An adaptive particle filter for detection and tracking of color objects

January 5, 2021

Report Author:
Weiqi Xu, weiqi@kth.se
Group Partner:
Oscar Palfelt, palfelt@kth.se

Abstract

This project is to implement particle filter for detection and tracking of color objects in the environment of Matlab. The large number of particles used in detection step will result in a longer computational time in tracking step. Thus, KLD-sampling method is integrated into the standard color particle filter to adjust the particle number automatically. Model update method is applied to deal with object's appearance changes in videos. The implementation is evaluated with videos from MOTChallenge.

Keywords: object tracking, detection, particle filter, KLD-sampling, color histogram

1 Introduction

Object tracking in videos is one of the main concern in the computer vision field. The tasks can be applied in automatic driving, surveillance, medical imaging and etc. However, real-world object tracking is challenging due to the occlusion, illumination and scale variation, shape transformation and other dynamic factors.

Recursive State Estimation, classified as stochastic methods use the state space to model the underlying dynamics of the tracking system [6]. Kalman filter is widely used in linear-Gaussian model with linear measurement. Weng et.al [5] use differential frame to extract moving objects and parameter adaptive Kalman filter for tracking. Particle filter is popular because of its robustness, accuracy and reliability with non-linear model and non-Gaussian noise[2]. Edge orientation histogram and color rectangle features are combined with particle filter to conduct single and multiple objects tracking in [6].

One main disadvantage of particle filter is its computational cost[2]. To improve the efficiency, Dieter Fox et.al[1] proposed a method to adapt the number of particles over

time, the KLD-sampling. This method determines the number of samples by bounding the distance between discrete sample-based approximation and true posterior.

Color is a robust property of non-rigid objects[3], and color histograms describe of the RGB color information in images. In this implementation color histogram is used as representation of target and measurements. In the real-world video, the illumination, object scale and appearance may change because of object's or camera's moving. Thus, adaptive target model is proposed to handle the target variation[4].

In this implementation we combine the standard particle filter with model updating and KLD-sampling. The adaptive color based particle filter is then integrated to a MATLAB project which can import videos, let users segment target object manually and conduct our proposed tracker. The experiment using video taken in sunny street shows that our tracker performs robustly under the circumstance of changing lights, scale transform and different angles. The remainder of this report is organized as follows: In the next section we introduce the outline of adaptive color particle filter. Then we state the detail to implement it in Section 3. Experiment results and analysis are given in Section 4. Conclusion is conducted in the final section.

2 Adaptive color particle filter

2.1 Color Distributions

Color histogram is a simple and robust property of non-rigid objects and is used in our tracking as measurement. Color histogram used in this implementation is a 3Xn matrix, which represents the color distributions respectively in RGB channels. We choose n as the number of bins. Bhattacharyya coefficient is used to evaluate the similarity between measurement and target distribution[4]. For two color histogram $p = \left\{p^{(u)}\right\}_{u=1\ldots m}$ and $q = \left\{q^{(u)}\right\}_{u=1\ldots m}$, the Bhattacharyya coefficient ρ can be computed as

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

Then the Bhattacharyya distance is defined to measure the distance of two histograms:

$$d = \sqrt{1 - \rho[p, q]} \tag{2}$$

2.2 Kullback-Lieber Distance

Particle numbers and estimation effect is a trade-off problem. A larger number of particles can prevent divergence, but increases computational cost. In order to optimize it, we use the Kullback-Lieber Divergence as a description of difference between target distribution and proposal distribution, and then adapt the number of particles until the K-L distance is less than a threshold. To approximate the discrete posterior, we divide the frame into bins.

We define bins occupied by samples as bins with support. From [1] we know that if particle number n satisfy,

$$n = \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \tag{3}$$

the K-L distance between MLE and target distribution will be less than ϵ with probability $1-\sigma$. By approximation, equation (3) can be written as:

$$n = \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \doteq \frac{k-1}{2\epsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3 \tag{4}$$

where $z_{1-\sigma}$ is the upper $1-\sigma$ quantile of the standard normal $N(0,1)$ distribution and k is the number of bins with support.

2.3 Adaptive Color Based Particle Filtering

The particle filter in this implementation is basically modified from what we use in LAB 2. Detailed description can be found LAB instruction. We rewrote the codes with the similar skeleton, added model update and KLD-sampling steps into the algorithms, and integrated the filter into video processing and visualization modules.

2.3.1 Motion Model and Prediction

The prediction step is consist of applying motion model and diffusion,

$$\bar{x}_t^m \equiv x_{t-1}^m + \bar{u}_t + \mathcal{N}(0, \Sigma_R) \tag{5}$$

Where Σ_R is the modeled process noise covariance matrix. We assume it's Gaussian noise. x_t^m is the state vector of sample m at time step t.

$$x_t^m \equiv [x_{t,x}; x_{t,y}; W; H]^T \tag{6}$$

Where $(x_{t,x}; x_{t,y})$ is the position of the sample; W and H are the window's width and height of each sample.

In the test video, the camera usually follows the target and fix it in the image center. Thus, objects in our case have no regular motion respectively to the frame and control signal $\bar{u}_t$ is defined as 0. Then the prediction step can be written as,

$$\bar{x}_t^m \equiv x_{t-1}^m + \mathcal{N}(0, \Sigma_R) \tag{7}$$

A larger process noise will be modelled since we use a fixed target model.

2.3.2 Measurement Model and Weighting

As stated above, we use color histogram calculated by each particle's window as measurements z_t^m, and the Bhattacharyya distance d as innovation, the presentation of the difference between observation and reference histogram. The likelihood function of an observation is given by

$$p\left(z_t^m \mid x_t^m\right) = \frac{1}{\sqrt{2\pi}Q}\exp(-\frac{1}{2}d^2Q^{-1}) \tag{8}$$

Since we have only one observation in each time step, the weighting process is

$$w_t^m = p\left(z_t^m \mid x_t^m\right) \tag{9}$$

Where w_t^m is the weight of sample m in time step t.

2.3.3 Model Update

The appearances of objects in videos may change due to illumination variance and different poses. Thus, the target distribution will be changed accordingly. We use the following law to update object models. In each time step we use the weighted mean state of proposal distribution as presentation of estimated objects:

$$E[S] = \sum_{n=1}^{N} w_t^n x_t^n \tag{10}$$

Objects which are occluded or mismatched will be discarded as outliers, i.e. mean state with probability lower than threshold will not be counted towards model update. The model will be updated when:

$$w_{E[S]} > w_T \tag{11}$$

Target distribution at time step t are association of the mean state histogram and target distribution at last time step $t-1$.

$$q_t^{(u)} = (1-\alpha)q_{t-1}^{(u)} + \alpha p_{E[S_t]}^{(u)} \tag{12}$$

Equation 12[4] shows that reference in the past has a decreasing contribution to the future frames and current weighted observation are updated into the target distribution.

2.3.4 KLD-sampling

In the object tracking problem, we need a large amount of particles scattered throughout the frame in the beginning to detect the target. Then with higher certainty of estimation, respectively less samples are enough to track the object and reduce the computational complexity. We use KLD-sampling method to generate an adaptive number of samples

in each time step. The key idea of KLD-sampling is limiting the distribution error under threshold. From [1] we know that we don't need to get a complete discrete posterior distribution, and with eq. (4) the number of bins with support k can determine the particle number.

At each iteration, we check the generated sample's distribution. If a sample falls into an empty bin, $k = k + 1$ and the corresponding sample number n will be updated. As soon as the number of current samples exceed the sample number n estimated by k, KLD-sampling stops and no new sample will be generated.

2.3.5 Systematic Sampling

As the KLD-sampling is only based on the number of bins with support but have no direct selection about sample weights, the systematic resampling is still applied to this filter to ensure estimation accuracy. The resampling algorithm used here is the same as LAB 2.

3 Implementation

3.1 Vedio Processing and Color Histogram

The implementation is done in Matlab environment with some real world videos. The vedios can be found at https://motchallenge.net/vis/MOT16-13, and ground truth of the target can be extracted from this website database. We use the VideoReader class to deal with the video frames.

Using function $drawrectangle()$, users can manually choose a object as target from the first frame. Then we use $histcount()$ function to calculate color histogram instead of using $imhist()$ which is encapsulated for image color histogram generation. $histcount()$ saves the running time since we can define the bins' edges. However, the space between bins, bins' width and bins' locations will be different for each particle if using $imhist()$, which increases computational amount and does not correspond to eq.(1). Since the test video has a large variation in colors, the number of bins for color histogram calculation, nc should be larger. When colors in videos have less variation, computational complexity can be reduced by using a smaller nc.

3.2 Adaptive Color Particle Filter

Algorithm 1: adaptive color particle

Inputs: S_{t-1}={<x_{t-1}, w_{t-1}>|i=1,...,n}, q_{t-1}
$S_t = \emptyset, n = 0, k = 0, M = 0$;
for *all b in H do*
  $b = empty$
endfor
do
  draw i with probability $\propto w^i_{t-1}$;
  $M = M + 1$;
  x^M_t=motion_model(x^i_{t-1});
  w^M_t=measurement_model(z_t, x^M_t);
  $S_t = S_t + < x^M_{t-1}, w^M_{t-1} >$;
  $q_t = (1-\alpha)q_{t-1} + \alpha p_{E[S_t]}$;
  if x^M_{t-1} *falls into empty bin b* **then**
    $k = k + 1$;
    $b = non - empty$;
    if $k > 1$ **then**
      $M_\chi = \frac{k-1}{2\epsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3$
    end
  end
  $Systematic - resampling$;
while $M < M_\chi or M < M_{\chi min}$;
Return: S_t, q_t

Particle positions are generated randomly in frame region and and window sizes are initialized the same as users segmentation for targets. All this parameter should be rounded to integer since frames are in grid form, and be bounded not to exceed the edge of frame. Number of particles M is initialized as 10000, which is big enough to scatter across the picture and detect the object. We define M_{min} as the minimum number of particles, in case particles shrink to a too small region to deal with kidnapped problem or sudden acceleration.

R is the model noise covariance matrix and assigned big values when the fixed motion model is applied. Q is the measurement noise and relatively low, since the error of operation with RGB image is much smaller than odometry.

According the eq.(4), the corresponding number of samples n is proportional to number of bins with support k. To get a preciser estimation, the distribution is divided into higher resolution grids with higher number of KLD bins, ns. The confidence level $1 - \sigma$ is set 0.99. We choose 0.3 for the threshold of distribution error ϵ.

u_{thresh} is the threshold for model updating step. Derived from eq.(11), we use Bhattacharyya distance d to represent w_t. The target model is updated when $d < U_{thresh}$, i.e. the object is not occluded or mismatched. The mean state contribution weight, α is set as 0.14.

Here is all the parameters' values we use:

TABLE 1: Parameter values used in this implementation

nc	16	M	10000
ns	100	M_{min}	600
R	$\begin{pmatrix} 15 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 \\ 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 13 \end{pmatrix}$	Q	0.1
u_thresh	0.12	α	0.14
δ	0.99	ϵ	0.3

3.3 Visualization and Presentation

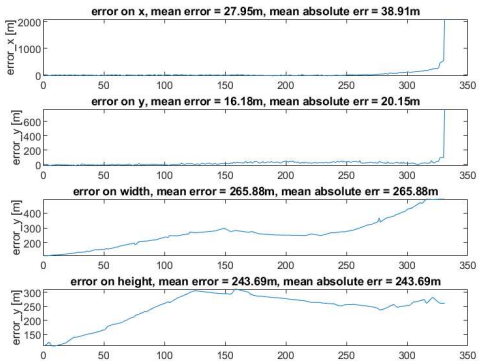
Mean states of particles are used to represent the estimation. For visualization, red rectangular is drawn with best match particle upon frames as estimation. Ground truth is drawn as blue rectangular.

4 Experiments

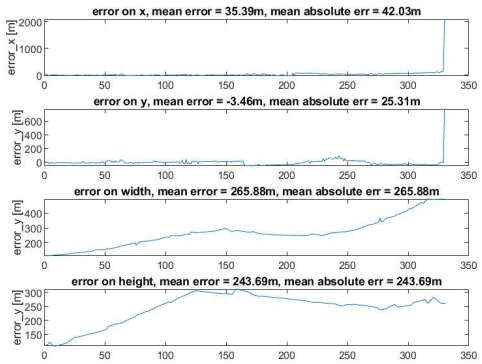
To illustrate the adaptive color based particle filter, we applied the method to track a blue car in a video and show the tracking results. To show how model uodating and KLD-sampling method affect the estimation results, we test the normal particle filter, particle filter with KLD-sampling, particle filter with model updating, and particle filter with KLD-sampling as well as model updating separately.

Figure 1 shows the evolution error of four algorithms. Particle filter with KLD-sampling and model update has the minimum position estimation error, while normal particle filter has the maximum. Compare the left column of Figure 1 and the right column, we can see that model update can effectively reduce the estimation error.

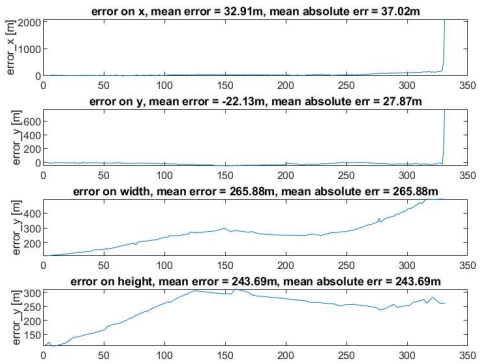
Figure 2 shows how reference histogram is updated over time.Figure 3 compares the evolution of mean state probability between implementation with model updating and



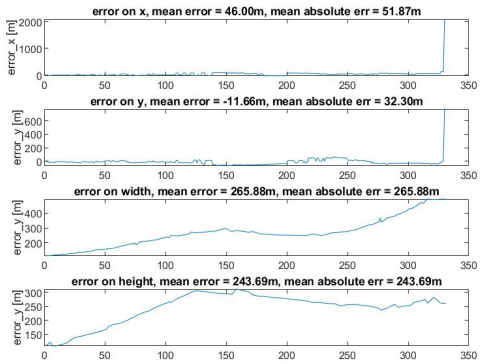
(A) KLD-smpling and model update



(B) KLD-smpling



(C) Model update



(D) Particle filter

FIGURE 1: The evolution error of different algorithms

without model updating. The likelihood of estimation drops rapidly as the car/camera starts moving and changing direction if model update is not applied. The two figures together show the importance of model updating. As illumination and pose of the target change over time, reference histogram, the target distribution will be updated, and using statics reference results in a decreasing estimation accuracy.

Figure 4 illustrates the tracking result straightforwardly. As the blue car moves, the scale of the target is increasing and the proportion of light-reflecting area is in variation. The first five frames show that our proposed method can track it pretty well with dynamic window sizes. Sub figure E proves that the tracker is still effective even when the car turns right and only a small part of it remains in the frame.

However, the model updating method is a bit of trade-off. When updating the target distribution, noise may be introduced by the mean state. Thus, the parameter of model updating threshold should be carefully selected. If the threshold is too strict, variation in target model would exceed it and update would be rejected. A large threshold cannot identify the occlusion or mismatching, leading to introducing noise to the target model. The

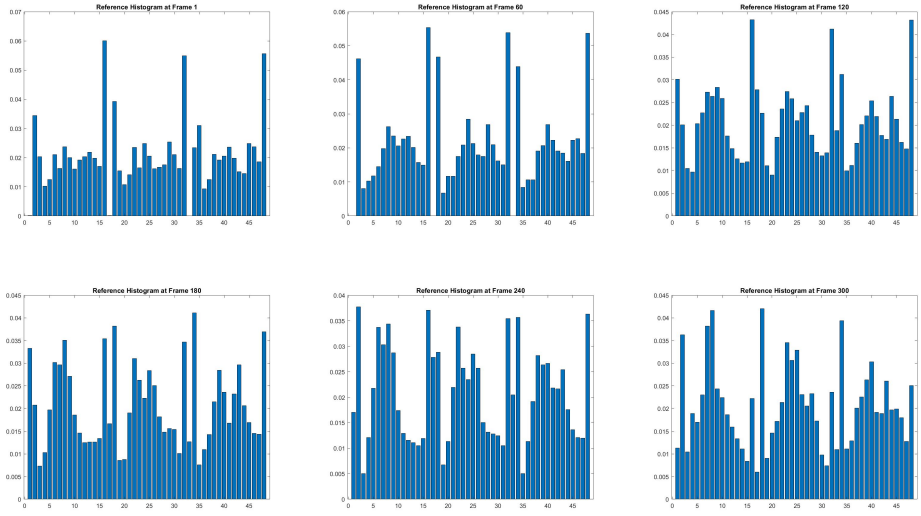
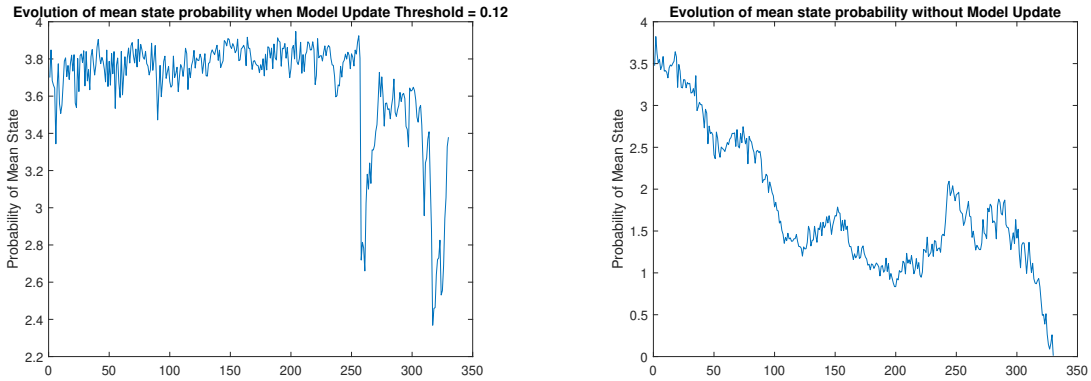


FIGURE 2: The evolution of reference histogram using KLD-sampling and model update



(A) KLD-smpling and model update (B) KLD-smpling

FIGURE 3: The evolution probabilities of mean states

sub figure F show the detriment using twice the defined threshold. Histogram of the road is updated into the reference by mean state and eventually resulting in mismatching on the road.

Figure 5 shows the effectiveness of KLD-sampling. The number of particles is reduced to the minimum number 600 after detection is completed in the first frames, which effectively decreases computational complexity.

(A) (B)

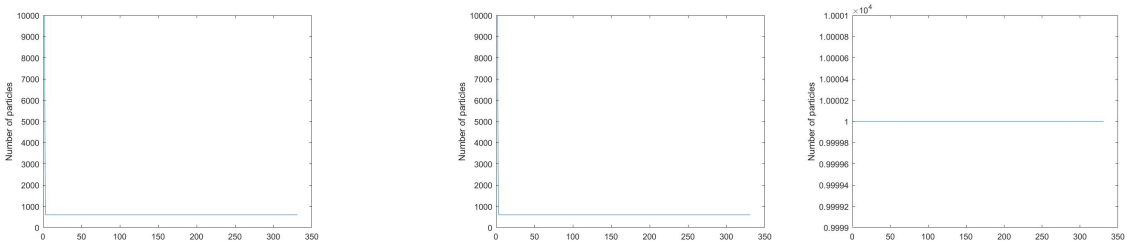
(C) (D)

(E) (F)

FIGURE 4: The tracking result in different frames. Update threshold in the first five frames is 0.14, while in the last one is 0.28

5 Conclusion

The proposed method adds model updating and adapting number of samples to standard particle filter. The tracker can successfully track moving, light reflecting objects with changes of camera angles and distance. Experiments show that model updating can effectively decrease the estimation error, i.e. increase probability of estimation. However, the process to update models can be further improved. For instance, when the car turns right, the right door area is not included in our reference histogram. Model update threshold should be carefully adjusted to optimize the tracking reliability while updating the reference, which may introduce additional noise to the model. Another remaining problem is



(A) KLD and model update (B) KLD (C) Particle filter

FIGURE 5: The evolution of number of particles using different algorithms. The left one is from particle filter with KLD-sampling and model update. The middle one is from particle filter with KLD-sampling. Both decrease to 600 at frame 3. The right one is from normal particle filter and keep in 10000

that the prediction and updating model for window size may be promoted as the error in window size stays high during evolution. The computational efficiency is greatly improved with KLD-sampling. The algorithms reduce the number of samples to 6% of the initial value (the minimum number of samples) when particles converge into a small area in the 3rd frame.

References

[1] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in neural information processing systems*, pages 713–720, 2002.

[2] Neil Gordon, B Ristic, and S Arulampalam. Beyond the kalman filter: Particle filters for tracking applications. *Artech House, London*, 830(5):1–4, 2004.

[3] M. Isard and J. MacCormick. Bramble: a bayesian multiple-blob tracker. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 34–41 vol.2, 2001.

[4] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. Object tracking with an adaptive color-based particle filter. In *Joint Pattern Recognition Symposium*, pages 353–360. Springer, 2002.

[5] Shiuh-Ku Weng, Chung-Ming Kuo, and Shu-Kang Tu. Video object tracking using adaptive kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190–1208, 2006.

[6] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 212–219. IEEE, 2005.