Learning Tactic-Based Motion Models of a Moving Object with Particle Filtering

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

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

***Abstract*— Learning motion models of a moving object is a challenge for autonomous robots. We address the particular instance of parameter learning when tracking object motions in a switching multi-model system. We present a general algorithm of joint parameter-state estimation based on multi-model particle filter. We apply the approach to a specific ball-tracking problem and extend the algorithm to learn model parameters in a dynamic Bayesian network (DBN). We show empirical results in simulation and in a team robot soccer environment, as a substrate for applying the learned models to object tracking in a team. The learning capability allow the tracker to much more effectively track mobile objects.**

I. INTRODUCTION

Many engineering applications are characterized by non-linear or linear dynamic systems with a few possible modes (models) [8]. 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 [6].

This paper 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 can actuate over the ball, e.g., grab and kick the ball, making the motion model of the ball very complex [11]. The good news is that we can acquire information about the ball motion from multiple sources besides the sensor. First, the robot's tactics provides valuable information and a tactic-based motion modeling and tracking algorithm is introduced in such scenarios [9]. Second, when the robot is playing a game as a member of a team, the team coordination knowledge provides further information that can be incorporated into the motion modeling and tracking process. We based our work upon a play-dependent tracking algorithm called play-based tracking [10].

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 paper, we present a novel method of automating the procedure of acquiring this probabilistic motion model. 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.

The paper is organized as follows. We first talk about related work to this paper. We give a brief description of the hybrid system model and joint parameter and state estimation. Next we show our algorithm of parameter-learning-based forward particle filtering and backward particle smoothing. We describe the learning algorithm, leading to our experimental results and conclusions.

II. RELATED WORK

There are several areas of previous work related to this research. We discuss them along the three main axes of our approach: (i) adaptive estimation and maneuvering targets; (ii) joint parameter and state estimation; (iii) learning algorithms on switching linear models.

Adaptive estimation algorithms are considered to deal with the uncertainty in a system. One type of uncertainty is the case of unknown inputs into the system, typically maneuvering targets. The other type is a combination of system parameter uncertainties with unknown inputs. One way to accommodate this is by modeling maneuvers as random process. Such approaches, including noise level switching and usage of autocorrelated noise are simple and approximate but quite effective sometimes. The second type of approach, input estimation, is implemented assuming the input to be constant over a certain duration. Such approaches, including variable state dimension (VSD), have difficulty providing noise reduction during the critical time of a maneuver. The third type of approach, multiple model (MM) algorithms assumes that the system behaves according to one of a finite number of models. The models can differ in noise levels or their structure. Among these MM algorithms, generalized pseudo-Bayesian (GPB) and interacting multiple model (IMM) are two suboptimal approaches [2]. GPB carries out merging after the measurement update step. IMM yields similar performance to GPB, but by merging after the hypothesis branching step, a lower complexity and computational load is

achieved. The fourth type, particle filter applied to the multi-model estimation problem, shows that the optimal multi-model solution can be obtained with a constant number of filters by taking the general Bayesian approach, and using more complex non-Gaussian densities. Further, since the particle filter is a general, recursive, Bayesian estimator, the approach is also directly applicable to nonlinear and non-Gaussian multiple-model case [13], [16]. Our approach is based on a multi-model particle filter; hypothesis branching is approximated by resampling from the system switching probability distribution function, and multi-model parameters are learned through tracking state variables.

Dealing simultaneously with both unknown fixed model parameters and state variables is a challenging problem. The initial idea is to augment the base state and put the unknown parameters in the framework of nonlinear state estimation. Then artificial process noise is added into the parameter equation. The key issues of this approach is that pretending the fixed parameters to be time-varying implies an artificial "loss of information". The kernel smoothing method resolved this issue by shrinkage of kernel locations to retain the mean $\bar{\theta}_t$ [12]. In this paper, we extend this algorithm to a multi-model situation.

Switching multi-model and specifically, switching linear dynamic systems (SLDS) has been studied in many fields like statistics and target tracking. Ghahramani [7] introduced a DBN-framework for learning and approximate inference in one class of SLDS models. A switching framework for particle filters applied to dynamics learning is described in [3]. An HMM-based approach is described in [4]. Three different approximate inference schemes: Viterbi, variational method, and GPB2 has been derived and applied to figure motion modeling [15]. Our approach uses the particle filter scheme, in which a framework for switching multi-model learning is presented.

III. The Problem of Learning Motion Models

A discrete-time hybrid system is given by:

$$\mathbf{x}_t = f_{t-1}(\mathbf{x}_{t-1}, s_t, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}) \quad (1)$$

$$\mathbf{z}_t = h_t(\mathbf{x}_t, s_t, \mathbf{n}_t) \quad (2)$$

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 $\mathbf{v}_{t-1}, \mathbf{n}_t$ are respectively Q_{t-1} and R_t. The model index parameter s can take any one of M values, where M is the number of models in the system.

If the model variable is governed by a discrete-state Markov chain with transitional probabilities

$$\pi_{ij} = P\{s_t = j | s_{t-1} = i\}, (i, j \in S), \quad (3)$$

where $S = \{1, 2, \cdots, M\}$, the transitional probability matrix $\Pi = \pi_{ij}$ is an $M \times M$ matrix, and we can represent such a system in Figure 1 [14].

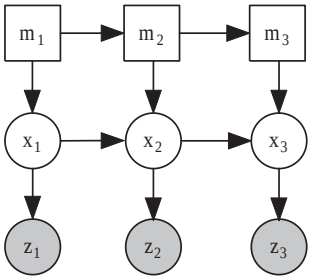

Fig. 1. A switching state-space model. For simplicity, we have omitted the input $\mathbf{u}$.

The problem of online parameter estimation can be represented by augmenting the parameters (θ) to the state space [2], [12]. This can be modeled as shown in Figure 2.

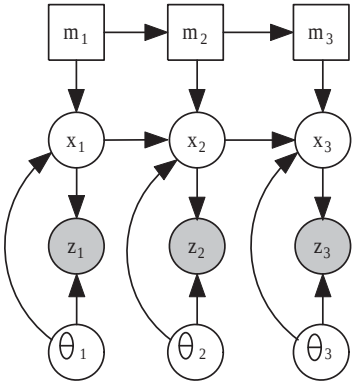

Fig. 2. Joint state-parameter estimation.

A. Tactic-Based Motion Model

When tracking is performed by a robot executing certain tasks acting over the target being tracked, such as a Segway RMP soccer robot grabbing and manipulating a ball, the motion model of the target becomes dependent on the robot's own actions. We have previously introduced a tactic-based motion model in such scenarios [9]. In short, tactics encapsulate individual robot behavior and instantiate actions through sequences of skills. Skills implement the focused control policy for actually generating useful actions. The approach includes the nonstandard information in terms of the robot's own behavior into a multi-model representation of the motion of the target.

In our soccer robot environment, we define three models, namely *Free-Ball*, *Robot-Grab-Ball*, *Robot-Kick-Ball*, to model the ball motion. Briefly, *Free-Ball* is a motion model that describes the ball's movement without external actuation. *Robot-Grab-Ball* and *Robot-Kick-Ball* are the two motion models that describe the robot's own actuation effects on the ball. The direction for how to infer which model to use and how to transition from one model to another (π_{ij}) are tactic-based. Therefore it is an extension of the ordinary jump Markov model. For detailed explanations about tactic-based motion tracking, please refer to [9].

We use dynamic Bayesian network (DBN) to represent the whole system for ball tracking. We apply the joint parameter-state algorithm to this problem and extend the algorithm to learn model parameters in a DBN. The new problem can be modeled as shown in Figure 3.

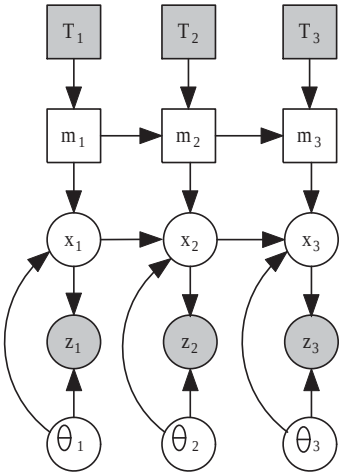

Fig. 3. Parameter learning in tactic-based motion modeling. For simplicity, we have omitted the infrared sensor observation.

B. Include Parameters into the State Space

Dealing simultaneously with both unknown fixed model parameters θ and state variables $\mathbf{x}$ is a challenging problem. The initial idea is to augment the base state and to put the unknown parameters in the framework of nonlinear state estimation. Then artificial process noise is added into the parameter equation. That is,

$$\theta_t = \theta_{t-1} + \zeta_t, \quad \zeta_t \sim \mathcal{N}(0, W_t) \tag{4}$$

for some specified variance matrix W_t and where θ_t and ζ_t are conditionally independent given the information set I_t. The key issue of this approach is that pretending the fixed parameters to be time-varying implies an artificial "loss of information." The posteriors are actually far too diffuse relative to the theoretical posteriors for the actual fixed parameters. The kernel smoothing method resolved this issue by a shrinkage of kernel locations to retain the mean $\bar{\theta}_t$ [12].

C. Particle Filtering for DBNs

Thus the problem of online parameter estimation can be represented by augmenting the parameters to the state space. In order to do state estimation and learn the model parameters, we need to be able to perform inference in the DBN. There have been several approximate inference techniques proposed for DBNs, but they are designed primarily for discrete domains. Sequential Monte Carlo methods are currently the only approach that allow us to perform filtering in general purpose hybrid DBN models. The particle filter is a Monte Carlo scheme for tracking and smoothing in dynamic systems [6]. It maintains the belief state at time t as a set of weighted particles $p_t^{(1)}, \cdots, p_t^{(N)}$, where each $p_t^{(i)}$ is a full instantiation of the tracked variables.

Thus in a sample-based framework, standing at time t, we have a sample of current states $\{\mathbf{x}_t^1, \cdots, \mathbf{x}_t^N\}$ and associated weights $\{w_t^1, \cdots, w_t^N\}$ that together represent a Monte Carlo importance sample approximation to the posterior $p(\mathbf{x}_t|I_t)$ [1], where $I_t = \{I_{t-1}, \mathbf{z}_t\}$ is the observed information set. When fixed parameter θ is added into the state, we now have a joint sample

$$\{\mathbf{x}_t^i, \theta_t^i : i = 1, \cdots, N\}$$

and associated weights $\{w_t^1, \cdots, w_t^N\}$. Let $\bar{\theta}_t$ and $\mathbf{V}_t$ be the Monte Carlo posterior mean and variance matrix of $p(\theta|I_t)$. The smooth kernel density is given by [12]

$$p(\theta|I_t) \approx \sum_{i=1}^{N} w_t^i \mathcal{N}(\theta|\kappa_t^i, h^2\mathbf{V}_t), \tag{5}$$

where $\mathcal{N}(\cdot|\kappa, \mathbf{S})$ is a multivariate normal density with mean κ and variance matrix $\mathbf{S}$, and h is chosen as a slowly decreasing function of N. The following shrinkage of kernel locations is introduced in [17]:

$$\kappa_t^i = a\theta_t^i + (1-a)\bar{\theta}_t^i, \tag{6}$$

where $a = \sqrt{1-h^2}$. An extension of [17] quantifies the undesirable "loss of information" and presented a way to determine h. $h^2 = 1 - ((3\delta - 1)/2\delta)^2$, where δ is a discount factor in (0,1], typically around 0.95 - 0.99. A general combined parameter and state estimation algorithm is introduced in [12].

D. The Learning Algorithm

Our approach is based on the above algorithm and extends it to a switching multi-model system. Suppose we have M models in the system, each model (or a subset of all the models) has unknown fixed parameters. Let $\theta = (\theta_1, \theta_2, \cdots, \theta_M)'$. Now standing at time t, we have a joint sample

$$\{\mathbf{x}_t^i, (\theta_{1,t}^i, \theta_{2,t}^i, \cdots, \theta_{M,t}^i)' : i = 1, \cdots, N\}$$

and associated weights $\{w_t^1, \cdots, w_t^N\}$.

The pseudo code of the algorithm is shown below.

LEARN-MODELS $(x_{t-1}^i, s_{t-1}^i, \theta_{t-1}^i, w_{t-1}^i, z_t, T_{t-1})$

```
select model k to update its parameters
for i ← 1 to N
    do calculate the i^th kernel location of model k
       κ^i_{k,t} ← a θ^i_{k,t-1} + (1 − a) θ̄^i_{k,t-1}
       draw s^i_t ∼ p(s_t | s^i_{t-1}, T_{t-1})
       μ^i_t ← E(x_t | x^i_{t-1}, θ^i_t, s^i_t)
       w^i_t ∝ p(z_t | μ^i_t, κ^i_t)
normalize-weight
[{−, −, i^j}_{i=1}^{N_s}] = RESAMPLE[{x^i_t, w^i_t}_{i=1}^{N_s}]
for i ← 1 to N
    do θ^i_{k,t} ∼ N(· | κ^j_t, h^2 V_{t-1})
       Draw x^j_t ∼ p(x_t | s^{i^j}_t, x^{i^j}_{t-1}, θ^{i^j}_t)
       w^j_t ∝ p(z_t | x^j_t, θ^{i^j}_t) / p(z_t | μ^{i^j}_t, κ^{i^j}_t)
       s^j_t ← s^{i^j}_t
normalize-weight
```

At the beginning, we select one of the models, say k, and we will update the parameters of this model at current time t. The selection criteria is to choose the model with maximum associated weights over all the particles. Other selection schemes are also considered, but we do not discuss them in this paper. For each particle, we proceed by first calculating the i_{th} kernel location of model k at line 3. Next we do a sequence of sampling following the DBN (lines 5-6). We then identify the prior point estimates of (x_t, θ) given by (μ_t^i, s_t^i). Weights are updated and normalized. We apply an extended version of the multi-model auxiliary particle filter algorithm, incorporating the parameter with the state. In detail, for each particle, we first sample a new parameter vector $\theta_{k,t}^i$ from the j^{th} kernel density at line 12. We sample a value of current state vector x_t^j from the system equation at line 13. We evaluate the corresponding weight and normalize.

IV. EXPERIMENTAL RESULTS

In this section, we test our algorithm both in simulation and in a team robot soccer environment. The simulated test verifies the efficiency of our proposed algorithm. We then give a brief description of our robot, followed by real robot test and results.

A. Simulated 2D Object Tracking

In this section, the following methods of motion tracking are illustrated and compared:

- Method A_0: switching kalman filter which deterministically know which model to use in each discrete time step (assumes the tracker knows the model sequence, which actually the filtering algorithm needs to estimate). We include this method only as a performance bound and we call it "magic KF" in the rest of this section.
- Method A_1: our multi-model joint parameter-state estimation
- Method A_2: IMM.

Each simulation run uses the same random variables for all the algorithms. To compare the performance of the algorithms A_1 and A_2, we use the comparison technique described in [2]. We take the comparison as a hypothesis testing problem. The decision whether A_1 is better than A_2 in the simulated scenarios is made upon the sample performance differences

$$\Delta_i = C_i^2 - C_i^1,$$

where C_i^k is the performance of algorithm k in run i. The hypothesis $H_1 : A_1$ is better than A_2 is accepted if

$$\mu = \frac{\bar{\Delta}}{\sigma_{\bar{\Delta}}} > \mu_0,$$

where

$$\bar{\Delta} = \frac{1}{N}\sum_{i=1}^{N}\Delta_i,$$

and

$$\sigma_{\bar{\Delta}} = \sqrt{\frac{1}{N^2}\sum_{i=1}^{N}(\Delta_i - \bar{\Delta})^2}.$$

Assuming the error in $\bar{\Delta}$ to be normal, we have $\mu_0 = 1.64$ for a 5% level. The performance of interest will be the mean square error in the estimate of one state component.

The simulation considers a target whose position is sampled every $T = 0.033s$. The initial condition of the target, with state

$$\mathbf{x} = [\xi \ \eta \ \dot{\xi} \ \dot{\eta}]'$$

is, with position and velocity units m and m/s, respectively

$$\mathbf{x}_0 = [10 \ 10 \ 0 \ 0]'.$$

Positions are the sole measurements collected according to the equation

$$\mathbf{z}_t = [1 \ 0 \ 1 \ 0]\mathbf{x}_t + \mathbf{n}_t$$

with $\mathbf{n}_t$ zero mean, white, independent of the process noise, and with variance

$$E[\mathbf{n}_t^2] = Q \cdot I$$

where $Q = 1m^2$.

Two types of motion models are employed. The first assumes constant velocity (with a decay of 0.95) in the Cartesian frame with small deviations in velocity on ξ-η axis by zeros-mean, Gaussian white noise with covariance Q_{cv}. The second model, employed to track the motion actuated by a robot, assumes a similar model except an external input $\mathbf{u}$ with large deviations in velocty by zeros-mean, Gaussian white noise with covariance Q_{ca}. The Markov transition probability matrix is defined as

$$\Pi = \begin{bmatrix} 0.8 & 0.2 \\ 0.25 & 0.75 \end{bmatrix}$$

and the initial probabilities are set to

$$p(m_0) = \begin{bmatrix} 0.95 & 0.05 \end{bmatrix}'$$

The first trial examines the RMS position error versus time for the magic KF, multi-model particle filter and IMM. In this trial, correct multiple dynamic models are provided with each algorithm, that is to say, the parameters is known to the trackers. The performance results are shown in Figure 4 where the three filtering algorithm achieved approximately the same performance. This is due to the small process and measurement noise.

In the second test, the process noise in the first model is multiplied by a factor of 10, and the process noise in the second model is multiplied by a factor of 20. Only the "magic KF" algorithm can "feel" this change. Algorithm A_1 takes the noise level in both models to be the unknown parameters and adapts parameters through joint parameter-state estimation. Algorithm A_2 does the same thing as in the first trial.

Figure 5 shows the RMS error curves corresponding to the three filters considered. From the graph, it is clear that the performance of IMM without learning is poor compared to the other two filters. Table I shows the test for the difference of the MSE between the two algorithms over several time intervals. Statistically significant improvements

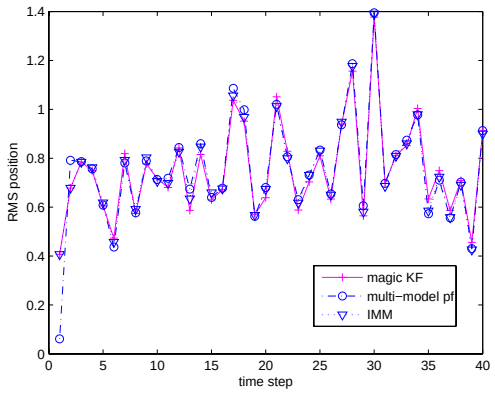

Fig. 4. Estimation errors in coordinate x for the three tracking filters with correct models(from 40 runs, known model transition).

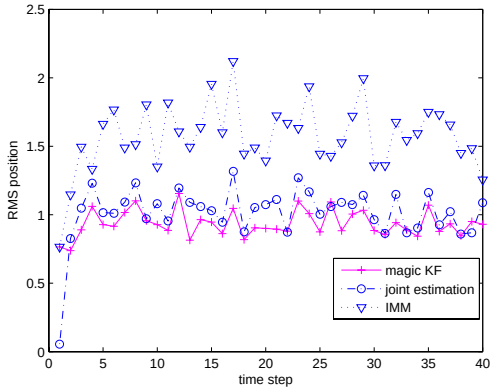

Fig. 5. Estimation errors in coordinate x for the three tracking filters (from 40 runs).

of our algorithm over the IMM without learning are observed in position over all the intervals considered. For this problem, it is sufficient to acclaim A_1 is better than A_2 at the 5% level. Figure 6 shows the parameter learning curves corresponding to the time step using algorithm A_1.

The next test examines the error performance of the particle filter for varying values of the number of particles. Figure 7 shows the efficiency at the final time, as a function of the number of particles for $500 \leq N \leq 4000$. As expected, we see an improvement in performance as the number of particles is increased. However, note that as N is increased beyond 3000, there is insignificant improvement in performance. Thus we use $N = 3000$ in our simulations, although note that the optimal value of N is scenario and parameter dependent.

TABLE I

TEST OF MEANS FOR COMPARISON FROM 40 RUNS

Time interval	Δ	$\sigma_{\bar{\Delta}}$	Test statistics
1-10	.476	.245	1.943
11-20	.597	.211	2.832
21-30	.569	.191	2.976
31-40	.580	.169	3.427
1-40	.555	.203	2.730

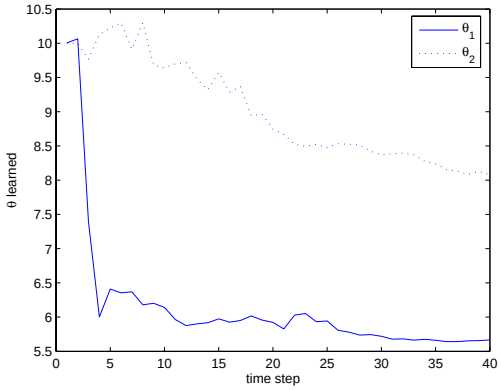

Fig. 6. Learning parameters of two different models.

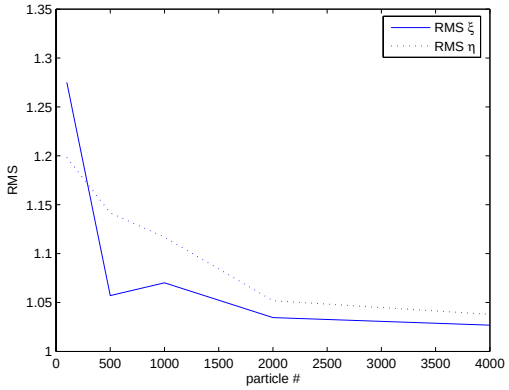

Fig. 7. RMS efficiency at the final time versus the number of particles (from 40 runs).

B. Segway RMP Soccer Robot

Segway RMP, or Robot Mobility Platform, provides an extensible control platform for robotics research. In our previous work, we have developed a Segway RMP robot base capable of playing Segway soccer [5]. The main sensor on our robots are two cameras. One is a pan-tilt camera mounted on the top of a customized unit. The other is a wide-angle camera. The infrared sensor is added as a secondary sensor to detect the ball when the ball is in the catchable area of the robot. Its measurement is a binary value indicating whether or not the ball is in that area. Our robot is also equipped with a catcher to trap the ball and a kicker to kick the ball.

C. Learning Motion Models

In the robot test, two Segway RMP soccer robots are included. One of the robots acts as the observer (A), who is executing the tactic *CatchBall*. The other robot acts as its teammate (B) who is executing the tactic *PassBall*. In this test, we assume there is no uncertainty on the tactic level. We are only interested to learn model parameters for each motion model conditioned on the given tactic. Each

experiment trial starts from the state that the robot B holds the ball and searches A. When B finds A, it passes the ball to A. A then aims at the ball and catches the ball when the ball is within its catchable area. The trial ends once the ball is being catched or runs out of the field without being received. In the beginning of each trial, A is at position (0,0) and B is at position (2.5, 0).

There are two kinds of ball motion models to be learned in this test. They are *Free-Ball* and *Robot-Grab-Ball*. We run 30 trials on the pair of robots. Vision sensor and infrared sensor logs are generated for off-line learning usage. Obviously there is only position information that can be obtained from both sensors. The velocity $(\dot{x}, \dot{y})$ is unobservable through the measurements. Robot A then reads the logs and runs our learning algorithm in each trial. The robot learns the measurement and process noise models

$$\theta = (R, Q_m) \qquad m = 1, 2 \tag{7}$$

(see Eqn. 1, 2 for details). Figure 8 shows the first standard deviation of the process noise along the x and y axes respectively. In the progression from the initial iteration to the final, the mean and variances are shifted significantly.

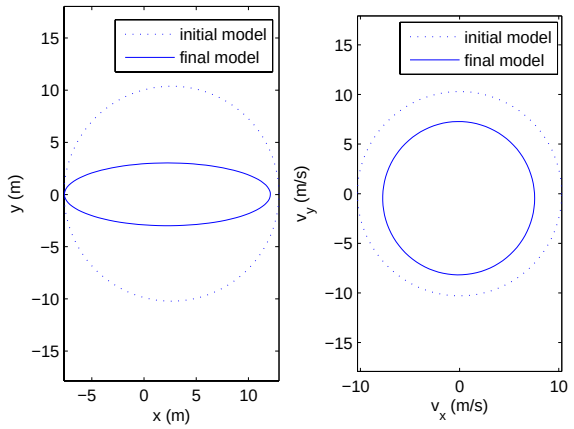

Fig. 8. Comparing the initial process noise *Free-Ball* model and the learned *Free-Ball* model .

To determine the performance of the learned motion model, we also performed an experiment to apply the learned process/measurement noise model to the existing tracking system. We then runs the similar trials, and use two different trackers simultaneously. One is the previous tracker, the other is the new tracker that includes the learned model parameters. We find that the tracker with learned model performs significantly better than the tracker with initial model.

V. CONCLUSIONS

Learning motion models of a moving object is a challenge for autonomous robots. We address the particular instance of parameter learning when tracking object motions in a switching multi-model system. We present a general algorithm of joint parameter-state estimation based on multi-model particle filter. We apply the approach to a specific ball-tracking problem and extend the algorithm to learn model parameters in a dynamic Bayesian network (DBN). We show empirical results in simulation and in a team robot soccer environment, as a substrate for applying the learned models to object tracking in a team. The learning capability allow the tracker to much more effectively track mobile objects.

VI. ACKNOWLEDGMENTS

This research was sponsored in part by the United States Department of the Interior under Grant No. NBCH-1040007 and the Boeing Corporation. The views and conclusions contained in this document are those of the authors only, and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution.

REFERENCES

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

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

[3] A. Blake, B. North, and M. Isard, "Learning multi-class dynamics," 1998.

[4] M. Brand, "An entropic estimator for structure discovery," 1998.

[5] B. Browning, J. Searock, P. E. Rybski, and M. Veloso, "Turning segways into soccer robots," *Industrial Robot*, vol. 32, no. 2, pp. 149–156, 2005.

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

[7] Z. Ghahramani and G. E. Hinton, "Switching state-space models," 6 King's College Road, Toronto M5S 3H5, Canada, Tech. Rep., 1998.

[8] ——, "Variational learning for switching state-space models," *Neural Computation*, vol. 12, no. 4, pp. 831–864, 2000.

[9] Y. Gu, "Tactic-based motion modeling and multi-sensor tracking." in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-05)*, 2005, pp. 1274–1279.

[10] Y. Gu and M. Veloso, "Multi-model motion tracking under multiple team member actuators," in *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press, 2006, pp. 449–456.

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

[12] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo Methods in Practice. New York*, J. F. G. D. F. A. Doucet and N. J. Gordon, Eds. Springer-Verlag, New York, 2000.

[13] S. McGinnity and G.W.Irwin, "Multiple model bootstrap filter for maneuving target tracking," *IEEE Trans. Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 1006–1012, 2000.

[14] K. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, 2002.

[15] V. Pavlovic, J. M. Rehg, and J. MacCormick, "Learning switching linear models of human motion," in *Neural Information Processing Systems*, Denver, CO, November 2000. [Online]. Available: http://www.cs.rutgers.edu/ vladimir/pub/nips00.pdf

[16] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter, Particle Filters for Tracking Applications*. Artech House Publishers, 2004.

[17] M. West, "Approximating posterior distributions by mixtures," in *Journal of the Royal Statistical Society*, 1993.