Rao-Blackwellised Particle Filter for Tracking with Application in Visual Surveillance

Xinyu Xu and Baoxin Li
{xinyu.xu and baoxin.li}@asu.edu
Department of Computer Science and Engineering
Arizona State University
Tempe, Arizona 85287, USA

Abstract

Particle filters have become popular tools for visual tracking since they do not require the modeling system to be Gaussian and linear. However, when applied to a high dimensional state-space, particle filters can be inefficient because a prohibitively large number of samples may be required in order to approximate the underlying density functions with desired accuracy. In this paper, by proposing a tracking algorithm based on Rao-Blackwellised particle filter (RBPF), we show how to exploit the analytical relationship between state variables to improve the efficiency and accuracy of a regular particle filter. Essentially, we estimate some of the state variables as in a regular particle filter, and the distributions of the remaining variables are updated analytically using an exact filter (Kalman filter in this paper). We discuss how the proposed method can be applied to facilitate the visual tracking task in typical surveillance applications. Experiments using both simulated data and real video sequences show that the proposed method results in more accurate and more efficient tracking than a regular particle filter.

1. Introduction

Visual tracking is an important step in many practical applications including video-based surveillance. In recent years, particle-filter-based visual tracking has been extensively studied (e.g., [1, 2, 3 and 4]). Particle filtering has been shown to offer improvements in performance over some conventional methods such as the Kalman filter, especially in non-linear/non-Gaussian environments [5], but the large number of samples required to represent the posterior density prevent its use in high dimensional state-space. However, in some cases, the model may have "tractable structure" with some components having linear dynamics and so can be marginalized out and analytically estimated using exact filters conditional on certain other components. The exact filters could be the Kalman filter, the HMM filter, or any other finite dimensional optimal filters [6]. This technique is called Rao-Blackwellisation. The resultant method is often called Rao-Blackwellised particle filter (RBPF).

Denote the state to be estimated as X_t and observation as Z_t with subscript t the time index. The key idea of RBPF is to partition the original state-space X_t into two parts R_t (*root variables*), and L_t (*leaf variables*), such that $p(L_{1:t}|R_{1:t},Z_{1:t})$ is a distribution that can be computed exactly , therefore an approximation of $p(R_{1:t}|Z_{1:t})$ using Monte Carlo methods yields straightforwardly an approximation of joint posterior $p(R_{1:t},L_{1:t}|Z_{1:t})$[1] . The justification for this decomposition follows from the factorization of the probability:

$$p(R_{1:t},L_{1:t}|Z_{1:t})=p(L_{1:t}|R_{1:t},Z_{1:t})p(R_{1:t}|Z_{1:t}) \quad (1)$$

If the same number of particles is used in a regular particle filter and an RBPF, intuitively the latter will provide better estimates. The reason for that is twofold: first, the dimension of $p(R_{1:t}|Z_{1:t})$ is smaller than $p(R_{1:t},L_{1:t}|Z_{1:t})$, and thus is presumably easier to handle; secondly, optimal algorithms may be used to estimate the 'tractable substructure'. This optimal algorithm could be Kalman filter, HMM filter, or Junction Tree algorithms. In [6] , the author pointed out that $p(L_{1:t}|R_{1:t},Z_{1:t})$ can be efficiently updated using Kalman filter when the initial uncertainty for leaves is Gaussian, and the CPDs (Conditional Probability Distributions) of the observation model and system dynamics for leaves are linear functions of the leaf states. In this paper we will show how Kalman filter is combined with particle filter to facilitate tracking in a typical surveillance application, and we will also show how a deterministic dependency relationship between leaves and roots can be exploited to achieve a better estimation for the linear parts.

RBPF has been applied in some state estimation problems. For example, in [7], Rao-Blackwellised particle filter is used to integrate out the subspace coefficients in an Eigen Tracking problem. In [8], in the problem of tracking multiple people using a combination of anonymous and id sensors, Rao-Blackwellised particle filter is used to estimate the locations and identities of multiple objects, with each particle representing a history of associations between object tracks and observations, Kalman filter is used to track an individual person. In [9], Freitas et al combine Kalman filter with particle filter for fault diagnosis for a mobile robot in Mars exploration, where Kalman filters are applied over continuous states and samples are obtained over discrete states. In [10], the non-linear ball motion model and robot location is tracked

using particle filter while ball location and velocity is estimated efficiently by Kalman filter. A more generalized discussion regarding marginalized particle filter for mixed linear/nonlinear state-space models can be found in [11].

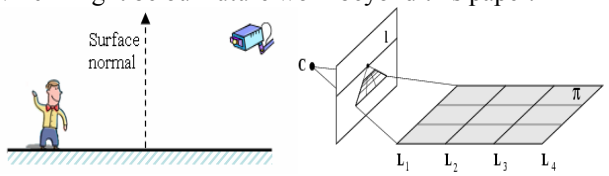
Although RBPF has been studied in the context of multi-target tracking [8], fault diagnosis for robot and robot localization [9, 10] and signal processing [12], its application in tracking for surveillance has yet to be fully explored. In particular, we believe that a thorough qualitative performance comparison between RBPF and a regular particle filter will help us to understand the advantages of RBPF. The key contribution of this paper is thus two fold: First, with video-based surveillance as a case study, we utilize the constraints imposed by typical camera-scene configuration to partition the original state space into two sub-spaces, and then we propose the RBPF algorithm for surveillance tracking; Secondly, experiments on both simulation data and real video sequence are carried out to gain quantitatively performance analysis. The improvements of proposed RBPF over regular PF manifest in three aspects: increased estimation accuracy, reduced particle numbers are needed to achieve the same level of accuracy, and reduced variance for estimates.

This paper is organized as follows. In Section 2, we describe all the details of the proposed RBPF algorithm. Section 3 reports experiments designed to compare the proposed algorithm and a regular particle filter on both simulated data and real video sequences. We conclude in Section 4 with a brief discussion about future work.

2. RBPF for Tracking in Surveillance

2.1 Partition the State Space

In typical surveillance applications, for most of the time, the tracked objects are constrained to move on a dominant plane (e.g. the ground), and the camera is usually higher than the tracked object. Fig. 1(a) illustrates such a scene configuration. Fig. 1(b) is a geometric representation for Fig. 1(a). In Fig. 1(b), suppose a person is moving on the ground plane π, and the ground is projected onto the image plane by camera C, l is the vanishing line for the ground plane. Any scene point projecting onto the vanishing line l is at the same distance from plane π as the camera center is from π [13]. If the scene point is farther from π than the camera, then the image point lies 'above' the vanishing line; and 'below' if it is closer to the ground than the camera. So if the to-be-tracked object is not higher than the camera to the ground, the image of the object will always lie 'below' the vanishing line, and when it moves towards to the camera, the size (or scale) of the object on the image will get bigger as the y coordinate on image plane gets bigger, and vice versa. From Fig. 1(b), we may see the dependence of the *scale change* of the object on the *translational motion* (or the x or y locations in the image domain). In these situations, the constraints due to the camera-scene configuration can be exploited to deduce the dependency relationship between state variables. It is also possible to exactly link the translational motion to the scale change (both considered in the 2-D image domain) by detecting the surface normal with respect to the camera (Fig.1 (a)), which might be our future work beyond this paper.



(a) (b)

Figure 1. (a) Illustration of a typical surveillance scene-camera configuration. (b) The dominant plane is projected onto the image plane.

Formally, in our work we use an ellipse to model the tracked object, and the following 8-D state model is used to describe the dynamics of the ellipse, following [14]:

$$\{x, \tilde{x}, y, \tilde{y}, H_y, \widetilde{H_y}, H_x, \widetilde{H_x}\}$$

where *(x,y)* represent the center location of the ellipse, $\{\tilde{x}, \tilde{y}\}$ represent the motion velocity, $\{H_y, H_x\}$ are the lengths of the ellipse half axes, and $\{\widetilde{H_y}, \widetilde{H_x}\}$ are the corresponding rates of scale change on the axes. With the above idea, the scale change of a moving object is related to its position along the y-axis (i.e., the vertical axis in the image domain). See Fig. 1(b), the length of the y-axis of the ellipse will become bigger as the y location of the object gets bigger (which means the object is moving toward the camera), while the length of the y-axis will become smaller as the y location gets smaller (which means the object is moving further from the camera). This facilitate us to partition the original 8-D state space into two groups: the root variables R containing the motion information (including location and velocity), which will be sampled by a regular particle filter, and the leaf variables L consisting of the scale parameters (including the rate of scale change), which will be estimated by an exact filter. These two groups are denoted by:

$$R = \{x, \tilde{x}, y, \tilde{y}\} \qquad L = \{H_y, \widetilde{H_y}, H_x, \widetilde{H_x}\}$$

In this work, the actual observations used for estimating the root variables, denoted by Z_t, are the color histogram within an ellipse specified by a sample and the intensity gradients along the ellipse boundary. Due to the noise in measurements from the image and the possibly irregular movements of the tracked object, the object's motion are typically better modeled by probabilistic multi-modal densities, especially as the observations cannot be linearly linked to the root state R. Therefore, particle filter is appropriate for sampling the density space of the root variables. On the other hand, the observation used for the

leaves, Y_t, are simply $\{H_y, H_x\}$, which does not directly rely on the color histogram or the intensity gradients used for root variables. Thus the observations form a linear relationship with state L. Moreover, the leaf state at time t-1, L_{t-1}, is able to be linearly projected to state at time t, L_t. Also we assume that the system and observation model for the leaf variables are driven by Gaussian random noise. As a result, Kalman filter can be used to optimally estimate the leaf variables. This results a Rao-Blackwellised particle filter that combines the representation power of a particle filter with the efficiency and accuracy of the Kalman filter.

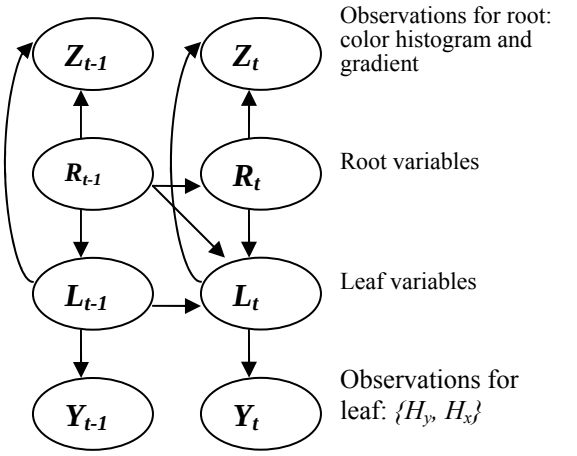


Figure 2 Dependency relationships between various components of the surveillance tracking problem.

In Fig. 2, the relationships between the state variables and the observations are graphically illustrated by a DBN [6]. In Fig. 2, the object motion (location and velocity) at time t, R_t, only depends on the previous motion R_{t-1}, while scale of the ellipse at time t, L_t, depends on previous ellipse scale, L_{t-1}, previous object motion R_{t-1} and current motion R_t. The observations for leaf at time t, Y_t, only depends on current scale L_t, while the observations for root at time t, Z_t, depends on both current object motion, R_t, and current scale L_t.

Now that the dependencies between different parts of the state space are defined, we can address the problem of filtering using Rao-Blackwell technique, which aims at computing the posterior over $<R_t, L_t>$ conditional on all observations. This is detailed in Section 2.2.

2.2 Algorithm Description

Fig. 3 illustrates the proposed RBPF algorithm. Just like regular particle filters, RBPFs represent posterior by a set of weighted samples: $S_t = \{s_t^i, w_t^i \mid 1 \le i \le N\}$. Each particle maintains not just a sample from $p(R_t|Z_t)$, which we denote by R_t^i, but also a parametric representation of the distribution $p(L_t \mid R_t^i, Z_t)$ which consists of the mean vector of the leaf state distribution, $\mu_t^i = E[L_t]$, and the estimation error covariance reflecting the variance of the leaf state distribution, $\sigma_t^i = E[(L_t^i - \mu_t^i)(L_t^i - \mu_t^i)^T]$ [6]. So each particle is represented by: $s_t^i =< R_t^i, \mu_t^i, \sigma_t^i >$. The proposed RBPF algorithm will sample the non-linear non-Gaussian motion R_t^i using particle filter, while apply Kalman filter to estimate the scale parameters μ_t^i and σ_t^i conditional on the motion state which have already been estimated. More specifically, RBPF technique will generate new sample set distributed according to Eq. (1) stepwise from right to left based on the previous sample set S_{k-1}. To do so, the following algorithm steps are performed for all particles at each time instant.

Input:: $S_{t-1} = \{< R_{t-1}^i, \mu_{t-1}^i, \sigma_{t-1}^i > \mid i = 1 \cdots N\}$ and Z_t.
For $i = 1{:}N$ do
1. Propagate samples
a) ***Sample object motion***:
$R_t^{i-} \sim p(R_t \mid R_{t-1}^i, Z_t) = p(Z_t \mid R_t)\, p(R_t \mid R_{t-1}^i)$
b) ***Kalman prediction***:
$L_t^{i-} \sim p(L_t \mid R_t^i, R_{t-1}^i, L_{t-1}^i, Z_t)$
2. Evaluate weight
a) ***Compute the color histogram:*** (8a)~(8b)
b) ***Compute the gradient:*** (9a)~(9c)
c) ***Compute the weight:*** (10a)~(10d)
$w_t^i \propto p(Z_t^{CH} \mid R_t^i, L_t^i) \bullet p(Z_t^G \mid R_t^i, L_t^i)$
End for loop
3. Select samples.
For $i = 1{:}N$ do
4. Kalman update: (11)
End for loop
5. Compute the mean state : (12)

Figure 3 RBPF for tracking in surveillance

1. *Propagate samples*
a) *Sample object motion* according to

$$R_t^{i-} \sim p(R_t \mid R_{t-1}^i, Z_t) = p(Z_t \mid R_t)\, p(R_t \mid R_{t-1}^i) \quad (2)$$

This means that a new object location and velocity will be propagated by the particle filter using a system motion model. A simple first order motion model was adopted in our work: $R_t^{i-} = TR_{t-1}^i + N_{t-1}$ where N_{t-1} is a random vector drawn from the noise distribution of the system. After this step, $s_t^i =< R_t^{i-}, \_, \_ >$. The "super minus" means that the corresponding variable is *a priori* estimate and _ denotes uninitialized value. In the regular particle filer, immediately after this step we are supposed to weight each sample by the left term which is the likelihood of the observation Z_t. But in the proposed RBPF algorithm, Kalman prediction is necessarily to be performed before weighting each sample, which is meant to analytically predict a new mean and covariance for the leaf variables

by making use of the fact that we already sample the object motion. This step is detailed in sub-step b).

b) *Kalman prediction* for leaf states according to

$$L_t^{i-} \sim p(L_t \mid R_t^i, R_{t-1}^i, L_{t-1}^i, Z_t) \quad (3)$$

Equation (2) and (3) follow directly from the two right terms in Eq. (1) and the graphical representation in Fig. 2. Eq. (3) is only an abstract expression, to actually apply Kalman filter, the system and measurement model have to be identified for the linear statistics. In our case, the two models are established as:

$$L_t^i = AL_{t-1}^i + w_t^i \quad (4) \qquad Y_t^i = HL_t^i + v_t^i \quad (5)$$

w and *v* is the system and measurement noise for Kalman filter respectively, with $p(w_t^i) \sim N(0, Q)$ and $p(v_t^i) \sim N(0, R)$. The system noise covariance Q and measurement noise covariance R might change with each time instant, but in our work they are set as fixed values with $Q_{4\times4} = 4I$ and $R_{2\times2} = 2I$. And

$$A = \begin{pmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad H = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \end{pmatrix} \quad (6)$$

Then Kalman prediction is performed by Eq. (7) (keep in mind $\mu = \{H_y, \widetilde{H_y}, H_x, \widetilde{H_x}\}$ and $Y = \{H_y, H_x\}$). The first four formulas in Eq. (7) perform prediction for the mean of leaf variables, the last two formulas are covariance and observation prediction respectively. The first formula means that each time the *y*-coordinate of the object increases β pixels from *t-1* to *t*, the half length of *y*-axis will be increased α pixels accordingly. Note that we have made use of the location prediction y_t^{i-} which was just sampled in step a). In the third formula, the scale change will keep the aspect ratio during tracking. In practice, parameter α and β should be adjusted according to the angle between ground plane and image plane, the more the angle, the bigger the scalar should be, and the higher the uncertainty of the scale change conditional on the location. The two parameters α and β influence greatly the estimation accuracy of the proposed algorithm as they control the change rate of sample ellipse size with respect to the object motion. In Section 3.1, simulation results show that accurate tracking is able to be maintained when the dependence scalar (α/β) falls into a range around the true dependency. That is, even when the dependence scalar is not as exact as the true value, the proposed RBPF algorithm is still able to have reasonable estimation accuracy. After this step, $s_t^i = < R_t^{i-}, \mu_t^{i-}, \sigma_t^{i-} >$.

2. *Evaluate weight* for each particle

This step is a regular particle filter step, once the location (obtained by ***motion sampling***) and size (obtained by ***Kalman prediction)*** of each hypothetical ellipse region are available, the following sub-steps will be accomplished:

$$\begin{aligned} H_{yt}^{i-} &= H_{yt-1}^{i} + \alpha * (y_t^{i-} - y_{t-1}^{i})/\beta \\ \widetilde{H_{yt}^{i-}} &\sim N(\mu_{\widetilde{H_y}}, \sigma_{\widetilde{H_y}}) \\ H_{xt}^{i-} &= \frac{H_{yt}^{i-}}{H_{yt-1}^{i}} H_{xt-1}^{i} \\ \widetilde{H_{xt}^{i-}} &\sim N(\mu_{\widetilde{H_x}}, \sigma_{\widetilde{H_x}}) \\ \sigma_t^{i-} &= A\sigma_{t-1}^{i} A^T + Q \\ Y_t^{i-} &= H\mu_t^{i-} \end{aligned} \quad (7)$$

a)*Compute the color histogram* for each sample region Γ located at $c = (x_t^{i-}, y_t^{i-})$ and with the size $(H_{yt}^{i-}, H_{xt}^{i-})$ by:

$$p_c^{(u)} = f \sum_{\theta_i \in \Gamma} k\left(\frac{\| c - \theta_i \|}{a}\right) \delta\left[h(\theta_i) - u\right] \quad (8a)$$

where δ is the Kronecker delta function and $h(\theta_i)$ assigns one of the *m*-bins of the histogram to a given color at location θ_i. Pixels which are closer to the region center are given higher weights specified by:

$$k(d) = \begin{cases} 1 - d^2 : d < 1 \\ 0 : otherwise \end{cases} \quad (8b)$$

This is essentially the same method as reported in [14].

b) *Compute the gradient* for each sample region Γ. Because the color cue would become fragile in some cases such as large degree of object rotation, or distracted by the background with similar color, gradient model is employed as a complementary cue to compensate for the dramatic color change and to help the ellipse tracker properly scale since the gradient model tend to get a strong response from the contour of the tracked object. The gradient of a sample ellipse is computed as an average over gradients of all the pixels located on the boundary:

$$g(\Gamma) = \frac{1}{N_\Gamma} \sum_{i=1}^{N_\Gamma} g(x_i, y_i) \quad (9a)$$

where the gradient at pixel (x_i,y_i) is established as the maximum gradient by a local search along the normal line (l_n) of the ellipse at location (x_i,y_i):

$$g(x_i, y_i) = \max_{(x_n, y_n) \in l_n} \{g(x_n, y_n)\} \quad (9b)$$

A simple operator is used to compute the gradient in *x*-axis and *y*-axis for point (x_n,y_n):

$$g_x(x_n,y_n) = I(x_n-2,y_n) + 2*I(x_n-1,y_n) - 2*I(x_n+1,y_n) - I(x_n+2,y_n)$$
$$g_y(x_n,y_n) = I(x_n,y_n-2) + 2*I(x_n,y_n-1) - 2*I(x_n,y_n+1) - I(x_n,y_n+2)$$

And finally the gradient at point (x_n,y_n) is computed as

$$g(x_n, y_n) = \sqrt{g_x^2(x_n, y_n) + g_y^2(x_n, y_n)} \quad (9c)$$

c) *Compute the weight*. One weight is based on color histogram similarity between hypothetical region and the target model:

$$G_c^i = \frac{1}{\sqrt{2\pi}\sigma_c} e^{-\frac{(1-\rho[p_{\Gamma(i)}, q])}{2\sigma_c^2}} \quad (10a)$$

where

$$\rho[p,q] = \sum_{u=1}^{m} \sqrt{p^{(u)} q^{(u)}} \qquad (10b)$$

p stands for the color histogram of a sample hypothesis in the newly observed image, and q represents the color histogram of the target model. Eq. (10b) essentially gives the Bhattacharyya coefficient between the target histogram and the hypotheses histogram. The larger ρ is, the more similar the distributions are. Another weight is based on the gradient $g(\Gamma)$:

$$G_g^i = \frac{1}{\sqrt{2\pi}\sigma_g} e^{-\frac{(1/g(\Gamma))^2}{2\sigma_g^2}} \qquad (10c)$$

Eq. (10c) matches to the intuitive requirement that a larger gradient should produce a larger weight.

The final weight for each sample is simply the average of two above weights:

$$w_t^i = \alpha G_c^i + (1-\alpha) G_g^i \qquad \text{with } \alpha=0.5 \qquad (10d)$$

If we assume the color histogram cue is independent of gradient cue, the final weight could also be computed as $w_t^i = G_c^i * G_g^i$.

3. ***Select samples.*** Multiply/discard samples $< R_t^{i-}, \mu_t^{i-}, \sigma_t^{i-}, w_t^i >$ proportional to its weight. After this step, $s_t^i =< R_t^i, \mu_t^{i-}, \sigma_t^{i-} >$. Because the weight is proportional to the observation likelihood $w_t^i \propto p(Z_t^{CH} \mid R_t^i, L_t^i) \bullet p(Z_t^G \mid R_t^i, L_t^i)$, i.e. the weight signifies how well the hypotheses predict the measurement, so step 2 and 3 are the places where true observations have been incorporated.

4. ***Kalman update*** for leaf variables. Kalman update is accomplished by Eq. (11) over the selected sample set. In the first formula of Eq. (11), K_t^i is called Kalman gain which aims at minimizing the ***a posterior*** error covariance. It weights the measurement Y_{t-1} more heavily as the measurement error covariance R approaches zero; On the other hand the predicted measurement is trusted more than the actual measurement as the ***a priori*** estimation error covariance σ_t^{i-} approaches zero. The second formula in Eq. (11) incorporates a new measurement Y_{t-1} into the ***a priori*** leaf state estimate to obtain an improved ***a posteriori*** leaf state estimate.

$$\begin{aligned} K_t^i &= \sigma_t^{i-} H^T (H \sigma_t^{i-} H^T + R) \\ \mu_t^i &= \mu_t^{i-} + K_t^i (Y_{t-1} - Y_t^{i-}) \\ \sigma_t^i &= \sigma_t^{i-} - K_t^i H \sigma_t^{i-} \end{aligned} \qquad (11)$$

After this step, $s_t^i =< R_t^i, \mu_t^i, \sigma_t^i >$.

5. ***Compute the mean state*** at time t:

$$E[S_t] = \sum_{i=1}^{N} w_t^i s_t^i \qquad (12)$$

3. Experiments

3.1 Simulation

We first test the proposed algorithm using synthetic data. In the simulation, a point moving on a 2-D plane with non-constant velocity generates an actual path; noise is then added to the actual path to simulate a noisy measurement of the actual path. At each time step, velocity in the x-direction was added by a Gaussian random number with zero mean and variance V, which takes any of the three values {0.5,1,2} and has the probability transition matrix as in Eq. (13). The state of the moving point at any time is given by s_t={x_t, u_t, y_t, v_t, *1*},

$$Pvx = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix} \qquad (13)$$

in which {x_t, y_t} corresponds to the position and {u_t, v_t} represents velocity in x and y direction. For easy notation, we introduce a fifth dummy dimension in s_t. In generating the path, we assume $v_t=6u_{t-1}$, this is to facilitate utilizing dependency relation between root and leaf in applying RBPF. The state is projected forward according to Eq. (14). In matrix A, S_s**randn()* denotes the noise added to the location, and $\psi(t)$ may return three possible values {1,2,3} (used as the index to V) according to probability matrix P_{vx}.

$$\begin{aligned} & s_t = A * s_{t-1} \\ \Leftrightarrow & \begin{bmatrix} x_t \\ u_t \\ y_t \\ v_t \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \Delta T & 0 & 0 & S_s * randn() \\ 0 & 1 & 0 & 0 & V(\psi(t)) * randn() \\ 0 & 6*\Delta T & 1 & 0 & S_s * randn() \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ u_{t-1} \\ y_{t-1} \\ v_{t-1} \\ 1 \end{bmatrix} \end{aligned} \qquad (14)$$

The observation at time t is Z_t, it is related to the state of the moving point by:

$$Z_t = C * s_t \Leftrightarrow Z_t = \begin{bmatrix} 1 & 0 & 0 & 0 & S_Z * randn() \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & S_Z * randn() \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ u_t \\ y_t \\ v_t \\ 1 \end{bmatrix}$$

where S_Z**randn()* denotes the noise added to location elements of observations.

After actual path is generated, the proposed RBPF and a regular PF are invoked to track this actual path, using the noisy path as the measurement. With the actual path as the ground truth, the performance of these two algorithms can be quantitatively compared. In the RBPF estimation, the root, leaf and observations are identified as:

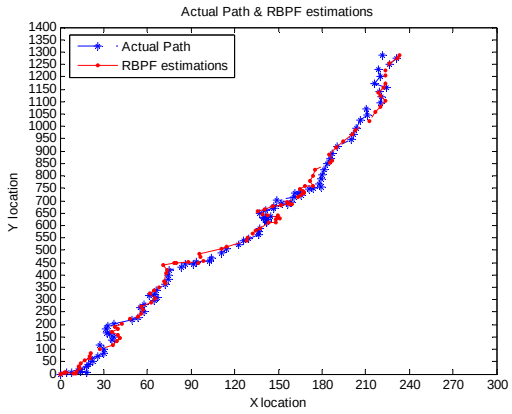
R_t= {u_t} L_t= {x_t, y_t, v_t} Z_t= {x_t, y_t}

The root is estimated by regular particle filter, and then the leaf variables are estimated using Kalman filter by assuming the dependency $v_t=5u_{t-1}$, which is different from the true dependency ($v_t=6u_{t-1}$) used in the path generation. This is intended to test if the proposed algorithm still

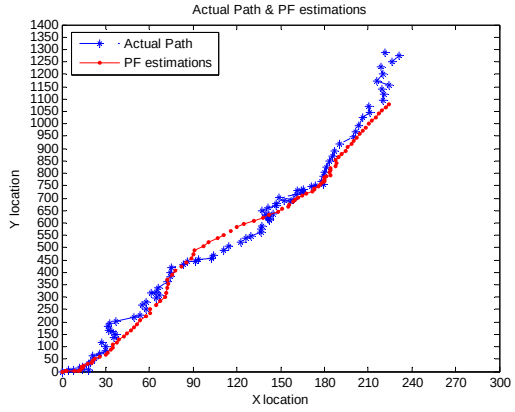
works fine even if the exact dependency model between the state variables is not available. The system and measurement model are the same with what have been presented in section 2.2 Eq. (4) ~ (5). Kalman prediction for the leaf variables is performed by:

$$x_t^{i-} = x_{t-1}^i + \Delta T * u_{t-1}^i$$
$$y_t^{i-} = y_{t-1}^i + \Delta T * 5 * u_{t-1}^i$$
$$v_t^{i-} = 5 * u_{t-1}^i$$

And Kalman update is the same as Eq. (11).

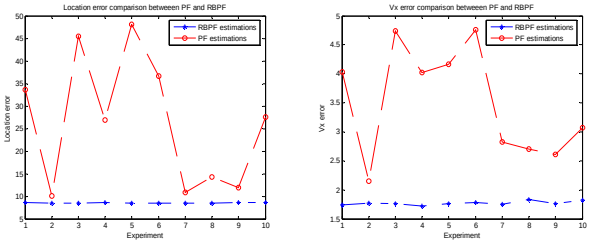


(a) Path estimated by proposed RBPF.



(b) Path estimated by a regular particle filter.

Figure 4. Paths estimated by two algorithms. Blue line marked by star is the true path; red line marked by dot is the path estimated by the RBPF or PF algorithm.



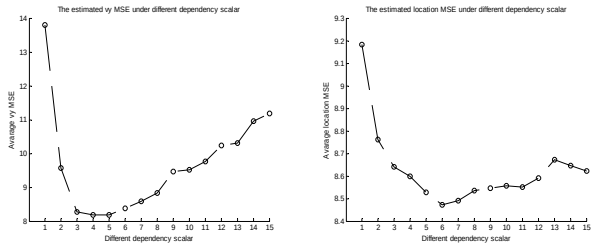
(a) Location error. **(b) Velocity error in x-axis**

Figure 5 Average error comparisons when running 10 times. The blue line with star is produced by RBPF, while the red curve with dot is from particle filter.

Fig. 4 shows the estimated paths by the RBPF and the regular particle filter. It can be seen that PF obviate from the true path in the curved part; In contrast, RBPF maintains good estimation all the way.

To obtain a statistical comparison between regular PF and RBPF, the experiment for both algorithms is repeated 10 times with different random initialization. 200 particles are used in all these experiments. In Fig. 5(a), the location errors for regular PF and RBPF are compared. In Fig. 5(b), the errors of *x*-axis velocity are compared. At a particular time *t*, the tracker's state is computed as the mean over all particles. Then the error (in terms of location or velocity) at time *t* is computed as the Euclidean distance between the estimated value and the ground truth value; the overall error is represented by the averaged Euclidean distance over all time steps. These two figures clearly show that the proposed RBPF significantly outperform regular PF.

As we mentioned in Section 2.2 algorithm step 1.b), the dependency relationship between leaf and root affects significantly the performance of our proposed RBPF algorithm. In practice, it is usually difficult to obtain an exact dependency model unless the surface normal between dominant plane and image plane is known, so it is necessary to analyze the effect of dependency model to the performance of proposed algorithm. For this purpose, we performed several simulations with the same setting as above. The true path is the same with the blue curve in Fig. 4(a), the dependency for the true path is vt=6ut-1. Then 15 different dependency models are used to estimate the actual path, with $v_t=S*u_{t-1}$, $S=1,2,\ldots 15$. We plot the estimated location and v_t error versus different dependency as shown in Fig. 6. We can tell that both v_t error (left figure *Y-axis*) and location error (right figure *Y-axis*) fluctuate within a small range when the dependency scalar (*X* axis) is around the true dependency 6; in contrast, the two errors are bigger when dependency is far form the true value. This result validates our conjecture that the proposed RBPF can maintain good estimation as long as the dependency relationship falls into a range around the true dependency.



(a) MSE for v_t **(b) MSE for location**

Figure 6. The MSE of the RBPF estimation with different assumptions about the dependency model.

3.2 Real data experiment

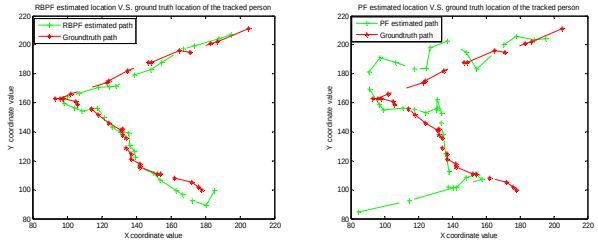
We have conducted extensive real data experiments to evaluate the performance of proposed RBPF algorithm

and compare it with a regular particle filter. In this section we present some sample results.

Figure 7.Outdoor human tracking. Left column: RBPF tracker. Right column: PF tracker. The green line is the gradient contour around the mean state. The frame number from top to bottom is 594,612,628,631 for both sequences.

One test scenario is outdoor human tracking. The test sequence for this scenario is from EC Funded CAVIAR project (http://homepages.inf.ed.ac.uk/rbf/CAVIAR/), which has also been used by PETS04 (Performance Evaluation for tracking and Surveillance in conjunction with ECCV2004). The video data were captured by a camera with wide angle lens along and across the hallway in a shopping centre. The ground truth data were also provided accompanying the video. The result in Fig. 7 reveals that when the person gets further from the camera and correspondingly, the size becomes smaller, the PF tracker would tend to deviate from the person and get stuck on the background. On the other hand, the RBPF algorithm maintains good tracking since it can effectively and efficiently update the scale change.

To quantitatively compare the performance of RBPF and PF based on Fig. 7 video sequence, we plot the estimated location (denoted by x and y coordinate in the image) by RBPF and PF verses the ground truth location, which is depicted in Fig. 8. It can be clearly seen that RBPF performs much better than PF.



(a) **(b)**

Figure 8. Comparison of RBPF and PF with ground truth. (a): RBPF estimation (green line) verses ground truth (red line). (b): PF estimation (green line) verses ground truth (red line). The video sequence is Fig. 7.

Figure 9. A person is moving in front of a shop window. Left column: RBPF tracker. Right column: PF tracker. The gradient contour is intentionally omitted so that the tracked region is easier to see with the interference of the reflection and the text on the window. In the original sequence (PETS2002 DATASET 1), the frame number is 75,79,84,87.

A more challenging human tracking test case is shown in Fig. 9. The task is made difficult by the reflections from the ground, the opposing window, occlusion due to the text on the window, etc. Nevertheless, the RBPF algorithm obtains good tracking, although the PF tracker was lost when encounters the text with different color.

Another test scenario in surveillance is vehicle tracking. One sample result is shown in Fig. 10, in which the car is involved in complex motion such as turning (rotation), translation, and large scale change. From Fig. 7 and Fig. 10, we notice that, the PF does not handle the tracking

well especially as the object becomes too small when it moves away from the camera. On the other hand, the proposed method is able to maintain the tracking in face of the dramatic scale change. In addition, our ongoing work shows that the variance of RBPF estimates is much less than that of PF estimates, and that only fewer particles would be required to achieve the same level of tracking accuracy.

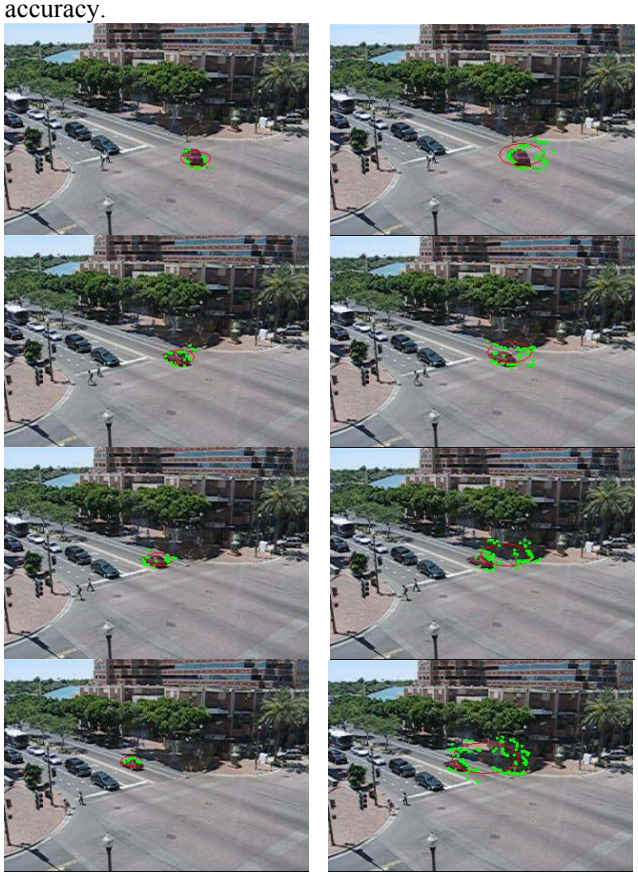


Figure 10.Vehicle tracking. Left column: RBPF tracker; Right column: PF tracker.

4. Conclusion and future work

In this paper, we proposed a tracking algorithm based on Rao-Blackwellised particle filter (RBPF). We discussed how the dependency between state variables imposed by typical surveillance application can be utilized to improve the efficiency and accuracy of a regular particle filter. Essentially, this is accomplished by partitioning the state variables into separate groups, with the linear parts being computed by Kalman filter and nonlinear part being estimated by particle filter. Experiments verify that the proposed method significantly outperforms a regular particle filter. As a future possibility, we are working on ways of automatically estimating the surface normal (see Fig. 1(a)) from the video so that a dependency model of the leaf variables on the root variables may be dynamically obtained. Additionally, the tracker can be avoided from being distracted by the static background through comparing the on-line gradient with a pre-stored background gradient map.

Reference

[1]A. Doucet, J. Freitas, and N. Gordon, *sequential Monte Carlo methods in practice*, Springer-Verlag, New York, 2001.

[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 of Signal Processing*, 2002, 50(2):174—188.

[3]N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation", Proc. Inst. Elect. Eng. F, vol. 140, no. 2, Apr. 1993, pp. 107--113.

[4]M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density", In *Europe Conference On Computer Vision (ECCV)*, 1996, pp. 343–356.

[5] S. Arulampalam, B. Ristic, "Comparison of the Particle Filter with Range Parameterised and Modified Polar ekf for Angle-Only Tracking", *Signal and Data Processing of Small Targets*, vol. 4048, 2000, pp. 288–299.

[6]K. Murphy and S. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks", *sequential Monte Carlo methods in practice*, Springer-Verlag, New York, 2001, pp.500-512.

[7]Z. Khan, T. Balch, and F. Dellaert, "A Rao-Blackwellized Particle Filter for Eigen Tracking", CVPR 2004.

[8]D. Schulz, D. Fox, and J. Hightower, "People Tracking with Anonymous and Id-sensors Using Rao-blackwellised Particle Filters", *In Intl. Joint Conf. on Artificial Intelligence* (IJCAI), 2003.

[9]N. de Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch and D. Poole, "Diagnosis by a Waiter and a Mars Explorer", in *Proceedings of the IEEE Special Issue on Sequential State Estimation*, 2003.

[10] C. Kwok, D. Fox, "Map-Based Multiple Model Tracking of a Moving Object", *RobuCup 2004*, pp18-33.

[11] T. Schon, F. Gustafsson and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models" *IEEE Transactions on Signal Processing*, 2004.

[12] T. Soma, K. Yosui and T. Mutsumoto, "Reconstructions and predictions of nonlinear dynamical systems by Rao-Blackwellised sequential Monte Carlo", IEEE International Conference on *Acoustics, Speech, and Signal Processing, 2003 Proceedings* (ICASSP '03).

[13] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision,* second edition, Cambridge University Press, March 2004, pp.220.

[14] K.Nummiaro, E. Koller-Meier, and L.V.Gool, "Object Tracking with an Adaptive Color-Based Particle Filter", *Proc. Symposium for Pattern Recognition of the DAGM*, pp.591-599, 2003.