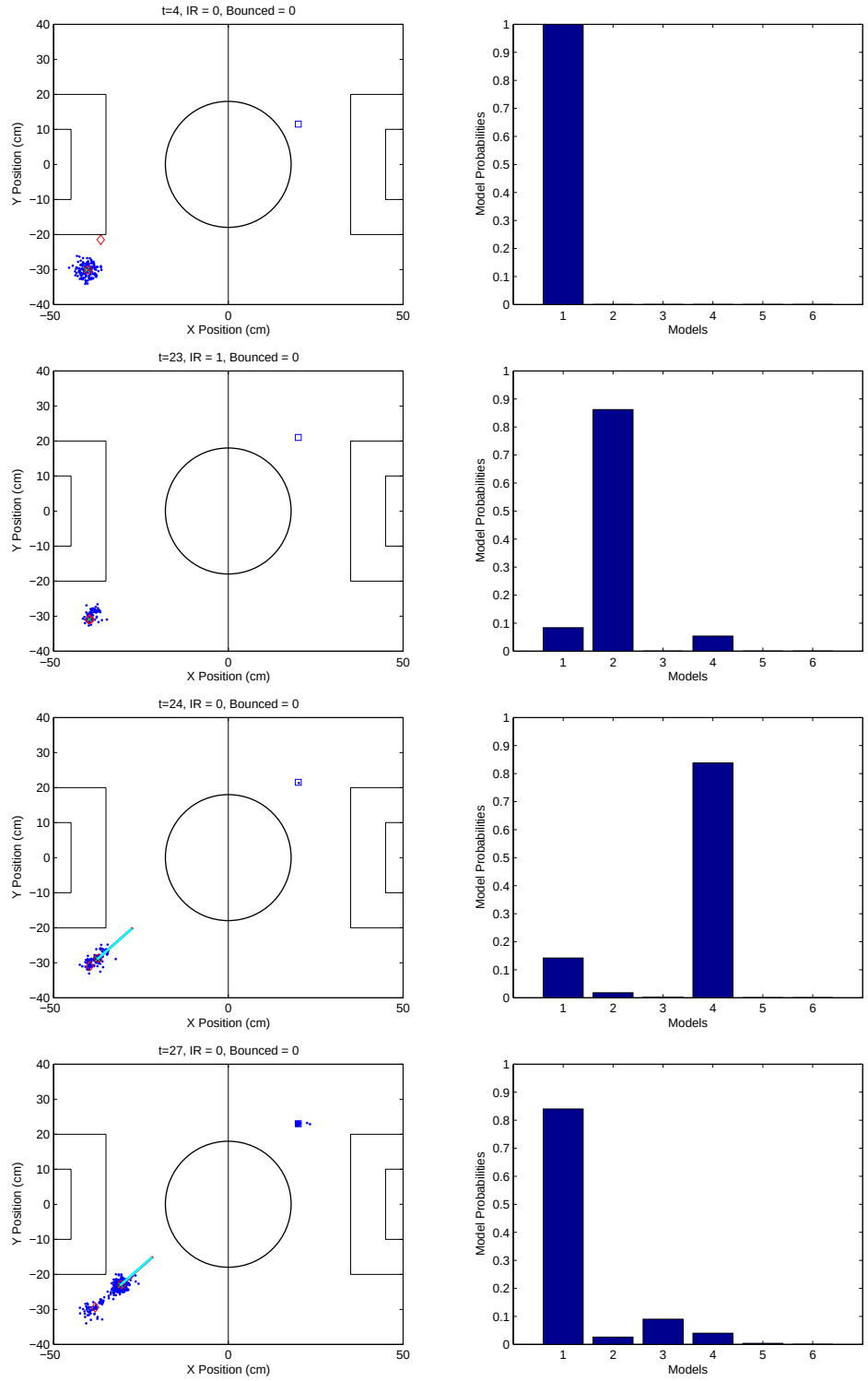

Figure 5.7: The snapshots of the particle cloud and the model probabilities at $t = 4$, 23, 24, and 27.

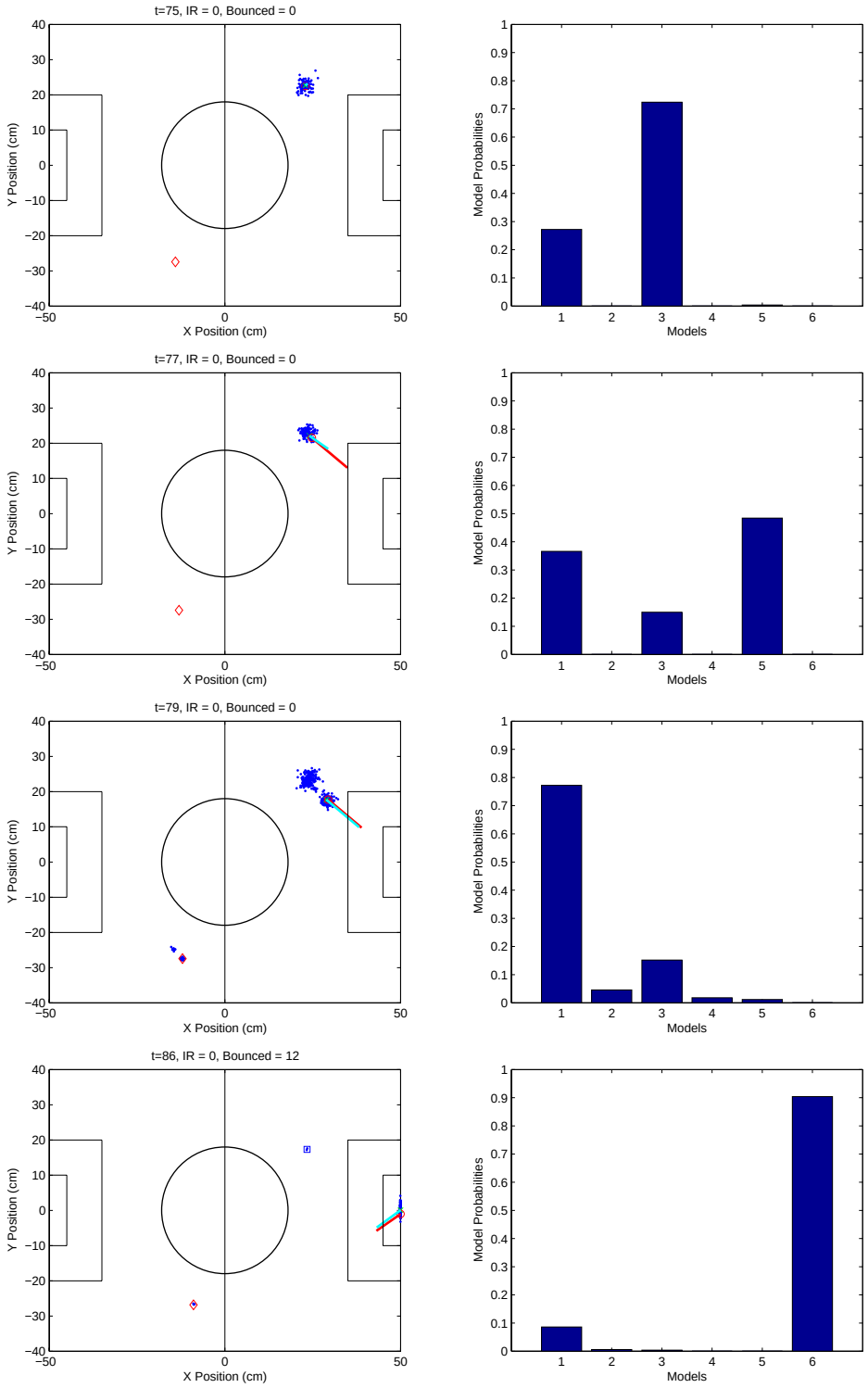

Figure 5.8: The snapshots of the particle cloud and the model probabilities at $t = 75$, 77, 79, and 86.

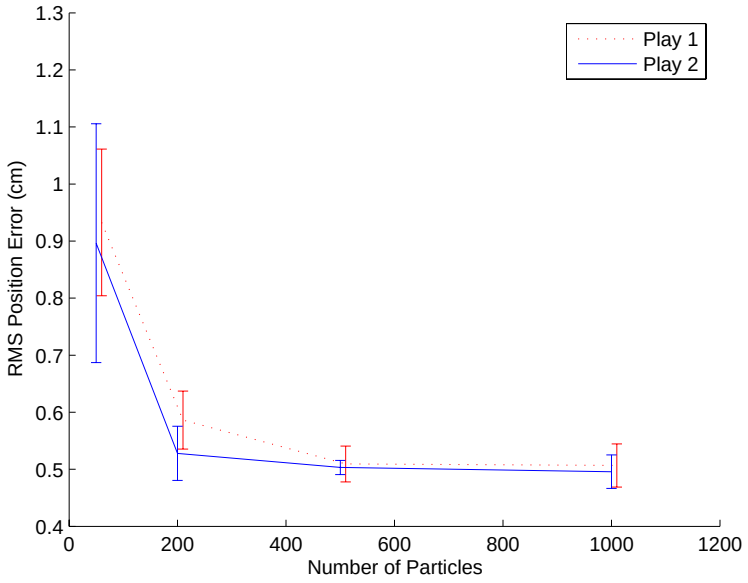

Figure 5.9: RMS position error versus the number of particles

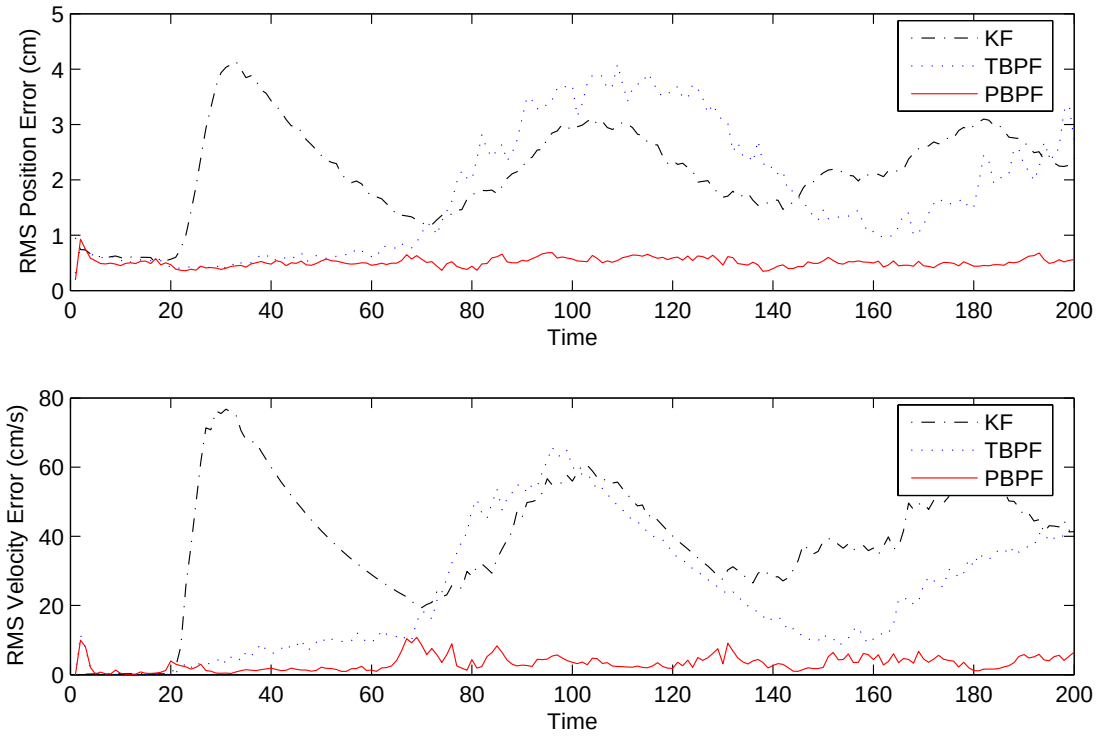

Figure 5.10: Play 1: RMS position error and velocity error versus time.

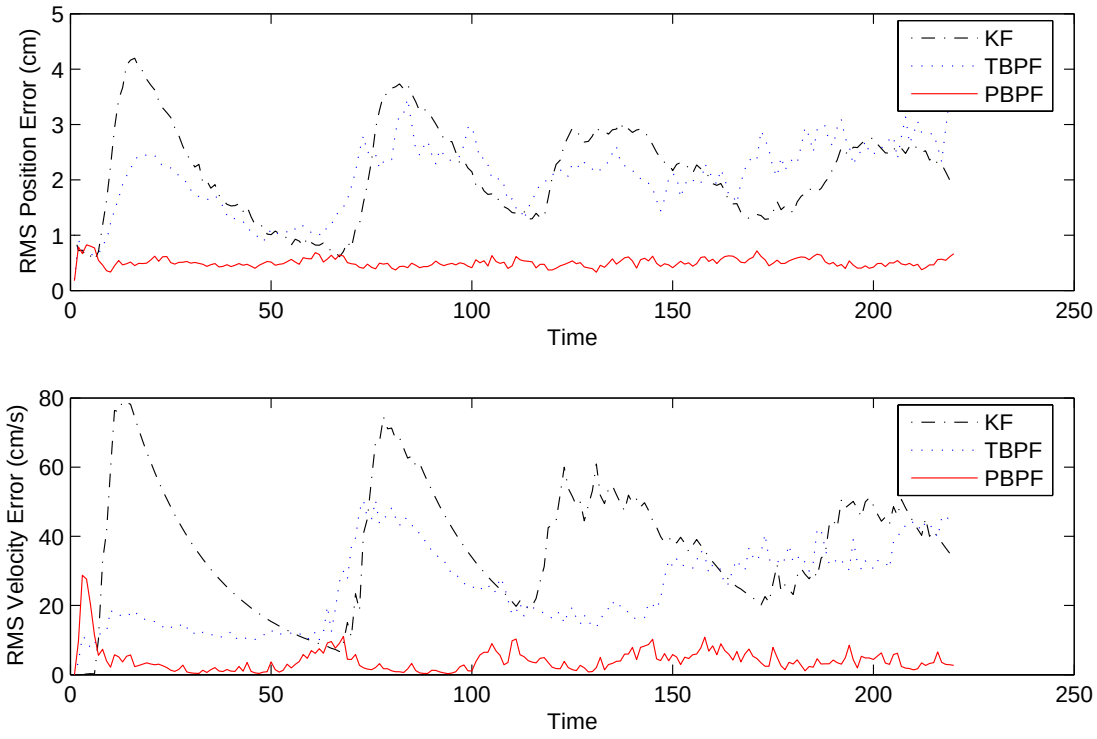

Figure 5.11: Play 2: RMS position error and velocity error versus time.

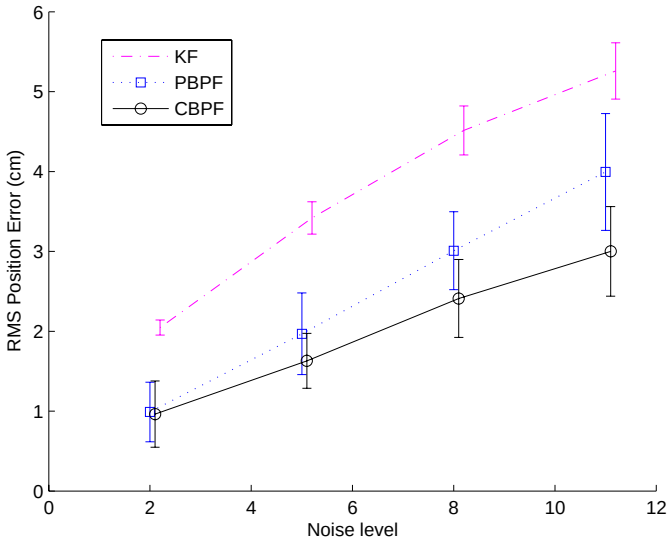

Figure 5.12: Play 2: RMS position error versus the amount of noise.

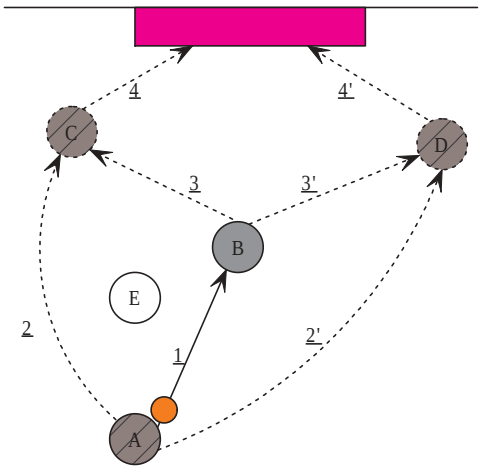

Figure 5.13: A demonstration of a naive team cooperation plan in offensive scenario. The digits along the lines show the sequence of the whole plan. The filled circle at position B represents the robot. The unfilled circle at position E represent an opponent player. The shaded circle represent the human teammember.

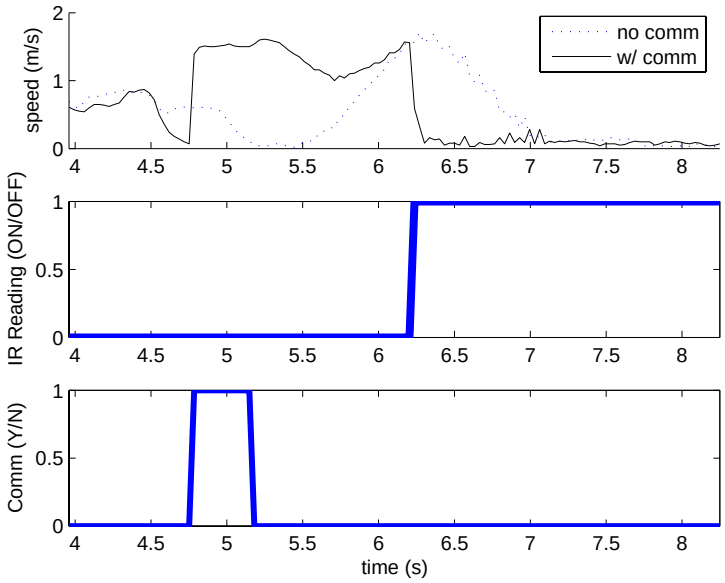

Figure 5.14: Ball speed estimation results from the multi-model tracker with and without including the communicated information.

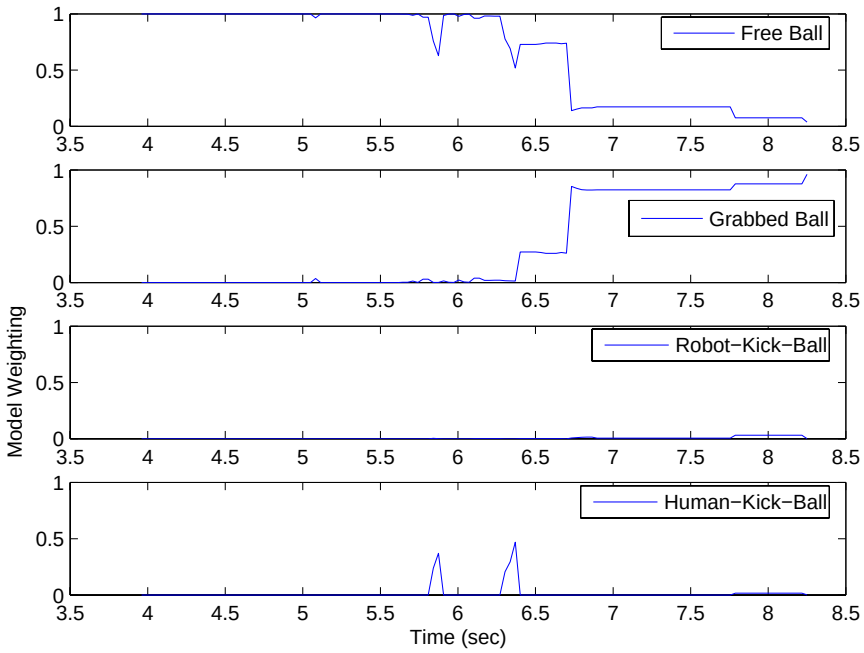

Figure 5.15: Model weightings when communication is disabled.

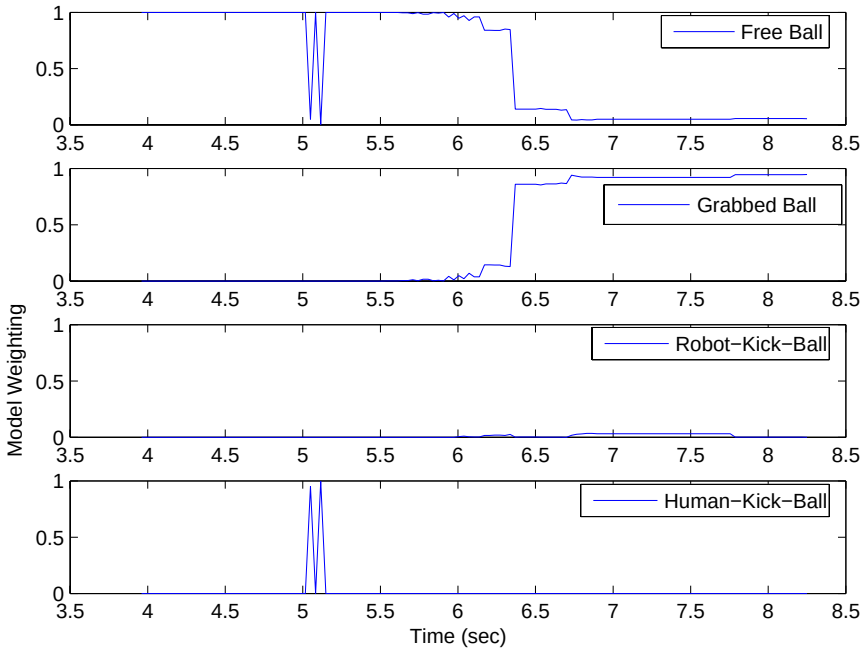

Figure 5.16: Model weightings when communication is enabled.

Chapter 6

Learning of Actuation Models

Many engineering applications are characterized by nonlinear or linear dynamic systems with a few possible modes (models) [14]. For example, an industrial plant may have multiple discrete modes of behavior, each of which has approximately linear dynamics. These problems are often referred to as jump Markov or hybrid-state estimation problems [11].

This chapter addresses estimating state and learning motion models in such a hybrid-state system. We are interested in tracking the ball in a robot soccer domain. This is a highly dynamic and multi-agent environment. All the robots in the field actuate on the ball, e.g., grab and kick the ball, making the motion model of the ball very complex [25].

Any model consists of one or multiple parameters. Usually the model parameters are set by a human expert, based upon the experience with the environment and the robot. In this chapter, we present a novel method of automating the procedure of acquiring this probabilistic motion model. We present a parametric system model. We apply particle filtering and extend the use of KLD-sampling to learning parameters in this model. The result is a fast algorithm which runs on-line and achieves accurate-enough learning of parameters.

This approach deals simultaneously with both unknown fixed model parameters and state variables. This not only relieves the work burden from the human expert, but can be very useful when the environment changes (e.g., moving from inside to outside). This approach can be applied to learn the motion model of the teammate or even the opponent, as a substrate for opponent modeling. Furthermore, this method provides a refined motion model based on the current one, resulting in more accurate tracking.

6.1 Parametric System Model

As described in Section 3.5, a discrete-time hybrid system is given by:

$$\mathbf{x}_t = f_{t-1}(\mathbf{x}_{t-1}, m_t, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}) \quad (6.1)$$
$$\mathbf{z}_t = h_t(\mathbf{x}_t, m_t, \mathbf{w}_t) \quad (6.2)$$

where f and h are the parameterized state transition and measurement functions; $\mathbf{x}_t, \mathbf{u}_t, \mathbf{z}_t$ are the state, input and measurement vectors at time t; $\mathbf{v}_{t-1}, \mathbf{n}_t$ are the process and measurement noise vectors. The covariances of $\mathbf{v}_{t-1}, \mathbf{w}_t$ are respectively Q_{t-1} and R_t. The model index parameter m can take any one of N_m values, where N_m is the number of models in the system.

A discrete-time hybrid system with unknown parameters is given by:

$$\mathbf{x}_t = f_{t-1}(\mathbf{x}_{t-1}, m_t, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}, \theta_{t-1}) \quad (6.3)$$
$$\mathbf{z}_t = h_t(\mathbf{x}_t, m_t, \mathbf{w}_t, \theta_t) \quad (6.4)$$

θ denotes the vector(s) consisting the unknown parameters, such as the variances of the noises and the coefficients of the functions f and h.

We are interested in learning the opponent actuation model in order to improve the tracking performance. In our Segway robot soccer setup, there are three related ball motion models in terms of actuation:

- *Passed-Ball*. The ball is passed from one player to another. Suppose we know the positions of the two players. We want to learn the initial speed μ of the ball and the kicking angle ϕ as shown in Figure 6.1. In this case, $\theta = (\mu, \phi)$.

- *Shot-Ball*. The ball is kicked by the player to the goal. In this case, we know that the ball is moving toward the goal. We want to learn the initial speed of the ball after the ball leaves the catcher of the player. That is, we want to know how hard the ball is being kicked. In this case, $\theta = \mu$. Therefore, this is a special case of learning *Passed-Ball*.

- *Grabbed-Ball*. The ball is grabbed by the catcher. In this case, we assume the ball moves with the player. If we know the position of the player, we can infer the position of the ball. We do not learn any parameters of this model.

6.2 Learning of Actuation Model

The challenges of learning the parameter $\theta = (\mu, \phi)$ of the actuation model are as follows.

- The object's motion consists of multiple motion models. We do not explicitly know which motion model the object follows.
- The actuation model of the opponent may not be fixed. We need a learning algorithm adapting to the opponent's actuation model dynamically and quickly.
- The actuation on the ball happens sporadically. The effect of the actuation changes the motion of the ball immediately. We need to identify the occurrences of the opponent's actuation. We need to learn the actuation parameters from the trajectory of the actuated ball.
- The computation time required by the learning algorithm is limited. The learning algorithm needs to be able to run on-line and its running time of each learning epoch needs to be less than the incoming sensor data interval, e.g., 33 milliseconds.

Considering the above challenges, we propose a sampling-based algorithm to learn the opponent's actuation model.

- We partition the parameter space into discrete bins and initialize the bins with uniform weights. The sum of the weighted bins represents the joint density estimation of the parameters.
- We extend the multi-model particle filter by augmenting each particle with another component: a bin id of the actuation model. We keep resampling the bin id after detecting the occurrences of actuation by evaluating the predicted observations vs the sensor measurement.
- We extend the use of KLD-sampling framework to learning and successfully decrease the computation time of learning and the state estimation process.

6.2.1 Partition of Parameter Space

Without losing of generality, we first consider the case that only one of the models contains unknown parameters. We partition the parameter space into b bins. Let $\rho = (\rho_1, \rho_2, \ldots, \rho_b)$

denote the true probability of each bin. We draw n parameter samples from the discrete distribution with b bins, which results in a multinomial distribution, i.e., $\Theta \sim \text{Multinomial}_b(n, \rho)$. $\Theta = (\Theta_1, \Theta_2, \ldots, \Theta_b)$ denotes the number of samples drawn from each bin. The sum of the number of samples from each bin equals to n.

$$\sum_{i=1}^{n} \Theta_i = n \tag{6.5}$$

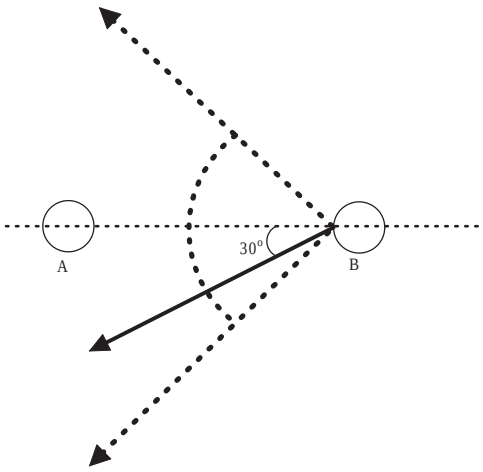

Figure 6.1: The illustration of the unknown parameter: passing angle.

For example, suppose we have only one unknown parameter $\theta = \phi$, which is the kicking angle of the opponent's actuation model when passing the ball. In Figure 6.1, robot B is passing the ball to A with a kicking angle $\phi = 30°, -45° \leq \phi \leq 45°$. If we use $b = 7$, the parameter space can be uniformly partitioned into 7 discrete bins listed in Table 6.1. The initial probability of each bin $\rho_0 = (\frac{1}{7}, \frac{1}{7}, \ldots, \frac{1}{7})$. The goal of our learning process is to approximate the true probability ρ of bins.

bin	1	2	3	4	5	6	7
ϕ (°)	-45	-30	-15	0	15	30	45
ρ_0	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{7}$

Table 6.1: Partition of the parameter space.

6.2.2 Likelihood Function

From the parametric system model in Section 6.1, we know that the likelihood of the measurement series $(\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T)$ from time 1 to T specified by the parameter θ is obtained by

$$\begin{aligned} L(\theta) &= p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T|\theta) && (6.6) \\ &= \prod_{t=1}^{T} p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \theta) && (6.7) \end{aligned}$$

The term $p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \theta)$ can be obtained by

$$p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \theta) = \int p(\mathbf{z}_t|\mathbf{x}_t, \theta) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \theta) d\mathbf{x}_t \tag{6.8}$$

where $p(\mathbf{z}_t|\mathbf{x}_t, \theta)$ is the measurement likelihood and $p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \theta)$ is the predictive distribution.

We use a set of particles presented in Section 4.3 to approximate the posterior distribution of $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$.

$$p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) \approx \sum_{i=1}^{N_s} w_{t-1}^{(i)} \delta(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{(i)}) \tag{6.9}$$

where N_s is the number of particles and $w_{t-1}^{(i)}$ is the weight associated with the ith particle at time $t-1$.

The predictive distribution is approximated by a one-step prediction

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \theta) &\approx \sum_{i=1}^{N_s} w_{t-1}^{(i)} \delta(\mathbf{x}_t - f_{t-1}(\mathbf{x}_{t-1}^{(i)}, m_t, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}, \theta)) && (6.10) \\ &= \sum_{i=1}^{N_s} w_{t-1}^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}) && (6.11) \end{aligned}$$

where f_{t-1} is the state transition function at time $t-1$.

Note that we compute the importance weight $w_t^{(i)}$ with respect to the observation $\mathbf{z}_t$ by

$$w_t^{(i)} = w_{t-1}^{(i)} \times p(\mathbf{z}_t|\mathbf{x}_t^{(i)}, \theta). \tag{6.12}$$

Therefore the term $p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \theta)$ can be approximated by

$$p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \theta) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} w_t^{(i)}. \tag{6.13}$$

The log-likelihood is given by

$$\log L(\theta) \propto \log \prod_{t=1}^{T} \sum_{i=1}^{N_s} w_t^{(i)} \tag{6.14}$$

$$= \sum_{t=1}^{T} \log \sum_{i=1}^{N_s} w_t^{(i)} \tag{6.15}$$

In the resampling step, we replicate particles in proportion to their weights. After resampling, all particles have an identical weight. So that, in a recursive way to compute $\log L(\theta)$, we count the number of particles with specific bin id. We update the probability of each bin (ρ) following the procedure as shown in Table 6.2.

```
1  for j ← 1 to b
2        do Θ_j ← Count the number of particles drawn from bin j
3           if Θ_j ≤ 1
4              then ρ_t^j ← ρ_{t-1}^j
5              else
6                   ρ_t^j ← ρ_{t-1}^j + log Θ_j
7  Normalize: ρ_t^j ← exp(ρ_t^j) / Σ_{j=1}^{b} exp(ρ_t^j)
```

Table 6.2: Update the probability of each bin.

6.2.3 Augment the State Vector of Particles

We extend the multi-model particle filter by augmenting a component $\theta_t^{(i)}$ to indicate the bin id of the unknown parameters of the actuation model: $p_t^{(i)} = \langle \mathbf{x}_t^{(i)}, m_t^{(i)}, \theta_t^{(i)} \rangle$. $\theta_t^{(i)}$ is dependent on $m_t^{(i)}$, i.e., the bin id indicated by $\theta_t^{(i)}$ corresponds to the motion model m. If $\theta_t^{(i)} = 0$, the particle i at time t does not related to any actuated motion. For example, if

$m_t^{(i)} = 1, \theta_t^{(i)} = 5$, the particle i at time t has been actuated by the opponent using bin id 5 associated with motion model 1.

The change of the bin id of each particle can be summarized as follows:

- **Initialization**. The initial value of the bin id of each particle is 0. No actuation is expected at the beginning of the learning process.
- **Birth**. When $m_t^{(i)} =$ *Passed-Ball* or *Shot-Ball*, a bin id is sampled from the current parameter distribution.
- **Death**. When $m_t^{(i)} =$ *Grabbed-Ball*, a bin id is reset to zeros. That is, after the ball is being caught by any player, the motion is changed. We stop the learning process.
- **No Change**. When $m_t^{(i)} =$ *Free-Ball*, *Bounced-Ball*, there is no change to the bin id.
- **Update**. At the end of each particle evolution, we resample particles and update their bin ids as shown in Table 6.2.

After resampling, we compute the weighted sum of the particles with the same bin id and update their corresponding log-likelihood of the measurements using Equation 6.15.

6.2.4 Adapting the Number of Particles

The key idea of KLD-sampling approach is to derive an upper bound of the error introduced by the particle-filter-based belief representation [13]. The number of particles is determined at each iteration of particle filtering to ensure the error between the true posterior and particle-based representation, namely the KL-distance is less than ϵ with probability $1 - \delta$.

As in the KLD-sampling algorithm applied to tracking, we adapt the number of modeling particles for learning the unknown parameters. Our approach chooses a small number of particles if no learning is performed. When learning opportunity is detected, our approach generates particles until their number is large enough to guarantee the KL-distance is less than the predefined bound.

For a given δ, the sample size needed to approximate a discrete distribution with an upper

bound ϵ on the KL-distance is [13, 20]:

$$n = \frac{k-1}{2\epsilon}\left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1}}z_{1-\delta}\right)^3 \tag{6.16}$$

where $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard normal distribution.

An update step of our actuation model learning algorithm is summarized in Table 6.3. We incrementally determine the number of supported bins k by checking for each sampled particle whether it falls into an empty bin or not. We assume that a bin has support if it contains at least one particle. We use Equation 6.16 to update the desired number of particles n_χ needed to learn the parameter θ. When the number of generated particle n is greater than the desired particle number n_χ, the stopping condition of generating new particles is satisfied.

$p_t = \{x_t^{(i)}, m_t^{(i)}, w_t^{(i)}, \theta_t^{(i)}\}_{i=1}^n$ represents the particles at time t. In each step, We start with a uniform distribution of ρ_0, which represents the initial joint distribution of parameters. p_t is set to an empty set. k denotes the number of supported parameter bins. n denotes the number of particles generated. We update the number of supported bins k for the predictive distribution after we generate a sample with certain motion model. The determination of k is done by checking for each generated sample whether its associated bin id falls into an empty bin or not.

6.3 Results

In this section, we evaluate the actuation model learning capability of our approach in simulated experiments.

In each trial, the ball's initial position is at (-40, -30) and its initial speed is set to zero. Two robot's initial positions are (-35, -20) and (20, 10) respectively. The two robots pass the ball to each other repeatedly until the end of the trial. We assume by default the passing robot use fixed actuation models in terms of kicking speed and kicking angle. We also test the learning result when the passing robot change their actuation models during the passing game.

An observing robot is located at a fixed position (45, -30), tracking and learning the actuation model used by the two robots.

Unless otherwise mentioned, the nominal filter parameters used in the simulation are listed in Table 6.4. The kicking speed and kicking angle vary in different trials. The process noise v and the sensor noise w are set to be identical in all models.

$v \sim \mathcal{N}(0, Q)$, the process noise covariance matrix

$$Q = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix} \tag{6.17}$$

$w \sim \mathcal{N}(0, R)$, the sensor noise covariance matrix

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.18}$$

Figures 6.2 and 6.3 display the output of tracker and learning result at $t = 3$ and 400 in one MC run, respectively. The upper-left figures shows the distribution of particles $(x_t^{(i)}, y_t^{(i)})$ at time t, with their weighted mean value $(\hat{x}_t, \hat{y}_t)$ represented by a green x-mark. The true position of the ball at time t (x_t, y_t) is represented by a red circle. The true velocity of the ball at time t $(\dot{x}_t, \dot{y}_t)$ is represented by the red line. The estimated ball velocity $(\hat{\dot{x}}_t, \hat{\dot{y}}_t)$ is represented by the cyan line. The upper-right figures present the probabilities of models. The two bottom figures show the learned distribution of the parameters: kicking speed and kicking angle.

Table 6.5 shows the preset parameters to be learned in the following experiments. From the final learning results of Figures 6.3, 6.4 and 6.5, we can see that the mean of the learned distribution is pretty close to the preset parameter values.

The next trial examines the learning performance in terms of the error between the learned parameter values and the preset parameter values. Figure 6.6 shows the parameter learning result over 30 MC runs. The learned parameters adapt to the unknown distribution in less than 25 seconds. That is, the approximate actuation model can be constructed by observing 2 or 3 passes. This quick adaptation ability is very useful in terms of learning opponent actuation model and responding with adversarial actions.

In the third trial, the passing robots change their actuation models during the passing game every 20 seconds. We test the tracking performance using three trackers overn 30 MC runs:

KF, PBPF without the actuation model and PBPF with the learned actuation model. The result is summarized in Table 6.6. By including the learned model, the tracker achieves significant tracking improvement in terms of position RMS and velocity RMS.

6.4 Summary

In this chapter, we present a parametric system model which describe the :

- We describe the parametric system model and the parameters we need to learn in the actuation model.
- We partition the parameter space into discrete bins and initialize the bins with uniform weights. The sum of the weighted bins represents the joint density estimation of the parameters.
- We extend the multi-model particle filter by augmenting each particle with another component: a bin id of the actuation model. We keep resampling the bin id after detecting the occurrences of actuation by evaluating the predicted observations vs the sensor measurement.
- We extend the use of KLD-sampling framework to learning and successfully decrease the computation time of learning and the state estimation process. The approach results in a fast and accurate on-line parameter-learning algorithm.
- We show the effectiveness of learning using simulated experiments. The tracker that uses the learned actuation model achieves improved tracking performance.

$p_t = \emptyset, n = 0, n_\chi = 0, n_{\chi_{min}} = 100, k = 0$
while $n < n_\chi$ or $n < n_{\chi_{min}}$
    do
        draw $m_t^{(i)} \sim p(m_t | m_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)})$
        if $m_t^{(i)}$ = *Passed-Ball* or *Shot-Ball*
            then draw $\theta_t^{(i)} \sim \rho_t$
        elseif $m_t^{(i)}$ = *Bounced-Ball* or *Free-Ball*
            then $\theta_t^{(i)} = \theta_{t-1}^{(i)}$
            else $\theta_t^{(i)} = 0$
        draw $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | m_t^{(i)}, \mathbf{x}_{t-1}^{(i)}, \theta_t^{(i)})$
        $w_t^{(i)} \leftarrow p(\mathbf{z}_t | \mathbf{x}_t^{(i)}, \theta_t^{(i)})$
        $p_t = p_t \cup (x_t^{(i)}, m_t^{(i)}, w_t^{(i)}, \theta_t^{(i)})$
        if $\theta_t^{(i)} \neq 0$ and $\theta_t^{(i)}$ falls into empty bin b
            then k = k+1
                b = non-empty
                if $n \geq n_{\chi_{min}}$
                    then $n_\chi = \frac{k-1}{2\epsilon}(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1}} z_{1-\delta})^3$
        $n = n + 1$
Calculate total weight: $w \leftarrow \sum[\{w_t^{(i)}\}_{i=1}^{N_s}]$
for $i \leftarrow 1$ **to** N_s
    do Normalize: $w_t^{(i)} \leftarrow w_t^{(i)}/w$
π = Resample$(\{w_t^i\}_{i=1}^{N_s})$
$\theta_t^{(\cdot)} = \theta_t^\pi$, $x_t^{(\cdot)} = x_t^\pi$, $m_t^{(\cdot)} = m_t^\pi$
$w_t^i = 1/N_s$
update ρ_t as in Table 6.2
return p_t

Table 6.3: Actuation model learning algorithm.

Parameter	Value	Description
Δt	0.033	The time interval between two consecutive vision frames
d	0.96	Speed decay
T	600	Total time steps in one Monte Carlo run
M	30	The number of MC runs
N_s	500	The number of particles
N_{thr}	$N_s/3$	Particle resample threshold
w_0	[0.9 0.1 0 0]	Initial model probabilities

Table 6.4: The nominal filter parameters used in the simulation.

Figure	μ (cm/s)	$\theta(^\circ)$
1.3	92	6
1.4	105	-10
1.5	85	4

Table 6.5: The preset parameters for Figures 1.3-1.5

Algorithm	RMS Position Mean (cm)	RMS Position Std (cm)	RMS Velocity Mean (cm/s)	RMS Velocity Std (cm/s)
KF	2.85	3.56	25.33	37.55
PBPF	2.18	3.10	20.33	26.05
PBPF w/ learning	1.50	2.10	9.02	10.87

Table 6.6: RMS error comparison for three trackers

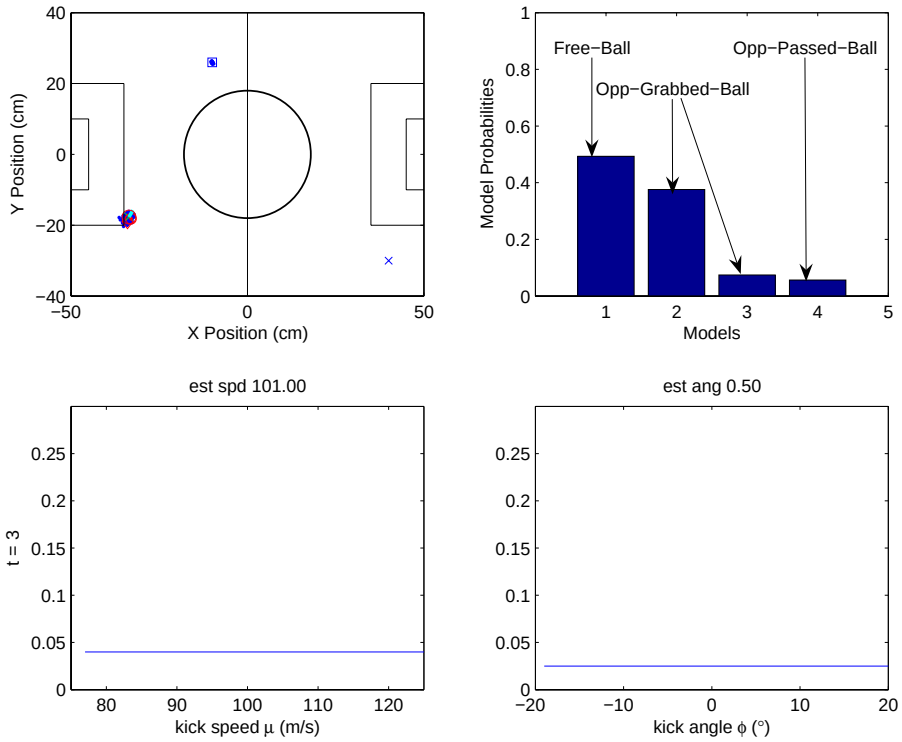

Figure 6.2: Learning result: t = 3

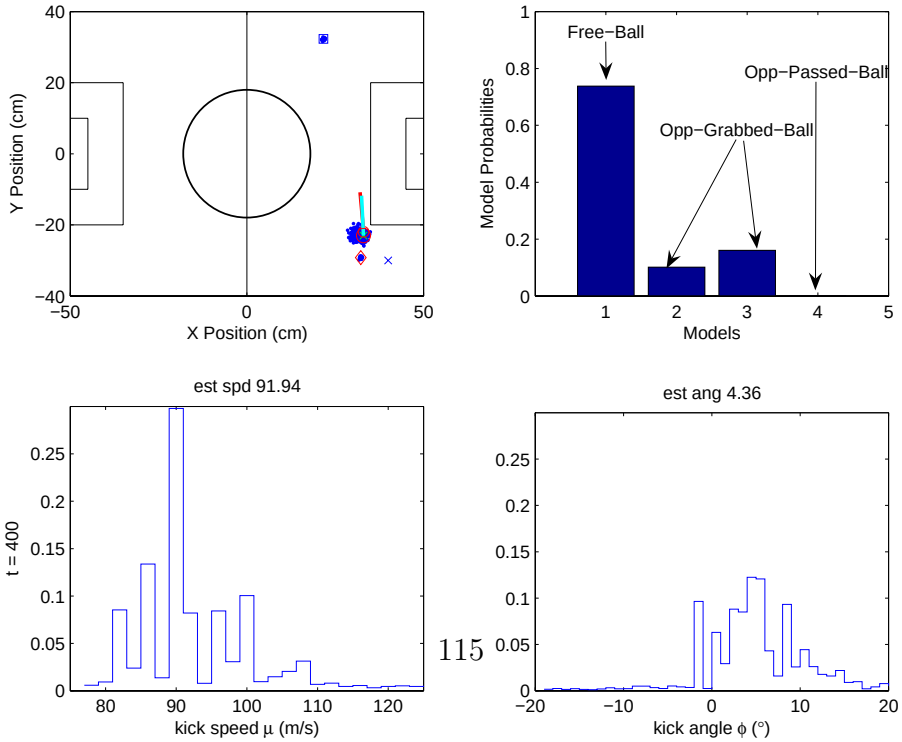

Figure 6.3: Learning result: $\mu = 92, \phi = 6, t = 400$

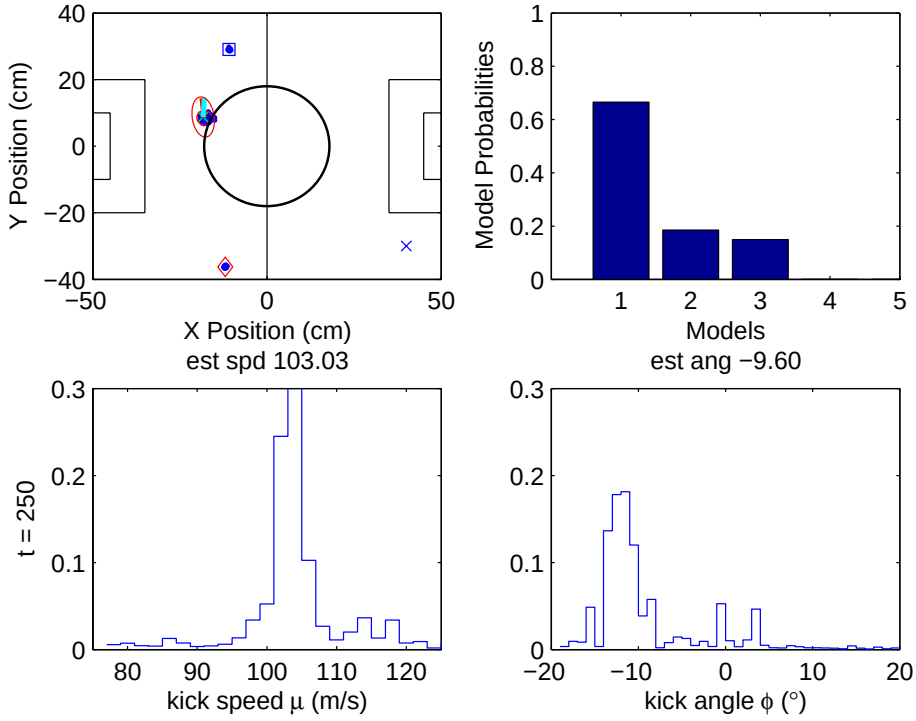

Figure 6.4: Learning result: $\mu = 105, \phi = -10, t = 250$.

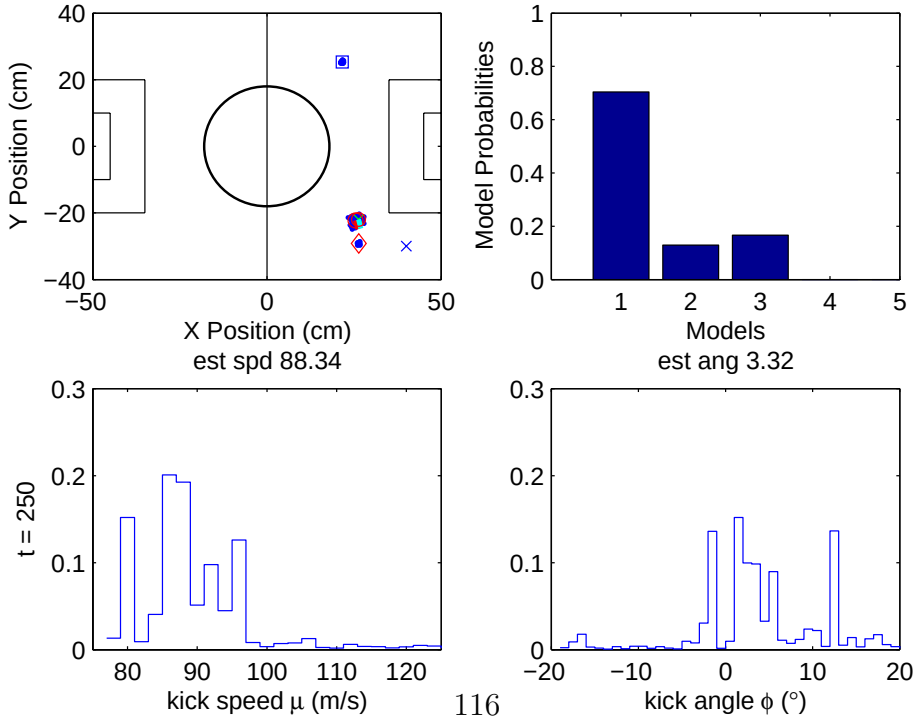

Figure 6.5: Learning result: $\mu = 85, \phi = 4, t = 250$.

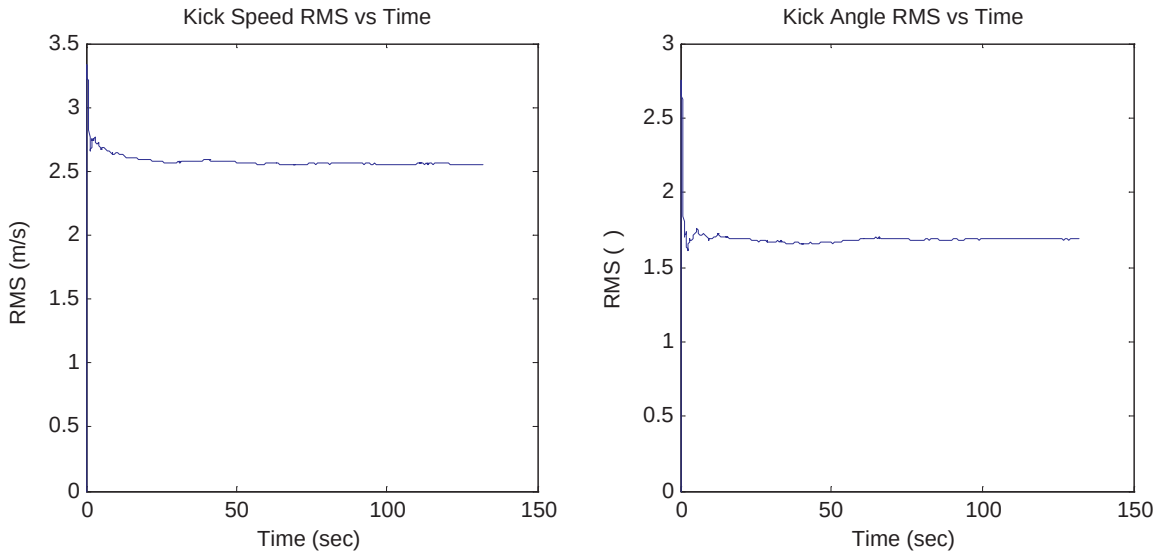

Figure 6.6: Actuation model learning result

Chapter 7

Categorization of Previous Work

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

7.1 Multi-Model Motion Tracking

Tracking moving objects using a Kalman filter is the optional solution if the system follows a linear model and the noise is assumed Gaussian [21]. 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 [3]. For nonlinear systems or systems with non-Gaussian noise, further approximations such as 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 in [6, 12, 27]. However, these approaches assume 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 $\mathrm{P}(m_t = i|m_{t-1} = j) = \pi_{i,j}$ where $i, j = 1, \cdots, N_m$. The uncertainties in the object tracking problem in this thesis do not have such a property due to the interactions between the robot and the tracked object. We contribute a tactic-based motion modeling (TBMM) method to solve the problem. We introduce the play-based motion modeling (PBMM) method when team coordination knowledge is available.

7.2 Cooperatively Tracking

Many efforts investigate the problem of state modeling and mapping with robot teams, e.g., [10, 37]. 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 [38]. 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 estimate of self and teammate information, which has been proven to be an effective solution to deal with this uncertainty [33]. Another interesting approach considers an environment where the robots are unable to cover the entire field with their own sensors and may be out numbered by the targets. The team of robots perform cooperatively tracking by communicating positive and negative information (where robots do not see targets) [31]. The object tracking error is reduced significantly in most instances. Our approach does not include communicated 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.

7.3 Motion Prediction

It is an important ability of an agent to perceive moving objects in its vicinity environment. In the robotics community, autonomous robots need to be able to predict the future motion of the mobile targets as well. Motion prediction in such scenarios is a difficult challenge because most objects are moving and there are physical interactions between different objects (e.g., a soccer robot holding and kicking a ball, or a robot pushing an obstacle).

Motion prediction is a research area with a wide range of applications, including video surveillance and robot navigation. Traditional research focuses on designing an a priori motion model to describe how the state of a particular object (e.g., position and velocity) changes over time when it is subject to a given control (e.g., acceleration). In order to predict the future motion of a particular object, its current state and control are estimated first (e.g., using the Kalman filter [21]). Next the estimated state and control are fed into the motion model to get future state estimates. This approach computes good motion predictions when the motion model used is faithful to describe the object motion and the state and control estimations are accurate. However, in real robot applications, e.g., robot

soccer, such conditions are hardly met [16]. We use the ball-tracking in robot soccer as an example. First, there are multiple robots that actuate the ball, which makes the motion of the ball highly noncontinuous. Any single motion model is impossible to faithfully describe the broken trajectory of the ball. Second, traditional motion prediction techniques are not able to get accurate control estimates because the actuation over the ball is coming from multiple robots. Therefore we are unable to feed the external control into the motion model using the traditional approach. Third, traditional prediction techniques perform well when the object is within the focus of visual scope at each time step. However, in a real environment, objects may have broken or even occluded trajectories, frequently leading to the object becoming lost to the tracker.

Recent motion prediction techniques are based on the idea that, for a given area, moving objects tend to follow typical motion patterns that depend on the objects' nature and the structure of the environment [39]. A typical motion prediction technique first uses a learning stage to observe and learn the typical motion patterns of the moving objects. It then uses the learned motion patterns to predict the future motion of a particular object. This approach leads to a better motion model since the learning stage clusters the similar observed trajectories and computes multiple representative trajectories [5]. It permits to take into account not only the current state of the object but also its past states. The weakness lies in the inability to use the known behavior models of the individual robot and the team of robots. By including such behavior knowledge and learned actuation model in this thesis, we can achieve more accurate motion prediction results.

In this thesis, we contribute a novel team-driven approach to incorporate models of robot-mobile-object-interaction, team plan and communication into the motion prediction algorithm. Our motion prediction method allows to have multiple predictions of the object's location rather than entirely depending on the most recent observation. Making use of multiple sources of information, we feed the team knowledge as an external control and introduce a probabilistic motion model. We also learn the model parameters in such a switching state-space model. We evaluate our resulting informed-tracking approach empirically in simulation and using a setup Segway soccer task. The input of the multiple single and multi-robot behavioral sources allows a robot to much more effectively visually track mobile targets with dynamic trajectories.

7.4 Tracking Using Prior Knowledge or Dynamic Information

There are several approaches incorporating some kind of prior knowledge related to the general problem of tracking under no actuation. For example, hard constraints on object position, speed or acceleration have been considered in tracking problems to improve tracking performance [41]. This kind of information is simple and easy to represent as a truncated density. The only thing to do is to sample from a truncated density using rejection sampling technique. Another situation is where a number of objects are moving in formation, and there is a strong dependency between the individual sensor measurements, which provide valuable information on object behavior. Actually this problem can be modeled as independent individual object motions superimposed on a common group effect. A model of this type was introduced in [28,34], in which the motion of the group and disposition of the measurement sources relative to the group are modeled as two separate components. In the 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 [1]. The algorithm is referred to as the variable structure multiple-model particle filter, since it adaptively selects a subset of modes that are active at a particular time. This approach outperformed 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. However, the tracker does not have actions on the targets in the group. So all the above approaches deal with the problem of tracking under no actuation. One of the effort reporting a tracking approach concerned with our problem of actuation over tracked objects is presented in [25]. Joint state estimation has 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. The play-based motion modeling can be flexibly integrated into our existing skills-tactics-plays architecture.

7.5 Summary

In this chapter, we discuss the related work along the four main aspects of our approach:

- Probabilistic state estimation.

- Cooperatively tracking by multiple robots.
- Motion prediction.
- Improvement of tracking performance through integration of prior knowledge or dynamic information.

Further related work when appropriate is occasionally discussed in the previous technical chapter of the thesis.

Chapter 8

Conclusions and Future Work

Our goal in this thesis is to explore methods to improve an agent's object tracking ability. We are interested in methods that take use of tactics, plays and communication to improve object tracking efficiency in the presence of multiple agents acting on a mobile object.

Our approach builds upon three main facts:

- An individual robot knows its own actions. If it is the case that a robot is manipulating the ball, the robot can predict the ball's motion using its own actions.

- Robots in a team collaborate according to pre-defined coordination plans or dynamic communication. If it is the case that a teammate is manipulating the ball, the robot that is not controlling the ball can predict the ball's motion by reasoning about the possible teammate actions. If the communication between teammates is allowed, the robot that is not controlling the ball can explicitly know what the action is from their teammates.

- Robots can learn the unknown actuator's action effects on the ball. At a low-level, robots can learn to know the actuator's kicking power and direction. At a high-level, robots can learn to know the actuator's kicking strategies, e.g., in a certain situation whether to shoot or to pass. If it is the case that an opponent robot is manipulating the ball, the robot in our team can predict the ball's motion as well using the learned action models during the game.

8.1 Conclusions

The main contributions of this thesis are:

- We incorporate a single robot and a team actuation models into a Dynamic Bayesian Network (DBN)-based temporal representation for tracking.

- We introduce several multi-model tracking algorithms based on: (i) the robot's own actions; (ii) predefined team plays; (iii) communicated team actions. Various information sources are channelled into the motion model of the system, namely tactic, play, communication, learned opponent actuation parameters and sensor observation.

- We introduce and implement the team-driven motion tracking framework. Team-driven motion tracking is a tracking paradigm defined as a set of principles for the inclusion of a hierarchical, prior knowledge and construction of a motion model. We illustrate a possible set of behavior levels within the Segway soccer domain that correspond to the abstract motion modeling decomposition.

- We present an empirical comparison between several tracking algorithms. We examine the performance of each algorithm according to a variety of metrics. We evaluate the new tracking algorithms in robot platforms, a human-robot team, and in simulated tests. We show the efficiency of the TBPF, PBPF and CBPF over single model tracking.

- We present a parameter learning algorithm to learn opponent actuation models. We describe the parametric system model and the parameters we need to learn in the opponent's actuation model. We partition the parameter space into discrete bins and initialize the bins with uniform weights. The sum of the weighted bins represents the joint density estimation of the parameters. We extend the multi-model particle filter by augmenting each particle with another component: a bin id of the actuation model. We keep resampling the bin id after detecting the occurrences of actuation by evaluating the predicted observations vs the sensor measurement. We extend the use of KLD-sampling framework to learniing and successfully decrease the computation time of learning and the state estimation process. We show the effectiveness of learning using simulated experiments. The tracker that uses the learned actuation model achieves improved tracking performance.

8.2 Directions for Future Work

This thesis opens up new interesting directions for further research:

- **Uncertainty of Human Teammate**. 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 might take such uncertainty into account. A better human team member modeling (for example, include intercepting the moving ball, mark a player, covering the goal) will also help. Another interesting work is 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.

- **Limited Trackers**. When there are limited tracking resources, i.e., there are more objects to be tracked than tracking resources. E.g., in Segway soccer, the robot needs to track several objects (including the ball, team members and opponents) using one pan-tilt camera. When multiple objects need to be tracked using limited trackers, the question of assigning different trackers to different objects is then crucial.

  - Instead of continuously scanning over the full field of view to locate the target, the tracker restricts its scanning area, on the basis of a real-time analysis of the current robot strategy, team plan and object trajectory in its memory, to a very narrow window precisely the size of the useful targets.
  - Instead of always focusing on one target, the tracker shifts its focus elegantly between several interesting targets.
  - Instead of using a manually tuned or fixed time on each target, the tracker learns to know how much time to spend on each target in order to ensure consistent tracking of each target.

- **High Dimensional Parameter Space**. In our current implementation of parameter space partition when learning opponent actuation model, we use a discrete distribution with a fixed bin size. If the parameter space contains high dimensional parameters, the fixed-size-partition method could lead to poor discretization. Because more samples may be used in "unimportant" parts of the parameter space. An interesting direction is to change the discretization over time using a KD-tree structure.

- **Application Opportunities in Other Domains**. The thesis contributions provide practical algorithms for tracking to be applied in domains with ever increasing robot-object interactions. First we need to identify the known behavior model of the

specific application. We apply the team-driven motion tracking framework and follow the introduced principles to decompose the motion modeling problem and take use of the behavior knowledge. Even in some application domains without robot-object interaction, e.g., Unmanned Aerial Vehicle (UAV), by including the cooperative tactics and communicated information, we can still reduce the tracking error by using the communication-based motion model.

8.3 Concluding Remarks

This thesis contributes a number of multi-model tracking algorithms in the presence of multiple agents acting on a mobile object. We introduce and implement the team-driven motion tracking framework. We illustrate a possible set of behavior levels within the Segway soccer domain that correspond to the abstract motion modeling decomposition. We demonstrate effective object tracking in single robot tasks with own actuation information, in domains with team actuation models, and for complex tasks with existence of opponents. We also demonstrate how a single, team or learned opponent actuation model could be integrated in a multi-model tracking framework. The thesis contributions provide practical algorithms for tracking to be applied in domains with ever increasing robot-object interactions.

Appendix A

Notations

Notation	Description
x_t	Object position in x coordinate at time t
$\hat{x}_t$	Predicted object position in x coordinate at time t
y_t	Object position in y coordinate at time t
$\dot{x}_t$	Object velocity in x coordinate at time t
$\dot{y}_t$	Object velocity in y coordinate at time t
d	Speed decay
$d_{c,l}(t_1, t_2)$	The normalized distance between the cth and lth measurement of z_{t_1} and z_{t_2}
f	State transition function in general
h	Measurement function in general
m_t	Model at time t
m'_t	Model of teammate at time t
m^i_t	Model of i^{th} particle at time t
$p^{(i)}_t$	The i^{th} particle at time t
q	Proposal function
s_t	Infrared sensor reading at time t
$\mathbf{u}_t$	Control input at time t
v	Additional information used by tactic-based motion modeling
$\mathbf{v_t}$	Process noise vector at time k
$\mathbf{w_t}$	Sensor noise vector at time k
$\mathbf{w_{c,t}}$	Sensor noise of the cth measurement at time t
$\mathbf{x}_t$	Estimated state at time t
$\mathbf{z}_t$	Sensor reading at time t

Notation	Description
$\mathbf{z}_{1:t}$	Sensor readings from time 1 to time t
$\mathcal{C}_t$	The communication message at time t
$E(X)$	Expectation of random variable X
F	State transition matrix
F_t^1	State transition matrix of model # 1 at time t
H	Measurement transition matrix
I_t	Observed information set at time t
J_t	The number of region of interest at time t
K_t	Kalman gain at time t
K_t^T	The transpose of Kalman gain matrix at time t
K_t	The number of target being tracked in the system at time t
M	The number of independent MC runs in simulated test.
N_{C_t}	The number of clutter points at time t
N_s	The number of particles
N_m	The number of motion models
$P_{t\|t-1}$	Covariance matrix at time t with accumulated sensor readings from time $t-1$
$P_{t\|t}$	Covariance matrix at time t with accumulated sensor readings from time t
$\mathcal{P}_t$	The play used at time t
Q	Process noise covariance
R	Sensor noise covariance
S	The total set of motion models: $S \triangleq \{1, 2, \cdots, N_m\}$.
$\mathcal{S}_t$	The set of region of interest at time t
S_t	Covariance matrix of the innovation term at time t
T	Total time steps in one Monte Carlo run
$\mathcal{T}_t$	The tactic used by the robot at time t
$\mathcal{T}_t'$	The tactic used by the teammate at time t
α_t	The association vector that indicate the measurement-to-target assignment
γ_t	The track-to-region association vector
ω	Weight
$\omega_t^{(i)}$	Weight of the i^{th} particle at time t
τ	The length of the sliding window
Δt	The time interval between two consecutive vision frames. $\Delta t = 1/30$ *sec* in all tests.
Π	Transitional matrix
π_{ij}	Transition probability
$\mathcal{N}(\mathbf{x}; \xi, P)$	A Gaussian density with argument $\mathbf{x}$, mean ξ and covariance P
θ_0	Initial model parameters
θ_t	Unknown model parameters at time t

Appendix B

Skill and Tactic Descriptions

Skill	Description
`aim`	look for the target and turn body to look at it
`go_near_ball`	move to a position near the ball and look at the ball
`grab_ball`	put down the catcher(finger)
`kick`	put up the catcher and use the kicker to kick the ball out
`search`	turn body and camera to look for the ball

Tactic	Description
`chase_ball`	follow the ball
`grab_and_kick`	go to the ball and grab it, then aim at the target and kick
`pass`	kick the ball toward a teammate.
`position_for_pass`	move to a position on the field to anticipate a pass.
`receive_pass`	move to a position to receive a pass.
`shoot`	kick the ball to go into the goal.

Bibliography

[1] M. Arulampalam, N. Gordon, M. Orton, and B. Ristic. A variable structure multiple model particle filter for gmti tracking. *Proc. 5th Int. Conf. Information Fusion*, July 2002.

[2] 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*, 50(2):174–188, February 2002.

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

[4] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*. Academic Press Inc, 1988.

[5] M. Bennewitz, W. Burgard, and S. Thrun. Learning motion patterns of persons for mobile service robots, 2002.

[6] Y. Boers and H. Driessen. Hybrid state estimation: a target tracking application. *Automatica*, 38:2153–2158, 2002.

[7] Brett Browning, James Bruce, Michael Bowling, and Manuela Veloso. Stp: Skills, tactics and plays for multi-robot control in adversarial environments. *IEEE Journal of Control and Systems Engineering*, 219:33–52, 2005.

[8] Brett Browning, Jeremy Searock, Paul E. Rybski, and Manuela Veloso. Turning segways into soccer robots. *Industrial Robot*, 32(2):149–156, 2005.

[9] Brett Browning, Ling Xu, and Manuela M. Veloso. Skill acquisition and use for a dynamically-balancing soccer robot. In *AAAI*, pages 599–604, 2004.

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

[11] A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[12] D.S.Angelova, T.A.Semerdjiev, V.P.Jilkov, and E.A.Semerdjiev. Application of monte carlo method for tracking maneuvering target in clutter. *Mathematics and Computers in Simulation*, 1851:1–9, 2000.

[13] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *I. J. Robotic Res.*, 22(12):985–1004, 2003.

[14] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[15] N. Gordon, D. Salmond, and A F M Smith. Novel approach to non-linear and non-gaussian bayesian state estimation. In *IEEE Proceedings-F*, 1993.

[16] Yang Gu. Tactic-based motion modeling and multi-sensor tracking. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-05)*, pages 1274–1279, 2005.

[17] Yang Gu and Manuela Veloso. Multi-model tracking using team actuation models. In *Proceedings of the 2006 International Conference on Robotics and Automation*, 2006.

[18] et al. H. G. Nguyen. A Segway RMP-based robotic transport system. In *SPIE Proc. 5609: Mobile Robots XVII*, Philadelphia, PA, October 2004.

[19] Kam-Chuen Jim and C. Lee Giles. How communication can improve the performance of multi-agent systems. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 584–591, New York, NY, USA, 2001. ACM Press.

[20] N. Johnson, S. Kotz, and N. Balakrishnan. *Continuous univariate distributions*, volume 1. John Wiley and Sons, New York, 1994.

[21] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:35–45, March 1960.

[22] R. E. Kalman and R. Bucy. New results in linear filtering and prediction theory. In *Trans. ASME. J. Basic Engineering*, pages 83:95–108, March 1961.

[23] Saphne Koller and Uri Lerner. Sampling in factored dynamic systems. *Sequential Monte Carlo Methods in Practice*, 2001.

[24] Chris Kreucher, Keith Kastella, and Alfred O. Hero III. Multi-target sensor management using alpha-divergence measures. pages 209–222, 2003.

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

[26] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[27] S. McGinnity and G.W.Irwin. Multiple model bootstrap filter for maneuving target tracking. *IEEE Trans. Aerospace and Electronic Systems*, 36(3):1006–1012, 2000.

[28] M. Morelande and S.Challa. An algorithm for tracking group targets. *Proc. Workshopp on Multiple Hypothsis Tracking: A Tribute to S.Blackman*, May 2003.

[29] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, 2002.

[30] William Ng, Jack Li, Simon Godsill, and Jaco Vermaak. A hybrid approach for online joint detection and tracking for multiple targets. *IEEE Aerospace Conferences*, 2005.

[31] M. Powers, R. Ravichandran, and T. Balch. Improving multirobot multitarget tracking by communicating negative information. In *Third International Multi-Robot Systems Workshop*, New Orleans, LA (US), 2005.

[32] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter, Particle Filters for Tracking Applications*. Artech House Publishers, 2004.

[33] Maayan Roth, Douglas Vail, and Manuela Veloso. A world model for multi-robot teams with communication. In *IROS-2003*, 2003. (under submission).

[34] D. Salmond and N. Gordan. Group and extended object tracking. *Proc. SPIE*, 3809, 1999.

[35] 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*, 22(2), 2003.

[36] Jeremy Searock, Brett Browning, and Manuela Veloso. Turning segways into soccer robots. *Proceedings of IROS'04*, September 2004.

[37] J. Spletzer, A.K. Das, R. Fierro, C.J. Taylor, V. Kumar, and J.P. Ostrowski. Cooperative localization and control for multi-robot manipulation. In *IROS 2001*, 2001.

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

[39] Dizan Alejandro Vasquez Govea and Thierry Fraichard. Motion prediction for moving objects: a statistical approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3931–3936, New Orleans, LA (US), April 2004.

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

[41] L.-S. Wang, Y.-T. Chiang, and F.-R. Chang. Filtering method for nonlinear systems with constraints. *IEEE Proc. - Control Theory and Appl.*, pages 525–531, November 2002.

[42] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.