The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

Novel Algorithms for Video Object Tracking

CHEN ZHANG

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Philosophy

February 2013

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

_____(signed)

_____CHEN ZHANG____ (Name of Student)

Publications

- Zhang Chen and Wan-chi Siu, “Novel Multi-Step Recursive Sampling Strategy for Particle Filtering in object tracking”, proceedings, pp.1067-1070, the 11th International Conference on signal processing, Beijing, China, Oct. 21-25, 2012.
- Zhang Chen and Wan-chi Siu, “New Multi-Step Sampling with Adaptive Sampling Patterns in Particle Filtering for Tracking in Surveillance Systems”, proceedings, pp.288-289, 2013 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, USA, 11-14 Jan. 2013.
- Zhang Chen and Wan-chi Siu, “Embedded Adaptive Multi-Step Recursive Sampling Strategy in Particle Filtering for Visual Tracking”, submitted to Pattern Recognition, UK.

ABSTRACT

Visual tracking is a way to trace the position of an object in a video sequence. It has been widely used in film industry, interactive gaming, surveillance and auto-robotics. In this thesis, we will initially introduce some existing approaches for target representation, such as templates, color histograms and orientation histograms. These models are then used to examine the similarity between a target and possible candidates. Let us recall some more tracking techniques. The template matching approach takes a template (a sub-image) for searching within a pixel region with the least distance to it. The mean shift is a gradient-decent method which indicates that the candidate with the highest similarity stays at the zero-gradient position. The particle filter follows the sequential Monte-Carlo method to form a discrete expression of posterior pdf (probability distribution function of possible states given the known observations), where the expectation is designated as estimate.

Experiment results illustrate that the particle filter is much robust than other two approaches. However, the computation is much more complex subsequently. In the research work, we propose a new sampling strategy for particle filtering in object tracking. A particle filter formulation is not always able to generate effective samples at the step of importance sampling. It requires heavy computation in order to achieve acceptable tracking results. We propose a multi-step recursive sampling method to replace the direct importance sampling. This relies on the feedback derived from the resampling procedure. New particles are sampled recursively from the existing particles with high weights. After several iterations, particles become densely populated, and this new sampling policy gives significant contributions to achieve accurate position estimation.

Besides, in order to strengthen the density around the probable area, we have to generate a good sampling pattern for the proposed candidates. Usually, the particle transition vector is simulated by element-wise Gaussian samplings in the literature, i.e. the covariance is a diagonal matrix. However, it cannot react properly due to the changing target motion. Therefore, we may apply a 2D predictive transition vector to update the pattern of the multivariate Gaussian sampling. Before this, several predictive models are analyzed for comparison. The adaptive averaging model wins over others which include the AR model and Taylor series expansion. Then it has been decomposed into rotational angle and magnitude, which help update the sampling pattern. Finally, the adaptively updating sampling pattern is able to establish a more appropriate searching region which can reach the real state. Experimental results indicate that the proposed method reduces computation substantially and it also preserves good tracking results comparable to other algorithms in the literature.

Also the static histogram-based representation sometimes may not be robust enough in observing really good candidates and serving the long tracking process. So we have adopted the structural histogram set to settle the problem of low-confident similarity map. During tracking, every partial histogram in the set is assigned a dynamic weight according to its performance. The weight dominates its contribution to the final summed similarity result. Meanwhile, the referenced target model is dynamic via selecting one in a candidate set. It helps ensure the model to follow the real changing appearance of target. Furthermore, the elements in the set can also be replaced by a real adorable estimate in a fixed interval. Experimental results illustrate that the new model can effectively track objects with clutter background. Acceptable tracking results are achieved for long videos.

Acknowledgments

Hereby, I wish to thank, first and foremost, my supervisor, Prof. Wan-Chi Siu, for the invaluable instructions, patient guidance and continuous encouragement during my whole study. He also provides me thought-provoking advice on life and future career with his professional knowledge, wisdom, positive attitude and dedication in science.

I am also indebted to my many colleagues who supported me with their inspiring discussions and collaboration. Their expert knowledge and kindly encouragement benefits me significantly in my two-year research. Because of them, I succeed in overcoming many difficulties in the study. Also the clerical staffs in the department of Electronic and Information Engineering have given a great environment for me to work in. They efficiently help settle my inconvenience of paperwork.

In addition, it gives me great pleasure in acknowledging the support and help of my family, my girlfriend and other friends for their solid support and insistent encouragement. They have always stood by me and dealt with stuff for me, so I can concentrate on my study.

Table of Content

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Introduction of the visual tracking

Visual object Tracking based on camera is an important topic in video monitoring and surveillance. It aims to capture the movement of one specific object and locate where it stays from time to time. Nowadays, because of fast developing techniques and creative innovations, more applications have been exploited, such as automatic CCTV surveillance system, traffic control, navigation, augmented reality, video compression and editing, etc. It needs analysis of the video contents, which brings in complex computation in processing. Thus in this research project, we aims at finding the optimal tracking results under limited computation environment. However, video tracking is not only an estimation process for optimal results, but is an integrated composition of research work with object detection, classification related and localization. Note that if the object cannot be separated out from the background, the tracking object from the videos is even more difficult in real-time application. Figure 1.1 gives a functional diagram of an object tracker system. It has been divided into three major sections as input, Tracker and output. The input part needs the fluent video stream in sequence and the target model of the object. As video frames in processing, several candidates are generated. They travel through the procedures of feature extraction, similarity measurement. Then the similarity results give the confidence of localization. Sequentially, the target is found in the video frames. Finally in the output part, we collect the past states to form a trajectory of the object. Also as object varying in appearance, its features need to be feedback to the model and tracker to capture the really closet candidate.

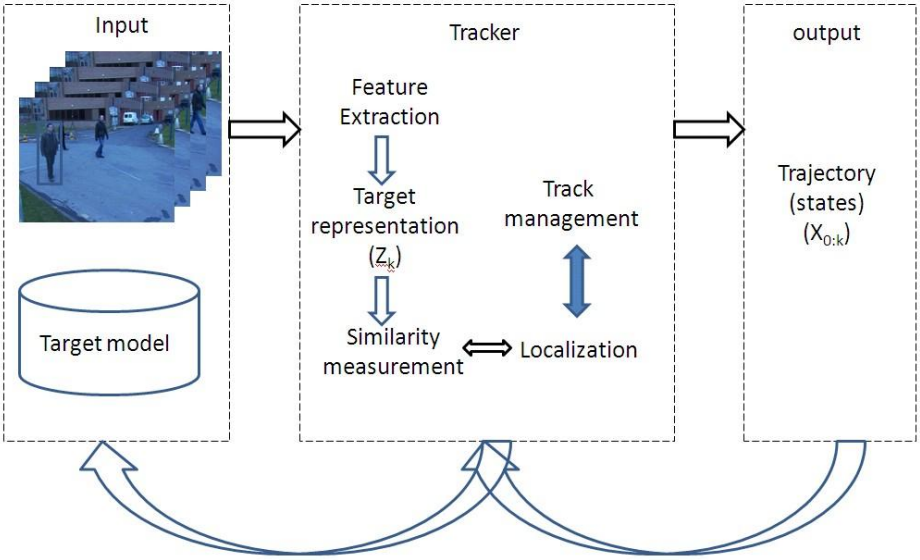


Figure 1.1 The structure of a video tracker

Let us start with some object identification and classification concepts for a tracking system. Features from one particular object have to be extracted. The area with the similar features then can be located as the possible object candidates. If multiple candidates are generated, the most similar one will be estimated as the final state of the object target. In other words, the features provide the evidence to classify the object candidates. A good formulation of processing the features ahead gives a good contribution to the representation of targets and candidates. However, not all features are successful in distinguishing between the real target in reference and candidates. Only the description of their characteristics has to be collected to form a set of records. Then, those records will act as the representatives for particular pixel regions, i.e. the image area owned by the target or other candidates. Therefore, target representation and feature extraction are tightly combined together. If different features provided, different representation forms will be obtained. As a result, these will significantly take influence on candidate evaluation, because they are used in similarity measures. Currently, it is still a big challenge for researchers exploiting out reliable features and a good

representation of target. If inappropriate features are extracted from images, the subsequent target representative will not be sufficiently unique and can be easily mistaken as part of background. The candidates are then incorrectly judged and the resultant estimate becomes a mistake. Equivalently, good features are desirable, which can effectively set up a proper description of the target. Candidates are therefore easily identified with great discrimination power. No doubt that the best candidate is absolutely most close to the real target. Finally, the estimate will approximate or equal to the current target state.

Also similarity measurement is essential in the tracking algorithms. It takes advantages of the representative of target model together with candidates, and computes similarities between them. These similarity values can be used to filter out the probable target and wrongly proposed candidates. Thus they are significant references for estimating the unknown state of target.

However, the useful similarity outputs rely on an effective candidate proposal. Candidates should be discriminated with each other and cover the possible region where target occurs. Then candidates are chosen with relatively higher possibilities. However, the candidate evaluation requires much more computation with heavy load to a real-time tracking system. Furthermore, the candidate generation and filtering process can also be regarded as a so-called localization procedure in the structure of visual tracking. The output is therefore a result of an estimated state for the object. Certainly it can include the tracking of multiple objects via processing each of them individually and taking advantage of some mutual relationship in different aspects. This part belongs to the tracking management.

In the following paragraphs, we will discuss furthermore the details of a complete tracking system. According to the met problems, we are motivated to settle them as the objectives.

1.2 Literature Review of the trends

1.2.1 Representation with reliable features for target and candidates

Object representation in a video frame is based on the pixel region covered by its image. The same pre-processing procedure is handled for special characteristics. However, they are hidden in pixels with confusing and complex intensities. So the concept of feature is involved to project the characteristics into different “dimensions”. Typically, the simplest feature is a template in frames, where the template is the image of the object. It has been widely used in the video coding systems. A template usually contains nearly the whole object image. However, it is easily corrupted by the noise in many circumstances [1, 2, 3]. In some papers [4], the template cooperates with the linear transformation matrix for perfect matching. Thus researchers have tried other methods to separate the features in different views, e.g. color, gradients or edges, scale, contour, motion and etc. [1, 5-7]. Histograms are therefore taken to collect features from different pixels to form a presentation in counting among bins. Color histograms [8, 9] and orientation histograms [10, 11] are popular tools for feature extraction. Color histograms encode pixel intensities into statistical distribution using bins. As not mixed with global information, the color histogram is robust in dealing with partial occlusion, rotation and scaling. Also the computation would be simplified with less data collected and processed. In chapter 2.1.2, we will explain its mechanism in details. Then the structural histogram [8, 12, 13] is considered in chapter 2.1.3 by referring more other features, especially, the spatial information. This solution is effective for high accurate similarity computation. But, the computational complexity is a few times to that of the single histogram.

Haar-like feature [14] has also been simply applied by researchers. The integral image can help create feature values using pixels with different patterns. Accordingly, the contents in the features can be collected to act as the representation of objects. Popular representations of objects include templates, histograms and vectors from different features.

1.2.2 Tracking algorithms

In recent decades, useful tracking algorithms also developed fast. Among all methods related, template matching [2-4] is the simplest and the most popular one, which has been widely used in video coding, image registration and pattern recognition. In visual tracking, one sub-image called template is extracted from the starting frame containing the pixel intensities. And in Chapter 2.2.1, the cost functions or score functions were derived to perform the comparison between the target template and the candidates in pixel region. The candidates are arranged to have a local searching region and to take full search or the advantage of fast searching algorithms. However, the simple template matching is relatively not satisfying in most cases, since object is changing with dynamic scenes. Also the template could mislead the tracking results, because pixel intensity values are easily corrupted by noises.

Mean shift [9, 15-18] is also usually adopted as a video tracking approach. Physically, it is an extension of Gradient descent method, as the increasing ramp becomes smoother and smoother. When the Gradient reaches zero, it comes to the top position which is considered as an estimate of its highest probability of occurrence. Normally, mean shift is used as one kind of clustering method [15]. However, since the similarity map is generated in a uni-modal distribution, the similarity could be recursively increasing until the motion stops. Some details on mean shift approach are described in chapter 2.2.2.

Then Bayesian filtering trackers [19] are totally different approaches from the previous approaches discussed so far. Under the assumption of Hidden Markov Model [20], they address the uncertainty problem by modeling the states and observations in one dynamic system with transitions and measurements. The goal of such filter is to estimate the posterior pdf where the object states probably occurring, given the known observations at current time. The estimation is performed recursively in two steps, namely prediction and updating.

The Kalman filter is one well-known Bayes filter [20]. Indeed, Kalman Filter needs more assumptions of linearity in state transition and Gaussian noises in systems. And the estimation process is a sort of analytical computation. However, the results are limited by these assumptions. And particle filter [18], in chapter 2.2.3, has no such assumption or limitation. Particle filter (PF) applies the sequential Monte-Carlo method [21, 22] to construct a discrete expression of posterior pdf (probability density function) given known past observations. The posterior probability distribution of the system is represented by a set of weighted random samples (called particles) and the prior distribution of the system is achieved by a Bayesian iterative evolution produced over these particles. The key is to use a number of discrete random samples (particles) to approximate the probability density function of random variables of the sample. Thus the expectation replaces the integral operation to achieve estimation of the system for minimum variance. With an increase in the number of particles the probability density function of the particles gradually approximate the probability density function thus the optimal Bayesian estimate is achieved.

The PF has also attracted recently the attentions from many researchers. The SIR (sequential importance resampling) particle filter, also known as Bootstrap filter, is most widely recognized as the standard particle filtering format [18], because it has simple and efficient

structure [22]. For the sake of convenience, the state transition probability is assumed to follow the proposal distribution, i.e. the pdf of self-proposed state transition in sequential importance sampling. Importance sampling is the significant sampling strategy adopted in the particle filtering. It uses the instances in the known distribution to describe a different distribution with weights. In practical tracking, the random walk transition model with multivariate Gaussian sampling is used in the algorithm to generate new particles/samples [23]. Indeed, the sampling process is not performed precisely following the exact state transition. Namely, the unknown distribution needs many samples in order to obtain precise description. Hence, it is unreliable for tracking using a small set of particles. Normally, a large amount of particles, which results in loss of simplicity and efficiency, are exploited in likelihoods to compute the weights as discrete coefficients of the posterior density.

Some recent works [18, 24-26] have succeeded on improving the proposal distribution by techniques, such as the unscented transform [24], extended Kalman filter [25] and mixture of Gaussians [26], which are closer to the prior transition density. However, they do not reduce the number of sampled particles and much computational cost is spent on processing particles separately. Since it may be ineffective for using randomly sampled particles, the mean shift [16] approach was proposed which can possibly optimize the moving positions and give better candidates. It not only results in better performance, but also reduces computations. Several papers have described its advantages by proposing similar approaches, i.e. the kernel particle filter (KPF) in [27], mean shift embedded particle filter (MSEPF) in [28]. Furthermore, the hybrid particle filter with mean shift (HY) [29] tracker is a typical example that uses mean shift to refine the particles, whilst its extended version is written in [30]. But it may have the drift problem after the mean shift process. Because the multi-modal similarity map has several

modes, particles can be misled to undesirable modes rather than the optimal state. Ref. [31] claimed that a compact association of PF and mean shift can effectively solve the drift problem. Before running the mean shift, the incremental Bhattacharyya coefficient [32] and matrix condition number are formulated to eliminate particles at positions with ill-posed conditions. The computational complexity is inevitably increased in settling this particular filtering operation. Moreover, block matching based motion estimation (ME) can be used for the prediction of target motion cue. In [33], the authors took advantage of this to optimize the transition distribution. Critically, this becomes effective after processing the ME operation. On the other hand, as motivated by multistage sampling in [34], MCMC(Markov Chain Monte Carlo) PF in [35] and Annealed PF in [36], the sampling may not involve any direct filtering operation or refinement of individual particles, but is done in stages or layers according to the feedback of previous samples. In [34], the proposal distribution is a continuous function that is initialized by unscented transform, and is then updated by an estimate of likelihood function from iterative sampling. The MCMC PF can also improve the efficiency of particles by Metropolis-Hastings (MH) algorithm [37] through acceptance and rejection in both burn-in and smoothing steps. However, the step number is relatively large in order to make particles converge to a high-likelihood region. The efficiency is not largely improved in realization. In the annealed PF [36], the simulated annealing sampling [38] is adopted for a layered discarding and redrawing scheme from the annealed weight computation. However, the annealing sampling is constrained by noises of the likelihood map. When an occlusion happens and there are not enough sampling layers, the results may drift far away easily. The final number of particles used for estimation is fewer than the replaced ones in the layers. Even though they are particles with high likelihood, the noisy likelihood map

can lead the estimation deviating from the optimal case. The posterior density becomes inaccurate subsequently. Thus it is inevitable that the estimate fluctuates heavily around the optimal state.

1.2.3 Transition model between states

As is known, the particle filter does not rely heavily on the transition prediction. However, an accurate transition prediction will still bring samples to the probable region with high density [18]. Then the weight extracted from each particle would lead to the most probable state by computing the expectation of the particle distribution with individual weight assigned. In the generic PF, we need not have the prediction but assume the particles transit randomly to the new vector at the next stage. By the way, particles are resampled at each stage to keep particles effective in case of sampling degeneration. In paper [39-41], some predictive methods, such as Taylor expansion and adaptive average transition model, have been proposed. And it needs further analysis whether their models are suitable to multiple video sequences with different situations. Besides, we also note that the conventional particle filtering approaches have little discussion on the sampling patterns. Most of them, e.g. [42-44], adopt the Gaussian sampling patterns with diagonal covariance matrix. However, they empirically set up fixed parameters for sampling patterns before implementation. Normally, these parameters are suitable only for one specific sequence. It is not robust in real object tracking. As the sampling pattern constrains the searching region and the samples density, the sparse density or a case with little overlapping between the target and searching region can lead to loss of tracking. In [45], Pan and Schonfeld noted this problem and addressed it by simultaneously adjusting the proposed variances. But they also neglected the mutual relation

between two dimensions of the vector in describing object position, i.e. the covariance may not be a diagonal matrix.

1.3 Objectives

The objectives of this research work are to develop an operational real-time video auto-surveillance system, combine with a well-reconstructed background extraction, form an accurate shape or type or structural identification for objects, and suggest a stable tracking process according to the real trajectory. Thus, we want to establish a new tracking method to handle single object tracking robustly and correctly. It is proposed to overcome difficulties in tough situations which are related to the changing scenes, deforming objects, occlusions and segmentation. Meanwhile it should be operated online while the video is playing. We mainly focus on modifying the particle filtering approaches to a simplified version by using fewer particles. The target can be well described with discriminative features and updated as object pose or environment changes. So a more stable tracking process is necessary for the object.

1.4 Organization of thesis

The rest of the thesis is organized as follows. In Chapter 2, we make a technical review on existing tracking algorithms, especially the particle filtering approaches for comparison. In Chapter 3, we proposed a novel multi-step recursive sampling strategy for particle filtering. Intuitively, sampling is done in steps in order to observe higher probable areas, which allows giving feedback and good convergence. Chapter 4 consists of the analysis on several predictive models for state transitions. And we embedded dynamic patterns in the new sampling strategy. The sampling pattern is updated according to the predicted transition vector which decreases the sampling scope effectively. In Chapter 5, we propose a new target

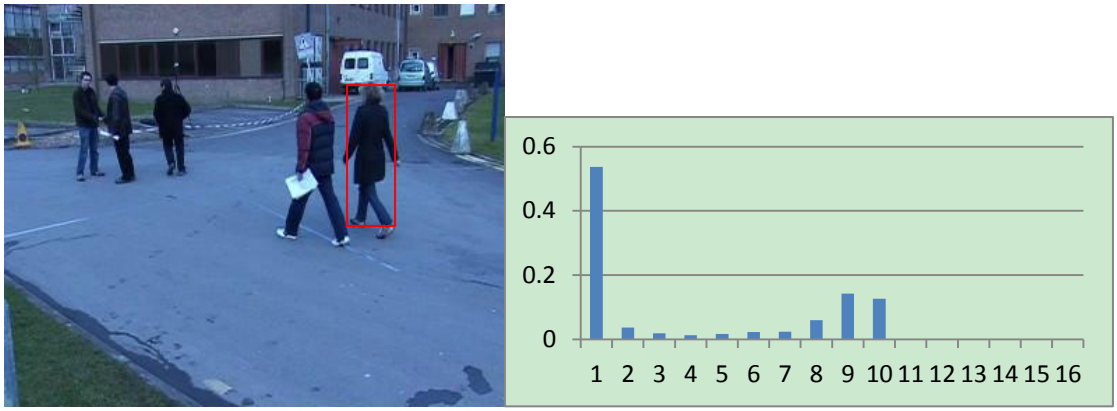
model as representation. A multi-part structural histogram set is formulated with corresponding partition weights. During the iterative tracking process, aiming at capturing varying object appearance, a new updating scheme is created to update the reference model of target. Chapter 6 concludes the research and discusses possible future research works in this area.

Chapter 2. Technical Review

2.1 Target representation and observation model

2.1.1 Templates

The most common target representation is the small template (Figure 2.1(a)), which is extracted from the frame with all pixels stored. Indeed, it is a small sub-image with all possible features which can be further exploited [2, 3]. Template matching is considered as the simplest tracking approach among the recent proposed ones. We need to match the template to an image frame, where template is a sub-image that contains the essential features of the target to be tracked. And according to real exploration, the most matched part in the frame can determine the right time position of the target. Furthermore, the searching candidate is transformed to cooperate with the referenced template. Typically, the transformation can contain translation, scaling and even affine transformation. Interpolation is allowed for better scaling to integer pixel coordinates. If its simplest target model is adopted without any further processing, the pixel intensities will mislead to obtain a wrong score of the comparison between the template and the candidates if noises are considered, such as the clutter background, partial occlusion, and body rotation in 3D circumstance.



(a) (b)

Figure 2.1(a) color template (pixel region in the red box) (b) color histogram (encoded with 16 bins)

2.1.1 Color histograms

To track an object in a video sequence, the special features of the target shall be obviously different from the other area in the frame picture. This approach of particle filter mainly relies on the color context, i.e. the correlated information from pixel intensities in color, which captures the important features apart from the background. Also color distribution in the target model normally achieves robustness against the rotation, partial occlusion, and the transition. Meantime, the color distribution (Figure 2.1(b)) is multiple bins in the form of a histogram with density where bin order can been seen as “quantized” pixel intensity. Normally, we applied the histograms using eight or more bins for every color channel.

As an initial stage, the target model built using a rectangular window, and the YUV or RGB values of which are counted in histograms for density calculation. In the following iteration computation, candidates with the same shape are used to match the reference one in similarity measurement.

To increase the reliability of the color distribution, when the boundary pixels are occluded by other objects or belong to the background, smaller weight coefficients are assigned to the pixel further away from the center position of the particle. Here, we have

$$k(r)=\begin{cases}1-r^2, & r<1\\ 0, & otherwise\end{cases}, \tag{2.1}$$

where r is the normalized distance from the region center. Thus, we increase the robustness against occlusion and background misleading [9].

The color histogram $p_y=\{p^u(y)\}_{u=1\ldots m}$ of each particle region at center position y is deduced as a set of densities of m bins. For example, in the u[th] bin, density $p^u(y)$ is expressed by

$$p^u(y)=f\sum_{i=1}^{I}k(\frac{\|(y-x_i)\|}{\sqrt{H_x^2+H_y^2}})\delta[h(x_i)-u] \tag{2.2}$$

where I is the number of pixels in the region, x_i is position of the particular i[th] pixel, $h(x_i)$ output the i[th] pixel's bin order, u is the current bin number from 1 to m, $\delta(.)$ output 1 if input is zero, otherwise 0 is the output, H_x and H_y are the half-size width and height of the model region, and f is the normalizing constant

$$f=\left(\sum_{i=1}^{I}k(\frac{\|y-x_i\|}{\sqrt{H_x^2+H_y^2}})\right)^{-1} \tag{2.3}$$

2.1.2 Structural histograms

The target window can be split into regions with regular rigid blocks (Figure 2.2) and we can compute multiple histograms (per rectangular window). Let the window center be the location of each block; in other words, the blocks are connected symmetrically around the center of window. This solution is effective for a specific application with clutter background.

And such a combined system of histograms forms a structural histogram [10, 46]. Indeed it can resolve the problems of the robustness and flexibility.

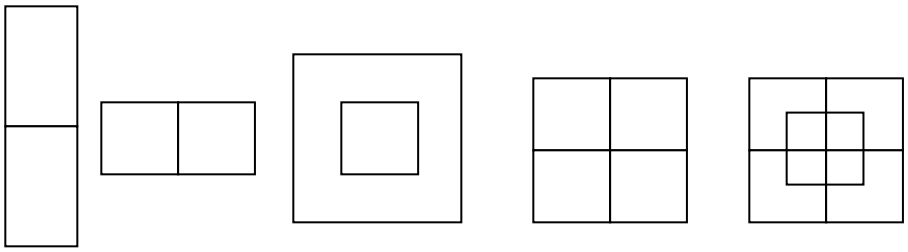


Figure 2.2 subdivisions for generating spatially sensitive appearance (e.g. histogram-based) representations.

Also for the same objective, the spatial information can be used for associating each bin of the histogram with the first two spatial moments of the pixel coordinates of the corresponding colors. Spatiogram therefore can be interpreted as a GMM (Gaussian Mixture Model) in the 2D Gaussian distribution and weighted by the bin of the histogram [46, 47].

2.2 Recent tracking algorithms

2.2.1 Template matching

Normally, the template matching can be defined as a method of parameter estimation. And it can help deduce the position of the template. A template is collected as a discrete function of pixel $T_{x,y}$, where x and y are the coordinates in the template window. If the noises of surrounding pixels follow the static rule, for example, the Gaussian distribution, they will have a mean value of zero and the standard deviation σ. So the probability of the pixels which varies among the templates can be defined as

$$p_{i,j}(x,y) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2}\left(\frac{I_{x+i,y+j} - T_{x,y}}{\sigma} \right)^2 \right], \tag{2.3}$$

where i and j are the coordinates of target candidate in the current frame, and $I_{x+i,y+j}$ is the pixel in the candidate corresponding to $T_{x,y}$.

Since the template is composed of many independent pixels, likelihood function can be derived as the exponential output of the L2 difference, which is

$$L_{i,j} = \left(\frac{1}{\sqrt{2\pi\sigma}}\right)^n \exp\left[-\frac{1}{2}\sum_{x,y\in T}\left(\frac{I_{x+i,y+j} - T_{x,y}}{\sigma}\right)^2\right]. \tag{2.4}$$

So the most matched part generally owns the maximum output of the likelihood function. To reduce the complexity, the logarithmic form can be interpreted for a reformed equation. Thus it can be defined as

$$L_{i,j} = n\left(\frac{1}{\sqrt{2\pi\sigma}}\right) - \frac{1}{2}\sum_{x,y\in T}\left(\frac{I_{x+i,y+j} - T_{x,y}}{\sigma}\right)^2. \tag{2.5}$$

We can observe that the second formulation emphasizes much more at the minimum value. Consequently, the maximum value of the likelihood is able to be generated subsequently. Indeed, the squared error controls the best match in the frame. Also it could be simplified by cancelling negligible terms to be

$$(i, j) = \arg\min\left(\sum_{x,y\in T}(I_{x+i,y+j} - T_{x,y})^2\right). \tag{2.6}$$

This cost function is capable in the tracking process. The minimum score obtained helps determine the optimal position of target. Still, eqn (2.6) can be modified in L1 norm

$$(i, j) = \arg\min \left(\sum_{x,y \in T} \left| I_{x+i,y+j} - T_{x,y} \right| \right) \tag{2.7}$$

where eqn (2.7) is the simplified version for pixel comparison of template.

Furthermore, the cross-correlation between the template and the current frame pixels can be maximized to obtain the best match. However, as $\left[\sum_{x,y \in T} (I_{x+i,y+j} T_{x,y}) \right]$ varies with position and is dependent on the size of the template and the image conditions. Thus an implementation of cross-correlation should be generally normalized.

Therefore, the equation is derived and modified as

$$CC = \frac{\sum_{x,y \in T} \left(I_{x+i,y+j} - \bar{I}_{i,j} \right) \left(T_{x,y} - \bar{T} \right)}{\sum_{x,y \in T} \left(I_{x+i,y+j} - \bar{I}_{i,j} \right)^2} \Rightarrow \frac{\sum_{x,y \in T} \left(I_{x+i,y+j} - \bar{I}_{i,j} \right) \left(T_{x,y} - \bar{T} \right)}{\sqrt{\sum_{x,y \in T} \left(I_{x+i,y+j} - \bar{I}_{i,j} \right)^2 \left(T_{x,y} - \bar{T} \right)^2}}, \tag{2.8}$$

where CC means the cross-correlation coefficients, $\bar{I}_{i,j}$ is the mean of the pixels value in the frame of the same size as the template window. $\bar{T}$ is the mean of the pixels of the template.[3]

All the equation (2.6), (2.7) and (2.8) can be taken to compute the possible position by searching the minimum or maximum output.

Then such approach will result in the problem of searching for the right corresponding position according to some deterministic procedures. Well, the searching methods related to the motion estimation could perform a great function in fast searching rather than the full searching approach.

2.2.2 Mean-Shift

It is a nonparametric statistical method for clustering and seeking the proper center position. And it also performs an iterative procedure that shifts each data point to the average of data points in its neighborhood [16, 17]. The dissimilarity between the target model (its color distribution) and candidates are expressed by a metric derived from the Bhattacharyya coefficient [16]. Therefore, the mean-shift method could be useful in clustering, mode seeking, probability density estimation, tracking, etc.

2.2.2.1 Kernel functions

The total algorithm brought in the kernel function that indicates how much the guessed value contributes to the estimation of the mean.

Generally, two main kernel functions are related in the real implementation.

Epanechnikov kernel:

$$K_E(x) = \begin{cases} \frac{1}{2} c_d^{-1}(d+2)(1-\|x\|^2) & if \|x\| < 1, \\ 0 & otherwise \end{cases} \tag{2.9}$$

where c_d is the volume of the unit d-dimensional sphere, x is the position vector.

Multivariate normal kernel:

$$K_N(x) = (2\pi)^{-d/2} \exp(-\frac{1}{2}\|x\|^2) \tag{2.10}$$

And the profile of the kernel should be introduced for later computation for convenience, which $k:[0,\infty) \to R$ such that $K(x) = k(\|x\|^2)$.

Kernel gradient:

Denote $g(x) = -k'(x)$; assume the derivative of k exists for all $x \in [0, \infty)$. The kernel function G(x) then can be defined as

$$G(x) = Cg(\|x\|^2), \tag{2.11}$$

where C is a normalization constant.

In reality, the g(x) is the necessary profile for mean shift procedure, since others only help define the density estimates.

2.2.2.2 General mean shift

The multivariate kernel density estimator $\hat{f}(x)$, with kernel K(x) and window radius (bandwidth) h, is computed in the form of the function given a set of n points $\{x_i\}_{i=1...n}$ in the d-dimensional space. We aim at finding the densest position in the distribution of points. Therefore, the density estimator at position *x* is established as the mean of kernel weights. They are assigned with distances from points to the designated position *x*.

$$\hat{f}(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{nh^d}\sum_{i=1}^{n} k\left(\left\|\frac{x - x_i}{h}\right\|^2\right), \tag{2.12}$$

Then, by converting the estimate of the density to the gradient form, we have

$$\begin{aligned}\hat{\nabla} f_k(x) \equiv \nabla \hat{f}_k(x) &= \frac{2}{nh^{d+2}}\sum_{i=1}^{n}(x - x_i)k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \\ &= \frac{2}{nh^{d+2}}\sum_{i=1}^{n}(x - x_i)g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)\end{aligned}$$

$$= \frac{2}{nh^{d+2}} \sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) \times \left[\frac{\sum_{i=1}^{n} x_i \cdot g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x\right], \tag{2.13}$$

where $\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)$ is assumed to be a nonzero function.

Note that the derivative of Epanechnikov profile is a uniform profile, which is

$$g_E(x) = -c_d^{-1}(d+2). \tag{2.14}$$

The derivative of a Gaussian profile remains a normal form, which is

$$g_N(x) = -\frac{1}{2\sqrt{(2\pi)^d}} e^{(-\frac{1}{2}x)} \tag{2.15}$$

Therefore, the mean shift vector M_G with kernel G(.) in eqn (2.14) or eqn (2.15) is

$$M_G(x) \equiv \frac{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) x_i}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x. \tag{2.16}$$

Also the density estimator $\hat{f}_G(x)$ at x with kernel G(.) can be defined as

$$\hat{f}_G(x) \equiv \frac{C}{nh^d} \sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right), \tag{2.17}$$

where C is a normalization constant. This definition helps to simplify the Motion vector expression in eqn (2.16).

And the equation can be simplified as

$$M_G(x) = \frac{Ch^2 \hat{\nabla} f_K(x)}{2 \hat{f}_G(x)} \tag{2.18}$$

It could be declared as an estimate of the normalized gradient of $\hat{f}_k$. Therefore, it can be used to obtain the estimate of $\hat{f}_k$.

Since the kernel has a monotonic decreasing profile, the approximated $\hat{f}_k$ will accordingly become a convergent sequence. Thus, the maximum value of the density estimator can be observed by recursive calculation. The Mean-shift vector in that case also converges to zero through mathematical proof.

When the mean-shift algorithm is applied to the video tracking process, the model relies on color distribution with histogram in color probability density. Thus the Bhattacharyya coefficient is taken in charge of the comparison with the target candidate of the object model.

2.2.2.3 Target model

To track an object in a video sequence, the special features of the target shall be obviously different from the other area in the frame picture. The target model is established by describing the histogram of pixel density in the selected object region. The description of target relies on the color context to apply capturing of the important features apart from the background. Also color distribution in the target model normally achieves robustness against the rotation, partial occlusion, and transition.

Meantime, the color distribution can be separated into m-bins to form specific histograms with density. In one implementation example, we applied the histograms using 8*4*4 bins for YUV values.

As in the color representation, partial occlusion happens sometimes. The probability of the color where the pixel belongs shall be assigned a weight, which is smaller for further location from the center of the kernel. It increases the robustness of the estimation, since the peripheral pixels are the least reliable, because of the influence of the occlusions or background.

In our implementation, the features of objects in the first frame would be extracted into target model establishment, such as

$$q^u = f\sum_{i=1}^{I} k\left(\left\|\frac{(y_c - x_i)}{h}\right\|^2\right)\delta[b(x_i) - u] \tag{2.19}$$

where y_c is the center position of the target (also the kernel); u is order of the bins; $\{x_i\}_{1...I}$ is the set of pixel positions in the rectangular target region; $b(x_i)$ exports the bin number of that particular pixel; f is the normalization constant.

Similarly, the target candidates, centered at y in the current frame, are described by that equation.

2.2.2.4 Distance minimization

Assume that the target will not charge a lot (stand similar) in the video sequence. Then, the most probable location of the target would always contain the minimum value in difference. As in mean-shift algorithm, it is aimed to maximizing the Bhattacharyya coefficient

equivalently. Firstly, the search of the new target position starts at the estimated location y_0 of the target in the previous frame. The color distribution $\{p(y_0)^u\}_{u=1...m}$ at the right position in the current frame has to be computed. Then to obtain the most similarity in the Bhattacharyya coefficient, it could be expanded by the Taylor series, such as

$$\begin{aligned}\rho[p(y),q] &\approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{p(y_0)^u q^u} + \frac{1}{2}\sum_{u=1}^{m} p(y)^u \sqrt{\frac{q^u}{p(y_0)^u}} \\ &= \frac{1}{2}\sum_{u=1}^{m}\sqrt{p(y_0)^u q^u} + \frac{C_h}{2}\sum_{i=1}^{n_h}\omega_i k\left(\left\|\frac{(y-x_i)}{h}\right\|^2\right)\end{aligned} \tag{2.20}$$

where

$$\omega_i = \sum_{u=1}^{m}\delta[b(x_i-u)]\sqrt{\frac{q^u}{p(y_0)^u}}\,. \tag{2.21}$$

As the first term in equation (2.20) is constant, the second one shall be maximized. It represents the density estimate with the particular kernel profile and weight of ω_i. The mean shift algorithm can be carried out iteratively to converge to the maximum value of Bhattacharyya coefficient. The resultant mean-shift vector describes the right motion of y_0 to the next location in the current frame.

2.2.2.5 Algorithm development

Mean-shift procedure for tracking

Table 2.1 Mean-Shift Algorithm

Given the target model $\{q_u\}$ and location y_0 of target in the previous frame:
(1) Initialize location of target candidate at the same position as y_0.

(2) Compute $\{p(y_0)^u\}_{u=1\ldots m}$, and $\rho[p(y_0), q]$.

(3) Compute weights, i= 1, …, nh.

$$\omega_i = \sum_{u=1}^{m} \delta(b(y_i - u) \sqrt{\frac{q_u}{p_u(y_0)}}$$

(4)Apply mean shift: compute new location z as

$$y = \frac{\sum_{i=1}^{n_h} \omega_i g\left(\left\|\frac{y_0 - y_i}{h}\right\|^2\right) y_i}{\sum_{i=1}^{n_h} \omega_i g\left(\left\|\frac{y_0 - y_i}{h}\right\|^2\right)}$$

Derive $\{p(y)^u\}_{u=1\ldots m}$, and $\rho[p(y), q]$

(4) While $\rho[p(y), q] < \rho[p(y_0), q]$, do $y \leftarrow \frac{1}{2}(y + y_0)$.

(5) If $\|y - y_0\|$ is less than 4, stop. Else, set $y_0 \leftarrow y$ and jump to step (1).

2.2.3 Introduction of Bayesian filtering framework

The tracking process can also be seen as the estimation of the unknown object state given current view in global scope. It can be modeled into the Bayesian filtering system. The dynamic system can be modeled with two equations in state transition and measurement.

$$\begin{cases} x_k = f_k(x_{k-1}, v_{k-1}) \\ z_k = h_k(x_k, n_{k-1}) \end{cases}, \tag{2.22}$$

where f_k is the evolution function which is used to introduce the (k-1)th state to next stage; h_k is the measurement function which is used to observe the state from existing image pixels; x_k and x_{k-1} are the current and previous states of object which declare position or appearance of the object; z_k and z_{k-1} are the kth current and previous measurement which is seen from the

existing knowledge, such as the object model for reference; v_{k-1} and n_{k-1} are the state noise and measurement noise (usually not Gaussian variables)[48, 49].

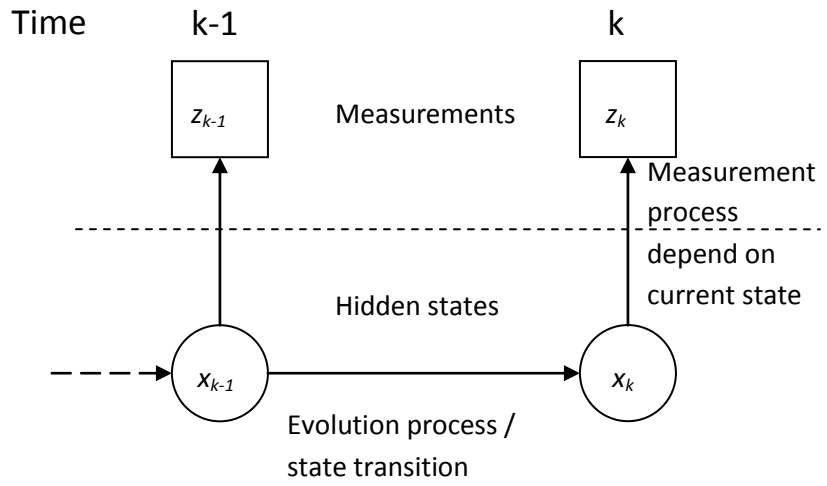


Figure 2.3 the diagram of a Bayesian filtering system

As shown in the flowchart, to estimate the real state x_k, the posterior probability should be obtained according to the Bayesian approach.

Generally, this approach is taken to compute the expectation of the state series $x_{0:k}$ using the formula:

$$E[f(x_{0:k})] = \int f(x_{0:k}) p(x_{0:k} \mid z_{1:k}) dx_{0:k} \tag{2.23}$$

where $x_{0:k}$ means a list of states from frame 0 to frame k, which actually stands for some mathematical results obtained via the history of the object going through stages 0 to k (stages usually refer to different time in this investigation). z_k is the observation at time k, which is usually the result of some measurement or deduction from the measured results, i.e., the target model in most cases. $z_{1:k}$ means a list of observations from time 1 to time k. In visual tracking process, $f(x_{0:k})$ output the historical locations of the object. So we need integrate all possible states at k stages to compute the expectation as the estimate. This is a general expression of

tracing object trajectory. We have object models $z_{1:k}$ in k frames. And we measure all individual candidates with their probability of occurring instances. It is a test to observe the confidence level whether candidate matches the model and possibility of grapping that hidden state. And the posterior density $p(x_{0:k} \mid z_{1:k})$ shows the probability of such state series given all existing observations.

Therefore, the posterior pdf was established to estimate the distribution of the states of the target given all the known previous measurements $z_{1:k}$. For simplicity, we will not estimate the trace of states $x_{0:k}$. Instead, every stage will be processed to observe its corresponding state.

Let us recall x_k is the target state at time k and $z_{1:k}$ denotes a set of past k observations. x_0 is the prior knowledge, z_k is the observation at time k, whereas in this case z_0 has no meaning. $x_{0:k}$ represents the sequence of the hidden states which is modeled by the Markov process with initial distribution $p(x_0)$ and transition function $p(x_k|x_{k-1})$. The observations $z_{1:k}$ are assumed to be conditionally independent when the state sequence $x_{0:k}$ exists. The marginal distribution $p(z_k|x_k)$ [18,19] indicates the likelihood function between the observation and state. In a video tracking process, we compute the expectation of the k^{th} state as the estimated result using the simplified formula:

$$E[f(x_k)] = \int f(x_k)\, p(x_k \mid z_{1:k})\, dx_k \tag{2.24}$$

where E[.] is the expectation of the input distribution, $f(x_k)$ is the general function that can output the parameters of the possible state, $p(x_k \mid z_{1:k})$ is the posterior density function which estimates the probability distribution of the target state given the known previous measurements $z_{1:k}$.

Practically, the tracking process works iteratively after the initial state, with the initial pdf $p(x_0|z_0) \equiv p(x_0)$.

Suppose the posterior density $p(x_{k-1}|z_{1:k-1})$ at last stage is available.

Prediction: If the initial distribution of the target is known, recursively, the predicted distribution of the next state can take the following equation:

$$p(x_k \mid z_{1:k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid z_{1:k-1}) dx_{k-1}, \quad (2.25)$$

where the prior density $p(x_k|z_{1:k-1})$ gives the prediction and the density $p(x_k|x_{k-1})$ declares the state transition. This prediction process should have the integral of all possible transitions from any cases of state x_{k-1}.

Update: In the update, we compute the posterior pdf from the predicted prior pdf and newly observation according to the Bayesian rule.

$$p(x_k \mid z_{1:k}) = \frac{p(z_k \mid x_k) p(x_k \mid z_{1:k-1})}{p(z_k \mid z_{1:k-1})}, \quad (2.26)$$

where $p(z_k|x_k)$ is the Likelihood given observation z_k, and the denominator is a normalizing constant. It has been applied as effective estimation of the hidden state. The Kalman filter is one extended application for a linear, discrete-time dynamical system. This filter predicts the state at some time and then obtains feedback in the form of (noisy) measurements. Thus, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations [25]. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, i.e. incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. In other way, the time update equations can be thought as a predictor, while the measurement update equations can be thought of as a corrector. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems.

The particle filter is another extended application of the Bayesian filtering framework. It is a deduced model for tracking object from a video sequence based on the sequential Monte Carlo methods. It is a hypothesis tracker, which approximates the filtered posterior distribution by a set of weighted particles.

2.2.4 Particle filter

2.2.4.1 Theory of particle filtering

For particle filter, the posterior probability density function (pdf) $p(x_k \mid z_{1:k})$ is characterized by a set of N weighted samples or particles, namely, $\{x_k^i, \omega_k^i\}_{i=1}^N$, where *i* is index for the particles, ω_k^i is the weight of particle x_k^i and is usually normalized as $\sum_i \omega_k^i = 1$. Then, practically, the posterior density [18] at state k can be approximated as

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N} \omega_k^i \delta(x_k - x_k^i), \tag{2.27}$$

where $\delta(.)$ command this formula to be expressed only by those discrete particles.

According to the principle of importance sampling, the sequential distribution of states $p(x_{0:k}|z_{1:k})$ can be obtained by the weighted importance density. Sample x_k^i is drawn with the importance density $q(x_{1:k}|z_{1:k})$. Subsequently, the weight can be defined, according to the importance sampling,

$$\omega_k^i \propto \frac{p(x_{0:k}^i \mid z_{1:k})}{q(x_{0:k}^i \mid z_{1:k})}. \tag{2.28}$$

where the particle weight ω_k^i is computed as the ratio between the instant probability density $p(x_k^i \mid z_{1:k})$ and the proposed importance density $q(x_k|z_{1:k})$.

Then, relying on eqn(2.28), the estimated state $\hat{x}_k$ is expressed as

$$\hat{x}_k = \sum_{i=1}^{N} \omega_k^i x_k^i . \tag{2.29}$$

Basic particle filter algorithm:

For i = 1: N

- Draw $x_k^i \sim q(x_k^i \mid x_{k-1}^i, z_k) = p(x_k \mid x_{k-1}^i)$.
- Assign weight to the i[th] particle, ω_k^i, according to the formula (2.29).

End For

Actually, the equation (2.28) could be further devised, since assuming the particle transition distribution (importance density), i.e. $q(x_k^i \mid x_{k-1}^i, z_k)$ approximates to the prior state transition $p(x_k \mid x_{k-1}^i)$ for each particular state, eliminating the two elements from the equation will not influence the proportional characteristic. And finally it becomes

$$\omega_k^i \propto \omega_{k-1}^i p(z_k \mid x_k^i), \tag{2.30}$$

where the $p(z_k \mid x_k^i)$ is the likelihood density, which shall be obtained through comparing the similarity of the features between target and each particle.

2.2.4.2 Systematic resampling

After several iterations of tracking process, except a small portion of particles, weights of the remaining particles is negligible. Thus a large number of computations are wasted in updating particles with small weights. Sometimes even only one left with acceptable weight performs as the effective particle and weight of others approximate to zero, which result in a degradation of the distribution. Resampling method can effectively reduce the particle

degradation. The general idea is that low-weight particles are removed, and then the vacancy is filled by concentration to the high-weight particles.

And it is time to take the resampling method to solve the degeneracy problem. Basically, we need to eliminate the low weight particles and concentrate on particles with large weights. It involves the generation of a new set of particles by resampling N times from the approximated posterior density. The new weights would be entirely reset to 1/N for all the particles.

Generally, we apply the most preferred resampling algorithm called systematic resampling.

Table 2.2 Algorithm of Systematic Resampling

$[\{x_k^{j*}, \omega_k^j, i^j\}_{j=1}^N] = \text{RESAMPLE}\ [\{x_k^i, \omega_k^i\}_{i=1}^N]$

- Initialize the CDF: $c_1 = 0$
- FOR *i = 2: N*
  - Construct CDF: $c_i = c_{i-1} + \omega_k^i$
- END FOR
- Start at the bottom of the CDF: *i = 1*
- Draw a starting point: $u_1 \sim U[0, N^{-1}]$
- FOR *j = 1:N*
  - Move along the CDF: $u_j = u_1 + N^{-1}(j-1)$
  - WHILE $u_j > c_i$
    - *i = i + 1*
  - END WHILE
  - Assign sample: $x_k^{j*} = x_k^i$
  - Assign weight: $\omega_k^j = N^{-1}$
  - Assign parent: $i^j = i$
- End FOR

2.2.4.3 Color-based particle filter

Compared with the standard SIR particle filter algorithm, the difference is described below by further complement for fulfilling the successful tracking. In convenience to explain this

approach, an example is set up. We will track an object with square shape in a gray-scale image sequence. And details will be introduced in the following paragraphs. The figure 2.4 is the starting frame of that particular sequence.

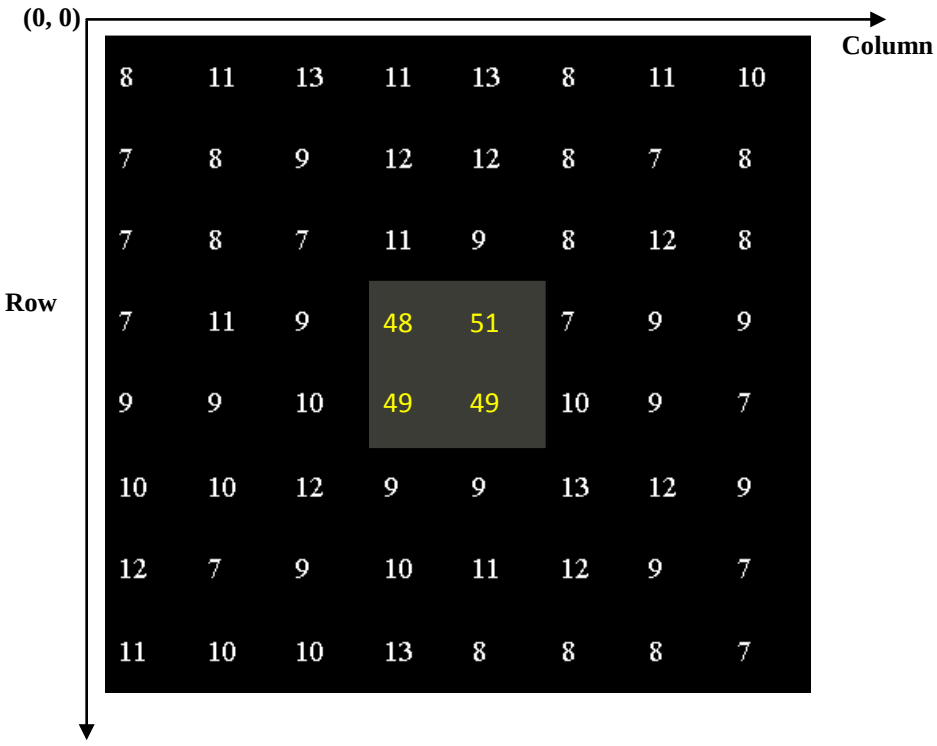


Figure 2.4 an 8×8 grey-level frame owns a view of object in size of 2×2.

2.2.4.3.1 The Object confirmation and representation

At the starting frame (frame 0), we want to confirm an object and its state. Here, figure 2.4 with pixel intensities shows the content in the frame. The black area is considered as the background region, while the grey region is chosen as the image projection of the object. The value in each grid indicates the pixel intensity, which the yellow number is obviously different from the background part.

1) Object representation:

From figure 2.4, it is known that this starting frame can help identify a clear recognition of an object (the rectangular part). So this particular view is used as the reference of the object, which would be represented by a color histogram.

Description of object state *x*:

Since the object is determined at first, we should have the object details stored in a state vector. It is the set of the location and window size, i.e. {object location in the row direction (x), object location in the column direction (y), window height (H), window width (W)}.

As shown in figure 2.4, the state vector at the beginning frame is x_0: $[x, y, W, H]^T = [4, 4, 2, 2]^T$. (the object location depends on the position of upper-left corner.)

Object modeling

Meanwhile, the referenced color histogram of the target should be established accordingly. We select 16 bins as total bin number. Since, the pixel intensity of a frame varies in the range [0, 255]. So the score of one pixel is computed with the expression:

$$Bin_I = \left\lfloor \frac{I \cdot 16}{256} \right\rfloor = \lfloor I/16 \rfloor, \tag{2.31}$$

where I is the luminance value of a pixel; Bin_I is the output bin order in a histogram, in other words, it can be seen as the quantized integer part. For example, $Bin_{50}=3$ and $Bin_{10}=0$.

The referenced color histogram of the object, i.e. composition of bins such as $q = \{q^u\}_{u=1\ldots16}$, is formed in a vector in $\Re^{16}$ subsequently. So in that figure, it is deduced from

$$q^u = f \sum_{i=1}^{W \times H} \delta[h(I_i) - u], \tag{2.32}$$

where $f=(W \times H)^{-1}$ is constant for normalization; u is the current bin order. $\delta(.)$ is the pulse function to match the pixel intensity to corresponding bin.

Then, as in this example, $q = [0\ 0\ 0\ 1\ 0\ 0\ \ldots 0]^T$.

2) Generation of the initial particles

As for convenience, it is assumed that, initially, particles are all staying in the same object position. Thus, the weights of the particles are typically the same with value 1/N, namely, this is the optimal situation that all particles could be transited with the exact movement of object.

The particle number N is determined empirically by testing and trials. Here, we took N = 16 in the example. Therefore, in the set of particles, every particle shares the same location as the object, which is (4, 4). The particle weight is 1/16. Every particle owns a set describing the particle information, e.g. {Row position of particle, Column position of particle, particle weight}. Correspondingly, in the vector form, for the i^{th} particle x_0^i at frame 0, it owns a vector $[4, 4, 1/16]^T$.

2.2.4.3.2 Tracking in the following frames

As video playing, frames are different since moving information is brought in. Generally, as mentioned, figure 2.5 below has illustrated the frame content in pixel intensity. The square object transits to an upper-right position. Here the primary work is to automatically estimate the hidden object position in this frame. And later frames will follow exactly the same procedures as this one. We will take this figure to analyze the detailed flow of computation

using particle filtering algorithm. Some data are introduced to generate an intuitive impression for this algorithm.

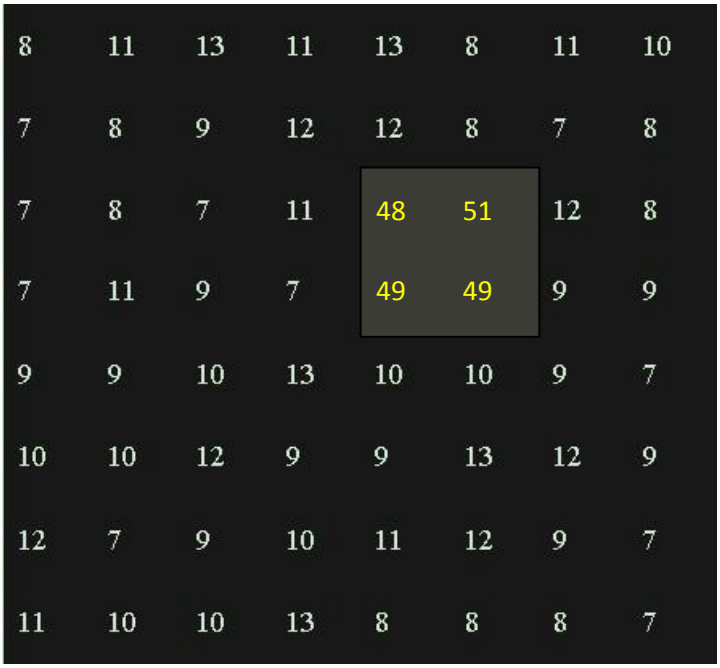


Figure 2.5 content of pixel intensity in the 2nd frame

1) Particle Transition

Since the proposal distribution of particle self-augmentation is assumed equal to the prior distribution $p(x_i \mid x_{k-1}^i)$. We can let the proposal transition follow rules of the normal distribution.

$$\left\{(x_{i,1}, y_{i,1}), w_1^i\right\} \rightarrow \left\{(x_{i,0} + N(0,2), y_{i,0} + N(0,2)), w_0^i\right\} \tag{2.33}$$

where ($x_{i,0}$, $y_{i,0}$) is the position of x_0^i; w_0^i is the particle weight. Then all the particles can be translated to the new positions.

For instance, we compute the random transition and list the new particles below in the table 2.3.

Table 2.3 Example of particles and their properties

Particle order	Position in rows	Position in columns	Weight not updated
1	4	4	0.0625
2	3	7	0.0625
3	5	3	0.0625
4	4	6	0.0625
5	3	1	0.0625
6	1	3	0.0625
7	4	3	0.0625
8	3	5	0.0625
9	4	3	0.0625
10	3	3	0.0625
11	2	2	0.0625
12	7	6	0.0625
13	7	7	0.0625
14	4	3	0.0625
15	5	5	0.0625
16	1	1	0.0625

2) Particle feature extraction

Color histogram $p(x_k^i)$ within the 2-D window with size of $width \times height$ is calculated according to the particle position x_k^i (still upper-left corner of the window) after the sampling transition. Then those histograms can also be derived similarly as computed in the reference frame. In Table 2.4, the histograms of all sixteen candidates are shown below. These histograms can be seen as collection of features for every candidate. Thus, if we measure these candidates to compare with the reference, the similarity computation should be adopted.

Table 2.4 normalized color histogram in particles. Each row indicates a 16-bin color histogram distribution.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.25	0.5	0	0.25	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0.75	0	0.25	0	0	0	0	0	0	0	0	0	0	0	0
5	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0.25	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0.25	0.5	0	0.25	0	0	0	0	0	0	0	0	0	0	0	0
10	0.25	0.25	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0
11	0.25	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0.25	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0.25	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3) Similarity measurement

Similarity of color histogram between particles and the target is compared using Bhattacharyya similarity coefficient measurement and its calculation formula is

$$\rho_i[p(x_1^i), q] = \sum_{u=1}^{16} \sqrt{q^{(u)} p_{i,1}^{(u)}}\,, \tag{3.34}$$

where q is the reference histogram of target while p belongs to the i^{th} particle.

That is, in the specific frame, we have

Table 2.5 Similarities in Bhattacharyya coefficients for existing particles.

particle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ρ	0.5	0	0	0.5	0	0	0	1	0.5	0.707	0	0	0	0	0	0

The Bhattacharyya distance therefore can be used:

$$d_i[p(y), p_0] = \left[1 - \rho_i(p(x_k^i), q)\right]^{0.5}, \tag{2.35}$$

As a result, the measurement of likelihood is then approximately given by an exponential function of the Bhattacharyya distance, such as:

$$p(z_k \mid x_k^i) \propto e^{-\lambda d_i^2[p(x_1^i),q]}, \tag{2.36}$$

where λ is adjustable to different scenes of video sequences (λ = 10 in this example).

Thus as in table 2.6, likelihood is then collected by these equations.

Table 2.6 Similarities in likelihoods for existing particles.

particle	1	2	3	4	5	6	7	8
Likelihood	0.0067	0	0	0.0067	0	0	0	1
particle	9	10	11	12	13	14	15	16
Likelihood	0.0067	0.0535	0	0	0	0	0	0

4) Particle weight update

Since resampling running iteratively, the weight updating performs,

$$\begin{aligned} w_1^i &\propto w_0^i p\left(z_1 \mid x_1^i\right) \\ &= \frac{1}{N} p\left(z_1 \mid x_1^i\right) \end{aligned} \tag{2.37}$$

After computation in eqn(2.37), the new weights still need normalization. So 1/N is not required here for simplification.

Then normalize w_1^i using the formula below.

$$w_1^i = \frac{w_1^i}{\sum_{j=1}^{N} w_1^j} \tag{2.38}$$

Update the weights of particles, e.g. in table 2.7,

Table 2.7 Corresponding weight of existing particles.

particle	1	2	3	4	5	6	7	8
Weight w_1	0.0063	0	0	0.0063	0	0	0	0.9310
particle	9	10	11	12	13	14	15	16
weight w_1	0.0063	0.0498	0	0	0	0	0	0

5) State estimation:

According to particle set $\{x_1^i, w_1^i\}_{i=1}^N$, the posterior distribution $p(x_1 \mid z_1)$ is approximated by

$$p(x_1 \mid z_1) = \sum_{i=1}^{N} w_1^i \delta(x_1 - x_1^i) \ , \tag{2.39}$$

and the state estimation is $x_1 = \sum_{i=1}^{N} w_1^i x_1^i$ (2.40)

In the example, let us bring those weights into eqn(2.39) and eqn(2.40).

$$x_1 = \sum_{i=1}^{N} w_1^i x_1^i = (3,5)$$

Sum up the particles vectors with weights. Then, the estimated target position is (3, 5). It is quite reasonable, because we can observe that the object has moved to (3, 5) in position.

6) Resampling

As known, particles transit for sampling in every frame. However, most of these particles will not follow the exact tracking routine. Therefore, the weights of such particles will be alleviated from time to time. When repeating for a few iterations, many particles contain negligible weights. It implies that a large computational effort is devoted to updating particles whose contribution to the approximated posterior density is almost zero. Therefore,

resampling is a must in applying particle filter for long tracking. It aims to discard particles with low weight and simultaneously replace them with others having high weights.

For a particle set $\{x_k^i, w_k^{(i)}\}_{i=1}^N$, the resampling algorithm is as shown below.

Set $c_1 = w_k^1$, and calculate the cumulative weight $c_i = c_{i-1} + w_k^i$.

Generate uniformly distributed random number $u : U[0, N^{-1}]$.

Set $i = 1$, for $j = 1 : N$

(i)If $u_j > c_i$, repeat $i=i+1$; else $x_k^j = x_k^i$, $w_k^j = N^{-1}$

(ii) $u_j = u + N^{-1}(j-1)$

Set $k = k+1$

After the special designed algorithm, the resampled particles have been centralized to high-weight positions. For instance, table 2.8 tells the final resampled particles. It can be observed that most of the particles concentrate in the position (3, 5). And the corresponding weight is replaced by the average 1/16.

Table 2.8 Properties of residual particles for next-frame particle sampling.

Particle order	Position in rows	Position in columns	Weight not updated
1	4	4	0.0625
2	3	5	0.0625
3	3	5	0.0625
4	3	5	0.0625
5	3	5	0.0625
6	3	5	0.0625
7	3	5	0.0625
8	3	5	0.0625
9	3	5	0.0625
10	3	5	0.0625
11	3	5	0.0625
12	3	5	0.0625
13	3	5	0.0625
14	3	5	0.0625
15	3	5	0.0625
16	3	5	0.0625

2.2.4.3.3 Algorithm procedures

Overall, this color-based Particle filter can be summarized in the table below. It has been proved efficiently in processing. However, still problems exist as mentioned in the introduction chapter.

Table 2.9 Summarized Algorithm of the Color-base PF.

Given the initial target model and particle set $\{x_0^i\}_{i=1}^N$

- Reference color distribution: q by eqn(2.2)
- Particle set: x_0^i uniformly distributed around the target location.
- Assign the particle weights to 1/N.

Perform the iteration steps when k does not reach the end:

- FOR i = 1: N (state transition of sampling particles)
  - Draw $x_k^i \sim q(x_k \mid x_{k-1}^i, z_k)$, $particle(x, y) \rightarrow (x + N(0,5), y + N(0,5))$
  - Keep the particle weight as the previous state one.
- END FOR
- FOR i = 1: N (Observe the color distribution)
  - Calculating the color histogram of each particle
  - Compute the Bhattacharyya distance between the particle and the referenced target model.
  - Approximate the Bhattacharyya distance into Likelihood computation.
  - Update the weight at current state from the eqn(2.37)
- END FOR
- Normalize the weight value with sum equal to 1, according to eqn(2.38)
- Apply the particles and the current normalized weights to estimate the output state through eqn(2.40)
- Resample using the **systematic resampling** algorithm

Stop tracking

Chapter 3. Embedded Adaptive Multi-Step Recursive Sampling Strategy

3.1 Introduction

We propose in this chapter a new multi-step recursive sampling in resampling algorithm [76,78]. It aims to find useful samples appropriately. Thus fewer particles are required for an acceptable estimation. Though the basic idea appears close to the Annealed PF [17], we have succeeded in preserving the tracking results more robustly and smoothly. In our approach, the particles at each step are formulated into likelihood computation and resampling procedures. The residual particles with the top weights are then drawn for the next sampling step. Thus some low-weight particles can be filtered, and only the more possible ones are kept to form the center of next sampling pattern. Iteratively, when the limited particle allocation is exhausted, global maximum position will be confirmed by estimating among these particles. Equivalently, an adaptive sampling pattern scheme is proposed. According to the results of previous tracking estimates and inertia, the pattern updates effectively. This creates better searching area. Also our scheme allows simple evaluation of tracking to control the number of particles, hence it saves computation cost. An early version on part of these ideas has been written in conference papers [28, 29].

The rest of the chapter is organized as follows. Section 3.2 gives our resampling modification and describes the details of our new sampling strategy, called Multi-Step Recursive Sampling in Resamping. The overall algorithm is expressed in section 3.3. Implementation and experimental results are reported in section 3.4. Finally, in section 3.5, we conclude the paper.

3.2 Multi-step sampling

3.2.1 Problem formulation

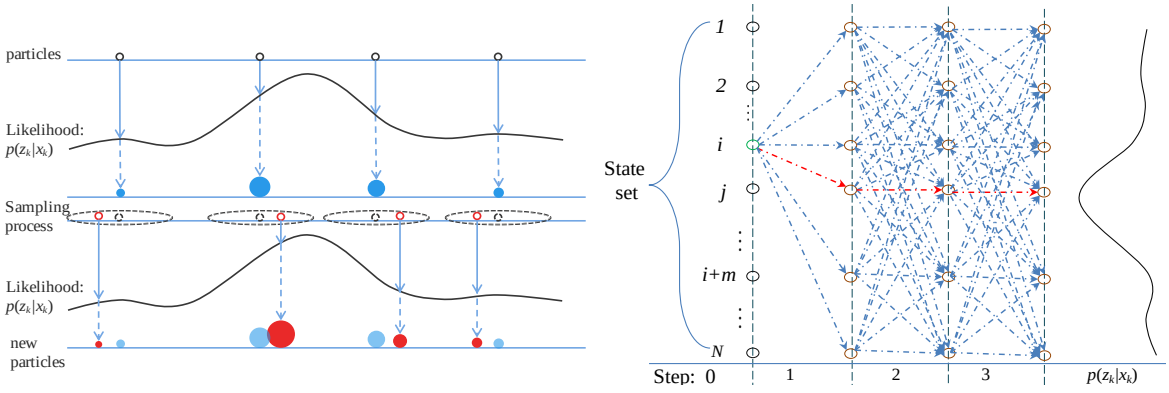
In our tracking algorithm, the objective is to find the closest estimate to the target, including the position and other parameters. Normally, those particles with high weights are considered as close candidates to the referenced target.

Hence, we assume that the highest weight, i.e. the highest posterior probability density in the discrete form, gives the most probable candidate. In the Hidden Markov Model, estimation is a method choosing the closest candidate to the hidden state given some prior knowledge. Weights of the particles are extracted from a comparison against the reference target model. Practically, the problem turns to observe the global maximum from those discrete weights. If the particle density is high enough after sampling, owning to the highest weight, the peak particle will be recognized as the estimate. However, error may arise from the computation of weights. Rather than simply searching for the highest-weight candidate, it is better to find the expectation of particles in a uni-modal weight distribution. If the sampled particles are mainly chosen from the high-likelihood region, where the particle density is high around the global maximum, the expected estimate will closely approximate the peak point with the real global maximum in weight.

This inspires us to propose the idea of multi-step recursive sampling to obtain high-weight particles as many as possible. Therefore, during one time interval, say at frame k, we will perform the sampling in chain actions after the initial predictive transitions of some residual particles. Sampling in a chain means drawing particles in a path with a few steps, which is

shown in figure 3.1 (a) and (b). Particles are the candidates of state. A state set contains all the possible candidates. As shown in figure 3.1 (a), it is the new sampling process between steps.

New particles at the next step are generated based on the ones existing and an appropriate sampling function (e.g. the Gaussian distribution) is represented by dash-line ellipses in figure 3.1 (a). From the likelihoods which are obtained from the density $p(z_k|x_k)$ function, we can observe that the new particle is more probable to stay in the high-likelihood region if its referenced particle (blue blob) is also in it. After several steps, these particles at every step can be linked together to form a possible path which is in the form of a particle chain.



(a) (b)

Figure 3.1 simulation result of recursive sampling: (a) example of sampling process between two steps, (b) possible ith chain path given the knowledge of likelihood distribution.

In figure 3.1 (b), the candidates (from 1 to N) in the state set are connected to candidates of the next steps, with the blue arrow lines. Each candidate also owns its likelihood from the density function, $p(z_k|x_k)$. Thus there are numerous possible routes in this example. Objectively, let us compose a chain with all leading particles and normally all high weights particles will stick around this chain. For example, the red arrows in figure 3.1 (b) represent the most prominent route. Unlike other less efficient algorithms, we have fewer steps in a chain, which can reduce the sampling particles and the complexity of computation.

3.2.2 Theoretical derivations

Since, after resampling and predictive transitions, the existing particles are considered as the starts of the chains. Each chain is labeled by an index for its start. Subsequently, the estimation process relies on the newly proposed particles in the chains. Hence, we define a new term $x_{k,0:S}^{i}$ as the ith particle chain which is a list of consecutive particle records $x_{k,s}^{i}$ from steps 0 to S, where S is the final step order of the chain, s is one instant of steps in [0, S], and i labels the particle chain which is equal to the index of a starting particle among all particles in the initial stage. Let us also denote $\omega_{k,0:S}^{i}$ as the weight of the particular particle chain $x_{k,0:S}^{i}$. Since the particles are samples according to the Markov property, such chain weight is proportional to the multiplication of every single weight. Namely only all high-weight particles can compose a good chain. It also tells that the particles approach the high likelihood in few steps. As $\omega_{k,0:S}^{i}$ is a certain combination of discrete particle weights, it dominates the quality of sampled chain of particles. When its value is higher, the chain is a better one in sampling. Hence, our objective is to let each chain obtain weights with values as high as possible. Particles with the promising route are most possibly staying in the high-likelihood area. Then convert eqn (2.27) into

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N} \sum_{s=0}^{S} \tilde{\omega}_{k,s}^{i} \delta(x_k - \tilde{x}_{k,s}^{i}) \tag{3.1}$$

where $\left\{ \tilde{x}_{k,0:S}^{i}, \tilde{\omega}_{k,0:S}^{i} \right\} = \arg\max\left(\omega_{k,0:S}^{i}\right),$

i.e. $\tilde{x}_{k,0:S}^{i}$ is the chain of particles with the maximized weight $\tilde{\omega}_{k,0:S}^{i}$ at time k. The highest weight $\tilde{\omega}_{k,0:S}^{i}$ indicates that particles in this chain will reach the high-likelihood region as soon

as possible and the individual weights ($\omega_{k,s}^{i}$) of particles will be high as well. Then, the weight $\omega_{k,0:S}^{i}$ can be expressed similar to eqn (2.37), i.e.,

$$\omega_{k,0:S}^{i} \propto \frac{p(x_{k,0:S}^{i} \mid z_{1:k})}{q(x_{k,0:S}^{i} \mid z_{1:k})}. \tag{3.2}$$

where $p(x_{k,0:S}^{i} \mid z_{1:k})$ is the posterior pdf output of the i[th] particular chain, and $q(x_{k,0:S}^{i} \mid z_{1:k})$ is the importance sampling strategy in pdf.

In eqn (3.2), the discrete instances of posterior density $p(x_{k,0:S}^{i} \mid z_{1:k})$ is separated into parts: the likelihood of the chain $p(z_k \mid x_{k,0:S}^{i}, z_{1:k-1})$, the Markov process in particle propagation with density $p(x_{k,1:S}^{i} \mid x_{k,0}^{i})$ at time k, the prior state transition pdf $p(x_{k,0}^{i} \mid x_{k-1}^{j})$ conditioned on the deterministically reordered particle x_{k-1}^{j} (after resampling) at time k-1 for sampling the sequence with particles $x_{k,0:S}^{i}$, and the posterior pdf $p(x_{k-1}^{j} \mid z_{1:k-1})$ at time k-1.

According to the equation of the posterior density updated from the prior via Bayes' rule as in eqn (2.25), then have,

$$p(x_{k,0:S}^{i} \mid z_{1:k}) = \frac{p(z_k \mid x_{k,0:S}^{i}, z_{1:k-1}) \cdot p(x_{k,0:S}^{i} \mid z_{1:k-1})}{p(z_k \mid z_{1:k-1})}$$
$$= \frac{p(z_k \mid x_{k,0:S}^{i}, z_{1:k-1}) \cdot p(x_{k,1:S}^{i} \mid x_{k,0}^{i}) p(x_{k,0}^{i} \mid z_{1:k-1})}{p(z_k \mid z_{1:k-1})}$$

$$= \frac{p(z_k \mid x_{k,0:S}^{i}, z_{1:k-1})}{p(z_k \mid z_{1:k-1})} p(x_{k,1:S}^{i} \mid x_{k,0}^{i}) p(x_{k,0}^{i} \mid x_{k-1}^{j}) p(x_{k-1}^{j} \mid z_{1:k-1})$$

Because $p(z_k \mid z_{1:k-1})$ is assumed to be constant,

$$p(x^i_{k,0:S} \mid z_{1:k}) \propto p(z_k \mid x^i_{k,0:S}, z_{1:k-1}) p(x^i_{k,1:S} \mid x^i_{k,0}) p(x^i_{k,0} \mid x^j_{k-1}) p(x^j_{k-1} \mid z_{1:k-1}). \quad (3.3)$$

As assumed in the Markov chain, the sequential particle propagation $x^i_{k,1:S}$ in the prior density can also be modeled as the multiplication of mutual state transitions, i.e.,

$$p(x^i_{k,1:S} \mid x^i_{k,0}) = \prod_{s=1}^{S} p(x^i_{k,s} \mid x^i_{k,s-1}).$$

Eqn(3.3) can then be derived as

$$\begin{aligned} p(x^i_{k,0:S} \mid z_{1:k}) &\propto p(z_k \mid x^i_{k,0:S}, z_{1:k-1}) p(x^i_{k,1:S} \mid x^i_{k,0}, x^j_{k-1}) p(x^i_{k,0} \mid x^j_{k-1}) p(x^j_{k-1} \mid z_{1:k-1}) \\ &= p(z_k \mid x^i_{k,0:S}) p(x^i_{k,1:S} \mid x^i_{k,0}) p(x^i_{k,0} \mid x^j_{k-1}) p(x^j_{k-1} \mid z_{1:k-1}) \\ &= p(z_k \mid x^i_{k,0:S}) \left[\prod_{s=1}^{S} p(x^i_{k,s} \mid x^i_{k,s-1}) \right] p(x^i_{k,0} \mid x^j_{k-1}) p(x^j_{k-1} \mid z_{1:k-1}) \end{aligned}$$

(3.4)

where $p(x^i_{k,0} \mid x^j_{k-1})$ is the initial sampling density based on residual particle x^j_{k-1}.

The importance density $q(.)$ distribution can be chosen by augmenting the existing particles from $q(x^i_{k,0:S-1} \mid z_{1:k})$ with the known proposal density $q(x^i_{k,S} \mid x^i_{k,S-1}, z_{1:k})$. Subsequently, it can further be factorized as the multiplication of the proposal Markov transitions $q_s(x^i_{k,s} \mid x^i_{k,s-1})$ together with the importance density $q_0(x^i_{k,0} \mid z_{1:k})$ of the beginning particle $x^i_{k,0}$. Note that $q_s(x^i_{k,s} \mid x^i_{k,s-1})$ declares the proposal sampling density of particle $x^i_{k,s}$ sampled from $x^i_{k,s-1}$ at the previous step, which is regardless of the observations $z_{1:k}$ .Thus the importance density of the chain can be factorized as

$$
\begin{aligned}
q(x_{k,0:S}^{i} \mid z_{1:k}) &= q(x_{k,S}^{i} \mid x_{k,S-1}^{i}, z_{1:k}) q(x_{k,0:S-1}^{i} \mid z_{1:k}) \\
&= \left[\prod_{s=1}^{S} q_s(x_{k,s}^{i} \mid x_{k,s-1}^{i}, z_{1:k}) \right] q_0(x_{k,0}^{i} \mid z_{1:k}) \\
&= \left[\prod_{s=1}^{S} q_s(x_{k,s}^{i} \mid x_{k,s-1}^{i}) \right] q_0(x_{k,0}^{i} \mid z_{1:k})
\end{aligned}
\tag{3.5}
$$

Consider the beginning of Markov transition, which follows the SIR particles filtering approach. Let us assume that the importance density function $q_0(.)$ is factorized as

$$
q_0(x_{k,0}^{i} \mid z_{1:k}) = q_0(x_{k,0}^{i} \mid x_{k-1}^{j}, z_{1:k}) q(x_{k\text{-}1}^{j} \mid z_{1:k-1}) \tag{3.6}
$$

where $q_0(x_{k,0}^{i} \mid x_{k-1}^{j}, z_{1:k})$ denotes the proposed particle transition density from the residual particle x_{k-1}^{j} at the previous stage, k-1, to the newly sampled one, $x_{k,0}^{i}$, at the current stage, k, and $q(x_{k\text{-}1}^{j} \mid z_{1:k-1})$ is the importance density of particle x_{k-1}^{j} with the previous observations provided.

Hence, substitute eqn (3.4), eqn (3.5) and eqn (3.6) into eqn (3.2), the subsequent weight update becomes,

$$
\omega_{k,0:S}^{i} \propto \frac{p(z_k \mid x_{k,0:S}^{i}) \left\{ \prod_{s=0}^{S-1} p(x_{k,s+1}^{i} \mid x_{k,s}^{i}) \right\}}{\left\{ \prod_{s=0}^{S-1} q(x_{k,s+1}^{i} \mid x_{k,s}^{i}) \right\}} \cdot \frac{p(x_{k,0}^{i} \mid x_{k-1}^{j}) p(x_{k-1}^{j} \mid z_{1:k-1})}{q_0(x_{k,0}^{i} \mid x_{k-1}^{j}, z_{1:k}) q(x_{k\text{-}1}^{j} \mid z_{1:k-1})} \tag{3.7}
$$

The likelihood of particle chain $p(z_k \mid x_{k,0:S}^{i})$ is given by the multiplication of the all individual likelihood outputs $p(z_k \mid x_{k,s}^{i})$. Therefore, we have

$$
p(z_k \mid x_{k,0:S}^{i}) = p(z_k \mid x_{k,0}^{i}) p(z_k \mid x_{k,1}^{i}) \ldots p(z_k \mid x_{k,S}^{i}). \tag{3.8}
$$

Consequently, the weight of particle chain $\omega_{k,0:S}^{i}$ can be simplified as

$$\omega_{k,0:S}^{i} \propto \left[\prod_{s=1}^{S} \frac{p(z_k \mid x_{k,s}^{i}) p(x_{k,s}^{i} \mid x_{k,s-1}^{i})}{q(x_{k,s}^{i} \mid x_{k,0:s-1}^{i})} \right] \omega_{k,0}^{i} \tag{3.9}$$

where $\omega_{k,0}^{i} = \dfrac{p(z_k \mid x_{k,0}^{i}) p(x_{k,0}^{i} \mid x_{k-1}^{j}) p(x_{k-1}^{j} \mid z_{1:k-1})}{q_0(x_{k,0}^{i} \mid x_{k-1}^{j}, z_{1:k}) q(x_{k\text{-}1}^{j} \mid z_{1:k-1})} \omega_{k-1}^{j}$.

Note that ω_{k-1}^{j} is available as a constant after resampling. The term $\omega_{k,0}^{i}$ is the beginning (s=0) weight of i[th] particle after random or predictive transition, and i also acts as an index of the same chain. Among s steps,

$$p(x_{k,s+1}^{i} \mid x_{k,s}^{i}) = q_s(x_{k,s+1}^{i} \mid x_{k,s}^{i}),$$

where $s \in [1, s]$ which assumes that the proposed distribution is equal to the prior transition distribution. The weight of particle chain $\omega_{k,0:S}^{i}$ is the multiplications of the likelihood at each step $p(z_k \mid x_{k,s}^{i})$ and the initial weight of the ith particle $\omega_{k,0}^{i}$ before the multi-step sampling.

$$\begin{aligned} \omega_{k,0:S}^{i} &\propto p(z_k \mid x_{k,S}^{i}) p(z_k \mid x_{k,S-1}^{i}) \cdots p(z_k \mid x_{k,1}^{i}) \omega_{k,0}^{i} \\ \omega_{k,0}^{i} &\propto p(z_k \mid x_{k,0}^{i}) \omega_{k-1}^{j} \end{aligned} \tag{3.10}$$

where ω_{k-1}^{j} is the weight of the residual particle after resampling at the last stage .

Recall the Vertibi algorithm, the most probable list of instances is the combination of those which have maximized the probability at corresponding steps. As the maximum of $\omega_{k,0:S}^{i}$ is essential for an efficient sampling, through the Vertibi algorithm, the expression in eqn (3.10) is converted into the consideration of maximizing the chain weight at each sampling step, i.e.

$$\begin{aligned} \omega_{k,0:s}^{i} &= \max_{x_{k,s}^{i}}[\, p(z_k \mid x_{k,s}^{i})\omega_{k,0:s-1}^{i}], \text{ when } s > 0 \\ \omega_{k,0}^{i} &\propto p(z_k \mid x_{k,0}^{i})\omega_{k-1}^{j}, \text{ when } s = 0 \end{aligned} \tag{3.11}$$

In other words, under a considerable number of samples, if we can obtain the particle with the highest likelihood at each step from eqn (3.11), the chain of particles can finally be converged to the area holding the optimal candidate. Also, the top-weight particles will let sampling at next step possibly have better performance in weight. Because, it is known that distribution of the Likelihood map is gradual against the spatial offset. The particles sampled from the top ones will more possibly receive high weight again. For example, the ellipse pattern of sampling in figure 3.1(a) indicates the 1-D Gaussian sampling intuitively. It can be observed that the high-likelihood point owns higher probability to sample new particles with still high likelihood.

The mathematical derivation has thus proved that the multi-steps sampling is effective to the estimation in discrete form. Because the optimal estimate is considered to have the highest sum of weights, it is certain that the appropriate particle chain should have the highest possibility. Also this particular chain must be the fastest route (fewest steps) towards the near peak region (the most probable position) of the estimate. Hence, our objective is to focus on finding the routes of particles with the highest weight. In other words, it can be considered as iteratively searching top particles with high likelihood which are drawn from the last top particles.

3.2.3 Resampling Modification

Let us recall that, for systematic resampling methods [22, 42, 45], high-weight particles are used to replace the low-weight ones. If weight of a particle is lower than the average, i.e. 1/N,

it will be replaced by a particle with higher weight. Hence, we can collect the times of replication for such particles, namely, $[\{x_k^{j*}, \omega_k^j, t^j\}_{j=1}^{N'}]$ = RESAMPLE $[\{x_k^i, \omega_k^i\}_{i=1}^{N}]$, where t denotes the times of repetition using the i^{th} particle to cover those low-weight particles. If t is bigger than zero, reorder the current i^{th} particle with a new index j in the remaining particles. Obviously, the repetition (t) is an intuitive representation of the weight. For instance, the particle with a weight of 2/N will repeat one more time to replace one particle with a weight lower than 1/N. Thus t is equal to two for this particle in the residual set. After resampling, as illustrated in Fig.1 (a), the recursive sampling will perform the next sampling process based on the residual particles in the new Gaussian sampling functions. The sampling priority depends on the repetition of residual particles. The most repeated particle owns the highest priority compared with others. It will be arranged to proceed the sampling firstly. The largest portion of particles will be allocated to samples around it. Fewer repetitions indicate lower priority. Then there will be fewer particles sampled from it. When the number of particles meets the given upper limit of allocation, the sampling will be terminated and the feedback in likelihood will be carried out. Details will be discussed in the next section.

3.2.4 Recursive sampling method

Intuitively, we sample a few particles as pioneers. They are examined by likelihood computation. Then we will resample them in terms of the density distribution. The next sampling step is performed based on the coarse density distribution. It is assumed that the area nearby higher-weight particles must be more probable to cover the high-likelihood region as well. Then according to the repetitions of the residual particles, new samples are randomly picked around them at the corresponding priority in the sampling order. Meanwhile, to enhance the role of higher-weight particles, the number of newly sampled particles around

one specific residual is made proportional to its repetition, e.g., No. ≈ repetition × 1.2. It is allocated to have more new samples generated from it. Equivalently, some local classifiers, say for example, the dissimilarity measure from multi-cue features, can be used to compute weights with better discrimination power for residual particles. Then sample allocations at the next step can concentrate mainly on particles with higher weights. Table II describes the details of the sampling strategy at each step. The input set $\{x^i_{k,s-1}, \omega^i_{k,s-1}, t^i_{s-1}\}_{i=1}^{N'_{k,s-1}}$ is the resampled particle set from the previous step. $N_{k,s-1}$ denotes the number of already sampled particles before step s. Based on the residual particles, new particles are generated around those with higher weights.

Table 3.1 Algorithm of Recursive sampling

$[\{x^j_{k,s}, \omega^j_{k,s}\}_{j=N_{k,s-1}}^{N_{k,s}}] = \text{RS}\,[\{x^i_{k,s-1}, \omega^i_{k,s-1}, t^i_{s-1}\}_{i=1}^{N'_{k,s-1}}, N_{k,s-1}]$

- Initialize: $N_{k,s} = 0$, $j = N_{k,s-1}$
- WHILE $N_{k,s} < Threshold\_Particle\_No \times 0.8 + N_{k,s-1}$
  - FOR $i = 1{:}N'_{k,s-1}$,
    - IF $t^i_{s-1} > Threshold\_repetition$ *//get rid of the sampling impoverishment*
      - $SampleRep = round\,(t^i_{s-1} \times 1.2)$
      - FOR a = 1: *SampleRep*
        - $x^j_{k,s} = x^j_{k,s-1} + V$,

          V is the multivariate Gaussian random vector with zero mean
        - $j = j + 1$
        - IF $j \geq N_s + N_{k,s-1}$
          - $N_{k,s} = j$
          - STOP, and jump out of all loops
        - END IF
      - END FOR
      - $t^i_{s-1} = 0$

    - END IF
  - END FOR
- *Threshold_repetition = Threshold_repetition - 1*
- END WHILE

This algorithm is designed for a recursive approach to reach the high-likelihood region. Additionally, we set up a mechanism for decreasing the threshold of repetition to continue the sampling from relatively lower weight particles for the sake of robustness in diverse sampling environment. We also define an upper-bounded particle number (N_s) at each step in order to regulate the sample size properly.

3.2.5 Online evaluation and operations

An evaluation procedure is carried out in each step. If the estimation result is sufficiently good at a certain step, the sampling process can be terminated. Hence, we can reduce further redundant sampling and similarity measurements. On the other hand, if the tracking result is bad, more particles are added step by step, until the evaluation is good again. The evaluation contains two parts: the result is

(i) Good & Converging, if $\left\|\hat{x}_{k,s} - \hat{x}_{k,s-1}\right\| < T_h \,\&\, \bar{L}_{k,s} > 0.9\bar{L}_{k-1}$, or

(ii) Bad, if $\bar{L}_{k,s} < 0.6\bar{L}_{k-1}$, then, N=N+$N_s$

where $\hat{x}_{k,s}$ is the estimate at the s^{th} step. Th (Th=2) is the threshold of the distance between two estimates. $\bar{L}_{k,s}$ is the average of the particles' likelihood at the s^{th} step. N is the size of particle set. N_s represent the number of generated samples at each step.

3.3 Overall algorithm

In general, the recursive sampling algorithm (in section 3.24) is combined with the modified resampling strategy (in section 3.23). We have used this composition to formulate our new particle filtering algorithm. Thus this Multi-Step Recursive Sampling in Resampling (MSRSIR) Algorithm can be illustrated in Table 3.2. Figure 3.2 shows the intuitive diagram of the algorithm.

It is mentioned that the approach keeps all used particles with their un-normalized weights, and then computes the expectation as the final estimate from all of them. This sampling approach involves more effective particles than simple random walk model. Meanwhile, some pioneer particles are more or less influenced by the residual particles (first-step sampling) of the last frame. Hence, making use of the influences of the current and last stages, the tracking results will be smoother on the overall trajectory rather than oscillating in the subjective evaluation.

Table 3.2 Algorithm of MSRSIR.

$\{x_k^{i*}, \omega_k^{i*}\}_{i=1}^{N_k}, \{x_k^j, \omega_k^j, t^j\}_{j=1}^{N'}$ =MRSIRPF [$\{x_{k-1}^i, \omega_{k-1}^i, t^i\}_{i=1}^{N'}, z_k$]

- Initialize: sampling stage s=0
- Sample N_0 particles by prediction and assumption ($N_{0=}\ N_{k,1}$)

  [$\{x_{k,0}^j, \omega_{k,0}^j\}_{j=1}^{N_{k,1}}$] = RS [$\{x_{k-1}^i, \omega_{k-1}^i, t^i\}_{i=1}^{N'}$,0]

  - Assign the particle a weight $\omega_{k,0}^j \propto p(z_k \mid x_{k,0}^j)\omega_{k-1}^j$
- Normalize those weights with sum equal to 1.
- Resample those $N_{k,1}$ particles

  [$\{x_{k,1}^j, \omega_{k,1}^j, t_1^j\}_{j=1}^{N'_{k,1}}$] = RESAMPLE [$\{x_{k,1}^i, \omega_{k,1}^i\}_{i=1}^{N_{k,1}}$]
- FOR s = 1: S (*Maximum*)
  - Sample again according to the resampled results

  $\{x_{k,s}^j, \omega_{k,s}^j\}_{j=N_{k,s-1}}^{N_{k,s}}$ =RS [$\{x_{k,s-1}^i, \omega_{k,s-1}^i, t_{s-1}^i\}_{i=1}^{N'_{k,s-1}}$,$N_{k,s-1}$]
  - Assign the particle a weight in eqn(9)
  - Normalize those weights with sum equal to 1
  - Resample these $N_{k,s}$ particles

    [$\{x_{k,s}^j, \omega_{k,s}^j, t_s^j\}_{j=1}^{N'_{k,s}}$] = RESAMPLE [$\{x_{k,s}^i, \omega_{k,s}^i\}_{i=1}^{N_{k,s}}$]
  - IF $N_{k,s} > N$ (N is the stopping threshold of particle No.)
    - $N_k = N_{k,s}$, and $\{x_k^j, \omega_k^j, t^j\}_{j=1}^{N'} = \{x_{k,s}^j, \omega_{k,s}^j, t_s^j\}_{j=1}^{N'_{k,s}}$
    - Stop iterations
- End FOR
- Collect all sampled particles in $\{x_k^{i*}, \omega_k^{i*}\}_{i=1}^{N_k}$

  ($\omega^{i*}{}_k$ is kept un-normalized.)
- Calculate the total weight of all sampled particles
- Normalize each particle.
- Compute the estimate using eqn(3.7)
- Update the sampling pattern by eqns (4.15), (4.16) and (4.17). //details are discussed in section 4.4

During the resampling procedure, other than Annealed PF, all existing particles should be brought in. It is an efficient way to address the partial occlusion and the drift problems. A good way of evaluating particles is necessary for searching the global maximum. If only a fixed few particles are considered in resampling, in the multi-modal likelihood distribution, the samples at the next step will be attracted by other modes and mislead the final estimate. However, as we consider all existing particles, more information of the likelihood distribution is observed. Thus, the sampling efforts at previous steps should be included for comparison in resampling. It ensures that all the top-weight residual particles are used in the next sampling.

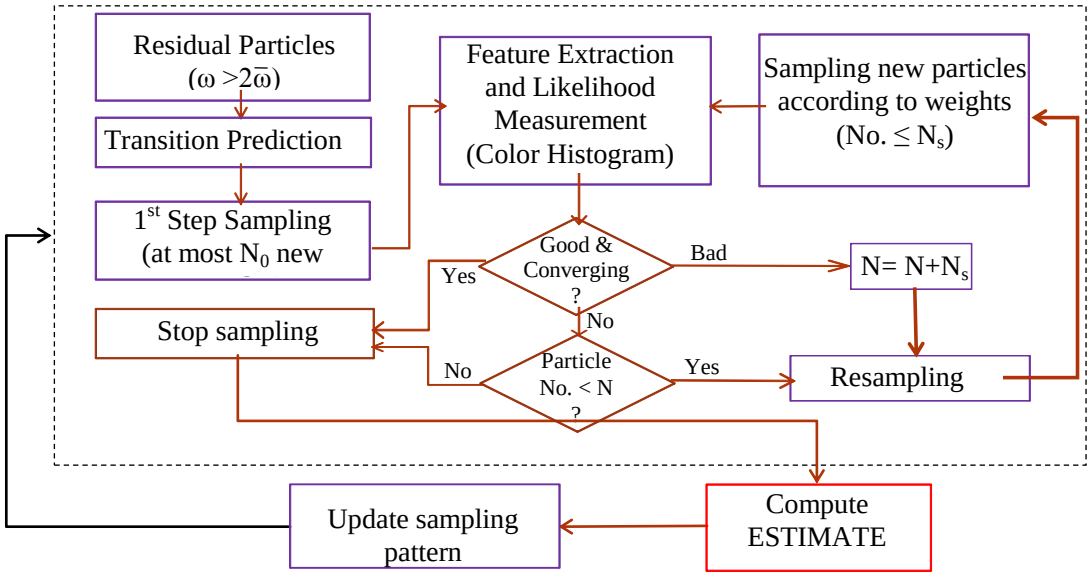


Figure 3.2 Float chart of our proposed method

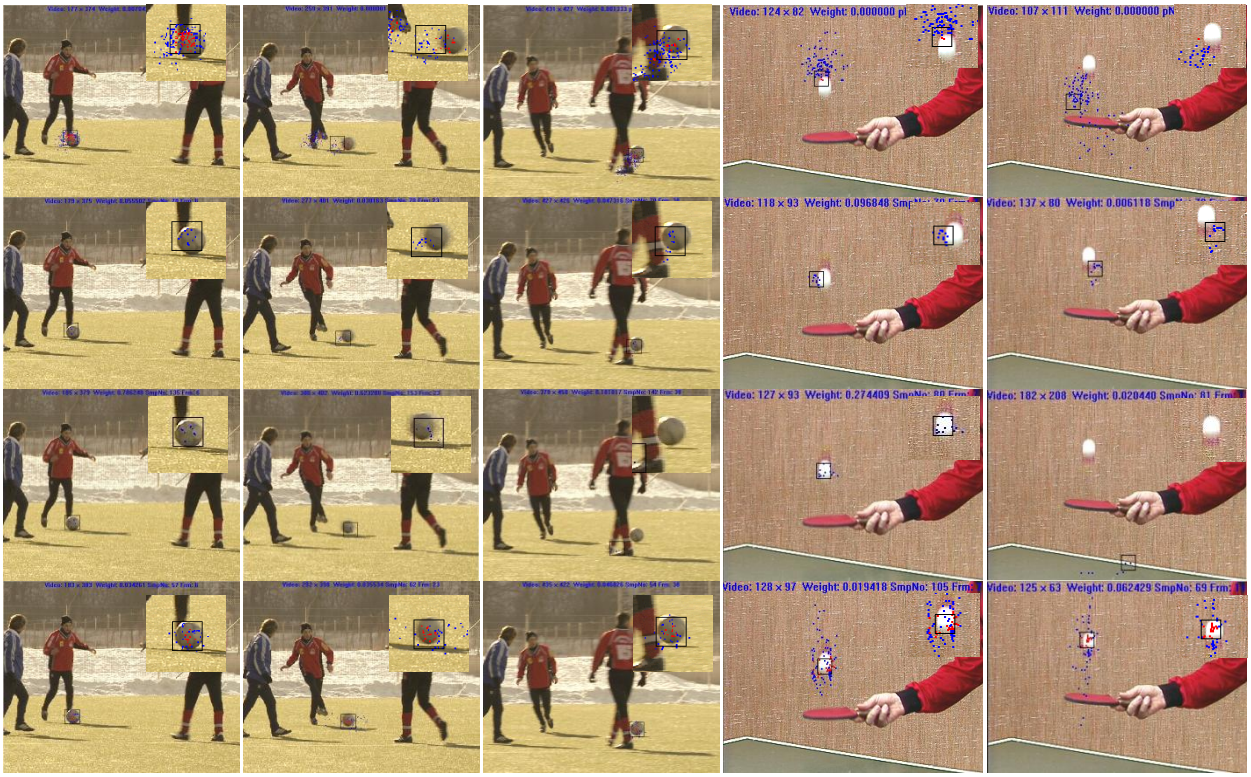


Figure 3.3 Comparisons of the tracking results in frames by SIR PF (a), Annealed PF (b), HY (c), and our proposed method (d) in the soccer (left) and table tennis (right) video sequence

Figure 3.4 Tracking results of the silver car (3 left columns) and red car (2 right columns) sequence using the four approaches, i.e., SIR PF (a), Annealed PF (b), HY (c), and our proposed method (d)

3.4 Simulation and Experiments

3.4.1 Implementation

Much experimental work has been done. For example, the algorithm was implemented to track the 2-dimensional positions (x, y) of a soccer ball, a ping pang ball or a car. As we only focus on localization, we need a clear target (easily to classify) as the subject to be tracked. Uniformly, the target is represented by its color histogram as the reference model [23, 50]. The model is not updating but keeps static in following tracking process. In the subsequent frames, particles are sampled to compute the similarity in the form of Bhattacharyya coefficient. The likelihood is the exponential output of the Bhattacharyya distance multiplied by a coefficient λ (30 in default) for a better discrimination of different particles.

Table 3.3 Parameters of four approaches in realization

Method	Particle No. (N)	Initial Sampling Pattern: $N(0, Q_0)$	Steps (S+1)	sample refinement
SIR PF [23]	200/100	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	1	No
APF [36]	14	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	5	Particle replacement in weight
HY [28, 29]	36	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	1	Mean shift (max iterations = 3)
Our Method	N_0=22, N_s=14, N=55	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	3-4	No

In the implementation, some parameters were predefined empirically as shown in Table 3.3., our target model is initially extracted from the beginning frame with a rectangular window e.g., 44*44 for a soccer, 20*20 for a ping pang ball, 40*28 for a grey car, and 40*24 for a red car.

Subsequently, the reference color histogram as the static target model is generated from that rectangular region. Then it used to measure similarities of samples in the following frame. Samples are generated with their locations. The static rectangular window will help extract the pixel regions centered at those positions. Soon, candidate histograms are deduced from such pixel regions and are applied to compute likelihoods against the reference model. These similarities will be converted into weights for estimation. Because our objective is to evaluate different approaches, we have to balance the efficiency and effectiveness of samples, namely the approaches should consume little computation in sampling and processing while tracking object correctly. We should select particles as few as possible to observe the acceptable results given fixed initial sampling pattern. Empirically, the particle numbers are arranged in Table 3.3. It is fair to every approach to find out the most efficient and effective way for respective algorithms.

3.4.2 Experimental Results

3.4.2.1 Subjective evaluation

The tracking results of several sequences are illustrated in figure 3.3 and figure 3.4. The black box represents the current estimates. The dots in the frame are the current distribution of particles. For example, in figure 3.3(a), using SIR particle filtering, most particles (blue dots) are spreading out in a large area with little contribution. The rest particles with weights higher than the average are drawn in red dots. However, the other three approaches make use of many fewer particles as shown in Table 3.3.

In the soccer sequence, the ball is stable with nearly no motion in the beginning frames (1st – 10th frame). Then, in frame 18, it is kicked with a rapid acceleration, and is later occluded by the legs. Note that, when the ball stands still, e.g. the first column of figure 3.3, the performance of

SIR PF (a) and Annealed PF (b) is worse than other two methods. They are partially offset to the real target. It is observed that SIR PF (a) samples particles in the offset area. Due to the noisy likelihood distribution, most of these particles are therefore wrongly contributing to the estimate. For APF (b), the overall density of particles is sparse in the possible target region. Without enough feedback of discrete weight distribution, it misleads the estimate to the lower-likelihood point. However, our method (by collecting all particles together) does not have such problem. After frame 18, most particles can not react to the fast transition in position, but are sampled around the previous estimate, such as the second column of figure 3.3(a). Also, the similarity result meets the multiple modes format in the likelihood distribution.

The generated multi-modal likelihood map increases the difficulty to capture the target. Hence, sampling in steps and particle refinement are inevitable for searching more effective particles in the right region. HY relies on mean-shift to move every particle iteratively to the position with higher likelihood. Annealed PF and our methods both sample the particles in steps. It is observed that all of these extended methods are positive in improving particle efficiency. Compared with 200 particles used in the SIR PF, fewer than 100 particles are utilized in the other three methods. It is observed that all of the four methods successfully catch the target again. But our method adapts to the abrupt acceleration faster than the other methods. As the person on the right-hand side starts moving, the legs occlude the soccer for a while. When the soccer comes out again, as shown in the third column of figure 3.3, it is beneficial for the algorithm making use of larger searching region and more particles to be involved, such as those in SIRPF (a). Though some particles find the correct target region as in the Annealed PF (b) and HY(c), due to the lag of others, results of the Annealed PF and SIR PF do not match well the target. Sometimes, the HY approach even loses track of the soccer ball as shown in figure 3.3 (c), as the mean shift traps the

particles around other modes in the map. However, the performance of the proposed method is still the best among all algorithms, by observing the soccer carefully.

In the table tennis sequence, we have examined again the effectiveness of algorithm in handling the motion abrupt change. The ball drops down (e.g. the fourth column of figure 3.3) and is pounced up (the fifth column of figure 3.3) subsequently. As the velocity changes dramatically, other methods lose the track of ball, while our approach keeps track of the ball. Let us also refer to the silver car sequence in figure 3.4. When the car speeds up and takes over by several similar cars, this gives the multimodal likelihood distribution. The SIRPF (a) uses many more particles and can present the same good results as ours. The APF (b) mistakes the neighbouring one as the target. HY(c) still holds the car but can not perfectly match the region of target. We have also applied the algorithms to the red car sequence. The red car stays at the center of the video frame with quite slow motion. All four algorithms are more or less capable of tracking the target. However, the fluctuation of the black box is significantly high in the APF (b) and HY (c). Occasionally, the estimate from HY (b) drifts away from the target. Our algorithm performs the best tracking as the SIRPF (a). Because of using the random sampling process, we have repeatedly performed the video tracking process many times. Hence, among these results, our method constantly gives stable tracking results as compared with other approaches.

3.4.2.2 Computation comparison

For simplicity, we take the number of used samples to represent the computation complexity for sampling efficiency. Since it is observed that every particle or samples are handled with the same work as pixel-wise feature extraction, similarity measurement. The repeated work has consumed the majority of computation sources. So the number of particles adopted is an intuitive

representation of computation efficiency. We have drawn the comparison figure in the thesis. Without doubt that mean-shift can also be considered to be computed like particles. Thus a feasible quantitative analysis of computation is utilized to illustrate our contribution of computation reduction. Moreover, this kind of comparison is regardless of computation speed of different machines. It is a relatively objective comparison of computation complexity.

Usually, the SIR PF requires the highest computation cost. Figure 3.5 illustrates that 200 particles have been used to pursue acceptable results in the soccer sequence whilst only 100 are required for others. If enough particles and proper searching region are provided, the final estimate is able to come close to the optimal one. However, it is a waste of computation in calculating some redundant particles as mentioned before. Meanwhile, the weight computation of particles is much complex, especially for large targets, since feature processing and similarity measurement need heavy computation.

Hence, the Annealed PF and HY were proposed to save computation by proposing fewer particles. The Annealed PF samples particles in layers and each layer contains the same particle number. In the soccer video sequence, totally, 70 (14 particles $\times$ 5 layers) particles were sampled and computed, which is a great reduction compared with that of SIR PF. For HY, since only 36 particles were proposed at each frame. Note that the mean-shift algorithm iteratively refines every particle by moving it to a new position with higher similarity. Approximately, the computation cost at the new position is similar to that of a new particle. For the simplification of comparison, the mean shift iteration can be regarded as the process required for at least one "particle". As shown on figure 3.5, the number of particle computations (real particles + MS iteration) for the HY algorithm varies from 72 to 162, depending upon the Mean-shift iterations.

In our evaluation work, the computation cost is also considered in the terms of "particles". On average, 136 units of "particles" were required in the soccer sequence; 80 were used for table tennis sequence; while there were 106 for silver car and 74 for red car. Figure 3.5 shows that our proposed method has succeeded in saving computation by multi-step sampling, much better than HY and SIRPF approaches. On average, 60, 74, 51 and 56 particles were sampled in these sequences, which are the fewest numbers of particles among all algorithms in most cases. Though it is not the fewest one in the table tennis sequence, it outperforms in terms of tracking quality with only a slightly increasing in particles. And for slow motion sequences, it can reduce numerous redundant samples. For example, it can save surprisingly up to 70% of the samples as compared with the SIR PF in tracking soccer. Hence, the computation cost can be the lowest comparing with other three methods. Meanwhile, the tracking result is the most successful among all the four approaches given such a limitation on computation.

Still the computation time of the four approaches has been examined by collecting their average processing time for tracking objects. As running in a computer, the real-time application depends on computational power of a PC, size of object and number of particles. If CPU of a PC is powerful and the object is relatively small, the real-time processing is reasonable. As in our simulation, our computer owns the specification with the Pentium® 4 CPU with clock frequency 3.4GHz and 3GB RAM. In the soccer and table tennis tracking, the program can handle the tracking with frame rate 7 frames/sec. The car sequence has speeded up their video playing as 25 frames/sec. In table 3.4, the average processing time of each frame is collected for comparison in these four methods. It is observed that our method still need least time to estimate the target while Mean shift approaches is not well optimized due to its longest processing time. Though our method has reduced numerous samples, the saved computation is not much apparent via

comparing with the original Particle filter (a) and APF (b). Approximately, 0.8 ms are saved in processing such four videos. However, the computation is heavily influenced by the object size. When object is much larger, it is an advantage for method using fewer samples, such as the method proposed by us. And the real-time application is much easier to be applied.

Table 3.4 Processing time for tracking objects using four approaches

Video	SIR PF (a)	APF (b)	HY(c)	Proposed (d)
Soccer seq.	16.3ms/frame	9.7ms/frame	18.2ms/frame	7.0ms/frame
Table tennis seq.	3.6ms/frame	3.2ms/frame	12.0ms/frame	3.4ms/frame
Silver car seq.	5.7ms/frame	5.6ms/frame	15.9ms/frame	4.9ms/frame
Red car seq.	5.4ms/frame	5.3ms/frame	15.8ms/frame	4.7ms/frame

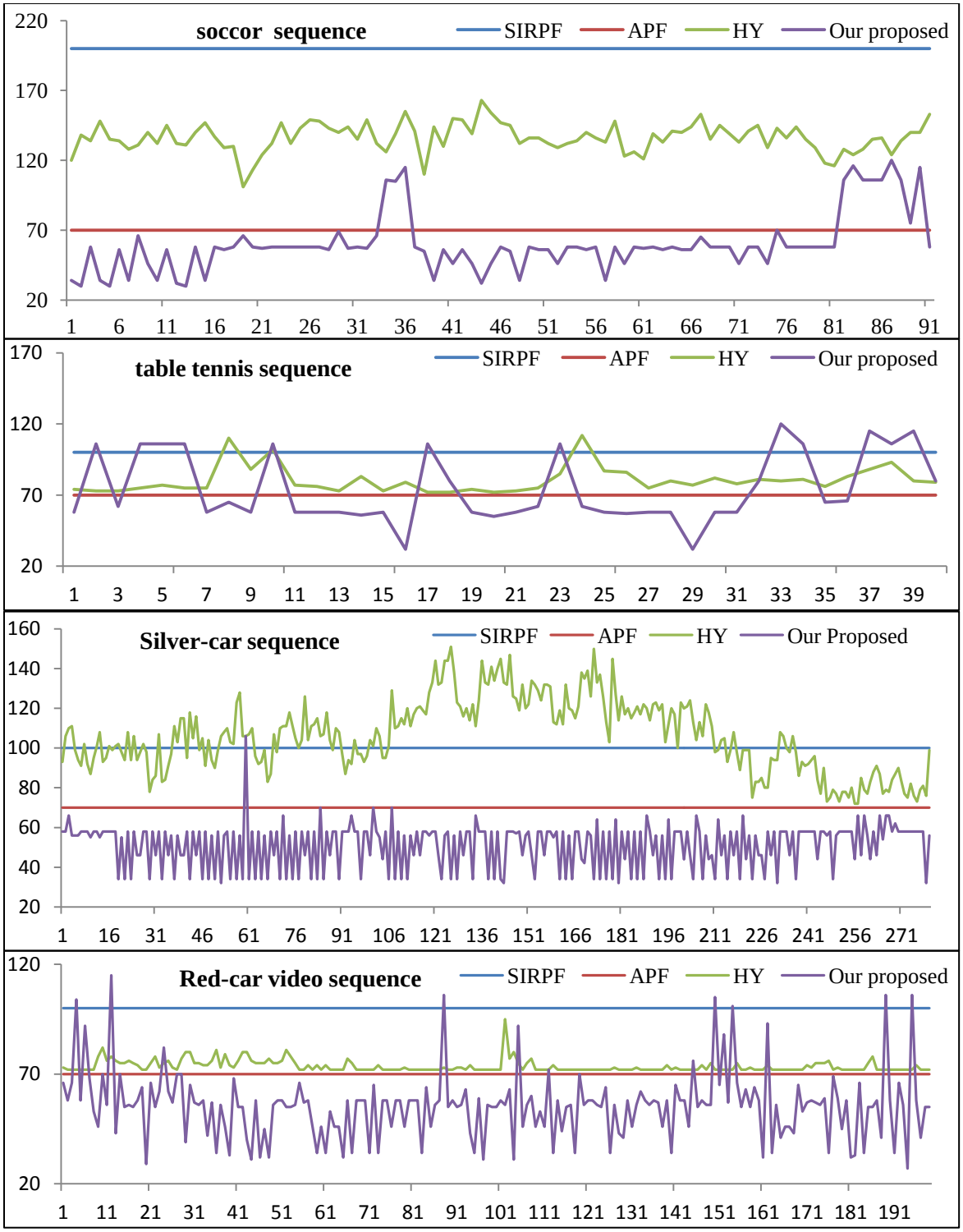


Figure 3.5 Comparison of the computational cost in "particle" respect to every frame in these sequences.

3.5 Conclusion

This chapter presented a new adaptive sampling approach for particle filtering. Rather than sampling all particles in one step, which refers to as resampling strategy, we proposed a multi-step recursive sampling method. It is designed to look for high-weight particles as many as possible in steps. Thus the sampled particles are more efficient in estimation, which is closer to the highly probable area. Experimental results have verified the effectiveness of the proposed method in reducing computation and improving robustness of our algorithm.

Chapter 4. Analysis of Sampling Patterns for Target Searching

4.1 Introduction

As observed, the Gaussian distribution is composed of the mean or expectation value with the covariance. In the Standard form, the mean is 0, namely, the samples are generated or sampled around the previous particles after resampling without any prior motion. It is known that, Gaussian distribution obeys the rule that the closer position to the mean, the higher probability of the samples occurring at that place. And the covariance matrix controls the shape of the sampling region, i.e. in the normal distribution, when the distance to mean position is less than twice of the standard deviation, 95% of the confidence has been determined that samples will occur in this area.

Generally, if we do not sketch out the possible region of estimate state, large sampling area is inevitable, which indicates a large variance for each parameter in the state. Otherwise, samples may miss some good candidate areas, which result in the wrong estimation as shown in figure 4.1. Furthermore, numerous samples would be randomly generated without positive impact to measurement. Since the good candidate region is rather small when compares with the search region/sampling region, it is recognized that lots of samples can be neglected in contributing to resultant estimate. In other words, some samples are created wastefully but increase computation complexity in statistical model. Furthermore, as the sampling pattern following the Gaussian distribution, probably, the best candidate area is out of region containing high sampling chance. As in figure 4.2, only few particles with high weights are certainly not enough to attract the

estimate moving in the right position, because of lack in effective samples. Therefore the intuitive solution is to accumulate more particles and sooner or later, enough effective particles could be collected, e.g. it is shown in figure 4.3. However, the tradeoff is apparent that heavy load of size in particle set would cause dramatically increasing in computation.

Figure 4.1, small sampling variance – lost in tracking.

Yellow and light blue points represent the particles sampled, while the yellow ones own the over-average weight and the light ones contain the below-average weight. The red square box confirmed the estimated position of the object which is far away from the actual position. And the green box is the last estimated position from the previous frame.

Figure 4.2 Large sampling variance – lost in tracking by 100 particles.

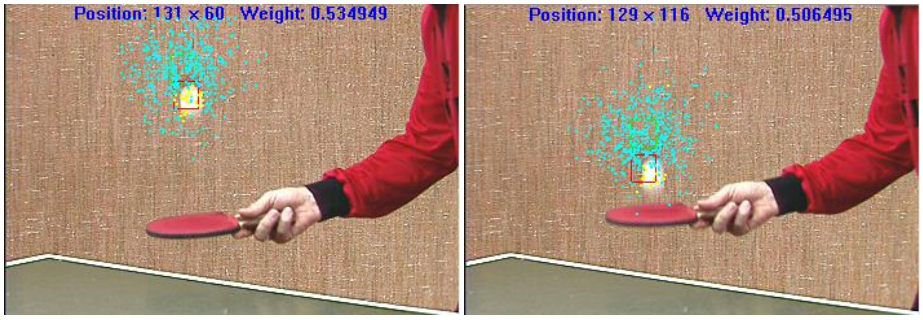


Figure 4.3 large sampling variance – keep tracking with 400 particles.

As the object moving frame by frame, sometimes we can observe its regulation of the movement from its trajectory. If the good prediction of target position accepted, the sampling will be easier by finding out good candidates.

As a result, we have to introduce a more or less reliable prediction method to coarsely shrink or narrow a region around some position to eliminate some impossible sampling area. Then several strategies have been adopted to improve the prediction rather than assuming the previous estimated state as the predicted state. And an adaptive sampling pattern scheme [77] is proposed. According to the results of previous tracking estimates and inertia, the pattern updates effectively. This creates better searching area.

4.2 Most accepted AR model

4.2.1 Formulation of AR(3) model

It is observed that the predicted transition has been applied by several auto-regressive models. Therefore, it is reasonable that the auto-regressive model can be accepted to apply in the prediction. One paper in [39] use AR model to perform smoothing operation.

$$x_k^- = C + \sum_{i=1}^{P} \varphi_i x_{k-i} + \varepsilon_k \ , \tag{4.1}$$

where x_k^- represents the state in prediction at stage k; C is a constant in the expression; x_{k-i} is the state with AR coefficient φ_i multiplied as regressive weight; P is the total order of relative states, e.g., if P = 3, this expression is typically called AR(3) model; ε_k is the white noise referred with normal assumption as Gaussian noise.

Verbally, the current term of the series can be estimated by a linear weighted sum of previous terms in the series. The weights are the auto-regression coefficients.

Of course, the key issue is to realize the model by computing the AR coefficients together with noises.

Since autoregressive model focuses on the wide-sense stationary process, we should assume the signal sequence still hold this assumption. In other words, the state chain $\{x_k\}_{k=1,\ldots,T}$ in set should be a typical WSS process. However, apparently, the state chain does not belong to the WSS process in the long term. And the training set contains the collection of previous estimated states in a good tracking procedure.

To simplify the AR expression, we neglect the constant value C by subtracting the mean value of the collected data in the training set. Therefore, all states in different stages would minus the average to construct the all-pole term fluctuating around zero.

$$\Delta x_k = x_k - \bar{x}, \bar{x} = \frac{1}{q}\sum_{i=1}^{q} x_{k-i}, \tag{4.1}$$

where $\bar{x}$ is the average of the sample set $\{x_{k-i}\}_{i=1,\ldots,q}$ with the size of q; Δx_k is the shifted samples with distribution centered around zero.

Then, the prediction equation could be reformulated as

$$\bar{x_k} = \bar{x} + \sum_{i=1}^{P} \varphi_i (x_{k-i} - \bar{x}) + \varepsilon_k \tag{4.2}$$

In order to estimate the best coefficients, many methods could be derived to obtain the solution. In our approach, we accepted the typical tool called Yule Walker equations. And those equations are formulated by autocorrelations together with the noise and AR parameters.

We could derive these equations by further exploiting the autocorrelation γ_m with shift m ($m \geq 0$) of its sequence.

$$\begin{aligned} \gamma_m &= E[\Delta x_k \Delta x_{k-m}] = E\left[\left(\sum_{i=1}^{P} \varphi_i (\Delta x_{k-i}) + \varepsilon_k\right)\Delta x_{k-m}\right] \\ &= E\left[\sum_{i=1}^{P} \varphi_i (\Delta x_{k-i} \Delta x_{k-m})\right] + E\left[\varepsilon_k \Delta x_{k-m}\right] \end{aligned} \tag{4.3}$$

We separately derive the two terms from right-hand side. We have assumed that the noise function outputs are mutually independent. And the signal Δx_{k-m} is independent to the noise ε_k when m is non-zero.

Then the second term could be derived to

$$E[\varepsilon_k \Delta x_{k-m}] = E\left[\varepsilon_k\left(\sum_{i=1}^{P}\varphi_i(\Delta x_{k-m-i}) + \varepsilon_{k-m}\right)\right]$$
$$= E\left[\varepsilon_k\left(\sum_{i=1}^{P}\varphi_i(\Delta x_{k-m-i})\right)\right] + E[\varepsilon_k(\varepsilon_{k-m})]$$

If $m \neq 0$, $E\left[\varepsilon_k\left(\sum_{i=1}^{P}\varphi_i(\Delta x_{k-m-i})\right)\right] = 0$ and $E[\varepsilon_k(\varepsilon_{k-m})] = 0$

If $m = 0$, $E\left[\varepsilon_k\left(\sum_{i=1}^{P}\varphi_i(\Delta x_{k-i})\right)\right] = 0$ and $E[\varepsilon_k(\varepsilon_k)] = \sigma^2$ which is the variance of the noise. Thus,

$$E[\varepsilon_k \Delta x_{k-m}] = \sigma^2 \delta(m),$$

where $\delta(m)$ is the Kronecker delta function.

And the first term can be analyzed to figure out the relation of AR coefficients with autocorrelation in different shifts.

$$E\left[\sum_{i=1}^{P}\varphi_i(\Delta x_{k-i}\Delta x_{k-m})\right] = \sum_{i=1}^{P}\varphi_i E[(\Delta x_{k-i}\Delta x_{k-m})] = \sum_{i=1}^{P}\varphi_i E[(\Delta x_k \Delta x_{k-m+i})] = \sum_{i=1}^{P}\varphi_i \gamma_{m-i}$$

At last, we summarize the derivations together.

$$\gamma_m = \sum_{i=1}^{P}\varphi_i \gamma_{m-i} + \sigma^2 \delta(m), \text{ s.t., } m \geq 0 \qquad (4.4)$$

And the Yule-Walker equations are created by assigning m with 0, 1, …, P. If P = 3, then the equation system is therefore expended as

$$\begin{cases} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_{-1} & \gamma_{-2} \\ \gamma_1 & \gamma_0 & \gamma_{-1} \\ \gamma_2 & \gamma_1 & \gamma_0 \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix} \\ \gamma_0 = \sum_{i=1}^{p} \varphi_i \gamma_{0-i} + \sigma^2 \end{cases} \tag{4.5}$$

And $\gamma_m = \gamma_{-m}$, since we assume the sample sequence belonging to the WSS process. So far, we have three unknown variables (AR coefficients) with three equations in settling them.

As for realization, the autocorrelation is modeled in the statistical method. We noticed that the autocorrelation were re-written to

$$\gamma_m = \frac{1}{N} \sum_{i=1}^{N} \Delta x_i \Delta x_{i-m} \ . \tag{4.6}$$

In our implementation of prediction using AR(3) model, an approach is established to estimate the next transition or next state information.

Table 4.1 An algorithm of iterative prediction of transition by AR(P) model

FOR k = 12: Frame_No

Store The individual estimated parameter in set along the time domain. $\{ \hat{x}_k \}_{\text{k= k-11, k-10, …k-1}}$

Then compute the average of the state $\bar{x} = \frac{1}{11} \sum_{i=1}^{11} \hat{x}_{k-i}$;

Assume $x_k^- = \bar{x} + \sum_{i=1}^{P} \varphi_i (\hat{x}_{k-i} - \bar{x}) + \varepsilon_k$, $\Delta x_k = \hat{x}_k - \bar{x}$, $\varepsilon_k \sim N(0, \sigma^2)$;

And stretch out the autocorrelations γ_0 , γ_1, γ_2

$\gamma_0 = \frac{1}{11} \sum_{i=k-11}^{k-1} \Delta x_i^{\,2}$, $\gamma_1 = \frac{1}{10} \sum_{i=k-10}^{k-1} \Delta x_i \Delta x_{i-1}$, $\gamma_2 = \frac{1}{9} \sum_{i=k-9}^{k-1} \Delta x_i \Delta x_{i-2}$

Apple the Yule-Walker equations for the AR coefficients φ_i and noise variance σ^2 .

END FOR

4.2.2 Analysis of applicable AR model for prediction

It is obviously that we hold the assumption of the sequence of state transition belonging to the WSS process.[40] Therefore, if, coincidently, the extracted sequence no longer fits that assumption, the computed AR coefficients would be helpless or even bringing in large error to confusion.

Thus we should propose a method to examine whether the AR coefficients are good or not.

Firstly, we noticed that WSS process owns some properties we should not avoid. [41]

1. A random process X(t) is said to be WSS if its mean and autocorrelation function are time invariant.
2. Autocorrelation is a symmetrical function only related to the time different τ, i.e. $R_X(-\tau) = R_X(\tau)$.
3. $|R_X(\tau)| \le R_X(0) = E(X^2(t))$, the "average power" of X(t). It can be proved by the Schwartz inequality.

$$(R_X(\tau))^2 = (E[(X(t)X(t+\tau)])^2 \le E[(X^2(t)]E[X^2(t+\tau)] = (R_X(0))^2 \quad (4.7)$$

Secondly, we rewrote the prediction expression in the simple form with noise unplugged.

$$f(k) = \sum_{i=1}^{3} \varphi_i f(k-i) + \varepsilon_k \quad (4.8)$$

Assume this model well describe the sequence by setting up a particular filter performed with this function. As the sequence belongs to the WSS process, so the filter should be stable and causal at the same time.

Then Apply to the expression with Z-transformation. And to fulfill the conditions, all the poles of the system function must be inside the unit circle, i.e. |z|<1.

$$F(z)=\sum_{i=1}^{3}\varphi_i F(z)z^{-i}+\varepsilon_z \longrightarrow F(z)=\frac{\varepsilon_z z^3}{z^3-\sum_{i=1}^{3}\varphi_i z^{3-i}}$$

To obtain the poles,

Since the pole must have at least one real number,

$$P(z)=z^3-\varphi_1 z^2-\varphi_2 z^1-\varphi_3=0;$$
$$P'(z)=3z^2-2\varphi_1 z^1-\varphi_2=0, \Delta'=4\varphi_1^2+12\varphi_2;$$
$$P'(-1)=3+2\varphi_1-\varphi_2;$$
$$P'(1)=3-2\varphi_1-\varphi_2;$$
$$P(-1)=1-\varphi_1+\varphi_2-\varphi_3;$$
$$P(1)=1-\varphi_1-\varphi_2-\varphi_3.$$

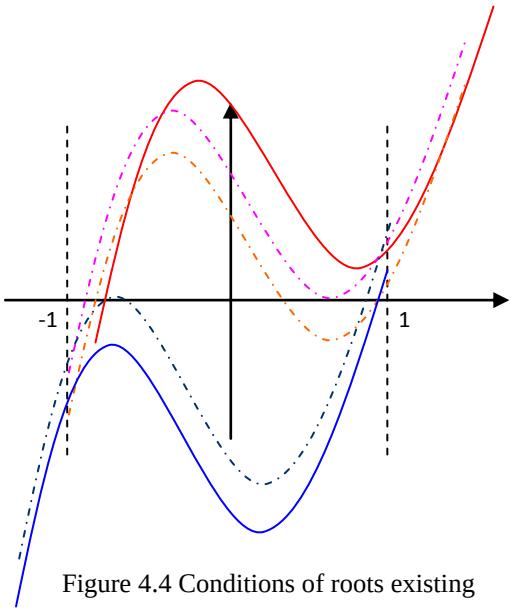


Figure 4.4 Conditions of roots existing

where P(z) is considered as the function of the denominator of F(z), P'(z) is the first-order differentiation of the function P(z). Apply 0 and 1 into both functions for consideration of the conditions. In figure 4.4, these curves are all the possible cases of the roots in the range of (-1, 1).

Nevertheless, P(-1)<0 and P(1)>0 should be fulfilled, thus

$$\begin{aligned} P(-1)&=1-\varphi_1+\varphi_2-\varphi_3<0 \\ P(1)&=1-\varphi_1-\varphi_2-\varphi_3>0 \end{aligned} \tag{4.9}$$

Then separate two situations in classification.

(1) If all the roots are the real numbers, i.e. $\Delta' \geq 0$,

then the derivatives in the both positions are increasing accordingly,

$$\begin{aligned} P'(-1) &= 3 + 2\varphi_1 - \varphi_2 > 0; \\ P'(1) &= 3 - 2\varphi_1 - \varphi_2 > 0; \end{aligned} \tag{4.10}$$

And the roots of $3z^2 - 2\varphi_1 z^1 - \varphi_2 = 0$,

$$\begin{aligned} |t_1| &= \left| \frac{\varphi_1 - \sqrt{\varphi_1^2 + 3\varphi_2}}{3} \right| < 1 \\ |t_2| &= \left| \frac{\varphi_1 + \sqrt{\varphi_1^2 + 3\varphi_2}}{3} \right| < 1 \end{aligned} \tag{4.11}$$

(2) If $\Delta' < 0$, the complex-conjugate roots exist with one more real root.

Since P(-1)<0 and P(1)>0 keep the real root belonging in the domain of (-1, 1), the complex root need verification of ensuring their magnitude smaller than 1.

Referring to the General solution of the cubic equation $P(z) = 0$,

$$S = \frac{\varphi_1\varphi_2}{6} + \frac{\varphi_1^3}{27} + \frac{\varphi_3}{2},$$
$$\Delta = S^2 - \left(\frac{\varphi_2}{3} + \frac{\varphi_1^2}{9} \right)^3$$

$$C_1 = \sqrt[3]{S + \sqrt{\Delta}}, C_2 = \sqrt[3]{S - \sqrt{\Delta}}$$
$$|z_2| = \left| \frac{\varphi_1}{3} + \frac{-1 + \sqrt{3}i}{2} C_1 + \frac{-1 - \sqrt{3}i}{2} C_2 \right| < 1$$
$$|z_3| = \left| \frac{\varphi_1}{3} + \frac{-1 - \sqrt{3}i}{2} C_1 + \frac{-1 + \sqrt{3}i}{2} C_2 \right| < 1$$

Then ,

$$\left(\frac{\varphi_1}{3}-\frac{C_1+C_2}{2}\right)^2+\frac{3}{4}(C_2-C_1)^2<1. \tag{4.12}$$

Then for those AR coefficients corresponding to the conditions, we could accepted them as good realization adaptive to the AR(3) model. Otherwise, they would be abandoned by taking other methods instead of applying the AR model.

4.3 Comparison with other simple predictive models

We have introduced a sequence of 2-component vectors to describe the estimated positions in two maps shown in figure 4.5. Apply the linear prediction to test whether these methods improve transition by moving the search region/sampling area closer to the optimal place.

Three properties of WSS state have constrained the prediction approach using AR models. We should find the most proper representation of the state. As a result, functionality of AR (3) model has been examined to observe the experimental results.

To be mentioned, we do not involve the conditions of proper AR coefficients for a BIBO system. Also, though noises are not related in the prediction, it can hardly be formulated regularly except the optimal white noise. So in the most cases, when noise is recovered into estimation, the estimation of predicted transition will be much away from the optimal one. Since the compared approaches are handled via predicting the next transition, the subsequent transition maps of X and Y dimensions have been plotted out in figure 4.6. It declares the real transitions of state changes. Then we carried out the AR model to predict the following transitions using the updating training set from previous optimal states.

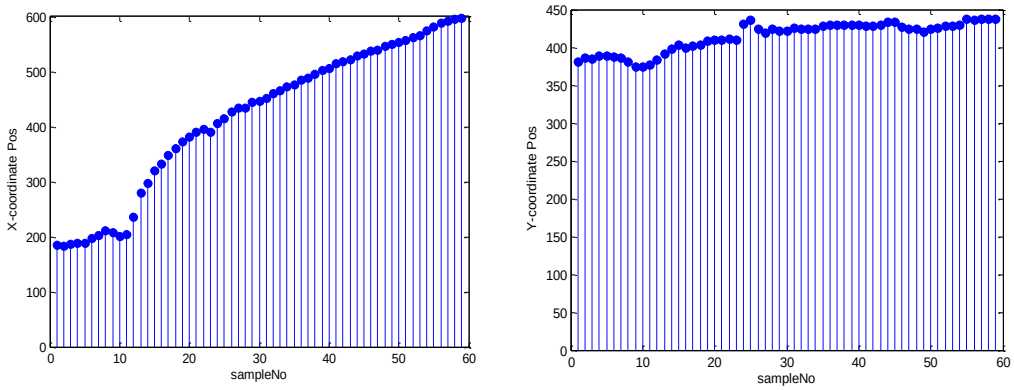


Figure 4.5 the elemental values of vectors in displaying separately along the time line. Left figure shows estimate positions in X-coordinate respect to time. Right Figure shows the estimate positions in Y-coordinate respect to time.

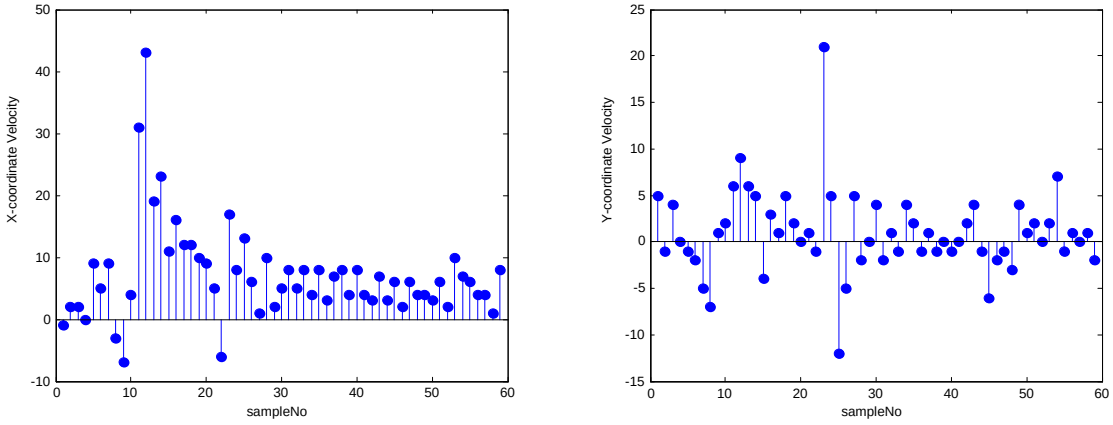


Figure 4.6 Difference vector known as transition vector is illustrated by the components individually. Left one is the transition in X-coordinate in stages, while the right one is the transition in Y-coordinates as time passing.

In our experiment, the dominant issue is that the closer prediction of transition to the optimal one, the better for late sampling strategy. Therefore, we should compare the length of distance in two dimensions. By the way, since the comparison involves some other methods, as for a better comprehension, the simple introduction is explained. In figure 4.7, the transition is predicted under the AR(3) model. And two components in the transition vector are computed independently.

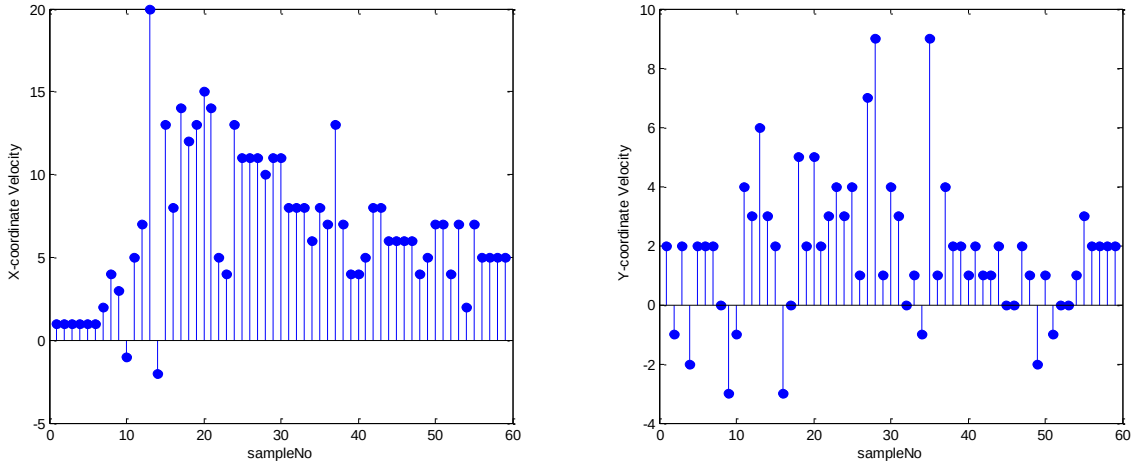


Figure 4.7 Predicted transition vectors by separately displaying in diagrams

And this prediction should be examined of the advantages and disadvantages. The distances are computed through comparing with the real transitions in figure 4.7. And this distance can be considered as the error between the prediction and the correct results. Meanwhile, we included some other methods into comparison, such as, Taylor extension by the 1st order derivative, Adaptive Average Transition Model, and also the no prediction by replacing with zero motion.

a. AR(3) model

   Details have been explained in the previous paragraphs.

b. 1st order derivative of Taylor series expansion

   ST(k) = 2×ST(k-1)-ST(k-2), where ST(k) is the state transition at time k

c. Adaptive Average Transition Model

   $\tilde{x}_k^- = \lambda \hat{x}_{k-1} + (1-\lambda)\tilde{x}_{k-1}^-$, where $\tilde{x}_k^-$ stands for the prediction of adaptive average transition from k-1 to k; and $\hat{x}_{k-1}$ is the corrected Transition from time k-2 to k-1.

The idea is obvious that average of the transitions covers the prediction, while the last transition still performs more or less heavy influence on the results. Actually, the coefficient λ, with value 0.2 in the experiment, is a tradeoff referring all previous average's function and the sudden eruption problem. If motion of the object is generally smooth with no giant sudden change, the last estimate is more or less similar to the previous average. The coefficient λ can be smaller. However, as sudden change occurs, the nearest estimate is a marvelous correction to the current prediction. Thus, the high weight cannot be ignored with coefficient larger enough.

d. Zero Motion

It obeys the original particle filter algorithm, since the particle transition is dominated only by the random generated Gaussian distribution, which also could be considered as the big noise. Then the prediction is not taken into account but only the previous state is kept as the center the searching region.

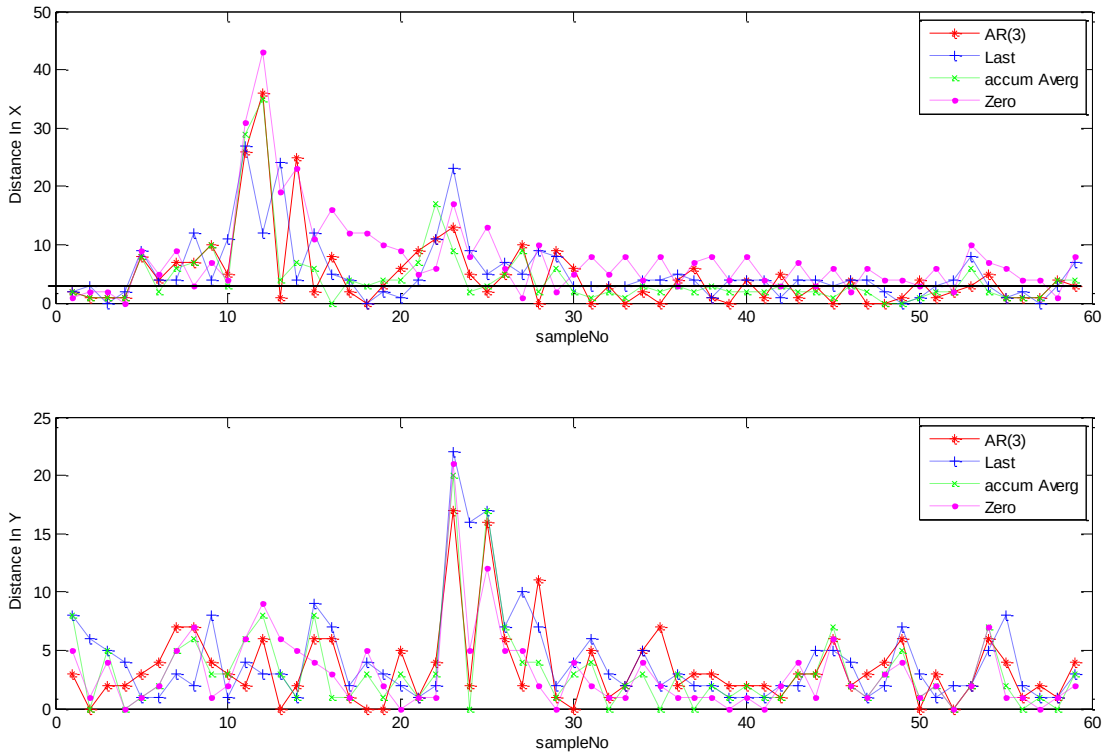


Figure 4.8 Length of difference of the predicted vector and the real transition in stages.

The upper figure shows the difference of the components in X-coordinate, while the lower figure is about the difference in Y-coordinate. In these figures, four lines in difference color have been drawn to compare with each other. The red solid line comes from method (a), the blue dotted line belongs to method (b), the Green line is the results from method (c), and the magenta line is from method (d). In figure 4.8, no method is extraordinarily better than others, as not clear line majorly stays in a lower level of distance. Obviously, zero-motion method (d) obtained the worst results, because its curve is on the top of others in most of time. However, when motion is slow, i.e. magnitude smaller than 5, it is still competitive against others. And it can be seen in Y map of figure 4.8, especially, in the sequence range between 30 and 50. Indeed, motion magnitude smaller than 2 is considered that the object is still but some perturbations exist, regarding to the size of object (much larger than 2×2). The method (b) relies on replacing the prediction with the previous estimated transition. It is similar to the method (c) except the averaging part. Actually,

averaging is a clever approach to smooth the fluctuation by introducing more information. It is observed that the method (c) not only perform better in the error distance, but also less fluctuated accordingly. The AR model method is also a good trial. Though it does not perform well in the rather small movement part (motion less than 4), it is perceived that AR model could exactly predict the little oscillated motion part from sample index 30 to 50 under considerable size of Motion. However, it still can not solve the problem of abrupt motion change (insert marvelous acceleration). Overall, the Method (c) and Method (a) are the comparatively best approaches in prediction. But in this experiment Method (c) is much smoother in prediction and takes the advantage of Method (a). As a result, Method (c) wins a bit. We found that the variance of noise is dynamic according to the Motion change after prediction. When some abrupt motion change occurs, the large variance would be introduced. In the frame domain of [10, 30], it is clear that much larger variances have been forced due to the fluctuated motion change.

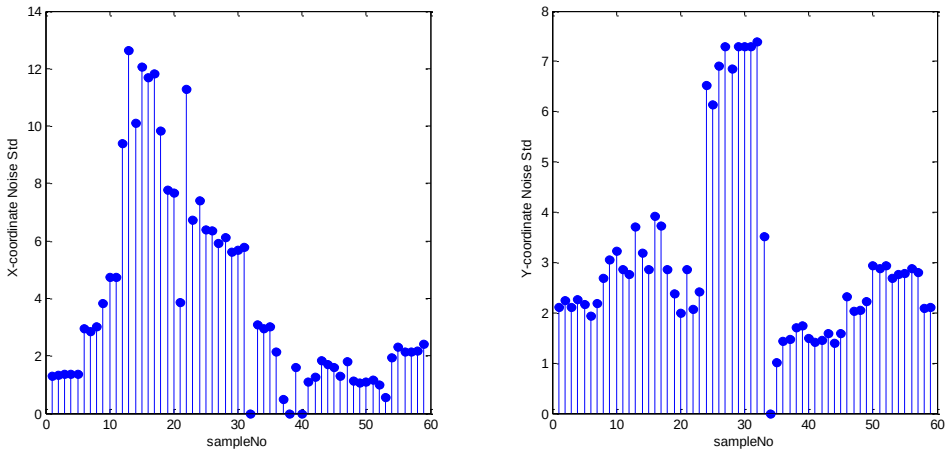


Figure 4.9 the standard deviation of noise after construction the AR(3) Model

Hopefully, we are considering bonding the variance with the coarse sampling. And it is convenient to stop using predetermined parameters but adaptive to the situation.

4.4 Adaptive sampling pattern

During the sampling process, the s^{th} step particle transition for generating new particle (i) is formulated as

$$x_{k,s}^{i} = \tilde{x}_{k,s-1}^{j} + \varepsilon_s, \tag{4.13}$$

where $\varepsilon_s \sim N(0, Q_k^s)$, which is sampled from the multivariate Gaussian distribution with covariance Q_k^s and zero mean. As we have mentioned that fixed variances are proposed for the sampling transition. It works in slow-motion videos. However, for videos with fast changes and active motions, we might have to manually adjust these parameters. Indeed the situation becomes difficult and does not allow good control for handling different video sequences. Furthermore, some large variances will create oversize sampling region which need more particles to fill in. Hence, the efficiency of particles is reduced. As shown below, we will propose our adaptive sampling pattern to adjust automatically for the time-varying requirement of sampling region due to the motion changes.

4.4.1 Adaptive averaging transition model

Similar to [10, 29], If a particle is constrained by a 2D location vector in the frame, the transition vector between stages can be predicted by

$$\bar{v}_k = \lambda \hat{v}_{k-1} + (1-\lambda)\bar{v}_{k-1}, \tag{4.14}$$

where $\bar{v}_k = (\bar{v}_{k,x}, \bar{v}_{k,y})$ stands for the prediction of adaptive average transition from k-1 to k; and $\hat{v}_{k-1}$ is the estimated transition from time k-2 to k-1. Accordingly, it belongs to the model of method (c) in the previous section. The idea is obvious that the average of transitions could cover

the prediction, but the last transition still perform more or less heavy influence on the results. Actually, coefficient λ, with value 0.2 in the experiment, is a tradeoff taking care of all previous average's function and sudden eruption problem.

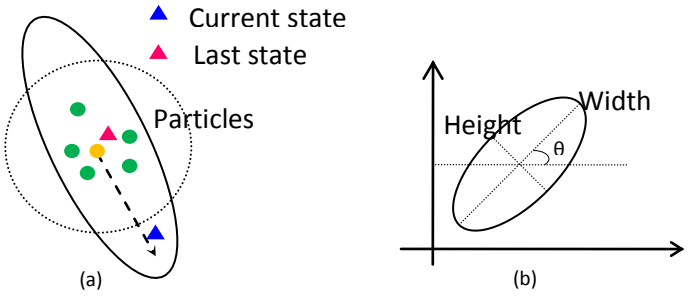


Figure 4.10 (a) Simulation of comparison between the new sampling pattern and the standard one. (b) The description of sampling pattern in diagram.

4.4.2 Adaptive covariance matrix

The covariance matrix Q can be Eigen-decomposed [51] into the following

$$Q_k^s = R_k{}^T P_k^s R_k, R_k = \begin{bmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{bmatrix}, P_k^s = \begin{bmatrix} \sigma_{w,k}^2 \cdot \rho^s & 0 \\ 0 & \sigma_{h,k}^2 \cdot \rho^s \end{bmatrix} \quad (4.15)$$

where R_k is the rotation matrix with count-clockwise angle θ to the positive horizontal direction, P_k^s is the diagonal covariance matrix, where the non-zero elements are the variance alongs the two perpendicular axis directions. P_k^s controls the shape of the sampling region. $\sigma_{w,k}^2$ and $\sigma_{h,k}^2$ are the starting variances on kth frame before sampling in steps. ρ = 0.8 is the constant to shrink the sampling region as step (s) incrementing. In figure 4.10(a), the color blobs are particles. The dash-line arrow shows the predicted state transition. The dash-line circle stands for the sampling region with 98% of confidence by the pattern only using the diagonal covariance matrix. Similarly, the solid-line ellipse declares our new pattern for sampling. According to figure 4.10

(b), it is observed that the sampling region owns its orientation, which is rotated by adding constraint R_k. Figure 4.10(a) shows our new pattern can limit some unnecessary transition by assigning lower probability. Conversely, the designated area will be more probably covered, which improve the sampling efficiency. Intuitively, the current state is more easily reached by using our adaptive pattern.

The rotation angle θ_k can be approximated by the orientation of the predicted transition vector obtained in eqn (4.13). If the vector length $D = \sqrt{\left(v_{k,x}^{-}\right)^2 + \left(v_{k,y}^{-}\right)^2} \geq 8$, i.e. the predicted transition distance is long enough, update θ_k with the predicted vector components $v_{k,x}^{-}$ and $v_{k,y}^{-}$, as

$$\theta_k \approx \arctan\left[\mu \frac{\left(\left|v_{k,y}^{-}\right| + 0.1\right)}{\left(\left|v_{k,x}^{-}\right| + 0.1\right)}\right], \ \theta_k \in [-0.5\pi, 0.5\pi] \tag{4.16}$$

where μ is the sign of $v_{k,y}^{-} \times v_{k,x}^{-}$.

Let us consider that the assumption of the law of inertia exists. The direction perpendicular to the motion vector has less probability with state occurring. Thus we can reform the pattern shape for more proper sampling region by adjusting the value of elements in Diagonal matrix P. Accordingly, we lengthen the axis (width) following orientation of the predictive vector, and shorten the perpendicular one (height).

$$\sigma_{w,k} = \begin{cases} \beta D, \ \beta D > \sigma_{w,0} \\ \sigma_{w,0}, \text{otherwise} \end{cases}, \sigma_{h,k} = \gamma \frac{\sigma_{w,0}\sigma_{h,0}}{\sigma_{w,k}} + (1-\gamma)\sigma_{h,0} \tag{4.17}$$

where β (β=1.2) is the coefficient which expands the pattern with higher probability of the vector. $\sigma_{w,0}$ and $\sigma_{h,0}$ are the initially known parameters of standard deviations, i.e., $\sigma_{w,0} = \sigma_{h,0} = 10$ in our implementation. γ (γ=0.2) is the multiplier that can compensate some extreme situation while

magnitude of the predictive vector is too large. At least the variance at the perpendicular direction should not be too small to deal with the motion diversity. Thus, combining eqns (4.15), (4.16) and (4.17), the new sampling pattern can be updated in the frames with acceptable tracking results. In our implementation, as for robustness, the vector orientation domain is uniformly quantized into 6 angle levels.

4.4.3 Experimental results

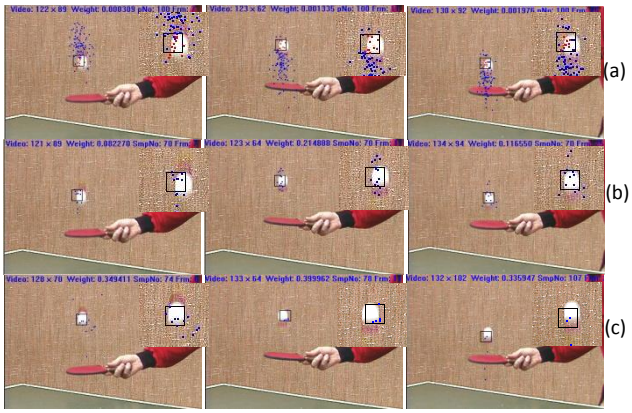


Figure 4.11 Tracking results of the table tennis sequence after applying adaptive Sampling patterns to the existing methods, i.e., SIR PF (a), Annealed PF (b), HY (c).

As the velocity of the targeted objects varies much in the four sequences, the empirical sampling parameters should be adjusted differently for each sequence. Otherwise, the sampling region may not be good enough to reach the target, or its sample density is too low, while both result in wrong samples. Hence we applied our adaptive sampling patterns to these existing three methods. The results are illustrated in figure 4.11. As an abrupt change happens, the adaptive pattern reacts to update itself. All of them are able follow the motion of the ball rather than losing it in figure 3.3. It indicates that the adaptive patterns can ensure the algorithm much more robust in handling varying motion situation.

To sum up, we have observed that our proposed method is the most stable algorithm in addressing sudden acceleration and occlusion problems and it also allows efficient computation.

Chapter 5. New Target Models and Observation Handling

5.1 Introduction

Generally, a target object in a video sequence is included in an image region of pixels. Template matching is a possible way for matching, but usually it is difficult to model the object. It is because the complex and confusing pixel region is difficult to handle. Many kinds of models have been established according to different features in need. The feature extraction and typical modeling methods have been described in the previous chapters. Currently, the color histogram is the most used representation of the target. Even though it is invariant to scale change and rotations, the similarity results between target and candidates usually can not correspond to the real ones. Therefore, we have improved the existing color histograms into a new model. First, the multi-partition color histograms [46, 51] are adopted to settle problems of partial occlusion. We segment the target window into non-overlapping blocks. Each partition of the target constructs one histogram. Subsequently, while computing the similarity, the effectiveness of different partitions is determined according to their weights. The weights dominate the influence of partitions to the whole similarity result. Second, we propose a new weighted histogram by enhancing the foreground and attenuating the interference of background. Note that the background information should be distinct from the foreground information. However, some features in the background still contain the foreground area. An improper use of the wrong features may result in indiscriminative similarity results. Therefore, we should select some important features and enhance them, while reduce the effects of features mixed in the background. Here, aiming at a better color histogram, we have designed a scheme that obtains redistributed bin weights for the histogram-based representation. The distance between the

foreground and background controls the weight assignment. As the model is different from the single color histogram, we should update the similarity measurement to fit this particular target model. It can be simplified into one expression for computational efficiency. Furthermore, in considering the appearance change of the target, only one static model is not practical enough for the long video sequence with non-rigid object in it. Therefore, we have exploited some dynamic multi-reference models which can automatically capture the appearance change. Meanwhile, as the object scale varies, the newly generated reference models are easily corrupted by the background information. Thus, a simple scale selection mechanism is proposed to find out the best matched target size.

5.2 Multi-partition target representation

It is mentioned that the structural histogram has been adopted by several researchers. We also adopt this as our starting stage to improve it in a better formulation. At beginning, the partition should be split out from a rectangular window. This has been introduced in the introduction part. We can divide the window into several non-overlapping partitions. And we do not limit the segmentation with just 2×2 structure. Let us predefine it to have $N_w \times N_h$ blocks, where N_w is generated from the width and N_h is obtained from the height. Therefore, it is useful for objects with different shapes. We can design different types of divisions for videos with different object tracking purpose. It is known that the partition contains partial features of the objects. As the object appearance changing, the features of each partition may vary significantly from the target. Thus we establish a weight-voting scheme to evaluate every partition. Those higher weight partitions have more interference on the similarity results. In this case some partial occlusion would give no harm in the similarity measure.

5.2.1 Description of representation

In this work, we still take the color histogram as our key form of feature representation. Some modifications have been carried out to perform a new structural histogram. For a window divided into M ($N_w \times N_h$) blocks, the structural histogram is actually a set of histograms. As an example in figure 5.1, the rectangular image region of a man is shown. It has been divided into 2×2 sub-regions (partitions). Then we should extract four histograms from each partition. Globally, also the overall region has its histogram. Therefore, totally we should have 5 (M+1) histograms forming a set of structural histograms. In figure 5.1, we stretched out these 5 elemental histograms.

Let us derive the representation in the term of structural histogram $\mathbf{p}(y) = \{p_i(y)\}_{i=0\ldots M}$. It composes of M+1 partial histograms. Separately, we can write their constructed their formulations as follows.

1) The overall histogram $p_0(y) = \{p_0^u(y)\}_{u=1\ldots m}$, which follows similar expressions as mentioned in chapter 2.1.1.

$$p_0^u(y) = f\sum_{j=1}^{I} k(y - x_j)\delta[h(x_j) - u], \quad f = \left(\sum_{j=1}^{I} k(y - x_j)\right)^{-1} \tag{5.1}$$

where I is the number of pixels in region 0, $h(x_j)$ gives the output the corresponding bin index of the pixel at x_j, u is the current bin number, k(.) is the spatially convex kernel function that output the coefficient between 0 and 1, and f is the normalizing constant.

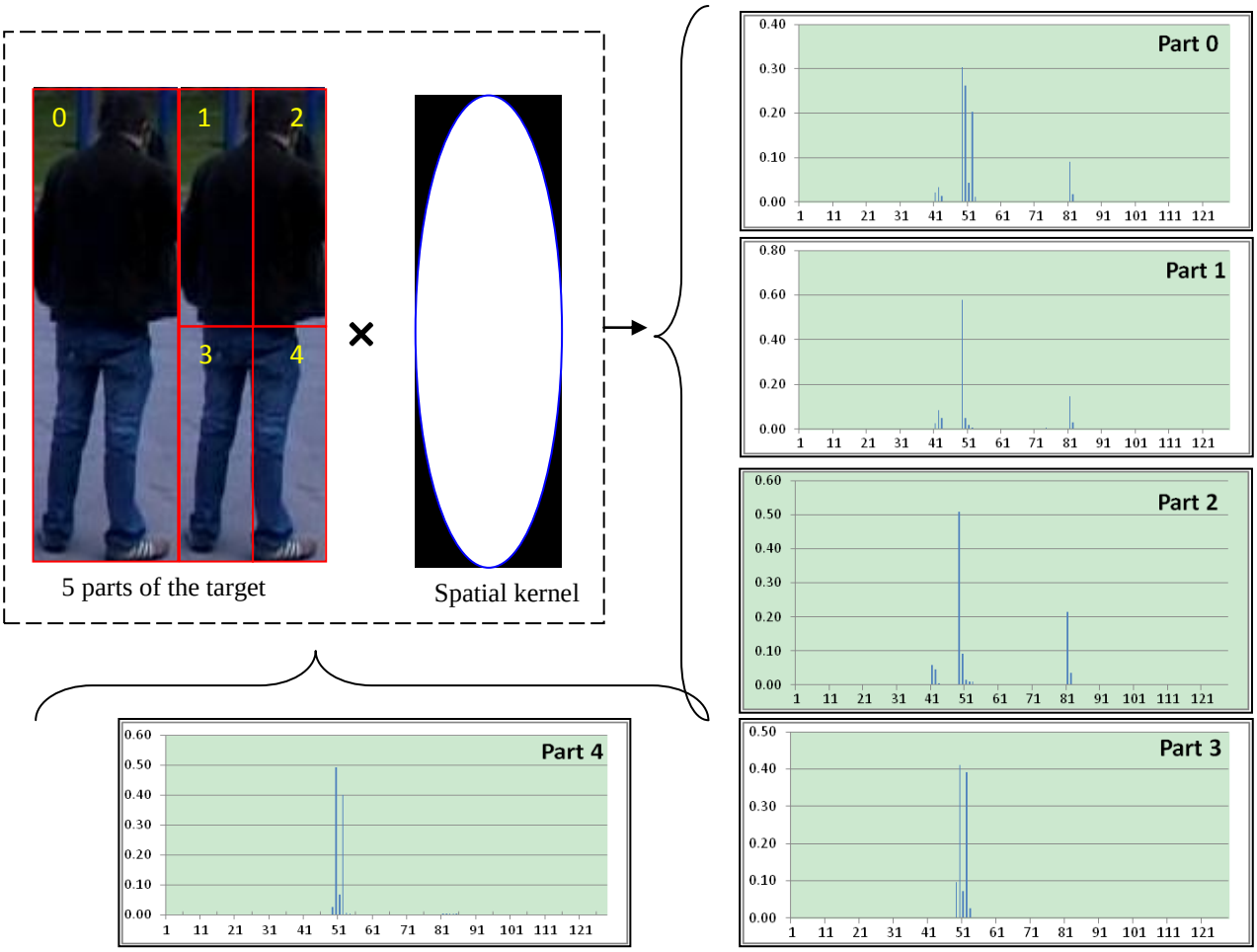


Figure 5.1 the stretched structural histogram set, which is composed by the 5 histograms (YUV 8×4×4) including one overall histogram in part 0 and partial histograms from part 1 to 4.

2) The i[th] partitioned histogram $p_i(y)=\{p_i^u(y)\}_{u=1\ldots m}$ (i>0)

$$p_i^u(y)=f_j\sum_{j=1}^{I_i}k(y-x_j)\delta[h(x_j)-u],\quad f_i=\left\{\sum_{j=1}^{I_i}k(y-x_j)\right\}^{-1} \tag{5.2}$$

where $p_i^u(y)$ is the value in bin u, I_i is the pixel number in the sub-region i, f_j is the current normalizing constant.

As a result, the structural histogram is the combination of those (M+1) histogram, which is used for the representation of target.

5.2.2 Partition weight

Indeed, the contribution of each partition varies as the object moves. Because the object will not keep in the same shape, some partitions fail in obtaining a good similarity, i.e. they are less effective than others. Then an algorithm should be generated to identify the deceasing contribution of partitions and reduce their weight accordingly. On the opposite side, if some partition can match perfectly with the reference, we shall give more weight to it to confirm it as the possible target. Therefore we should add weight parameters for partitions. Let us denote w_i as the symbol of weight for the i[th] partition ($i \in [1, M]$).

Initially, as no evaluation of the partitions is given, we can predefine their weights. In paper [53], authors did some similar work by assigning spatial weights for partitions. In our scheme, it is slimily done by the following assignment

$$w_i^{init} = w_i^{sp}, \tag{5.3}$$

where the initial weight w_i^{init} of the i[th] partition is its spatial weight w_i^{sp}. Note that w_i^{sp} is the ratio that is the percentage of pixel weights occupied in this part compared with that of the whole window. Mathematically, $w_i^{sp} = \frac{f}{f_i}$, where f is the normalizing constant for histogram formulation of the target window, and f_i is also the normalizing constant for the i[th] partition.

After initialization, the weights should be updated according to their performance in the final estimate. Note that the weight can not meet sudden changes in case of wrong estimate at some time. The weight is gradually updated by the current likelihood. Thus, the new weight of the i[th] partition can be derived as an updating process, i.e.,

$$w_i^k = 0.2 \times w_i^{k-} + 0.8 \times w_i^{k-1}, \tag{5.4}$$

where w_i^k is the assigned i[th] partition weight at frame k, w_i^{k-1} is the contribution response for i[th] part after frame k-1.

Then, w_i^{k-1} is computed by the equation below, namely,

$$w_i^{k-} = c w_i^{sp} L_i^{k-1},\ c = \left\{ \sum_{i=1\ldots M} w_i^{k-1} \right\}^{-1} \tag{5.5}$$

where, w_i^{k-1} is the contribution response after frame k-1, L_i^{k-1} is the likelihood output of the ith part given the estimated target, and C is the normalizing constant.

As shown in eqn(5.5), we take the likelihoods of partitions to update their weights. Thus, when some part gradually losing its features stored in the reference one, its weight is forced down to a lower level. The similarity results will not change significantly for a varying target in the temporary domain. Subsequently, those close candidates will be kept rather than discarded.

5.2.3 Background-foreground-weighted histogram

Since the clutter background misleads our target identifications, people tried methods to reduce its harmful influence. In [9], Comaniciu et al. proposed a background-weighted histogram (BWH) to decrease the interference of background features. They derived a simple histogram of background region and select only the salient feature (e.g. color bins) from the Target and candidates. Technically, they attempt to decrease the weight of prominent background features. Ning et al. in [54] pointed out their incorrect use of BWH and enhanced it. In [55], the authors improved this via a comparison between the color histograms of foreground and background.

However, its mechanism is complex in setting up thresholds and equations. Let us propose our Background-foreground-weighted histogram. It works well and gives stable tracking of object with enhanced prominent features.

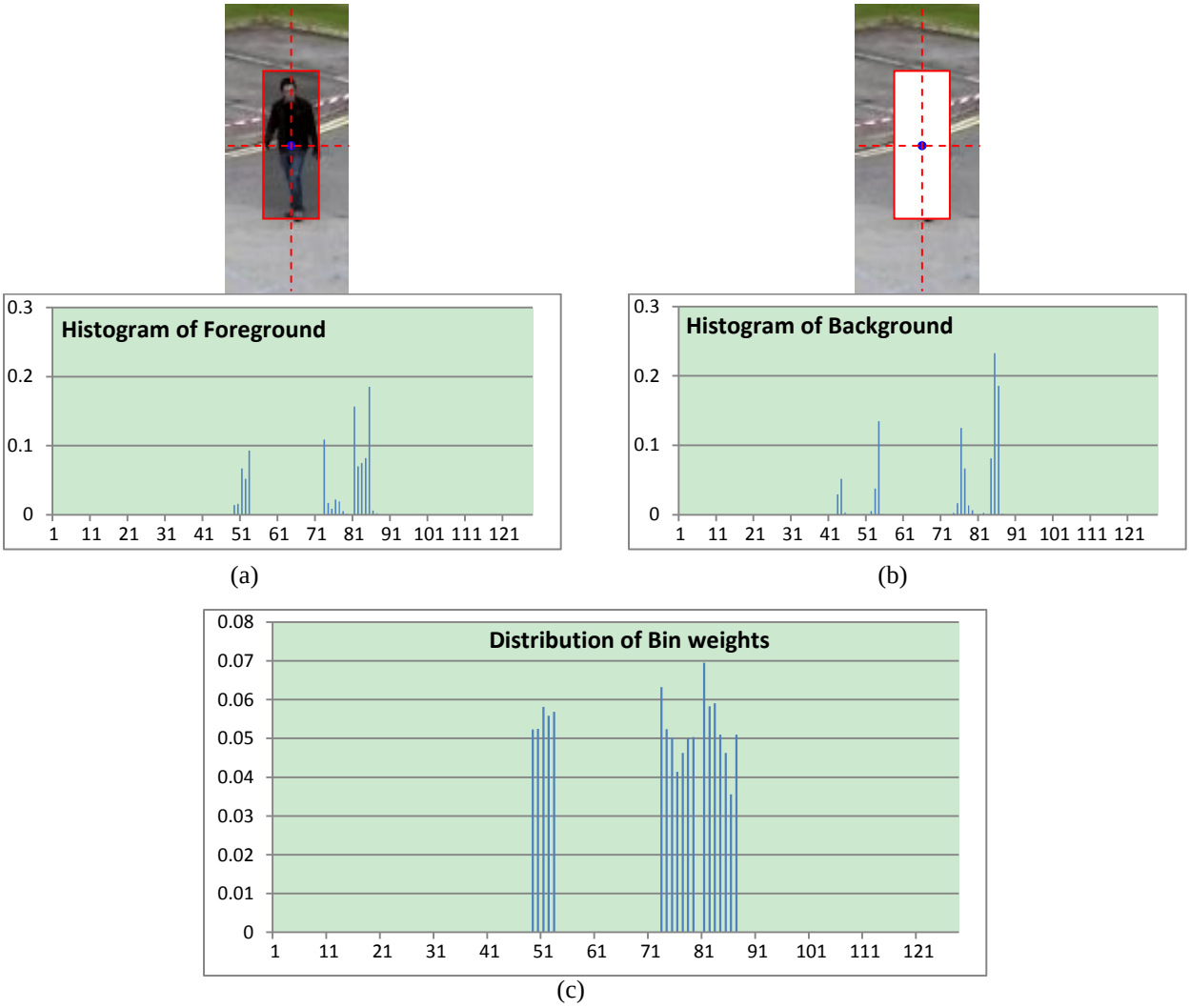


Figure 5.2 An intuitive example for Bin weight distribution extracted from background and foreground. The upper-left image in (a) with pixel region covered by gray color is the foreground window. The histogram below is the corresponding part of representation. The upper-right image in (b) excluding the white region is the background area with histogram shown underneath. The diagram in (c) illustrates the output Bin weight distribution.

It is good if the background histogram can also be extracted. We select a rectangle (such as in figure 5.2(a)) with the double size of the object in the center area. The pixel region except the object window (foreground) is recognized as the background area (in figure 5.2(b)). Then

according to the process mentioned before, we can obtain its color histogram without embedding the kernel in the spatial domain. Let us denote $p_{BG}(y)$ as the color histogram of background area. It can be observed in figure 5.2 that the background contains some pixels with the same color bins in the foreground regions. We proposed an equation to decrease its interference and enhance some bins which occur occasionally in the background. Its result is shown in figure 5.2(c). As for an example, we take $p_0(y)$ in the structural histogram of foreground. Consequently, the bin weight U_0^u can be derived as

$$U_0^u = \begin{cases} C_{U_0} \exp\left[\lambda_U \left(p_0^u(y) - p_{BG}^u(y)\right)\right], & \text{if } p_0^u(y) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

where λ_U is the constraint constant to limit the domain of bin weight in U_0. In our implementation, we select λ_U equal to 2.0. Thus U_0^u will not own too small or too large value ($U_0^u \in (0.13, 7.39)$), namely, the feature integrity can be preserved. Meanwhile, the "special" bins, which is different from background ones, can perform more in the role of similarity measurement process. C_{U_0} is the normalizing constant, where $C_{U_0} = \left\{ \sum_{u=1\ldots Max} U_0^u \right\}^{-1}$. (5.7)

As a result, the formulation of the structural histogram should be updated by adding this bin weight into histograms, i.e.

$$\begin{aligned} p_0^u(y) &= f\left[\sum_{j=1}^{I} k(y - x_j)\delta[h(x_j) - u]\right] U_0^u, \\ p_i^u(y) &= f_i\left[\sum_{j=1}^{I_i} k(y - x_j)\delta[h(x_j) - u]\right] U_i^u, i > 0 \end{aligned} \tag{5.8}$$

where f and f_i are the new normalizing constants.

5.3 Simplification of likelihood computation for new target model

From these sections, we can finally establish a new target model both for the reference (q) and candidates (p) in eqn (5.8). They should be compared to output a similarity result. The Bhattacharya coefficient (BC) is the most popular similarity measurement approach. We also take advantage of it as the core algorithm. It is known that we have multiple histograms. The integration of similarity result should be exploited.

Firstly, we propose an integrated Bhattacharyya coefficient ρ_M. It is computed to gather the average of the partial histograms. So, it can be written as

$$\rho_M = \sum_{i=1\ldots M} \rho_i w_i^k \ , \tag{5.9}$$

where the integrated BC ρ_M is weighted the average of the Bhattacharyya coefficients (ρ_i) of corresponding partitions, w_i^k is the weight of the ith partition on frame k.

Meantime, we have to compute the similarity of the overall histogram $p_0(y)$. We give the symbol ρ_0 to represent this original BC.

Finally, the effective Bhattacharyya coefficient ρ can be derived from making use of the integrated BC ρ_M and original BC ρ_0.

$$\rho = \frac{\rho_M + \lambda_\rho \rho_0}{1 + \lambda_\rho} \ , \tag{5.10}$$

where λ_ρ is the parameter that controls the significance of original BC. For simplicity, $\lambda_\rho = 1$.

Respectively, the likelihood (L) is another explanation of similarity measurement. In our work, it is the exponential output when Bhattacharyya Distance (D) is used as input.

$$L = \exp(-\lambda_{sim} D) \,, \tag{5.11}$$

where $D = 1-\rho$, and λ_{sim} is the parameter which adjusts the slope of likelihood difference. λ_{sim} is selected empirically, e.g. λ_{sim}=10, from our wide experimental testing results.

This Likelihood equation can then be further decomposed as

$$\begin{aligned} L &= \exp\left[-\lambda_{sim}\left(1-\frac{\rho_M + \lambda_\rho \rho_0}{1+\lambda_\rho}\right)\right] \\ &= \exp\left[-\lambda_{sim}\left(\frac{1-\rho_M}{1+\lambda_\rho}\right)\right] \cdot \exp\left[-\lambda_{sim}\lambda_\rho\left(\frac{1-\rho_0}{1+\lambda_\rho}\right)\right] \\ &= \exp\left[-\lambda_{sim}(1-\rho_M)\right]^{\frac{1}{1+\lambda_\rho}} \cdot \exp\left[-\lambda_{sim}(1-\rho_0)\right]^{\frac{\lambda_\rho}{1+\lambda_\rho}} \end{aligned} \tag{5.12}$$

Indeed the two terms in the braces can be identified as the Likelihood value of some candidates to the model. Note especially for the second term, $\exp\left[-\lambda_{sim}(1-\rho_0)\right] = L_0$. L_0 is the likelihood value for candidate with the overall histogram p_0.

In eqn(5.12), the first term can be substituted by eqn(5.11), we then have,

$$\begin{aligned} L &= \exp\left[-\lambda_{sim}(1-\rho_M)\right]^{\frac{1}{1+\lambda_\rho}} \cdot L_0^{\frac{\lambda_\rho}{1+\lambda_\rho}} \\ &= \exp\left[-\lambda_{sim}\left(1-\sum_{i=1\ldots M}\rho_i w_i^k\right)\right]^{\frac{1}{1+\lambda_\rho}} \cdot L_0^{\frac{\lambda_\rho}{1+\lambda_\rho}} \end{aligned} \tag{5.13}$$

As $\sum_{i=1\ldots M} w_i^k = 1$,

$$L = \exp\left[-\lambda_{sim}\left(\sum_{i=1\ldots M}(1-\rho_i)w_i^k \right) \right]^{\frac{1}{1+\lambda_\rho}} \cdot L_0^{\frac{\lambda_\rho}{1+\lambda_\rho}}$$

$$= \exp\left[-\sum_{i=1\ldots M}\lambda_{sim}(1-\rho_i)w_i^k \right]^{\frac{1}{1+\lambda_\rho}} \cdot L_0^{\frac{\lambda_\rho}{1+\lambda_\rho}}$$

$$= \left\{ \prod_{i=1\ldots M} \exp\left[-\lambda_{sim}(1-\rho_i) \right]^{\frac{w_i^k}{1+\lambda_\rho}} \right\} \cdot L_0^{\frac{\lambda_\rho}{1+\lambda_\rho}} \quad (5.14)$$

Actually, $\exp\left[-\lambda_{sim}(1-\rho_i) \right] = L_i$, which is the likelihood of ith partition while the corresponding Bhattacharyya Coefficient is ρ_i.

At last, the final likelihood L is simplified as the multiplication of element-wise likelihoods with their power components.

$$L = \left\{ \prod_{i=1\ldots M} L_i^{\frac{w_i^k}{1+\lambda_\rho}} \right\} \cdot L_0^{\frac{\lambda_\rho}{1+\lambda_\rho}} \quad (5.15)$$

where L_i and L_0 are likelihoods of elemental histograms in the structural histogram set, w_i^k is the weight of histogram from ith partition. λ_ρ is the constant parameter that controls the role of the similarity resulting from the overall histogram.

Via eqn (5.15), we can simplify our program code with simpler structure and fast processing speed. Also, this expression intuitively tells us the role of each element in the structural histogram of the target representation. Meanwhile, those partition weights can be seamlessly involved in the similarity measurement.

5.4 Target model updating

5.4.1 Linear updating by successful estimate

In the previous implementations, the static model was employed to act as reference in similarity measurement. However, experiments have indicated that the similarity of estimate decreased as time goes on. Because the object would not keep a steady view in the camera, and their appearance varies. Thus, the so-called online updating is inevitable to be used to preserve the long-time tracking. Researchers in [51, 56] have studied the linear updating process. The histogram-based representation of the target model can be updated by that of the estimate. A coefficient α contains the linear relation with the current estimate view and the reference model. Thus, New_reference = (1- α) Old_reference + α Current_estimated_object. This updating algorithm happens when the current estimated object is much similar to the reference but still a little different from it, e.g. the similarity value or likelihood probability is larger/higher than a particular threshold Th_r (Th_r = 0.3). Empirically, many authors select 0.1 as the constant coefficient.

In implementation, the reference model q_k at frame k is updated by the previous model q_{k-1} and the estimate p_{k-1}, i.e.,

$$q_k = \alpha p_k + (1-\alpha) q_k . \tag{5.16}$$

Accordingly, the new reference model in histogram is more close to the changed object appearance. However, this approach contains its limitations. Firstly, in the real sequence, the object appearance change must not follow the linear updating process. The estimate can not be regarded as the real target state. Therefore, error still exists in the estimate. When the update repeats for many times, the problem of error propagation is inevitable. Even worse, the new

reference model will lose characteristics of the target, namely, the model can not represent the object well. Second, several constraints need to be satisfied when allowing updating the reference model. However, the intended conditions cannot cover all the situations. If the object has suddenly changed its appearance and later its keeps steady, according to the condition provided before, no updating process is handled. Indeed the current model is wrongly recorded. Consequently, the similarity measurement is meaningless because of incorrect comparison to the target. So we should design a new flexible and feasible target model for the updating scheme.

5.4.2 Dynamic target model selected in sets

Here, we will propose our new Target Model updating algorithm. It includes a brand new symbol (T) to represent the "reference set". All reference models are contained in this set. Each time, we will pick up the most probable one to perform as the next reference for similarity measurement. Consistently, the dynamic reference model needs update in the set. Some useless reference candidates can be replaced by the new one generated from the estimates. Still, the estimates of past frames are stored in a back-up set, which is called "reference back-up set". It can be denoted by the symbol B. The dynamic model update approach is motivated by paper [57], where the authors used a set of templates as the object model, which are updated according the assigned weights and a similarity threshold. We modified it to fit our histogram-based model. Comparing against using a set of templates, we only pick up the most matched reference candidate as the object model at a time. Meanwhile, during the tracking process, we collect the used times of those candidates acting as standard reference. The fewest used candidate will be discarded and replaced by an existing estimate. Details are discussed in the following sections.

5.4.2.1 Reference set and candidate model selection

Initially, we should establish a new reference set. In frame 0, due to no temporal information provided, some references **q** (R = 8 is the number of references in our implementation) are manually selected around the designated position. Among these references, one in the structural histogram set is exactly extracted at this position. The others are generated via perturbing only a few pixels around it. We use Gaussian random variables to answer the perturbation. For example, in figure 5.3, sequences of small images for car, soccer and humans are shown. They are used to extract the histogram-based models for corresponding video sequences. We can identify their images which come from different positions. Therefore, these references are different between each other in the set. Thus the reference set is created at the beginning frame before the tracking process is started.

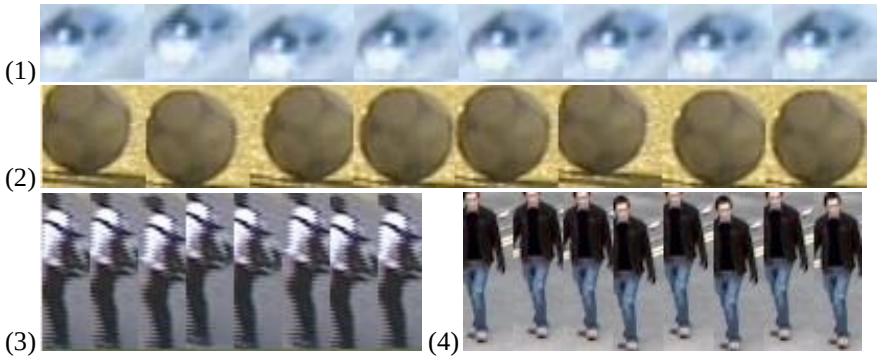


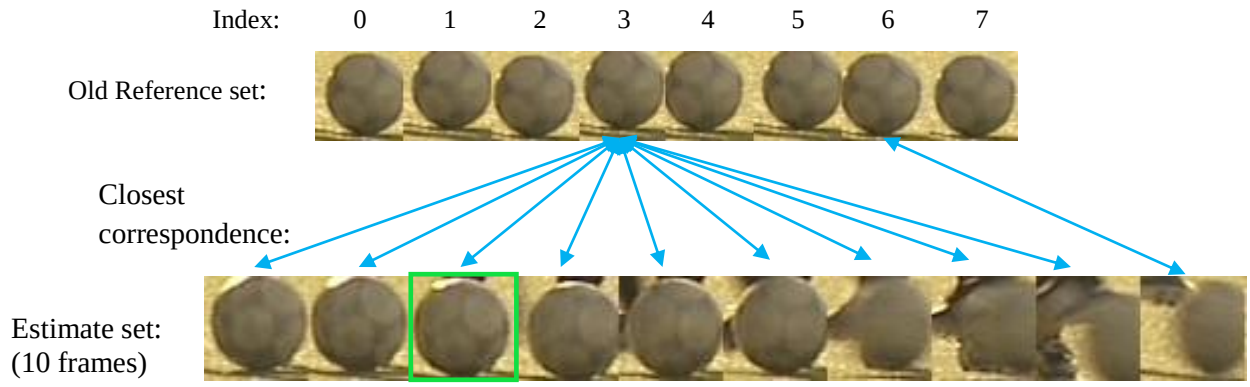
Figure 5.3 The templates of the reference set. Each template will extract its structural histogram as candidate model. The sequence in (1) shows car images. (2) contains the soccer templates. (3) and (4) are images of human beings for reference.

We then have to select one of these references to act as the current designated reference. Above all, we have to admit an assumption that the neighboring two frames will have only little variation in appearance. Given this condition, we can select the appropriate reference model based on the estimate results at the last frame. Suppose, before k^{th} frame starting, we examine all the references in the set by computing the similarity results with the estimated "target". The top similar one will be adopted as the next reference. Also we will record the least similar one.

Subsequently, when applying reference set updating, the most frequently recorded reference will be replaced by another newly proposed.

5.4.2.2 Back-up set for reference entry

As mentioned, we have an online-reference updating approach. However, it is not reliable to track object with only those fixed initial references. In real application, the candidate reference can also be replaced in order to adapt to the varying pose change. A back-up set (B) is created to contribute to its online learning method. To achieve this, we segment the video into parts with a fixed time interval F, e.g. F=10 frames. The set is filled by estimates with structural histograms in one interval. When the set is full, we seek the local maximum of similarity result with its corresponding reference and decide whether the reference set can be updated. If the local maximum is larger than a threshold (Th_r), we select this estimate to replace one in the reference set. Otherwise, no replacement is made, namely, the reference set is not updated.



Frame_Index	10	11	12	13	14	15	16	17	18	19
Max_Sim_Index	3	3	3	3	3	3	3	1	3	6
Min_Sim_Index	6	6	6	6	6	6	6	2	6	3
Max_Sim	0.72	0.82	0.98	0.97	0.45	0.81	0.95	0.24	0.36	0.47

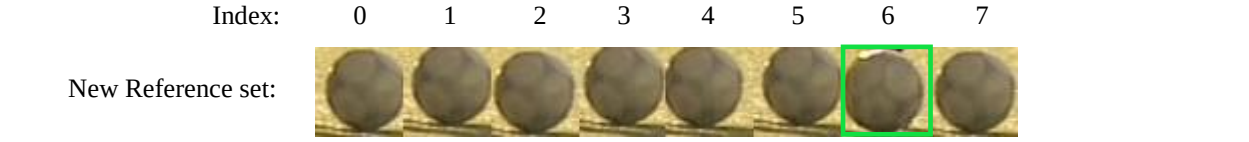


Figure 5.4 Example of target model updating and entry of new candidate reference.

For example, in one time interval from frame 10 to 19, we can see the operations in figure 5.4. In this particular interval, the estimate will be carried to examine the similarity with all candidate references. The most matched one will be considered as the closest correspondence to the estimate. Then in the next-frame tracking, the "official" reference will be this one (the third reference). It repeats until the end of the interval. After that, we will take advantage of the full set as the back-up set. Only when the top similarity among all elements is larger than the Th_d (0.15), the most similar estimate (at frame 12) will be inserted to the reference set with the index of the discarded one, e.g. 6 in this example. Because it happens mostly as the least similar reference, it is regarded as the weakest reference adapting to the new changing target and environment.

5.4.3 Overall dynamic target model updating approach

This model update scheme is summarized in table 5.1. It includes the previously mentioned details. We not only apply this model selection approach but also adopt the linear updating for closer reference to the current target appearance. At the starting frame, we choose the random distributed positions in a small area for creating reference models. Thus, this method keeps the objects still with the major appearance and preserves some differences with each other. As sequence moves on, the reference model is updated according to similarity feedbacks, which is shown in the form of likelihood. As the object changes gradually, the temporarily neighbouring target views must have similar appearance. We take this property to select the most matched reference and also further update it linearly. Because the reference set can not cover long-time object models, we can replace some low similar reference with the newly proposed estimate after every fixed interval. Experimental results have proven the robustness and effectiveness of our approach.

Table 5.1 the algorithm of Reference Model selection and Reference set updating

Initialization:

At frame 0, given the initial state vector y_0 and pixel region I_0 in rectangle window.

- Initial target model in structural histogram $p(y_0)$ can be extracted from I_0.
- Create a reference set T_0 by position perturbations and histogram Extraction,
  i.e., $T_0 = \{\mathbf{q}(y_{0,i})\}_{i=0,\ldots R}$, where $y_{0,i} = y_0 + \Delta y_i$ and Δy_i is a small Gaussian random vector.

While the frame index k>0,

Target Model Update:

$\hat{y}_k$ is newly estimated vector at frame k. the reference set is $T_k = \{\mathbf{q}(y_{k,i})\}_{i=0,\ldots R}$.

- Apply the estimate to compute Likelihoods ($L(\hat{y}_k, y_{k,i})$) with all references.
- Find the reference containing maximum likelihood. (its index is marked by j)
  $j = \arg\max\left(\left\{L\left(\hat{y}_k, y_{k,i}\right)\right\}_{i=0,\ldots,R}\right)$
- **If** $L(\hat{y}_k, y_{k,i}) > Th_r = 0.3$,
  - **then**, the next-frame reference $\mathbf{q}(y_{k+1})$ is obtain by linearly updating newly selected reference $\mathbf{q}(y_{k,j})$ with estimate $\hat{y}_k$, where $q(y_{k+1}) = \alpha q(y_{k,j}) + (1-\alpha)\, p(\hat{y}_k)$.
  - **Otherwise,** $q(y_{k+1}) = q(y_{k,j})$.
- **END IF**
- Fill histogram-based representation of estimate $\hat{y}_k$ into the reference back-up set $B_n = \left\{p(\hat{y}_k)\right\}_{k\in[nF,(n+1)F-1]}$.
- Write down the maximum likelihood $L(\hat{y}_k, y_{k,j})$, its reference index j, and the index containing the minimum likelihood.

Reference set update :

- **IF** k=(n+1)F-1,
  - Examine the reference set and observe the most frequent index *l* with minimum likelihoods for past F estimates.
  - Examine this interval and find out the representation of the maximized likelihood in records. Suppose t is the optimized index from estimates.
  - **If** its likelihood $L(\hat{y}_t, y_{t,j}) > Th_d = 0.15$,
    - Update the reference set. $\mathbf{q}(y_{k,l}) = P(\hat{y}_t)$
    - Otherwise, no update.
  - **End if**
- **END IF**
- Pass the reference set to the next frame, where $T_{k+1} = \{\mathbf{q}(y_{k,i})\}_{i=0,\ldots R}$

5.5 Target scale estimation

It is known that histogram-based representation is invariant to scale and rotations. In other words, if the kernel matches well with the target window, the histogram will be almost the same to that of newly scaled one. In our work, scale is denoted as "scl". For example, in figure 5.5, if we create a luminance (value in [0, 1]) histogram for a rectangular target region, the half-sized region gives a histogram with very similar pattern. We can check that both regions can have nearly the same histogram output as we apply a matched kernel with the same scale format.

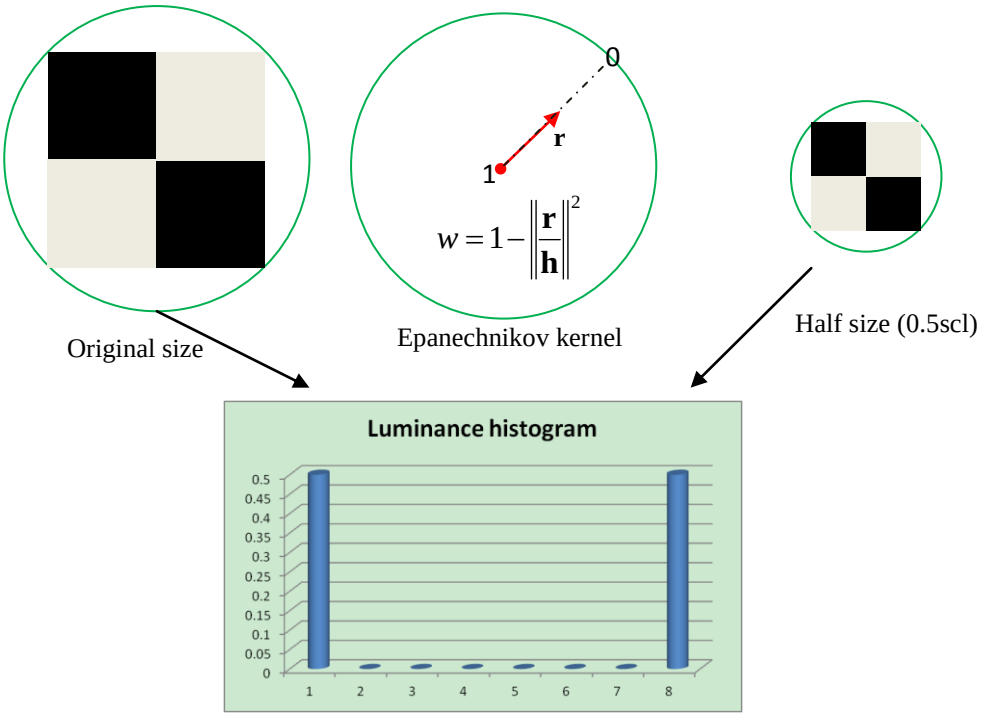


Figure 5.5 An example that a scaled target having the same histogram representation if the spatial kernel follows its scale change

Reference [9] has stressed the importance of scale invariance. Via running target localization algorithm separately in three different scaled bandwidths (**h**, 0.9**h** and 1.111**h**), this approach successfully estimates the target scale. In reference [58], it has been further discussed by exploring a scale space for better choice of scale. Motivated by this idea, we come out a method

by applying different kernels to a target region for a kind of representations in both matched and mismatched formulation. Since it is observed that the outlier pixels in the target region have weights close to zero, namely, their contribution to the histogram can approximately neglected.

Therefore, as in figure 5.6, we indicate multiple histograms after applying differently scaled kernels correspond to the reciprocally scaled object regions. In the particle filtering approach, the particles are sampled to evaluate its similarity and weight. It will only choose one scale for all particles to the corresponding pixel region for feature extraction. Size of such regions is applied with the estimated scale at the previous frame. We have then to collect similarity results through comparing the obtained representation with those three scaled-kernel models. Figure 5.6 tells that we select three kernels for a reference image region. Different scaled models are thus created to compare with candidates for best match. As the variation of the three scales is little, the image of the candidate, i.e. transparent and blue region, still contains nearly all characteristics of the target, which are valuables pixels. Also the outlier pixels are negligible since the spatial kernel limits these pixel contributions approximate to zero. Therefore, the candidates using one specific kernel can feedback their matches to scaled models. The maximum similarity result and corresponding target scale are confirmed to reflect the current scale of that candidate. In order to preserve the robustness, the determination of real scale change is voted by every possible particle. Then similarities of all particles (candidates) are converted into normalized weights. We compute the weighted scale average as the estimate. However, in case of misleading by failure candidates, only those with weights higher than two times of the average weights can be included in the scale computation.

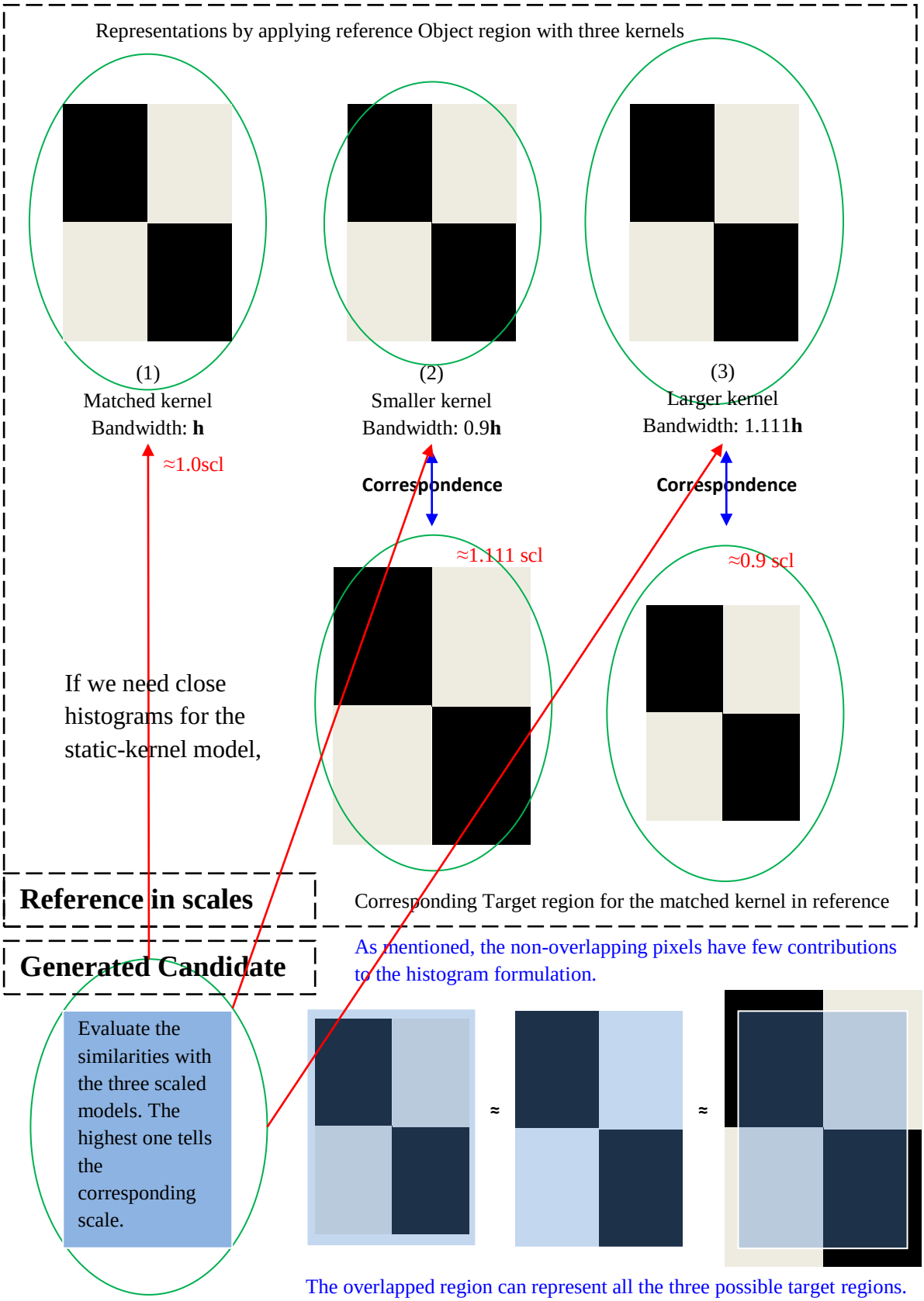
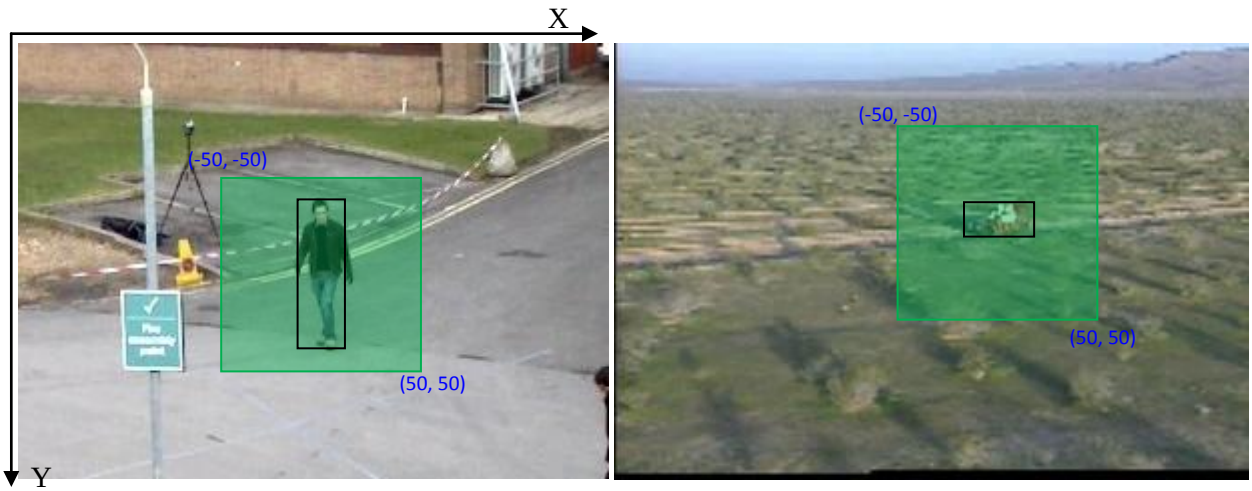


Figure 5.6 Intuitive diagram of scale estimation scheme for one candidate

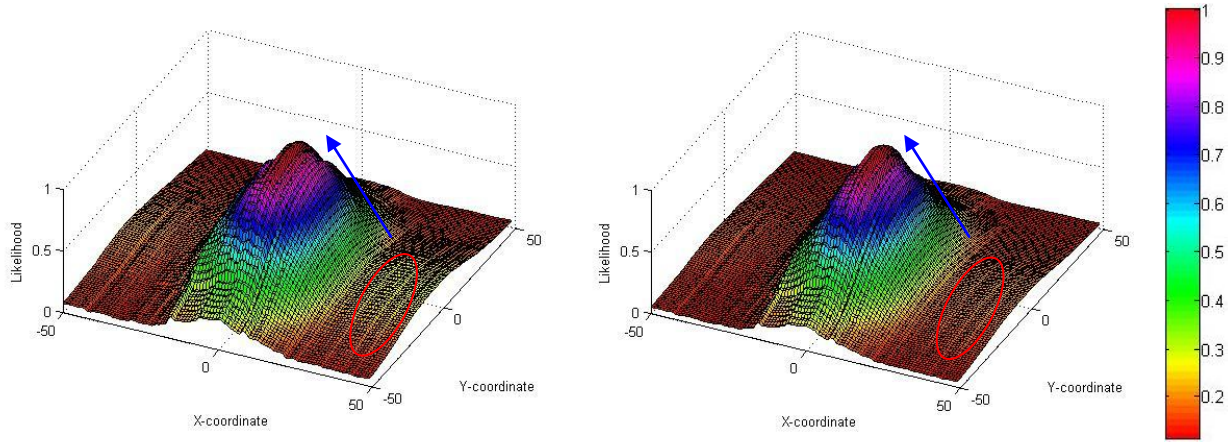
5.6 Experiments

First, we have to prove the effectiveness and robustness of our target representation scheme (i.e. Background-foreground-weighted structural histogram set). We choose two kinds of objects in computing the likelihood map in the square region with a size of 10000 pixels. As mentioned, the likelihood is computed by an exponential function with the power of Bhattacharyya Distance and parameter λ which controls discrimination. λ is multiplied with the Bhattacharyya Distance. In this experiment, we take $\lambda = 5$. It is known that the quality of likelihood map is a significant part in our evaluations. Likelihood map feedbacks coefficients for target candidates, which judges the candidate's contribution. In figure 5.7, a man and a red car are target representation for evaluation. In one particular frame, pixels in the black rectangle window are considered as foreground pixels, while others belong to background. We therefore model the representations in four formats, which include an original color histogram, a background-foreground-weighted (BFW) histogram, structural histogram set and BFW structural histogram set. In order to create a likelihood map, we drew a green box (size: 100×100) around the target center. Pixels in this transparent green box can be considered as particles to compute the likelihood. Each particle owns its pixel region like the target region for feature extraction and representation. Finally, figures in figure 5.8 and figure 5.9 illustrate the likelihood maps in three-dimensional plots. In the X-Y plane, the two coordinates are both ranged in [-50, 49], where the point (-50, -50) is located at the left-upper corner of the box in figure 5.7. They are converted into color maps which introduce colors to discriminate mutual differences in values, where pink is close to the value 1 and orange occupies small value near to 0. Also, the relations between the spatial distance and likelihood are obviously shown in these 3D diagrams.

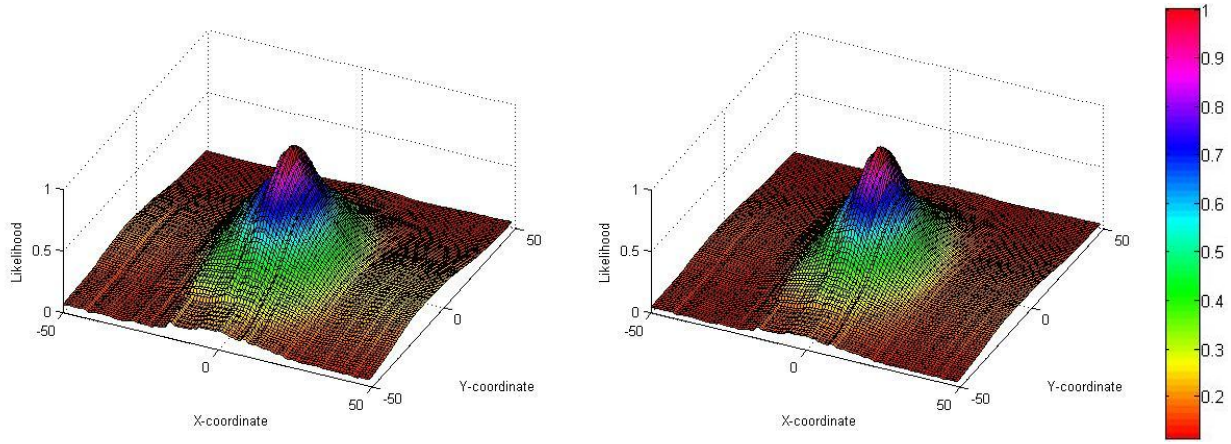


(a) (b)

Figure 5.7 Illustration of the examples for likelihood map extraction within area covered by transparent green color. The image (a) holds a human target in a clear background. And (b) contains the car in the clutter background.



(1) (2)



(3) (4)

Figure 5.8 Likelihood maps for the man in figure 5.7(a) through different Target representations, such as (1) by original color histogram, (2) by Background-foreground-weighted histogram, (3) by multi-partition color histogram set, and (4) by our proposed representation, i.e. Background-foreground-weighted structural histogram set.

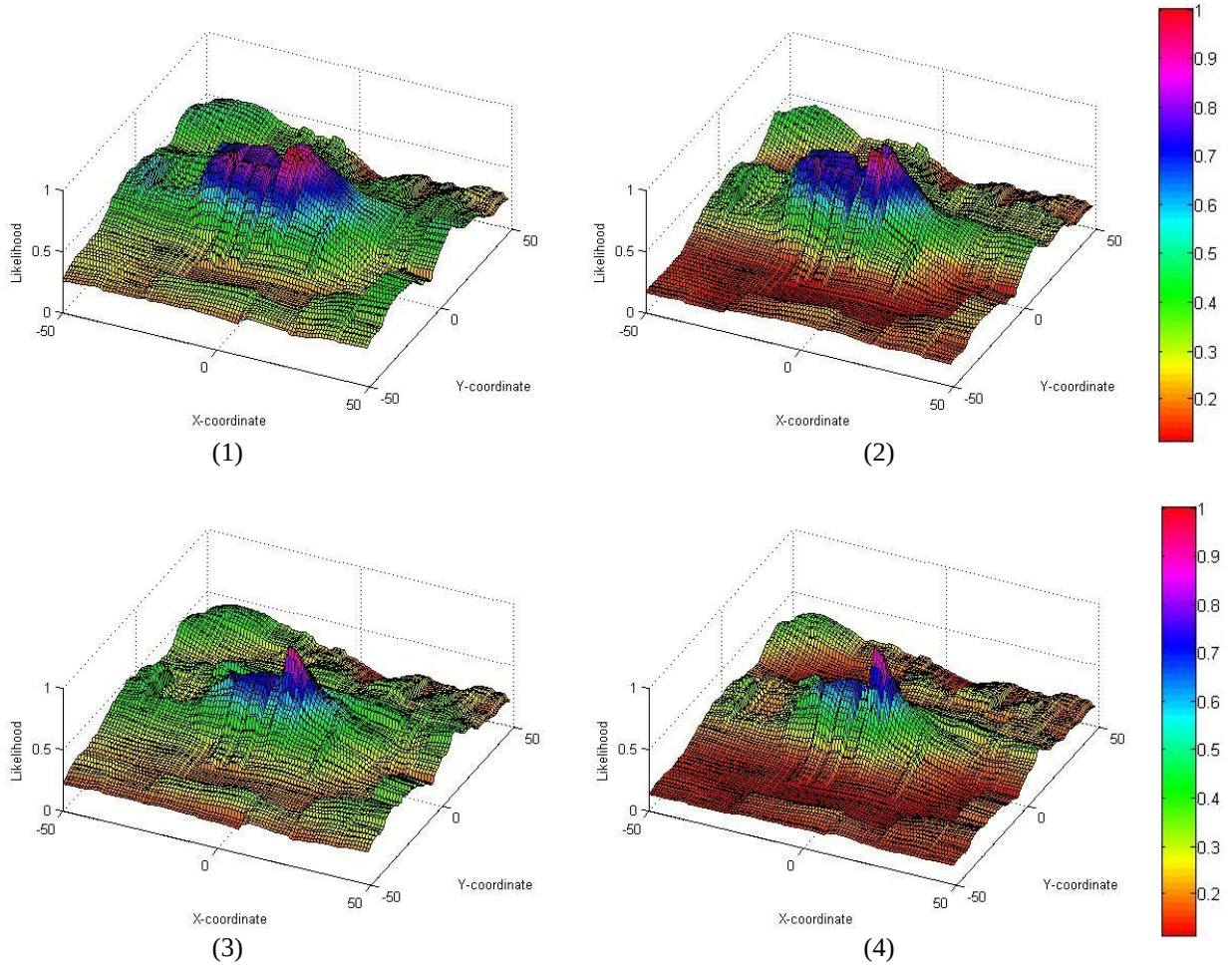


Figure 5.9 Likelihood maps for red car in figure 5.7(b) through different Target representations, such as (1) by original color histogram, (2) by Background-foreground-weighted histogram, (3) by multi-partition color histogram set, and (4) by our proposed representation, i.e. Background-foreground-weighted structural histogram set.

In figure 5.7(a), as a man standing in the road, the background is clear with smooth regions. The existing original histogram can produce an acceptable uni-modal likelihood distribution. However, the climbing slope is not smooth when slightly approaching the center (where the peak stays) in the X-coordinate direction. After processing BFW procedures, the map is better in smoothness of the spatial domain. Also the absolute background position is attenuated in likelihood, namely the value is closer to zero. Therefore its performance is certainly better than the original one. When we adopt the multi-partition structural histograms, the likelihood map shows more discriminative results as pixels spreading. The slopes are much sharper than that of the one histogram representation. The slopes are still not smoothly sliding up or down. After integration of the structural histogram set and the BF weights for bins, the best uni-modal distribution is observed. They are not only the mostly discriminative but can also preserve the slope with smooth sliding surface. So this good likelihood map can improve our sampling efficiency in tracking algorithms, especially the weight feedback from the map gives the best match to the real image. In a different video source, we take the red car in figure 5.7(b) as another target for example. Because of the clutter background in the image, the likelihood map is not uni-modal any more. Figure 5.9 shows the results of these four approaches in comparison and contrast. It is seen that the BFW operations successfully decrease the similarities of candidates in the background area. Thus these candidates will not be coherent with mistaken weights and contribute wrongly to the estimate. Meanwhile, the multi-partition structural histogram set effectively attenuates the neighbouring modes and highlights the global maximum at the peak area. So our model still performs the best among all four approaches.

Second, as partition weight is dynamically updated together with the reference model, we have to evaluate our updating scheme in the temporary dimension. We carried out the visual tracking in

one video sequence, i.e. PETS_view_1. Based on the figures from figure 5.10 to 5.13, we can have comparison among the four methods i.e., our proposed representation with proposed updating scheme, proposed representation with linear updating model, BFW histogram with proposed updating scheme and BFW histogram with linear updating model. It is observed that the linearly update scheme can not react well to the real target pose change and sudden occlusion. So the involved approaches in figure 5.11 and figure 5.13 fail in restoring the object tracking after partial occlusion in frame 22 and frame 162, which are pointed out by red arrows. Even worse, the updated model can lead the tracking to a wrong target in the both figures. For representation of the single BFW histogram, the target information is not enough for tracking in a complex situation. It can be easily confused by other objects since the model is not sufficiently discriminative. We can see in figure 5.12, though a great target updating model is adopted, this approach still loses target for times. Above all, our target representation with selective model and dynamic reference update in figure 5.10 outcomes better performance in a real-time tracking. It conquers the difficulties of partial occlusion and real-time pose change. Still the tracking results are preserved well.

Thirdly, the adaptive scale estimation is integrated into our target model. It has been proved working well in the walking sequence. The results are shown in figure 5.14. Three scales are handled for three models aiming at one target. The candidates are evaluated to see which is the most matched. This is a proper model which can react to a scale change for that candidate. After computing as frames moves on, the averaged scale will be brought in as the current scale after estimation. In figure 5.14, it is shown that the rectangle window turns to smaller size as frames passing by. The recorded references and past few estimates are illustrated in the left-down corner. Also, the black window captures the man perfectly.

Figure 5.10 Tracking results by our selectively updating model using proposed target representation format

Figure 5.11 Tracking results by our linearly-updated target model using proposed target representation format

Figure 5.12 Tracking results by BFW histogram with selectively updating model

Figure 5.13 Tracking results by linearly-updated BFW histogram

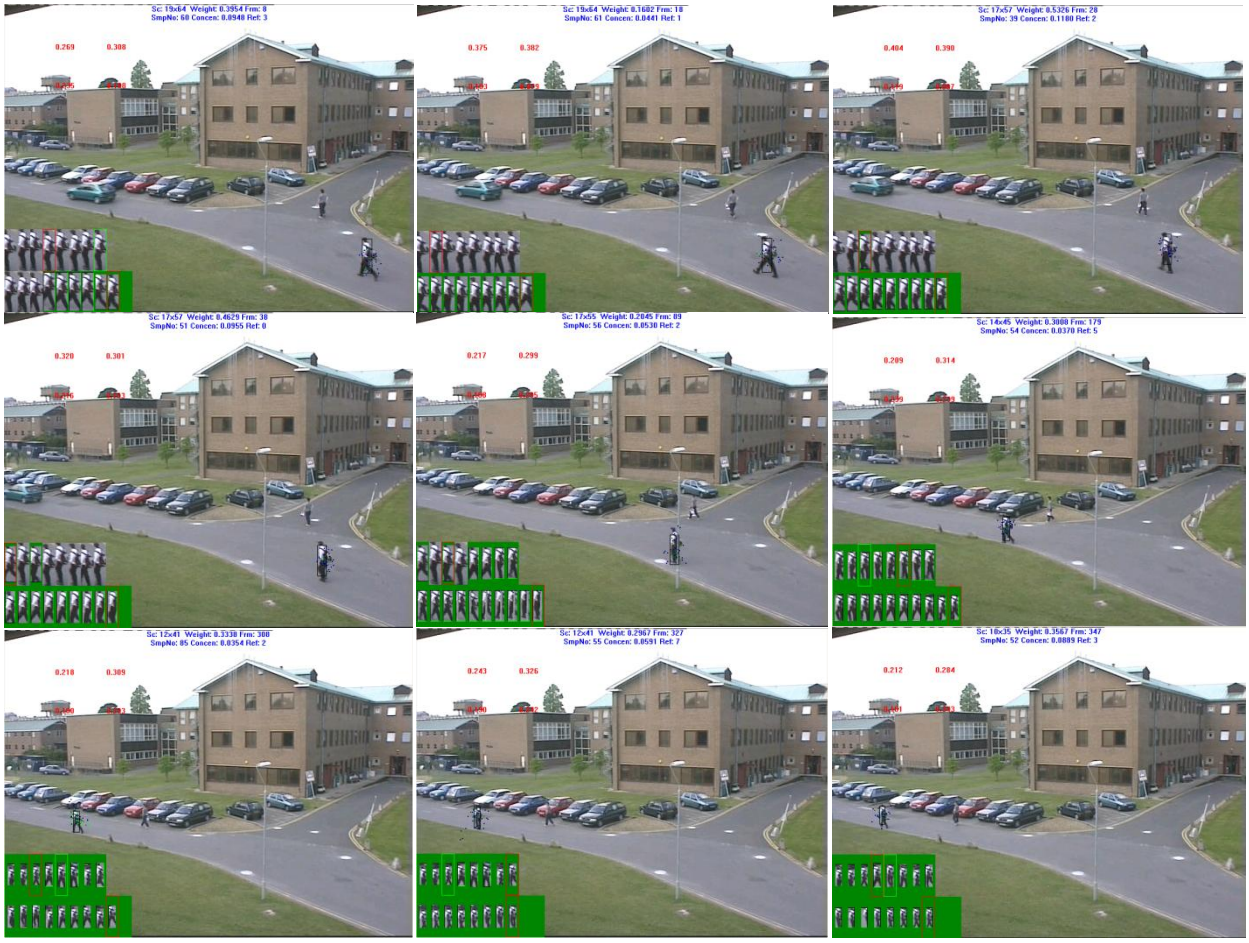


Figure 5.14 Tracking results by involving scale estimation into our Target models

5.7 Conclusion

This chapter gives a new target model based on the structural histogram set. Every partial histogram in the set is assigned a dynamic weight correspondingly. The new representation can effectively produce a better confident similarity map, which is closer to the real map. Meanwhile, the modified similarity measurement can adapt to the newly established model. Since the static model is not suitable for long-time tracking, we have to apply a model updating strategy. The referenced target model will be selected from a candidate set. The set is subsequently updated after every fixed time interval. Results of our extensive experiments show that the new model has better performance when two objects with backgrounds are handled for examining similarities in the specified areas. Indeed, our proposed method succeeded in classifying the clutter background apart from the real possible object area. The new updating scheme can support the target appearance changes, and the tracking process is much more robust and successful. Finally, we have also adopted a simple scale estimation approach. It has been tested to partially useful for solving the scale varying problem. To sum up, our approach relies on the existing estimate to be candidate models. However, in some complex video scenes, the object varies a lot and tracking will be not reliable. Thus not enough good estimates are provided for candidate models in selection. Then the updating scheme cannot follow the change of target. Meanwhile, the tracking results become worse given of those wrong models. So in the future, we will extend our exploration into the detection field, which can monitor the tracking quality while provides better appearance with sufficient confidence.

Chapter 6. Conclusion and Future Work

6.1 Conclusion

In this thesis, we focus our research major on two topics, namely better sampling strategy and better target modeling. We have obtained improvement by proposing our new methods. Details are described in the previous chapters.

Chapter 3 gives a new adaptive sampling approach for particle filtering. Rather than sampling all particles in one step, which refers to as resampling strategy, we propose a multi-step recursive sampling method (MSRS). It is designed to look for high-weight particles as many as possible in steps, while it neglects most of the low-weight ones. Thus the sampled particles, which are denser and closer in the highly probable area, are more useful for estimation. In chapter 4, the predictive models of transition are analyzed for considering a better density of sampling in the proper region. The adaptive averaging model is the simplest and best of all four approaches, especially if we include the AR model. It is then applied into the computation of the shape of a possible sampling region. In other words, the sampling pattern is adjusted corresponding to the average transition. The adaptive patterns can then contribute to handle varying motion models. The sampling region by such patterns corresponds to motion prediction for observing better particles. Also our proposed method (MSRSIR) can adaptively adjust the number of iterations according to the evaluation after each step. Experimental results have verified the effectiveness of the proposed method in reducing computation and improving robustness of our algorithm.

Note that the particle filtering approach has been widely explored in visual tracking and surveillance. The development of the engineering applications is hindered by its computation

complexity. Our approach gives a more efficient structure of the particle filtering algorithm via a better sampling scheme. It can be considered as a sparse sampling for more accurate estimation of posterior density. It will be much more effective and efficient when higher dimensional state space is required, such as those in articulated motion tracking. Interestingly, our approach can be extended to improve the mean shift approach as well with better particle refinement.

In Chapter 5 we have proposed a new target model and updating approach. An object is segmented into parts. All of these partitions contribute to an existing structural histogram set. Partition weight is provided to reduce the giant drop of the candidate similarity. The similarity measurement is accordingly modified to fit the new model. This model has been proven to have better performance with a feedback of better estimate. In the temporal dimension, the model keeps updating after every fixed interval. It captures the pose change of a target or illumination change. The tracking results are significantly improved in terms of accuracy and robustness. Moreover, the new model has better description of target characteristics. Thus it can be used partially to identify different targets by inserting different thresholds. This is also a part of the research in pattern recognition.

6.2 Future work

In the future work, the noisy likelihood map can be integrated by fusing multiple cues in similarity measurements, such as making use of the SIFT features [59, 60] and histogram of gradient (HOG) [61]. For sake of reducing computation, hierarchical likelihood measurement with different discriminate levels can be established as appropriately robust and accurate guidance of sampling. Not all of the particles need to measure the likelihood from all cues. Hierarchical levels may ensure efficient particle sampling or the transition stays on the most

probable area. Furthermore, the multi-modal problem needs a fast classifier to identify the global mode. Accordingly, fewer samples should be attracted by those useless modes. Some deterministic optimization methods, such as mean shift and other Expectation Maximization (EM) ideas, may be helpful to replace some particle transitions by random sampling. If the confident likelihood-map is occupied, their efficiency can be better than Gaussian random walk. It is also a good research direction to investigate an optimal scheme aiming at getting the best choice for particle generation and optimization.

References

[1] Gonzalez, R. C. and Woods, R. E. Digital Image Processing, 3rd ed., Prentice Hall, Upper Saddle River, NJ, 2008.

[2] Jianbo Shi, Tomasi, C., "Good features to track", Computer Vision and Pattern Recognition, 1994. Proceedings, Computer Vision and Pattern Recognition, IEEE Computer Society Conference on , vol., no., pp.593-600, 21-23 Jun 1994.

[3] Von Mark S. Nixon, Alberto S. Aguado, "Feature Extraction and image processing", pp. 164-168, Newnes Publisher, 2002.

[4] Matthews L., Ishikawa T., Baker S., "The template update problem", Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.26, no.6, pp.810-815, June 2004.

[5] Lindeberg T., "Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics", 21(2):224-270, 1994.

[6] Canny John , "A Computational Approach to Edge Detection," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.PAMI-8, no.6, pp.679-698, November 1986.

[7] Comaniciu D., Meer. P., "Mean shift: a robust approach toward feature space analysis", Pattern Analysis and Machine Intelligence IEEE Transactions on, vol. 24, no. 5, pp. 603-619, 2002.

[8] Birchfield S., "Elliptical head tracking using intensity gradients and color histograms", Proceedings, Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, pp.232-237, 23-25 Jun 1998.

[9] Comaniciu D., Ramesh V., Meer P., "Kernel-based object tracking," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.25, no.5, pp. 564- 577, May 2003.

[10] W. T. Freeman and M. Roth. "Orientation histograms for hand gesture recognition", International Workshop on Automatic Face- and Gesture- Recognition, IEEE Computer Society, Zurich, Switzerland, pp. 296-301, June 1995.

[11] Dalal N., Triggs B., "Histograms of oriented gradients for human detection", Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, vol.1, pp.886-893, 25-25 June 2005.

[12] Lowe D.G., "Object recognition from local scale-invariant features", Computer Vision, The Proceedings of the Seventh IEEE International Conference on, vol.2, pp.1150-1157, 1999.

[13] Birchfield S.T. and Sriram Rangarajan, "Spatiograms versus histograms for region-based tracking", Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, vol.2, pp. 1158- 1163, 20-25 June 2005.

[14] Lienhart R., Maydt J., "An extended set of Haar-like features for rapid object detection", Proceedings, Image Processing, International Conference on, vol.1, pp. I-900- I-903, 2002.

[15] Yizong Cheng, "Mean shift, mode seeking, and clustering", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.17, no.8, pp.790-799, Aug 1995.

[16] Comaniciu D., Ramesh V., and Meer P., "Real-time tracking of non-rigid objects using mean shift", Proceedings, Computer Vision and Pattern Recognition, IEEE Conference on, vol. 2, pp. 142–149, June 2000.

[17] Comaniciu. D. and Meer. P., "Mean shift: a robust approach toward feature space analysis", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 5, pp. 603-619, 2002.

[18] Arulampalam M.S., Maskell S., Gordon N. and Clapp T., "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking", Signal Processing, IEEE Transactions on, vol. 50, no.2, pp. 174–188, 2002.

[19] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", Proceedings of the IEEE, Vol. 77, No. 2, February 1989.

[20] Gary Bishop and Greg Welch. "An Introduction to the Kalman Filter". University of Norh Carolina SIGGRAPH 2001 course notes. ACM Inc., North Carolina, 2001.

[21] Isard M. and Blake A., "CONDENSATION—Conditional density propagation for visual tracking", International Journal of Computer Vision, vol. 29, no. 1, pp. 5-28, 1998.

[22] Doucet A., de Freitas J.F.G. and Gordon N.J. "Sequential Monte Carlo methods in practice", New York: Springer-Verlag, 2001.

[23] Nummiaro K., Koller-Meier E. and Gool, L. V., "A Color-Based Particle Filter", Proceedings, 1st International Workshop on Generative-Model-Based Vision (in conjunction with ECCV02), Denmark, pp. 53-60, Jun 2002.

[24] Merwe, R., Doucet, A., Freitas, N. and Wan, E., "The unscented particle filter", Technical Report CUELYF-INFEh'GAR 380, Cambridge University Engineering Department, August 2000.

[25] A. M. Johansen and A. Doucet, "A note on the auxiliary particle filter", Statistics and Probability Letters, vol. 78, no.12, pp. 1498-1504, September 2008.

[26] Kotecha, J.H. and Djuric, P.M., "Gaussian sum particle filtering", Signal Processing, IEEE Transactions on, vol.51, no.10, pp. 2602- 2612, Oct. 2003.

[27] Cheng Chang and Ansari, R., "Kernel particle filter for visual tracking", Signal Processing Letters, IEEE, vol.12, no.3, pp. 242- 245, March 2005.

[28] Caifeng Shan, Tieniu Tan and Yucheng Wei, "Real-time hand tracking using a mean shift embedded particle filter", Pattern Recognition, Vo. 40, Issue 7, pp 1958-1970, July 2007.

[29] Maggio, E. and Cavallaro, A., "Hybrid Particle Filter and Mean Shift tracker with adaptive transition model", Proceedings, Acoustics, Speech, and Signal Processing (ICASSP'05), International Conference on, Philadelphia, PA, USA, vol.2, pp. 221- 224, March 18-23, 2005.

[30] Emilio Maggio and Andrea Cavallaro, "Accurate appearance-based Bayesian tracking for maneuvering targets", Computer Vision and Image Understanding, vol. 113, iss. 4, pp. 544-555, April 2009.

[31] Anbang Yao, Xinggang Lin, Guijin Wang and Shan Yu, "A compact association of particle filtering and kernel based object tracking", Pattern Recognition, vol. 45, iss. 7, pp 2584-2597, July 2012.

[32] Anbang Yao, Guijin Wang, Xinggang Lin and Xiujuan Chai, "An incremental Bhattacharyya dissimilarity measure for particle filtering", Pattern Recognition, vol. 43, iss. 4, pp 1244-1256, April 2010.

[33] Bouaynaya N. and Schonfeld, D., "On the Optimality of Motion-Based Particle Filtering," Circuits and Systems for Video Technology, IEEE Transactions on, vol.19, no.7, pp.1068-1072, July 2009.

[34] Bohyung Han, Ying Zhu, Comaniciu, D. and Davis, L.S., "Visual Tracking by Continuous Density Propagation in Sequential Bayesian Filtering Framework", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.31, no.5, pp.919-930, May 2009.

[35] Khan, Z., Balch, T. and Dellaert, F., "MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 27, no. 11, pp. 1805–1819, November 2005.

[36] J. Deutscher, A. Blake and I. Reid, "Articulated body motion capture by annealed particle filtering", Proceedings, Computer Vision and Pattern Recognition (CVPR'00), International Conference on, Hilton Head, SC, USA, pp. 126–133, 2000.

[37] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm", The American Statistician, vol. 49, no. 4, pp. 327-335, 1995.

[38] Tokdar S. T. and Kass R. E., "Importance Sampling: A Review", Wiley Interdisciplinary Reviews: Computational Statistics. vol. 2, issue. 1, pp. 54-60, Jan 2010.

[39] Andrieu C., Davy M., Doucet A., "Efficient particle filtering for jump Markov systems. Application to time-varying auto-regressions," Signal Processing, IEEE Transactions on , vol.51, no.7, pp. 1762- 1770, July 2003

[40] Priestley M.B., "Non-linear and Non-stationary Time Series Analysis", Academic Press, 1988, ISBN 0-12-564911-8

[41] Monson H. Hayes, "Statistical Digital Signal Processing and Modeling", Wiley, 1996, ISBN 0-471-59431-8

[42] Gustafsson F., "Particle filter theory and practice with positioning applications", IEEE Aerospace and Electronic Systems Magazine, vol. 25, no. 7, pp. 53-82, 2010.

[43] Kotecha J.H. and Djuric P.M., "Gaussian particle filtering", IEEE Transactions on Signal Processing, vol.51, no.10, pp. 2602- 2612, Oct. 2003.

[44] S. K. Zhou, Chellappa R., Moghaddam B., "Visual tracking and recognition using appearance-adaptive models in particle filters", Image Processing, IEEE Transactions on, vol.13, no.11, pp.1491-1506, Nov. 2004.

[45] Pan P. and Schonfeld D., "Dynamic Proposal Variance and Optimal Particle Allocation in Particle Filtering for Video Tracking", Circuits and Systems for Video Technology, IEEE Transactions on, vol.18, no.9, pp.1268-1279, Sept. 2008.

[46] Maggio E. and Cavallaro A., "Multi-part target representation for color tracking", Image Processing, IEEE International Conference on , vol.1, no., pp. 729-32, 11-14 Sept. 2005.

[47] Hanzi Wang, Suter, D., Schindler, K. and Chunhua Shen, "Adaptive Object Tracking Based on an Effective Appearance Filter", Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.29, no.9, pp.1661-1667, Sept. 2007

[48] Doucet, Arnaud, Simon Godsill, and Christophe Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering". Statistics and computing 10.3: 197-208, 2000.

[49] Chen Zhe, "Bayesian filtering: From Kalman filters to particle filters, and beyond", Statistics : 1-69, 2003.

[50] Kin-Yi Yam, Wan-Chi Siu, Ngai-Fong Law, and Chok-Ki Chan, "Effective bi-directional people flow counting for real time surveillance system", Proceedings, Consumer Electronics (ICCE), Las Vegas, 2011 IEEE International Conference on , USA, pp. 863-864, 9-12 Jan. 2011.

[51] Khan, Z.H., Gu, I.Y. and Backhouse, A.G., "Robust Visual Object Tracking Using Multi-Mode Anisotropic Mean Shift and Particle Filters", Circuits and Systems for Video Technology, IEEE Transactions on, vol.21, no.1, pp.74-87, Jan. 2011.

[52] Hol Jeroen D., Schon Thomas B. and Gustafsson Fredrik, "On Resampling Algorithms for Particle Filters", Proceedings, 2006 IEEE Nonlinear Statistical Signal Processing Workshop, Cambridge, UK, pp.79-82, 13-15 Sept. 2006.

[53] Islam, M.Z., Chi-Min Oh, Jun Sung Lee, Chil-Woo Lee, "Multi-part histogram based visual tracking with maximum of posteriori", Computer Engineering and Technology (ICCET), 2010 2nd International Conference on , vol.6, no., pp.V6-435-V6-439, 16-18 April 2010.

[54] Ning J., Zhang L., Zhang D. and Wu C., "Robust mean-shift tracking with corrected background-weighted histogram", Computer Vision, IET, vol.6, no.1, pp.62-69, Jan. 2012.

[55] Jaideep Jeyakar, R. Venkatesh Babu, K.R. Ramakrishnan, "Robust object tracking with background-weighted local kernels", Computer Vision and Image Understanding, Vol. 112, Iss. 3, pp. 296-309, December 2008.

[56] Yuanwei Lao, Junda Zhu, Zheng Y.F., "Sequential Particle Generation for Visual Tracking", Circuits and Systems for Video Technology, IEEE Transactions on , vol.19, no.9, pp.1365-1378, Sept. 2009.

[57] Xue Mei, and Haibin Ling, "Robust Visual Tracking and Vehicle Classification via Sparse Representation", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.33, no.11, pp.2259-2272, Nov. 2011.

[58] Collins R.T., "Mean-shift blob tracking through scale space", Proceedings, Computer Vision and Pattern Recognition, IEEE Computer Society Conference on , vol.2, no. 2, pp. 234-240, 18-20 June 2003.

[59] Lowe D.G., "object recognition from local scale-invariant features", Computer Vision, The Proceedings of the Seventh IEEE International Conference on , vol.2, no., pp.1150-1157 vol.2, 1999.

[60] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features", Computer Vision–ECCV 2006, pp. 404-417, 2006.

[61] Adam A., Rivlin E., Shimshoni I., "Robust Fragments-based Tracking using the Integral Histogram", Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on , vol.1, no., pp. 798- 805, 17-22 June 2006.

[62] Dalal N., Triggs B., "Histograms of oriented gradients for human detection", Computer Vision and Pattern Recognition, IEEE Computer Society Conference on , vol.1, no., pp.886-893, 25-28 June 2005.

[63] Yuan Li, Haizhou Ai, Yamashita T., Shihong Lao, Kawade M., "Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Life Spans", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.30, no.10, pp.1728-1740, Oct. 2008.

[64] Brown Robert Grover and Hwang Patrick Y.C., "Introduction to Random Signals and Applied Kalman Filtering", the third edition, New York: John Wiley & Sons. ISBN 0-471-12839-2, 1996.

[65] Fontmarty, M., Lerasle, F. and Danes, P., "Data fusion within a modified annealed particle filter dedicated to human motion capture", Proceedings, Intelligent Robots and Systems, IEEE/RSJ International Conference on , San Diego, CA, USA, pp.3391-3396, 2007.

[66] A. M. Johansen and A. Doucet, "A note on the auxiliary particle filter", Statistics and Probability Letters, 78(12):1498-1504, September 2008.

[67] N. Whiteley and A. M. Johansen, "Recent developments in auxiliary particle filtering," in Inference and Learning in Dynamic Models, Barber, Cemgil, and Chiappa, Eds. 2010, Cambridge University Press

[68] Musso C., Oudjane N., and LeGland F., "Improving Regularized Particle Filters", in Chap. 12 of "Sequential Monte Carlo Methods in Practice", edited by Doucet, A., de Freitas, N, and Gordon, N., Springer, New York, 2001.

[69] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, "The Unscented Particle Filter", in Advances in Neural Information Processing Systems (NIPS13), (T. G. Dietterich T. K. Leen and V. Tresp, eds.), Dec. 2000.

[70] Xiuzhuang Zhou, Yao Lu, Jiwen Lu, Jie Zhou, "Abrupt Motion Tracking Via Intensively Adaptive Markov-Chain Monte Carlo Sampling", Image Processing, IEEE Transactions on , vol.21, no.2, pp.789-801, Feb. 2012

[71] Min Li, Tieniu Tan, Wei Chen, Kaiqi Huang, "Efficient Object Tracking by Incremental Self-Tuning Particle Filtering on the Affine Group", Image Processing, IEEE Transactions on , vol.21, no.3, pp.1298-1313, March 2012

[72] Erkut Erdem, Sévérine Dubuisson, Isabelle Bloch, "Visual tracking by fusing multiple cues with context-sensitive reliabilities", Pattern Recognition, Vol. 45, Iss. 5, pp. 1948-1959, May 2012.

[73] Anbang Yao, Guijin Wang, Xinggang Lin and Xiujuan Chai, "An incremental Bhattacharyya dissimilarity measure for particle filtering", Pattern Recognition, Vol. 43, Iss. 4, pp. 1244-1256, April 2010.

[74] Leichter I., "Mean Shift Trackers with Cross-Bin Metrics", Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.34, no.4, pp.695-706, April 2012

[75] Junseok Kwon and Kyoung Mu Lee, "Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive Basin Hopping Monte Carlo sampling", Computer Vision and Pattern Recognition, IEEE Conference on , vol., no., pp.1208-1215, 20-25 June 2009.

[76] Zhang Chen and Wan-Chi Siu, "Novel Multi-Step Recursive Sampling Strategy for Particle Filtering in object tracking", proceedings, signal processing, the 11th International Conference on, Beijing, China, pp.1067-1070, 21-25 Oct., 2012.

[77] Zhang Chen and Wan-Chi Siu, "New Multi-Step Sampling with Adaptive Sampling Patterns in Particle Filtering for Tracking in Surveillance Systems", proceedings, pp.288-289, Consumer Electronics (ICCE), 2013 IEEE International Conference on, Las Vegas, USA, 11-14 Jan. 2013.

[78] Zhang Chen and Wan-chi Siu, "Embedded Adaptive Multi-Step Recursive Sampling Strategy in Particle Filtering for Visual Tracking", submitted to Pattern Recognition, UK.