

Chapter 16

Tracking Multi-Objects in Web Camera Video Using Particle Filtering

Yang Weon Lee

Abstract This paper presents new methods for efficient object tracking in web camera video sequences using multiple features and particle filtering. Particle filtering is particularly useful in dealing with nonlinear state space models and non-Gaussian probability density functions. We develop a multi-objects tracing system which considers color information, distance transform based shape information and also nonlinearity. We examine the difficulties of video based tracking and step by step we analyze these issues. In our first approach, we develop the color based particle filter tracker that relies on the deterministic search of window, whose color content matches a reference histogram model. A simple HSV histogram-based color model is used to develop this observation system. Secondly, we describe a new approach for moving multi-object tracking with particle filter by shape information. The shape similarity between a template and estimated regions in the video scene is measured by their normalized cross-correlation of distance transformed images. Our observation system of particle filter is based on shape from distance transformed edge features. Template is created instantly by selecting any object from the video scene by a rectangle. Finally, in this paper we illustrate how our system is improved by using both these two cues with nonlinearity.

Keywords Particle filter · Multi-target tracking · Condensation · Video image

16.1 Introduction

To track an object in a video sequence, the object or its features must be identified first [1]. An object is defined by the features it exhibits. To detect the motion of an object, the movement of the features must be tracked. After identifying the motion

Y. W. Lee (✉)

Department of Information and Communication Engineering, Honam University,
Seobongdong, Gwangsan-gu, Gwangju 506-714, South Korea
e-mail: ywlee@honam.ac.kr

of the features, a prediction model must be formed such that the position of the features can be predicted in the next frame. This paper considers edges for tracking since edges are the most prominent features of an object.

The motion of an edge is parameterized by (θ, u_f, u_b, d) , where θ is the angle made by the edge, u_f is the foreground velocity, u_b is the background velocity, and d is the perpendicular distance between the edge and the center of the block. The moving edge (or the motion edge) can be completely characterized by these parameters. It is assumed that the object, and hence the edge, moves with the foreground velocity. The foreground and background velocities help in modeling the occlusion and disocclusion effects due to the movement of the object. Tracking of the motion edge can now be re-defined as prediction of the motion edge parameters for the next frame using the information available from the previous frames. In order to predict and track the parameters, a motion model must be defined, followed by a prediction algorithm both of which are defined in the subsequent sections. In this paper, a square neighborhood of seventeen pixels in each dimension is used throughout the model.

16.2 Tracking Algorithm

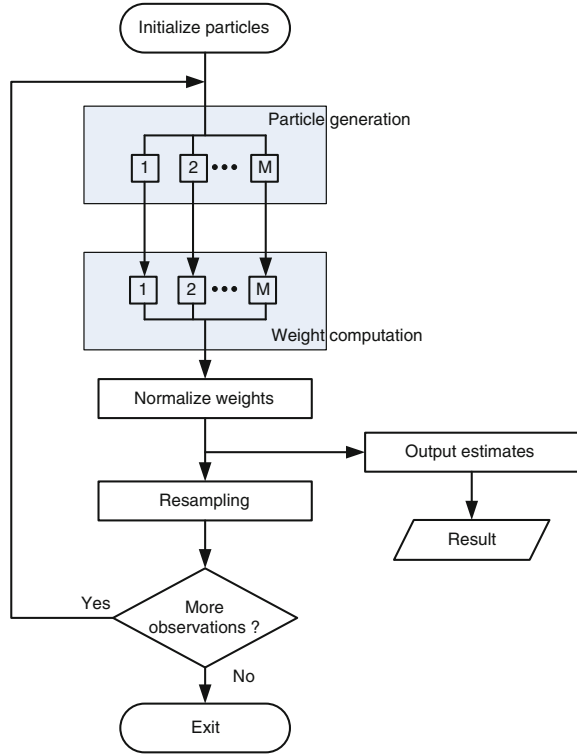
16.2.1 Particle Filter Algorithms

The particle filter approach to track object, also known as the condensation algorithm [2, 3] and Monte Carlo localization, uses a large number of particles to ‘explore’ the state space. Each particle represents a hypothesized target location in state space. Initially, the particles are uniformly randomly distributed across the state space, and for each subsequent frame, the algorithm cycles through the steps illustrated in Fig. 16.1:

1. Deterministic drift: particles are moved according to a deterministic motion model (a damped constant velocity motion model was used).
2. Update probability density function (PDF): determine the probability for every new particle location.
3. Resample particles: 90 % of the particles are resampled with replacement, such that the probability of choosing a particular sample is equal to the PDF at that point; the remaining 10 % of particles are distributed randomly throughout the state space.
4. Diffuse particles: particles are moved a small distance in state space under Brownian motion.

These results in particles congregating in regions of high probability and dispersing from other regions; thus the particle density indicates the most likely target states. See [4–8] for a comprehensive discussion of this method. The key strengths of the particle filter approach to localization and tracking are its scalability (the

Fig. 16.1 Process of particle filter calculation



computational requirement varies linearly with the number of particles), and its ability to deal with multiple hypotheses (and thus more readily recover from tracking errors). However, the particle filter was applied here for several additional reasons: It provides an efficient means of searching for a target in a multi-dimensional state space. It reduces the search problem to a verification problem, i.e., is a given hypothesis face-like according to the sensor information? It allows the fusion of cues running at different frequencies.

16.2.2 Application of Particle Filter for Multi-Object Tracking

In order to apply the particle filter algorithm to hand motion recognition, we extend the methods described by Huang [2]. Specifically, a state at time t is described as a parameter vector: $s^t = (\mu, \phi^i, \alpha^i, \rho^i)$ where μ is the integer index of the predictive model, ϕ^i indicates the current position in the model, α^i refers to an amplitude scaling factor and ρ^i is a scale factor in the time dimension.

(1) Initialization

The sample set is initialized with N samples distributed over possible starting states and each assigned a weight of $1/N$. Specifically, the initial parameters are picked uniformly according to:

$$\left. \begin{aligned} \mu &\in [1, \mu_{max}] \\ \phi^i &= \frac{1-\sqrt{y}}{\sqrt{y}}, y \in [0, 1] \\ \alpha^i &= [\alpha_{min}, \alpha_{max}] \\ \rho_i &\in [\rho^{min}, \rho^{max}] \end{aligned} \right] \quad (16.1)$$

(2) Prediction

In the prediction step, each parameter of a randomly sampled s_t is used with s_{t+1} determined based on the parameters of that particular s_t . Each old state, s_t , is randomly chosen from the sample set, based on the weight of each sample. That is, the weight of each sample determines the probability of its being chosen. This is done efficiently by creating a cumulative probability table, choosing a uniform random number on $[0, 1]$, and then using binary search to pull out a sample [1]. The following equations are used to choose the new state:

$$\left. \begin{aligned} \mu_{t+1} &= \mu_t \\ \phi_{t+1}^i &= \phi_t^i + \rho_t^i + N(\sigma_\phi) \\ \alpha_{t+1}^i &= \alpha_t^i + N(\sigma_\alpha) \\ \rho_{t+1} &= \rho_t^i + N(\sigma_\rho) \end{aligned} \right] \quad (16.2)$$

where $N(\sigma_*)$ refers to a number chosen randomly according to the normal distribution with standard deviation σ_* . This adds an element of uncertainty to each prediction, which keeps the sample set diffuse enough to deal with noisy data. For a given drawn sample, predictions are generated until all of the parameters are within the accepted range. If, after, a set number of attempts it is still impossible to generate a valid prediction, a new sample is created according to the initialization procedure above.

(3) Updating

After the prediction step above, there exists a new set of N predicted samples that need to be assigned weights. The weight of each sample is a measure of its likelihood given the observed data $Z_t = (z_t, z_{t1}, \dots)$. We define $Z_{t,i} = (z_{t,i}, z_{(t-1),i}, \dots)$ as a sequence of observations for the i th coefficient over time; specifically, let $Z_{(t, 1)}, Z_{(t, 2)}, Z_{(t, 3)}, Z_{(t, 4)}$ be the sequence of observations of the horizontal velocity of the left robot, the vertical velocity of the left robot, the horizontal velocity of the right robot, and the vertical velocity of the right robot, respectively. Extending Black and Jepson, we then calculate the weight by the following equation:

$$p(z_t | s_t) = \prod_{i=1}^4 p(Z_{t,i} | s_t) \quad (16.3)$$

$$p(z_{t,i} | s_t) = \frac{1}{\sqrt{2\pi}} \exp \frac{-\sum_{j=0}^{\omega-1} (z_{(t-j),i} - \alpha^* m_{(\phi^* - \rho^* j),i}^{\mu})^2}{2(\omega - 1)} \quad (16.4)$$

where ω is the size of a temporal window that spans back in time. Note that ϕ^* , α^* and ρ^* refer to the appropriate parameters of the model for the group of robots in question and that $\alpha^* m_{(\phi^* - \rho^* j),i}^{(\mu)}$ refers to the value given to the i th coefficient of the model μ interpolated at time $\phi^* - \rho^* j$ and scaled by α^* .

16.3 Experiment Result

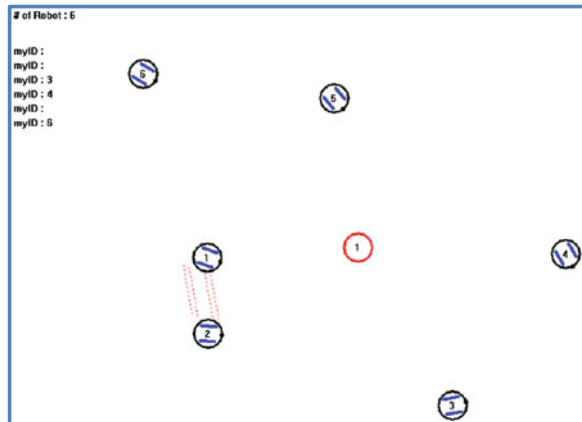
To test the proposed particle filter scheme, we used general web camera. The coefficient of particle filter is $\mu_{\max} = 2$, $\alpha_{\min} = 0.5$, $\alpha_{\max} = 1.5$, $\rho_{\min} = 0.5$, $\rho_{\max} = 1.5$ to maintain the 50 % scale. Also, the other parameters are $\sigma_{\alpha} = \sigma_{\phi} = \sigma_{\rho} = 0.1$. The value of ω in Eq. 16.4 is 10.

We first performed off-line experiments to evaluate the performance of the proposed dynamic object tracking method using video image as shown in Figs. 16.1 and 16.2. Out of the 80 object track path model sequences, 40 were randomly chosen for training, 16 from each class. The remaining track sequences were used to test. The overall recognition rate is 97.6 %. Following off-line experiments, we implemented a hand control interface based on dynamic moving object recognition in the real environments. The control interface worked well in dynamic environments.

To test the proposed particle filter scheme, we also used MATLAB (Math Works, Natick, MA, USA) and Visual Studio (Microsoft, Redmond, WA, USA). MATLAB was used for simulation of the particle filter and Visual Studio was used

Fig. 16.2 Experiment environment



Fig. 16.3 Object recognition**Fig. 16.4** Initial position of multi-object

to calculate the mutual localization of the inter objects. Through the experiment, we confirmed that the accuracy of multi-object localization using the particle filter is greater than localization using only sensor information. Therefore, it is necessary to use the particle filter to localize the object when there is no information except video information. This is shown in Figs. 16.3, 16.4, 16.5 and 16.6.

16.4 Conclusions

Tracking Multi-Objects in Web Camera Video Using Particle Filtering is an important technology for intelligent HCI. In this paper, we have developed the real time multi-object tracking using web camera. By applying particle filtering, we

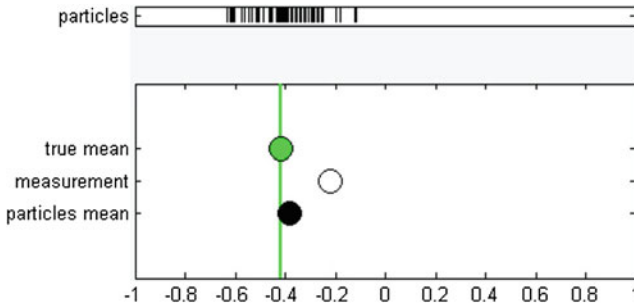


Fig. 16.5 Result of experiment by particle filter

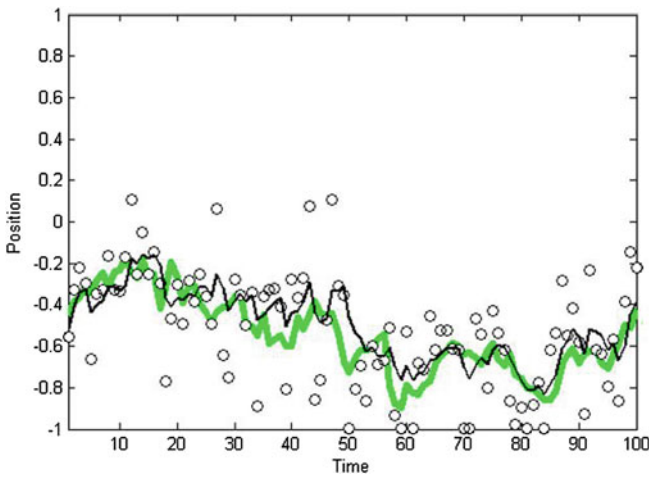


Fig. 16.6 Tracking result of experiment by particle filter

implemented a real time automatic object tracking system. Our approach produces reliable tracking while effectively handling rapid motion and distraction with roughly 75 % fewer particles. We also present a simple but effective approach for multi-object recognition. The tracking control interface based on the proposed algorithms works well in dynamic environments of the real world. Although the proposed algorithm is effective for multi-object tracking, further investigation should be conducted to verify its effectiveness in other tracking problems, especially the higher dimensional problems such as 3D articulated object tracking, as the number of particles required in high dimensional space is more prohibitive.

References

1. Lee YW (2011) Automation of an interactive interview system by hand gesture recognition using particle filter. *Int J Marit Inf Commun Sci* 9(6):633–636
2. Huang DS, Ip HHS, Law KCK, Chi Z (2005) Zeroing polynomials using modified constrained neural network approach. *IEEE Trans Neural Networks* 16(3):721–732
3. Kang CG, Kim DH (2011) Designing of dynamic sensor networks based on meter-range swarming flight type air nodes. *Int J Marit Inf Commun Sci* 9(6):625–628
4. Zhao ZQ, Huang DS, Jia W (2007) Palmprint recognition with 2DPCA + PCA based on modular neural networks. *Neurocomputing* 71(1–3):448–454
5. Liu J, Wu J (2001) *Multi-agent robotic systems*. CRC Press, Boca Raton
6. Yeo TK, Hong S, Jeon BH (2010) Latest tendency of underwater multi-robots. *J Inst Control Rob Syst* 16(1):23–34
7. Lee YW (2010) Implementation of code generator of particle filter. *Int J Marit Inf Commun Sci* 8(5):493–497
8. Lee YW (2008) Development of tracking filter for the location awareness of moving objects in ubiquitous computing. *Int J Marit Inf Commun Sci* 6(1):86–90