OBJECT TRACKING BY FINITE-STATE MARKOV PROCESS

Lan Dong, Stuart C. Schwartz

Dept. of Electrical Engineering, Princeton University
E-mail: {ldong, stuart}@ princeton.edu

ABSTRACT

The general problem of object tracking can be modeled as a Markov process and solved by computing probability distributions of the possible object states, followed by MAP estimation. This paper presents a new framework for the efficient estimation of the probability distribution of the states. In contrast to particle filters, where the possible states are numerous and random, we limit the possible states to a finite candidate set which is guaranteed with high probability to contain the true state of the object. After the problem is reduced to a finite-state Markov process (FSM), forward filtering is used to estimate the distribution of the object state. Moreover, the Viterbi algorithm can also be used to estimate the most likely state sequence. We test the new framework by both these methods and compare the tracking results. Experimental results show the effectiveness and efficacy of the proposed algorithm.

Index Terms— object tracking, finite-state Markov process, forward filtering, Viterbi algorithm, particle filters

1. INTRODUCTION

Real-time object tracking in complex environments is one of the key problems in visual surveillance. There are two major categories of algorithms: bottom-up process which usually consists of target representation and localization and top-down process which usually deals with evaluation of different hypotheses. Bottom-up processes like mean-shift [1] and blob model [2], are usually more sensitive to occlusion, noise and inconsistent movement. Top-down process is mostly filtering and data association which model the discrete-time dynamic system as Markov process through state distribution estimation. Kalman filter [3], Extended Kalman filter [4], particle filter [5] all have been used for the purpose of tracking objects. The probabilistic framework of these methods improves the robustness of the tracker. While Kalman filters are restricted to Gaussian distributions, particle filters can propagate more general distributions, which is a more practical assumption.

Particle filtering is based on Monte Carlo methods. The current probabilistic density of the object state (usually position and velocity) is represented by a set of random samples with associated weights and the new density is propagated based on these samples and weights and the observation. In order for the algorithm to converge to the correct density, the number of particles used must be large (at least hundreds). With the increase of dimensionality of the state space, the algorithm becomes much more computationally complex. In many real-time applications only a small percentage of the system resources can be allocated for tracking. Therefore, it is desirable to keep the computational complexity of a tracker as low as possible.

The main contribution of this paper is to introduce a new framework for efficient estimation of the general probability distribution of the object states for top-down object tracking process. The central idea is to generate at each time a set of candidate states which are guaranteed with high probability to contain the true state of the object we want to track. As we reduce the state space to a finite number of candidate states, the tracking problem can be modeled by a finite-state Markov process (FSM). This probabilistic framework brings the advantage of combining color, motion of the object and motion of the scene together in a principled manner. Assuming the density of the states is non-zero only for those candidate states, forward filtering is used to efficiently estimate the distribution of the object state and the tracker is estimated according to maximum a posterior (MAP) criterion. The Viterbi algorithm, alternatively, can be used to directly estimate the most likely state sequence. As the number of candidate states is much smaller than the number of particles for most applications, the computational complexity of our algorithm is much lower than that of particle filters.

The success of the algorithm depends on the process for generating candidate states so that the true state is always included in the set. This requirement is indeed satisfied in most scenarios due to sophisticated foreground segmentation methods. We give an example at the end of the paper that even if for some frames the candidate set does not include the true state, the probabilistic framework is able to re-localize the object when the true state is included at some later time for persistent frames.

The paper is organized as follows. The description of the FSM tracking is discussed in Sec.2. The experimental results are presented in Sec.3, followed by Sec.4, with conclusions.

2. OBJECT TRACKING BY FSM

FSM, among which Hidden Markov Model (HMM) is one specific kind, has been used for object tracking in a way different from ours. In [6], the whole range of the state space is divided into a finite number of cells with each cell associating with one state of the Markov chain. In [7], HMM is used spatially in one frame to find the contour of the object. Our method limits the states space to a set of candidate states and applies FSM temporally.

2.1 Markov Process Formulations

A Markov process is a temporal probabilistic model in which the state of the process is described by a single discrete random variable. It is a widely used model for top-down object tracking algorithms [3]-[5]. In the context of object tracking, the state at

time t is the information characterizing the object (usually location and velocity), which is denoted $\mathbf{x}_t$ and its history is $\mathbf{x}_{1:t} = \{\mathbf{x}_1,...,\mathbf{x}_t\}$. Similarly, the set of image features (observation) at time t is $\mathbf{z}_t$ with history $\mathbf{z}_{1:t} = \{\mathbf{z}_1,...,\mathbf{z}_t\}$. Note that there is no assumption (linearity or Gaussianity) about densities in the general treatment.

The object dynamics is assumed to form a Markov Chain:

$$p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

which means that the new state is only dependent on the previous state and not on any earlier ones. The conditional distribution $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ is called the transition model.

The assumption for observation $\mathbf{z}_t$ is that it depends only on the current state and not on the previous states or observations:

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}) = p(\mathbf{z}_t \mid \mathbf{x}_t)$$

It is called the observation model because it describes how the features are affected by the actual state of the object.

In addition to the transition model and observation model, we assume the distribution over initial states is uniform. These three distributions give us a specification of the complete joint distribution over all states. An illustration of the state diagram is given in Fig. 1 where the arrow indicates direct influences.

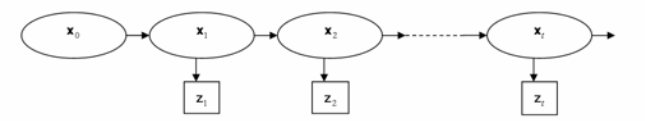


Figure 1: Markov process illustration

Once the probability density of the state(s) $\mathbf{x}_t$ ($\mathbf{x}_{1:t}$) given all the measurements up to that moment $\mathbf{z}_{1:t}$ is known, an optimal solution to the object tracking problem is the maximum *a posteriori* probability (MAP) estimation of the state.

After we give specific transition and observation models for tracking in Sec. 2.2, in Sec. 2.3, we discuss how to reduce the state space to a set of candidate states in order to use techniques such as forward filtering [8] and the Viterbi algorithm [9] to find the optimal solution of estimation, described in Sec. 2.4 and 2.5 respectively.

2.2 Transition Model and Observation Model

As dynamics of moving objects are often modeled as a second order process, we define the state of the object to be

$$\mathbf{x}_t = \begin{pmatrix} \mathbf{X}_t \\ \mathbf{X}_{t-1} \end{pmatrix}$$

where $\mathbf{X}_t$ is the position of the object at time t and $\mathbf{X}_t - \mathbf{X}_{t-1}$ is the velocity of the object. As the motion of the object will satisfy some temporal motion continuity constraints, we can predict the new position by current position and velocity of the object as $\mathbf{X}_p = \mathbf{X}_{t-1} + (\mathbf{X}_{t-1} - \mathbf{X}_{t-2})$ and consider the probability of a new position monotonically decreasing with the distance from the previously predicted position. A convenient distribution is the Gaussian with centre located at the predicted position:

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = p(\mathbf{X}_t, \mathbf{X}_{t-1} \mid \mathbf{X}_{t-1}, \mathbf{X}_{t-2}) \propto \exp(-\frac{\left\|\mathbf{X}_t - \mathbf{X}_p\right\|^2}{2\sigma_m^2})$$

The main image feature used in our algorithm is the color histogram of the object. Color information is a powerful feature to characterize the appearance of tracked objects since the color feature of an object or a region of interest often forms a very persistent localization cue.

The observation model must be defined in such a way as to favor color histograms close to the reference histogram of the object we are trying to track. Using the same color model as in [10], we denote by $\mathbf{h}_{\mathbf{x}}^c$ and $\mathbf{h}_{\text{ref}}^c$ with $c \in \{\text{R,G,B}\}$ the normalized color histogram of the image patch at position $\mathbf{X}$ and histogram of reference object in the red, green or blue (RGB) channel. The distance metric is based on the Bhattacharyya similarity coefficient:

$$D(\mathbf{h}_1, \mathbf{h}_2) = (1 - \sum_{i=1}^{B} \sqrt{\mathbf{h}_{i,1}\mathbf{h}_{i,2}})^{1/2}$$

where i denotes each bin of the histogram and B is the total number of bins. Then the observation model can be defined as:

$$p(\mathbf{z} \mid \mathbf{x}) \propto \exp(-\sum_{c\in\{R,G,B\}} D^2(\mathbf{h}_x^c, \mathbf{h}_{ref}^c) \Big/ 2\sigma_C^2)$$

When the object is composed of patches of distinct color, we can keep the relative spatial arrangement of these different patches by splitting the object region into sub-regions, each of which has its own color histogram. Assume each sub-region is rigidly connected and the colors in different sub-regions are conditionally independent, we can modify the observation model as:

$$p(\mathbf{z} \mid \mathbf{x}) \propto \exp(-\sum_{c\in\{R,G,B\}} \sum_{j=1}^{N_R} D^2(\mathbf{h}_{j,x}^c, \mathbf{h}_{j,ref}^c) \Big/ 2\sigma_C^2)$$

where $\mathbf{h}_{\text{j,ref}}^c$ is the histogram of sub-region j and N_R is the total number of sub-regions.

2.3 Candidate Selection

As the observation model is non-Gaussian, we are not able to use a Kalman Filter to estimate the state. Instead, we need to estimate the probability density function (pdf) of $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$. A possible state can be any pixel location in the current frame and previous frame. If there are 320x480 pixels, for example, for each frame, the size of the state space will be as big as $(320\text{x}480)^2$, which is formidable in term of computational complexity. Particle filters have reduced the space by careful sampling in the state space, while we try to decide a deterministic candidates set so that the true state is always included. The same idea has been used in [9].

We generate the candidate states by foreground segmentation. Assume we are using a stationary camera, the detection of foreground can be achieved by comparing each new frame with a representation of the background. Block based, rather than pixel based modeling of background can take advantage of the spatial correlation of the pixels and are robust to illumination change and noise. Adopting the DCT-based segmentation approach developed in [11], two features are used for each 8x8 block: The DC feature gives information on the average intensity and the AC feature, the square sum of low frequency AC coefficients, gives information on the energy of the block. After comparing these two features with the background features, a foreground block is detected and a candidate object state (position) is found if the image patch (same size as the object) at that position contains enough foreground blocks. This segmentation method has been shown to be invariant to noise and small scene changes, which is detailed in [11].

In most situations, the object is part of foreground segments. Thus, the true state is always included in the candidate set. In the case when the candidates do not contain the true state, as may be caused by occlusion or the object leaving the scene, the probabilistic framework can still guide the tracker back to the objects when it re-appears and persists for several frames.

2.4 Forward Filtering

Having set up the structure of a generic temporal model, we can formulate the object tracking problem as different inference tasks according to different applications.

For online-tracking, one commonly used inference task is to estimate the posterior distribution over the current state, given all evidence to date, which is $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$. Then MAP estimation is performed to find the optimal state. The theoretically optimal distribution computation is provided by the recursive Bayesian filter which solves the problem in two steps: first, the already computed pdf of the states at time $t-1$ is projected forward to time t, as determined by the dynamic equation; then it is updated using the new evidence $\mathbf{z}_t$.

Prediction step:
$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}) = \sum_{\mathbf{x}_{t-1}} p(\mathbf{x}_{t-1}, \mathbf{x}_t \mid \mathbf{z}_{1:t-1}) = \sum_{\mathbf{x}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1})$$

Updating step:
$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) \propto p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}) = p(\mathbf{z}_t \mid \mathbf{x}_t) \sum_{\mathbf{x}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1})$$

This whole process is called forward filtering. The second term in the summation is the already computed pdf of the last time step. Thanks to the candidate selection, we can efficiently compute the summation because the number of states is limited (or equivalently, the values of pdf of non-candidate states are all zero). If the average number of candidate states is n, as we need to compute the density for each current candidate state, the computational complexity at one time step is $O(n^2)$. The computational complexity of particle filters is $O(N)$, where N is the number of particles used. As N is usually much bigger than n^2, the computational load is greatly reduced.

Forward filtering can be shown to be equivalent to particle filters under the finite state assumption by choosing some appropriate proposal distribution for generating new samples. The proof can be found in a future journal version of this paper.

2.5 Viterbi Algorithm

Another inference task is to find the posterior distribution over the state sequence given all evidence to end, which is $p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T})$. Application for this task is off-line tracking where both the past and the future information can be used to estimate the path of the object through the whole video. Usually it is not advisable to compute the density for each possible state sequence explicitly as is done in forward filtering. There is a Viterbi algorithm for finding the MAP estimate of the state sequence directly.

The Viterbi algorithm is actually a forward dynamic programming problem. The recursive relationship between the most likely path to state $\mathbf{x}_t$ and most likely path to state $\mathbf{x}_{t-1}$ is:

$$\max_{\mathbf{x}_1 \cdots \mathbf{x}_{t-1}} p(\mathbf{x}_{1:t-1}, \mathbf{x}_t \mid \mathbf{z}_{1:t}) \propto \max_{\mathbf{x}_1 \cdots \mathbf{x}_{t-1}} p(\mathbf{x}_{1:t-1}, \mathbf{x}_t, \mathbf{z}_t \mid \mathbf{z}_{1:t-1})$$
$$= \max_{\mathbf{x}_1 \cdots \mathbf{x}_{t-1}} p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}) p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}) p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}, \mathbf{x}_t)$$
$$= p(\mathbf{z}_t \mid \mathbf{x}_t) \max_{\mathbf{x}_{t-1}} \left(p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \max_{\mathbf{x}_1 \cdots \mathbf{x}_{t-2}} p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}) \right)$$

At the end, the algorithm will give the probability for the most likely sequence reaching each of the final states. One can then select the most likely sequence overall by going back from the final state to internal states. It is again clear that without a good candidate selection process to limit the number of possible states at each time step, it is difficult to implement the Viterbi algorithm which needs to evaluate the path to each possible state. The computational complexity is also $O(n^2)$ if the average number of candidate states for each time step is n.

3. EXPERIMENTAL RESULTS

The first video is an indoor scene with two people walking towards each other and then passing by one another. Figure 2 shows how the state-density evolves as tracking of the occluded person progresses by forward filtering. Initial distribution is simply uniform in the absence of any measurements, shown in frame 1. The density is only non-zero where the two people are, based on foreground segmentation. As tracking progresses, one peak is much higher than the other (for some frames, the other peak is missing on the figure since it is relatively too small), indicating where the object we want to track is. Though the tracker was designed to track just one person, one can easily extend it to track multiple objects by the peaks of the state-density function.

We also test using the Viterbi algorithm to estimate the most likely state sequence of the object. The resulting object position vs. time is plotted in figure 3 with solid line, together with the result of forward filtering by the dotted line.

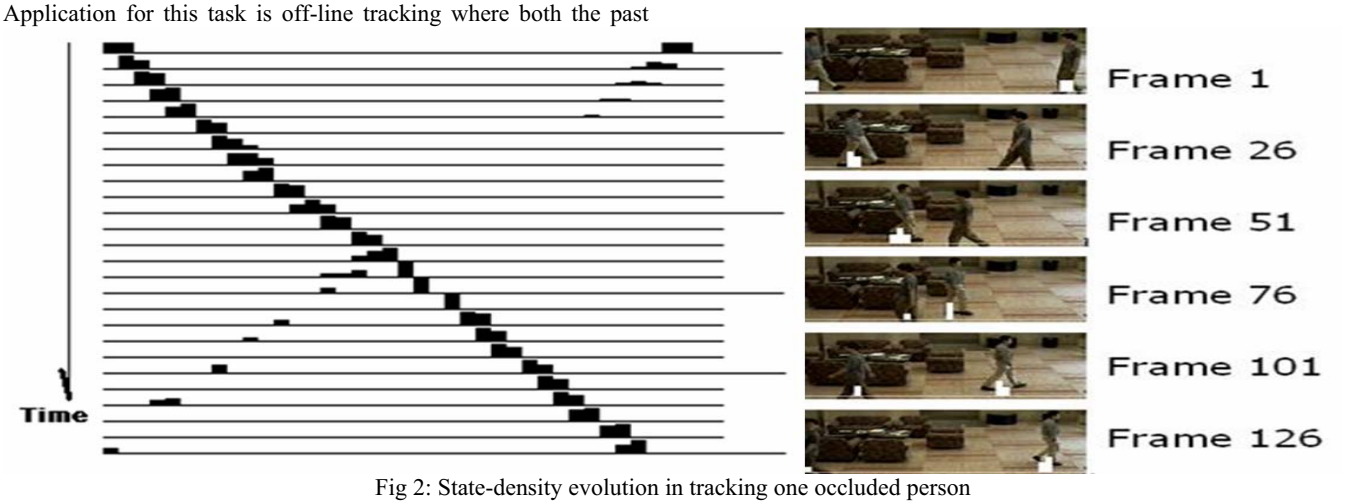


Fig 2: State-density evolution in tracking one occluded person

These two methods differ only for the first several frames where initialization occurs and the frames where the occlusion occurs. The Viterbi algorithm gives a smoother trajectory of the object position because it takes into account future information.

However, in case there is need to update the reference object model during tracking, the Viterbi algorithm is not applicable because it does not know the object location until the end of the video while forward filtering can estimate object location and update object model at each time step.

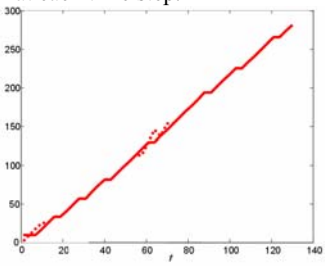
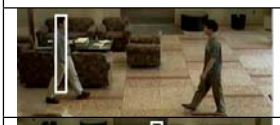
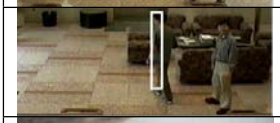
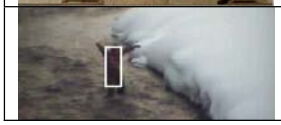


Figure 3: Tracking result for Viterbi algorithm (solid line) and forward filtering (dotted line)

We test both the forward filtering and Viterbi algorithm on two more sequences. The second video has the same background as the previous one but with different object motion. Two people walk towards each other, stop, turn around and walk back. The third video has a river with running water. Thanks to our block-based foreground segmentation, we avoid labeling the running water and successfully select the person as the candidate. The tracking result is shown in Table 1 where the average distance (DT) of ground-truth object to tracker-submitted object is used to indicate the performance. (DT=1 is 8 pixels as the candidate is block based.) The performance of Viterbi algorithm is a little better than forward filtering as expected.

For each video, the average number of candidates is less than 10 for each frame. Compared with the computational complexity of particle filters, where usually hundreds of particles are used, our algorithm greatly saves computational load and achieves real-time operation.

Table 1: Tracking performance of different videos

	Total # frames	forward filtering	Viterbi
	132	0.15	0.08
	300	0.1	0.05
	1041	0.2	0.15

As previously indicated, the success of the algorithm depends on the process for generating candidate set for which the true state is included. This requirement is satisfied for the testing videos discussed above. The following example illustrates the situation where for some frames the candidates do not include the true state.

In the video of two moving people in a parking lot, when the one we want to track is occluded by a tree in the background for some frames, only the other person is considered as the candidate so the tracker goes to the other person. When the one we want to track re-appears, it takes some frames for the density over the true state to accumulate and when the density is big enough, the tracker goes to the right person. Figure 4 shows some selected frames.

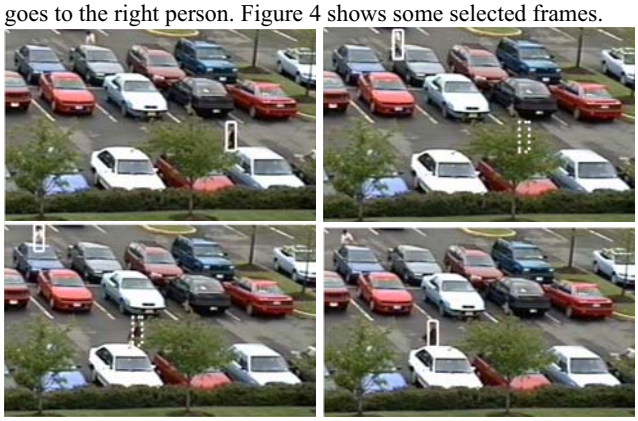


Figure 4: Sample frames (Frame 41, 56, 96,111): solid rectangle indicates tracker, dotted rectangle indicates true object location

4. CONCLUSIONS

In this paper, we modeled the object tracking task by a finite-state Markov process by reducing the state space to a candidate set, which is determined by block-based foreground segmentation. The probabilistic framework combines color, motion of the object and motion of the scene together in a principled manner. Forward filtering is used to estimate the probability distribution of the states and the Viterbi algorithm is used to estimate the most likely state sequence. Either of these two methods can be used for tracking according to different applications. The algorithm achieves very good performance and is a good alternative for particle filters when computational complexity is required to be low.

5. REFERENCES

[1] D. Comaniciu, V. Ramesh, P. Meer, “Real-Time Tracking of Non- Rigid Objects using Mean Shift”, IEEE Proc. of CVPR, Vol.2, pp.142 – 149, 2000

[2] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, “Pfinder: Real-time tracking of the human body”, IEEE Trans. Pattern Anal. Machine Intl., Vol.19, pp.780–785, 1997

[3] Y. Boykov, D. Huttenlocher, “Adaptive Bayesian recognition in tracking rigid objects”, IEEE Proc. Conf. on CVPR, Vol.2, pp.697–704, 2000

[4] R. Rosales, S. Sclaroff, “3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions”, IEEE Proc. Conf. on CVPR, pp.117–123, 1999

[5] M. Isard, A. Blake, “Condensation - Conditional density propagation for visual tracking”, Intl. J. of Computer Vision, vol.29, no.1, 1998

[6] X. Xie, R.J. Evans, “Multiple target tracking using hidden Markov models”, IEEE Intl. Radar Conf., pp.625 – 628, 1990

[7] Y. Chen, Y. Rui, T. Huang, “JPDAF-based HMM for real-time contour tracking”,IEEE Proc.Conf.CVPR, Vol.I, pp.543–550, 2001

[8] S.J. Russell, P. Norvig, Artificial Intelligence—A modern approach, second edition, Prentice-Hall, 2005

[9] F. Pitie, S.A. Berrani, A. Kokaram, R. Dahyot, “Off-line multiple object tracking using candidate selection and the Viterbi algorithm”, IEEE Proc. Intl. Conf. ICIP, Vol. 3, pp. 109-112, 2005

[10] P.Perez,J.Vermaak,A.Blake, “Data Fusion for Visual Tracking with Particles”, Proc. of the IEEE, Vol. 92-3, pp.495 –513, 2004

[11] L. Dong, S.C. Schwartz, “DCT-Based Object Tracking in Compressed Video”,IEEE Proc.ICASSP,Vol.2, pp. 665 -668, 2006