Non-Rigid Object Tracking Using Particle Filters

Hui Pang
TSCRM

Abstract

Based on methods proposed by several other papers, we have built an easy-to-implement probabilistic object tracking algorithm of our own. Our algorithm can track single non-rigid object, even under changing environment. We build our algorithm under particle filter framework, and measure weight by Bhattacharyya distance between color histograms. We choose particle filter because it is not limited to linear Gaussian models, and choose color histogram because it is a rather robust representation for objects. We also update the initial target model judging by the weight corresponding to the sequential estimation because even color histogram of the object may change over time.

Indexed Terms:

Non-rigid object tracking, particle filter, color histogram, Bhattacharyya distance

I INTRODUCTION

Object tracking is useful in computer vision, for it is the start of many other applications. The definition of object tracking problem is given by paper [4], that is, with a representation of the interesting object as the initial target model, the object tracking algorithm needs to be able to estimate the following state in next frames.

The challenge of object tracking lies in the real-life demands. In a real-life tracking scenario, we want our algorithm to be **computational efficient, robust, able to handle nonlinearity** and **to compensate for drifts**. We want our algorithm to be computationally efficient and resource-adaptive, to save the valuable computing sources to do more primary tasks. We also want it to be robust to common environmental changes, such as motion, scale, occlusion, illumination etc [3] [6]. Besides, its nonlinearity with non-Gaussian noises can make elsewhere useful applications like Extended Kalman Filter fail [8]. Also, the appearance of the object may also be different from what we first saw due to environment changes, so we need to gradually integrate the following estimations into the initial one.

For the computational efficiency, we will be implementing a method with particle filtering, known as Condensation algorithm [9] in computer vision community. As the classification of tracking algorithms in paper [6] goes, it is a "top-down" tracking method, with filtering and data association. It only needs to compute certain areas so it saves the computation of the whole frame. Particle filtering is also known as a resource adaptive method [1], so it is possible to adjust the number of particles according to allocated computation resource in future optimization.

For the nonlinearity, we still have particle filtering as the weapon. It is not affected much by the hidden states, and can handle multiple hypotheses at the same time, therefore can keep track of object even with dramatic changes [2].

For the robustness, color histogram is appealing, for it is simple to implement and invariant to scale, partial occlusion and rotation [2]. It remains even when spatial structure changes [8].

For the drifting from initial state, we need to update the target model as the tracking goes by. For discriminative models like ours, we must integrate our trusted samples into the initial state to build more robust algorithms [4]. In our algorithm, we simply use static threshold to select estimation with large-enough-to-be-trusted weight for target updating [2].

As for contributions, there have been enough visual tracking algorithms to form a proper benchmark [4]. Our only contribution is a rather simple implementation of color-based tracking algorithm based on several widely accepted papers introduced below.

As for related work, paper [4] provides a review on online tracking algorithms, and its website of benchmark is priceless to provide not only the testing datasets but also the ground truth to evaluate our experiment results. Paper [6] introduces the definition of object tracking and its classification, as well as the color histogram we are using in this report. Paper [8] convinces us of the advantage of color histogram and particle filter. Our main particle filtering algorithm is from the baseline of this paper, and we borrow some of important parameter settings. However, we use a lower order of state space models, and we put a target update algorithm to compensate for drifting as an improvement. Paper [2] provides an easily implemented target update algorithm.

The outline is as follows. Section I is a short introduction of the object tracking problem, the reason why color histogram and particle filter are good choice, as well as the necessity of target update and related work. Section II explains the main theory and mathematical representation for the algorithm. Section III explains the initialization method

and results. Section IV concludes the report with a discussion, and possible ways to improve our algorithm in the future.

II METHOD

We have decided to use a particle filtering framework and a weight based on Bhattacharyya distance between color histograms. To obtain this, we need to know the following theory and mathematical representations for: **A.** the approximation of estimation for each step; **B**. the state space transition model [8]; **C**. definitions and number of bins for color histogram; **D**. weight based on color histogram distances; **E**. update of the target model to compensate for drifts from the initial state.

The different steps will be introduced in sequent order in this part.

A. PARTICLE FILTER

Particle filter, aka Monte Carlo filter, aka Condensation algorithm, can track multiple hypotheses at the same time. Experiment results from paper [8] show that it can reliably avoid background clusters, and even recover after partial or complete occlusions.

To get the sequential estimation for our algorithm, we will write the observation step on Bayes theorem. Use the denotation from the textbook [1], x_t represents the current state at t, while z_t the measurement. We did not write it like the simple first-order Markov model in the textbook but explicitly write all the previous states to indicate the "hidden states" , and the Bayes recursion is as follows:

$$p(x_{t+1} \mid z_{0:t+1}) \propto p(y_{t+1} \mid x_{t+1}) \int_{x_t} p(x_{t+1} \mid x_t) p(x_t \mid y_{0:t}) \, dx_t$$

If we can make it a first-order Markov process with Gaussian noise we can write it in the good old Kalman filter way. However, it is stated in [8] that in tracking problem, the noise may not be Gaussian, and the belief is nonlinear. However, we can approximate the belief in the nonparametric way by a sample set of particles. We can decide the number of M particles according to our computational cost and make it resource-adaptive. To approximate the sequential expectation, we calculate a weighted sum of particles. We use

$$E[S_t] = \frac{1}{M} \sum_{n=1}^{M} \pi_t S_t^{(n)} \approx E(x_t \mid z_{0:t})$$

as the estimation at time t. The state, weight and dynamic model are explained in later sections.

B. STATE SPACE AND DYNAMICS

In our model, we estimate not only the position and velocity, but also the scale, as a known "hidden" state. It is proposed in paper [8] that color feature is insensitive to rotation when the scale nears 1.

We choose the state representation as a rectangular patch in an image at time t. We consider the center of the patch as the position component in state, and difference between neighbor states as velocity component. The initial scale is set to the default 1.

The state at time t is

$$S_t = [x, y, v_x, v_y, s] \ ,$$

where x, y for the location of the rectangular patch, v_x, v_y the velocity, s the corresponding scale change.

For the propagation process, a first-order auto-regressive dynamics is applied.

$$S_t = A \bullet S_{t-1} + R_{t-1}$$

, where A is the state-space transition matrix, and R_{t-1} is Gaussian.

C. COLOR HISTOGRAM

Color distributions are suitable representations for their robustness. It remains more or less the same with deformation, rotation and partial occlusion [2]. One good way to represent it is using histograms in color space, known as color histogram.

The number of bins of the color histogram should neither be too small nor too large. Too few bins will cause confusion with not-alike colors, while too many bins is not good for computation. We use the default RGB color space in MATLAB and using $8 \times 8 \times 8$ bins proposed in [2] and works rather well. With strong illumination changes, we can put a HSV model.

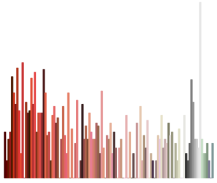


Fig 1. Color histogram of region of interests

By drawing the color histogram with corresponding color for each bar, we can see the red components peaks in this color histogram. It is because red is the dominant color in the small green rectangle, which we can obviously see. This particular useful illustrating code is provided by Chaitanya Sai Gaddam

D. WEIGHT BASED ON BHATTACHARYYA DISTANCE

We will also need a measurement for distance between two distributions, the similar way we use Mahalanobis distance to measure distance between a point and a distribution in textbook. We also would like to get large weights for small distances between distributions.

In mathematics, there are two commonly used measurements to measure distances between two subsets: the Bhattacharyya distance and Hausdorff distance. The latter, while does not need to compute an explicit correspondence, is sensitive to outliers [5]. It is therefore not ideal for our purposes. Bhattacharyya distance however, is popular among computer vision research field, for it has a clear geometric interpretation, scale-invariant, and can avoid the singularity problem of the chi-square test when comparing empty histogram bins [6][7].

The Bhattacharyya coefficient ρ for normalized discrete distributions is defined as [7]

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

where m is number of bins.

Bhattacharyya distance is defined as

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

They are both bounded within [0,1] when p and q are normalized.

We compare our target model and sampling areas by calculating the Bhattacharyya distance of their normalized color histograms. This is actually where the computation bottleneck lies.

We also want to set a measurement to get large weights for small distances. It can be achieved by the exponential weighing model proposed in [8] using square of distance:

$$\pi \propto \exp(-\lambda \bullet d^2), \lambda = 20$$

where the parameter λ = 20 is a rule-of-thumb value.

E. MODEL UPDATE

There are already many MATLAB codes to tracking single fixed color, which are a bit akin to localization in static environment. However, the real-life cause is more like localization in dynamic environment with moving landmarks. Therefore, we need to update the reference color histogram with trusted observations.

We first set a rule-of-thumb threshold to decide if the new observation is trustful enough. That is, too small a weight of a new observation can be outlier resulted from occlusion or noise [2], therefore should be discarded to avoid polluting the reference. We then set a static weight to decide the contribution of the trusted new observation. This way we introduce a forgetting process where the contribution of any frame decreases exponentially as the observation goes [2]. The weight in our implementation is static. However, we can improve the update algorithm later by introducing a dynamic weight with the foreground-background segmentation using mixture of Gaussians.

Algorithm 1 The reference update algorithm

for observation $E[S_t]$ at time t with corresponding confidence $\pi_{E[S_t]}$

if $\pi_{E[S_t]} > \pi_T$

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

else

$$q_t^{(u)} = (1-\alpha) \bullet q_{t-1}^{(u)}$$

end if

In our implementation, we only set one single reference for all particles. However, we can try to improve our results by set several references, and use the data association technique we learnt in location problem with several landmarks [1]. Consider a scenario where we want to track a human face. We can select several angles of the face as references, and decide a better fit by calculating the maximum likelihood.

III EXPERIMENTS

A. INITIALIZATION

To implement this algorithm, we first need to find a proper representation of the target object as the initial state. As is summarized in [2], there are three common ways to form a given prior: manual initialization, automatic initialization via given histogram or object detection algorithm. In our algorithm, we take manual initialization, the easy way to form prior knowledge. We use a GUI for user to specify the initial state by drawing a rectangular area over the region of interest in a user-specified frame. The object must be fully visible, and the selected area needs to cover distinct color features to serve as a decent color histogram reference.

To evaluate our results quantitatively, and knowing that tracking problems are sensitive to initialization [4], we use the datasets downloaded from the website of published benchmarks and test our algorithm by initializing the tracking with ground truth and compare our estimations with ground truth.

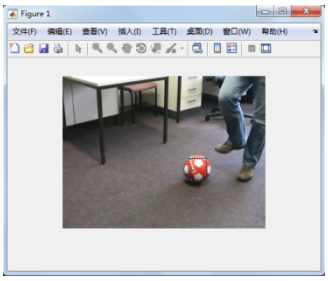


Fig 2. GUI window for user to specify the initial state

IV RESULT

Based on the parameters proposed above, we select some short colored picture sequences on the published benchmark website. The selected region of interests will stay in upper left corner for every frame. The blue rectangle clusters are particles that represent multiple hypotheses. The single trafficlight-colored rectangle represents the sequential expectation, with green indicating large likelihood, yellow medium, red small and likely to lose track of the object.

To illustrate the robustness of our implementation, we use MountainBike, Human8, Bird2 and Dog sequences. The picture sequences, labels and ground truths are from the website of [4].

For Dog, it has been manually labeled as SV(Scale Variation), DEF(Deformation), OPR(Out-of-Plane Rotation). We can see that the rectangle bounds tight despite the puppy change its shape by hopping and running. It is mainly because the color distribution of the back fur is very distinctive and its color distribution is rather invariant to shape and scale changes.

For MountainBike, it has be manually labeled as IPR(In-Plane Rotation), OPR(Out-of-Plane Rotation), and BC(Background Clutters). To evaluate the robustness to IPR and OPR, we can compare the first 30 frames, where the rider drives on a slope and rotates. While the color distribution does not change much, the pose of the rider experiences a 45 degree change. We can see however, the expectation of the state does not change much in size and tracks the rider rather closely. As for BC, we can see that while the rider and the desert are of similar colors to our naked eyes, the algorithm still can tell the object from background.

For Bird2, it has been manually labeled as OCC(Occlusion), DEF(Deformation), FM(Fast Motion), IPR(In-Plane Rotation), OPR(Out-of-Plane Rotation). We can see that partial occlusion and fast motion tends to cause the particle to spread wide in order to capture and bound the fast moving object again.

For Human8, it has been manually labeled as IV(Illumination Variation),SV(Scale Variation) and DEF(Deformation). We can see due to illumination changes, the particles almost lose the man to the shadow and very dark windows. However, our algorithm can update the target model to compensate for changes over time. Our algorithm recovers tracking quickly. We can use the HSV color space for better robustness to illumination.

To visualize the accuracy and the recovery speed of our tracking algorithm, we calculate the number of overlapping pixels of ground truth rectangle and estimation rectangle. It can be seen in *fig.7* that our algorithm can track non-rigid object rather closely. Its robustness also makes it able to recover for complete losing track of object, as is noted is Human8 at frame 90, there is no overlapping pixels between estimation and ground truth, but the tracking recovers nonetheless.

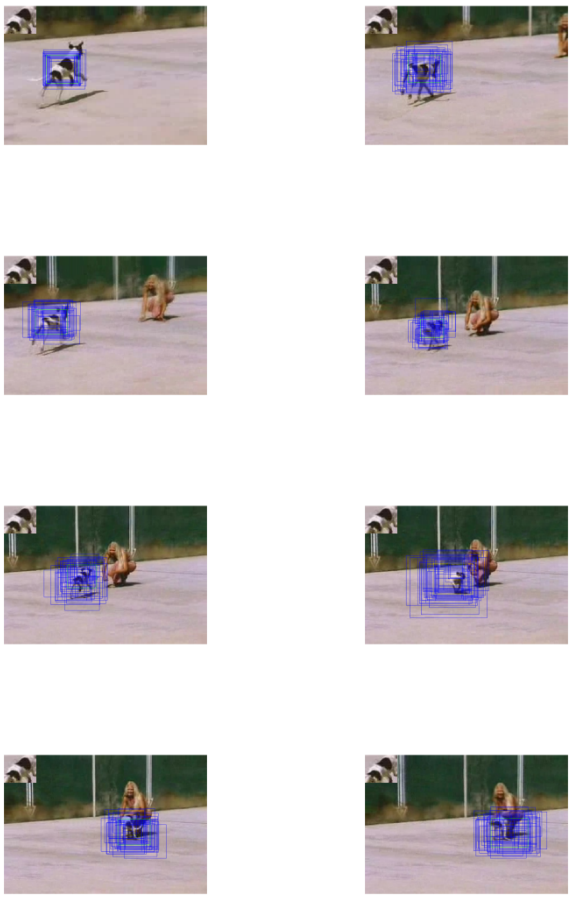


Fig 3. Eight frames out of Dog dataset

It can be seen that despite the running, the particles still track the distinct color of the fur.

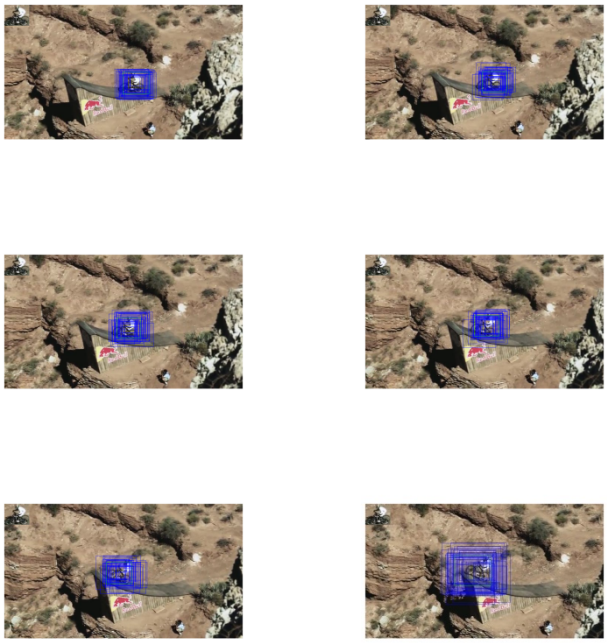


Fig 4. Six frames out of the first 30 frames for MountainBike dataset

It can be seen that even with similar colors in the background, 45 degree rotation and slight scale change, the particles track object rather closely.

Fig 5. 8 frames out of Bird2 dataset

It can been seen that particles tend to spread to capture occluded or fast moving object

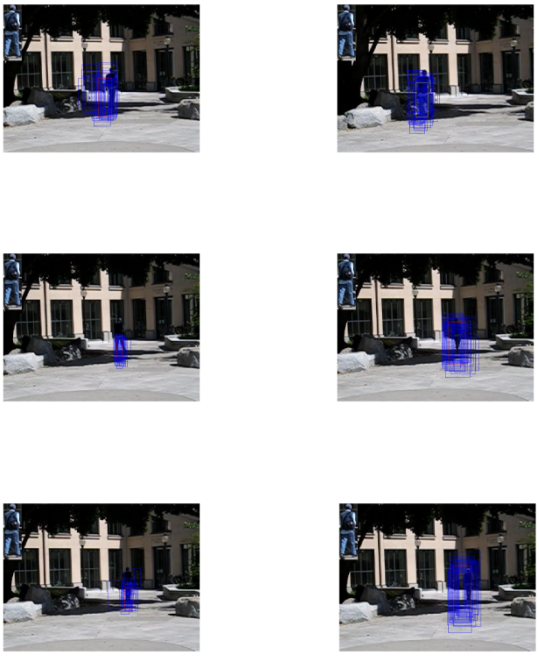
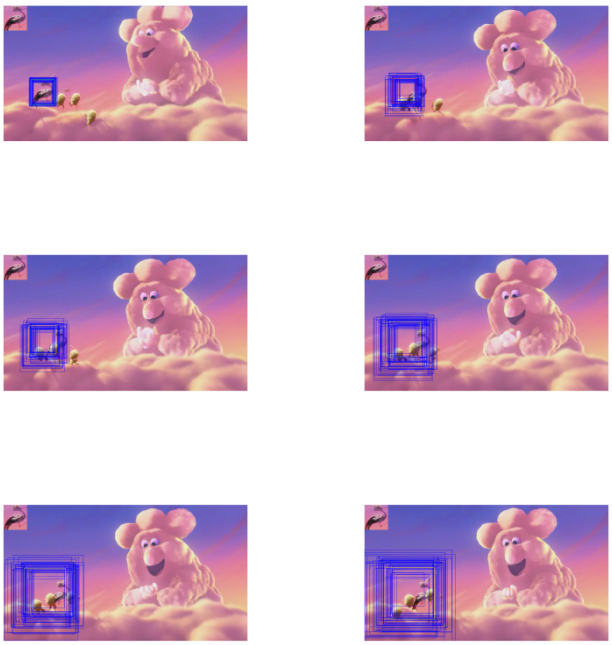


Fig 6. 6 frames out of Human8 dataset, during which a man is crossing a shadow

It can be seen that somewhere in the shadow, the particles almost lose the man to the dark shadow and window. However, the robustness of color histogram makes it possible to recover from it.



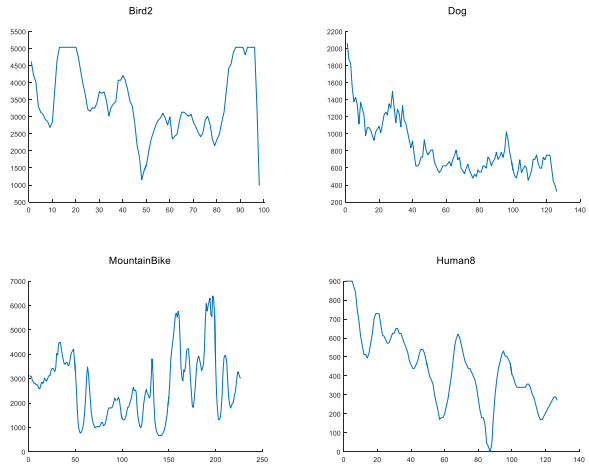


Fig 7. Overlapping pixels between ground truth and sequential tracking results.

Combined with the videos, it can be seen that our algorithm is able to track non-rigid objects and recover from complete losses.

V SUMMARY AND CONCLUSION

By following the literatures on visual tracking, we implement a color-based particle filtering algorithm which also updates the target. The weight is measured by the Bhattacharyya distance between histograms. Results have proven this algorithm is robust to rotation, scale change, occlusion, fast motion and deformation.

For future improvement, we can improve our algorithm by integrating features other than color histograms into the particle filtering framework, such as Local binary patterns, for particle filter is versatile with its weighing model. Also, we can use more complex model for target update, such as mixture of Gaussian. Besides, we can use lower languages to speed up computation for real-time tracking.

REFERENCES

[1]. Thrun, S., Burgard, W., and Fox, D. (2005).Probalistic robotics. MIT Press.

[2]. Nummiaro, Katja, Esther Koller-Meier, and Luc Van Gool. "An adaptive color-based particle filter." *Image and vision computing* 21.1 (2003): 99-110

[3]. Yang, Hanxuan, et al. "Recent advances and trends in visual tracking: A review." *Neurocomputing* 74.18 (2011): 3823-3831.

[4]. Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang. "Online object tracking: A benchmark." *Computer vision and pattern recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013.

[5]. Huttenlocher, Daniel P., Jae J. Noh, and William J. Rucklidge. "Tracking non-rigid objects in complex scenes." *Computer Vision, 1993. Proceedings, Fourth International Conference on*. IEEE, 1993.

[6]. Comaniciu, Dorin, Visvanathan Ramesh, and Peter Meer. "Kernel-based object tracking." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25.5 (2003): 564-577.

[7]. Kailath, Thomas. "The divergence and Bhattacharyya distance measures in signal selection." *Communication Technology, IEEE Transactions on* 15.1 (1967): 52-60.

[8]. Pérez, Patrick, et al. "Color-based probabilistic tracking." *Computer vision—ECCV 2002*. Springer Berlin Heidelberg, 2002. 661-675.

[9]. Isard, Michael, and Andrew Blake. "Condensation—conditional density propagation for visual tracking." *International journal of computer vision* 29.1 (1998): 5-28.