JPDAF Based HMM for Real-Time Contour Tracking

Yunqiang Chen
chenyq@ifp.uiuc.edu
ECE , Univ. of Illinois
Urbana, IL, USA 61801

Yong Rui
yongrui@microsoft.com
Microsoft Research
Redmond, WA 98052

Thomas S. Huang
huang@ifp.uiuc.edu
ECE , Univ. of Illinois
Urbana, IL, USA 61801

Abstract

Tracking objects using multiple cues yields more robust results. The well-known hidden Markov model (HMM) provides a powerful framework to incorporate multiple cues by expanding its observation. However, a plain HMM does not capture the inter-correlation between measurements of neighboring states when computing the transition probabilities. This can seriously damage the tracking performance. To overcome this difficulty, in this paper, we propose a new HMM framework targeted at contour-based object tracking. A joint probability data association filter (JPDAF) is used to compute the HMM's transition probabilities, taking into account the inter-correlated neighboring measurements. To ensure real-time performance, we have further developed an efficient method to calculate the data association probability via dynamic programming, which allows the proposed JPDAF-HMM to run comfortably at 30 frames/sec. This new tracking framework not only can easily incorporate various image cues (e.g., edge intensity, foreground region color and background region color), but also offers an on-line learning process to adapt to changes in the scene. To evaluate its tracking performance, we have applied the proposed JPDAF-HMM in various real-world video sequences. We report promising tracking results in complex environments.

1. Introduction

Many real-world applications, e.g., video surveillance [13] and video conferencing [14], require robust visual object tracking. Unfortunately, robust and efficient visual tracking in complex environments is still an open problem even after years of research.

To discriminate targets from clutter, various image cues have been proposed. For example, object contour is used in [9], face template is used in [4], and color distributions are used in [5, 24]. Because each of the above features may not be robust enough individually, more and more researchers are resorting to multiple visual cues [2, 18, 10, 20]. The main difficulty for multi-feature based tracking is, however, how to probabilistically integrate and adapt the multiple cues when objects and background gradually changetheir appearance.

Hidden Markov model (HMM) [17] provides a powerful and efficient way to incorporate multiple features by expanding the observation vectors. However, extending the HMM structure from handling 1D time series to 2D imagery data is challenging. Pseudo-2D-HMM (embedded HMM) has been proposed for character recognition [12], face recognition [15] and template matching [19]. A two-level HMM is defined, which consists of a set of super states, along with a set of embedded states. The super states model the horizontal dimension while the embedded states model the vertical dimension. The difficulty of applying this approach to real-time object tracking is the large number of parameters to be trained.

In this paper, we propose a new type of HMM targeted at real-time object tracking. We use parametric shape to model object contours. Observations are collected along the normal lines of the contour (see Figure 1) as in [9]. We define the HMM states to be the contour point location along each normal line. If we have N pixels on a normal line, we have N states in the HMM. This representation allows us to convert the contour in 2D image plane into an easier-to-solve 1D problem. Furthermore, this new HMM formulation allows us to develop a powerful object tracking framework in the following way:

- Multiple tracking cues, e.g., edge intensity, foreground /background region properties, can easily be integrated into the tracking process probabilistically.
- In addition to using the standard contour smoothness constraint to compute the state transition probabilities, we further develop a joint probability data association filter (JPDAF) to encode richer inter-relationships between neighboring measurements, thus leading to more accurate transition probabilities. The resultant new HMM is termed JPDAF-HMM.
- A robust on-line training process similar to the training of traditional HMM (Baum Algorithm) is also proposed to adapt the observation model through time.

The rest of the paper is organized as follows. In Section 2, we present a 1D HMM framework that can easily incorporate multiple visual cues. Specifically, we will discuss the 1D contour representation, a multi-cue observation model, and a transition probability model based on the contour smoothness constraint. The transition probability is one of the most important components in a HMM. More accurate transition probabilities lead to more accurate and robust tracking results. In Section

3, in addition to the standard contour smoothness constraint as used in Section 2, we develop a JPDAF to encode richer interrelationships between neighboring measurements, thus leading to more accurate transition probabilities. An efficient algorithm for calculating JPDAF is also presented for real-time performance. In Section 4, We present an adaptive learning process to update the observation models through time to handle dynamic environments. The overview of the tracking diagram is also presented. To evaluate the performance of the proposed approach, we have implemented a real-time tracking system, and applied it on various real-world video sequences. We report promising tracking results in Section 5. Concluding remarks are given in Section 6.

2 Contour tracking using HMM

For tracking non-rigid objects in complex environments, the active contour model has been proved to be a powerful tool [11, 21, 16]. For real-time tracking, it is imperative to have an efficient optimization method to find the best contour. A common difficulty of the active contour model is its deficiency in recursively refining the contours in the 2D image plane [1, 6]. In fact, because of the aperture problem, only the deformations along the normal lines of the contours can be detected. So, we can restrict the contour searching to a set of normal lines to the predicted contour position (see Figure 1). Let $\phi = 1, ..., M$, be the index of the normal lines and $\lambda = -N, ..., N$, be the index of pixels along a normal line. Furthermore, let $\rho_\phi(\lambda)$ denote the image intensity at pixel λ on line ϕ. That is,

$$\rho_\phi(\lambda) = I(x_{\lambda\phi}, y_{\lambda\phi}) \tag{1}$$

where $(x_{\lambda\phi}, y_{\lambda\phi})$ is the corresponding image coordinate of the pixel λ on the ϕth normal line. $I(x_{\lambda\phi}, y_{\lambda\phi})$ is the image intensity at $(x_{\lambda\phi}, y_{\lambda\phi})$.

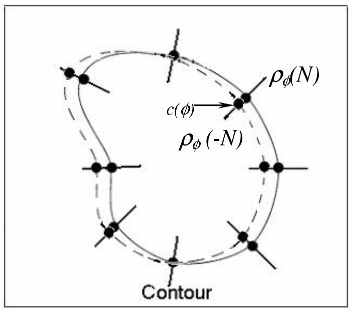

Figure 1. Illustration of the 1D contour model: *At frame t, the solid curve is the predicted contour based on the tracking results on previous frames. The dashed curve is the true contour that we want to find. A set of measurements are collected along the M normal lines of the predicted contour. $c(\phi)$ is the true contour point on the ϕth normal line. The true contour can be found if we can detect all the $c(\phi), \phi \in [1, M]$.*

Each normal line has $2N+1$ pixels, which are indexed from $-N$ to N. The center point of each normal line is placed on the predicted contour and indexed as 0. If the prediction were always accurate, the detected contour points on all normal lines should be exactly at the center, i.e., $c(\phi) = 0, \forall\phi \in [1, M]$. In reality, however, we need to find the true contour point $c(\phi)$ based on the measurements. Note that instead of representing the contour by a 2D image coordinate, we can now represent the contour by a 1D function $c(\phi), \phi = 1, ..., M$.

To detect the contour points accurately, different cues (edge intensity, color model of the foreground and background) and prior constraints (e.g. contour smoothness constraint) can be integrated by using HMM. The hidden states of the HMM are the true contour point on each normal line, (denoted as $\mathbf{s} = \{s_1, ..., s_\phi, ..., s_M\}$). The observations of the HMM, $\mathbf{O} = \{O_1, ..., O_\phi, ..., O_M\}$, are collected along each normal line ϕ. A HMM is specified by the number of states (in our case, $2N+1$), the observation model $P(O_\phi|s_\phi)$, and the transition probability $p(s_\phi|s_{\phi-1})$. Given current state s_ϕ, the current observation O_ϕ is independent of previous state $s_{\phi-1}$ and previous observation $O_{\phi-1}$. In addition, because of the Markovian property, we have $p(s_\phi|s_1, s_2, ..., s_{\phi-1}) = p(s_\phi|s_{\phi-1})$. This is shown in Figure 2.

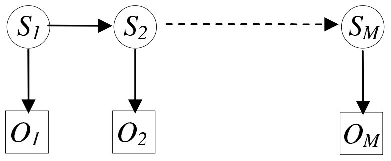

Figure 2. Graphic model of contour tracking

In Section 2.1., we will give detailed description on how to incorporate multiple cues into the observation model. A simplified state transition model is then discussed in Section 2.2 based on the contour smoothness constraint. Given the observation model and the state transition probabilities, in Section 2.3, we present how to find the optimal contour efficiently via the Viterbi algorithm [17].

Different from other contour models [7] that also resort to the Dynamic Programming to obtain the global optimal contour, HMM offers an elegant way to integrate multiple visual cues and a probabilistic model adaptation formula (shown in Section 4) to adapt itself to the dynamic environments, which is very important for a robust tracking system.

2.1. Observation likelihood of multiple cues

The observation on line ϕ (represented as O_ϕ) can include multiple cues, e.g., pixel intensity (i.e., $\rho_\phi(\lambda), \lambda \in [-N, N]$) and edge detection (i.e., $\mathbf{z}_\phi$) along the line.

First, the observation likelihood model of the edge detection results $\mathbf{z}_\phi$ can be derived following similar approaches in [9]. Because of noise and image clutter, there can be multiple edges along each normal line. Let J be the number of detected edges ($\mathbf{z}_\phi = (z_1, z_2, ..., z_J)$). Of the J edges, at most one is the true contour. We can therefore define $J + 1$ hypotheses:

$$\begin{aligned} H_0 &= \{e_j = F : j = 1, ..., J\} \\ H_j &= \{e_j = T, e_k = F : k = 1, ..., J, k \neq j\} \end{aligned} \tag{2}$$

where $e_j = T$ means the jth edge is associated with the true contour, $e_j = F$ otherwise. Hypothesis H_0 therefore means that none of the edges is associated with the true contour.

With the assumption that the clutter is a Poisson process along the line with spatial density γ and the true target measurement is normally distributed with standard deviation σ_z, we can obtain the edge likelihood model as follows:

$$p(\mathbf{z}_\phi | s_\phi = \lambda_\phi) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma_z q\gamma} \sum_{m=1}^{J} \exp(-\frac{(z_m - \lambda_\phi)^2}{2{\sigma_z}^2}) \tag{3}$$

where q is the prior probability of hypothesis H_0.

In addition to the edge likelihood model, other cues about the region properties of the foreground and background, e.g., mixture color models, can easily be integrated into the HMM framework. Let $p(v|FG)$ and $p(v|BG)$ represent the color distribution for the foreground (FG) and background (BG), respectively. The posterior probabilities $P(BG|v)$ and $P(FG|v)$ can be derived: (Assume $p(FG) = p(BG)$)

$$P(BG|v) = \frac{p(v|BG)}{p(v|BG) + p(v|FG)} \tag{4}$$

$$P(FG|v) = \frac{p(v|FG)}{p(v|BG) + p(v|FG)} \tag{5}$$

If $s_\phi = \lambda_\phi$ is the contour point on line ϕ, we know that the segment $[-N, s_\phi]$ is on the foreground and the segment $[s_\phi + 1, N]$ is on the background. Combining the edge likelihood model and the color posterior probabilities, we have the following multi-cue observation likelihood function:

$$\begin{aligned} P(O_\phi | s_\phi) = p(z|s_\phi) \cdot & \prod_{i=-N}^{s_\phi} P(FG|v = \rho_\phi(i)) \\ \cdot & \prod_{i=s_\phi+1}^{N} P(BG|v = \rho_\phi(i)) \end{aligned} \tag{6}$$

Other cues can also be integrated in a similar way. As we will show in Section 4, our proposed HMM framework also allows us to update the foreground/background color model in a probabilistic way through time.

2.2. Computing transition probabilities

In addition to the observation model discussed in the previous sub-section, another important component in HMM is the transition probability. It determines how a state transits to another state. In this section, we will use the standard contour smoothness constraint to derive the transition probability. We will defer a much more sophisticated smoothness constraint based on region properties to Section 3.

Here we follow the philosophy in traditional snake model [1], which use an internal energy term to penalize the roughness of the contour. But instead of using an internal energy in an optimization framework, we encode this constraint in transition probabilities. To achieve this, the smoothness constraint has to be represented in a causal form. In Figure 1, we can see when the normal lines are dense (e.g., 30 in our experiments), the true contour points on adjacent normal lines tend to have the same displacement from the predicted contour position (indexed as 0 on each normal line). This correlation is causal and can be captured by transition probabilities $p(s_\phi | s_{\phi-1})$:

$$p(s_\phi | s_{\phi-1}) = c \cdot e^{-(s_\phi - s_{\phi-1})^2/\sigma_s^2} \tag{7}$$

where c is a normalization constant and σ_s is a predefined constant that regulates the smoothness of the contour. This transition probability will penalize sudden changes of the contour points between adjacent lines, hence resulting in a smooth contour. The best contour can be obtained by the Viterbi algorithm described in the following section.

2.3. Best contour searching by Viterbi algorithm

Given the observation sequence $\mathbf{O} = \{O_\phi, \phi \in [1, M]\}$ and the transition probabilities $a_{i,j} = p(s_{\phi+1} = j | s_\phi = i)$, the best contour can be found by finding the most likely state sequence $\mathbf{s}^*$. This can be efficiently accomplished by the Viterbi algorithms:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} P(\mathbf{s}|\mathbf{O}) = \arg\max_{\mathbf{s}} P(\mathbf{s}, \mathbf{O}) \tag{8}$$

Let's define

$$V(\phi, \lambda) = max_{s_{\phi-1}} P(O_\phi, s_{\phi-1}, s_\phi = \lambda) \tag{9}$$

Using the Markov conditional independence assumptions, it can be recursively computed as follows:

$$\begin{aligned} V(\phi, \lambda) =& P(O_\phi | s_\phi = \lambda) \\ & \cdot \max_j P(s_\phi = \lambda | s_{\phi-1} = j) V(j, \phi - 1) \end{aligned} \tag{10}$$

$$\begin{aligned} j^*(\phi, \lambda) =& P(O_\phi | s_\phi = \lambda) \cdot \\ & \arg\max_j P(s_\phi = \lambda | s_{\phi-1} = j) V(j, \phi - 1) \end{aligned} \tag{11}$$

with the initialization $V(1, \lambda) = \max_{s_1} P(O_1|s_1)P(s_1)$, where the initial state probabilities $P(s_1) = \frac{1}{2N+1}, s_1 \in [-N, N]$. The term $j^*(\phi, \lambda)$ records the "best previous state" from state λ at line ϕ. We therefore obtain at the end of the sequence $\max_{\mathbf{s}} P(\mathbf{O}, \mathbf{s}) = \max_\lambda V(M, \lambda)$. The optimal state sequence $\mathbf{s}^*$ can be obtained by back tracking j^*, starting from $s_M^* = \arg\max_\lambda V(M, \lambda)$, with $s_{\phi-1}^* = j^*(s_\phi^*, \phi)$. The computation cost of the Viterbi algorithm is $O(M \cdot (2N + 1))$. Unlike traditional active contour model [1, 6], this method can give us the optimal contour without recursively searching the 2D image plane. Given the best state sequence $\mathbf{s}^* = \{s_1^*, ..., s_M^*\}$, we denote the corresponding image coordinate of the best contour point s_ϕ^* on line ϕ by $[x_\phi, y_\phi]$.

To increase the stability of tracking, we reduce the degree of freedom of the contour by fitting it to a parametric shape. Different kinds of parametric shape can be used, e.g. B-Spline or ellipse, etc. In our experiment, we use ellipse to approximate the object contours. Then we have:

$$ax_\phi^2 + by_\phi^2 + cx_\phi y_\phi + dx_\phi + ey_\phi - 1 = 0 \tag{12}$$

which can be representation in matrix form: $A \cdot \mathbf{f} = \mathbf{b}$, where

$$A = \begin{bmatrix} x_1^2 & y_1^2 & x_1 y_1 & x_1 & y_1 \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ x_M^2 & y_M^2 & x_M y_M & x_M & y_M \end{bmatrix}$$

and $\mathbf{b} = [1, 1, ..., 1]^T$. The parameters of the best-fit ellipse $\mathbf{f}^* = [a, b, c, d, e]^T$ can be obtained by the least mean square (LMS) solution:

$$\mathbf{f}^* = (A^T A)^{-1} A^T \mathbf{b} \tag{13}$$

The above ellipse representation $\mathbf{f} = [a, b, c, d, e]^T$ is convenient mathematically. But there is no clear physical interpretations of the five parameters. In tracking, a different 5-element ellipse representation is normally used:

$$\theta = [x, y, \alpha, \beta, \Phi]$$

where (x, y) is the center of the ellipse, α and β are the lengths of the major and minor axes of the ellipse, and Φ is the orientation of the ellipse. Because $\mathbf{f}$ and θ are two representations of the same ellipse [8], we use them interchangeably in the paper.

3. Improving transition probabilities: The JPDAF-HMM

Transition probability is one of the most important parameters in an HMM. It directly affects the best state sequence estimation. In Section 2.2, we have derived a simplified way of computing transition probabilities based on the contour smoothness constraint. Even though simple, it does not take into account all the information available to us. Specifically, the contour smoothness constraint only considers the contour points themselves. All the other pixels on the measurement lines are ignored. Unfortunately, considering the contour points only can be dangerous. This is especially true when noise is presented in the measurement, which is almost always the case in reality. In this section, we develop a JPDAF-based method to encode not only the contour smoothness constraint, but also the region smoothness constraint observed on all the pixels along the normal lines. To ensure real-time performance, we further develop an efficient JPDAF algorithm by using dynamic programming. The resulted JPDAF-HMM runs comfortably in real-time.

3.1. Encoding region smoothness constraint using JPDAF

Under normal condition, pixel intensity values of human body parts (e.g., face or head) change smoothly inside their regions. It is therefore a reasonable assumption that in human tracking, the foreground and background have smooth region properties so that the measurements in Equation (1) on two adjacent lines are similar. Let s_ϕ and $s_{\phi+1}$ be the contour points on line ϕ and line $\phi + 1$, respectively. These two contour points segment the two lines into foreground segments and background segments. Based on the region smoothness assumption, not only should s_ϕ and $s_{\phi+1}$ be close to each other, all the other pixels on the two lines should also match well. To ensure the region smoothness constraint, we need a joint probability data associate filter (JPDAF) to conduct the line matching. That is, it is not a single point to single point matching problem, but rather a $(2N + 1)$ points to $(2N + 1)$ points matching problem. By considering all the pixels along the lines together, we can have much more robust matching results. The transition probabilities based on this new matching paradigm are therefore much more accurate. Let $D^F(i, j)$ and $D^B(i, j)$ be the matching distances of the foreground ($[-N, i]$ on line ϕ and $[-N, j]$ on line $\phi+1$) and background ($[i+1, N]$ on line ϕ and $[j + 1, N]$ on line $\phi + 1$), respectively. A more accurate transition probability can then be defined to replace the simplified one used in Section 2.2 (see Equation 7):

$$\begin{aligned} log(p(s_2|s_1)) = & D^F(s_1, s_2) + D^B(s_1, s_2) \\ & + (s_2 - s_1)^2/\sigma_s^2 \end{aligned} \tag{14}$$

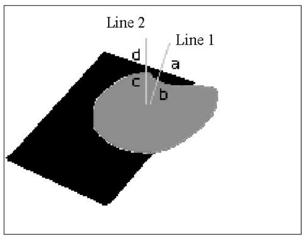

Figure 3. An illustration of the region smoothness constraint: *both 'bc' and 'ad' can be good contour candidates that separate the two lines into well matched foreground and background. On the other hand, 'ac' is a bad choice because there will be no good match for either foreground or background.*

The region smoothness concept can be illustrated by a synthesized image in Figure 3. There are two regions where the lighter region represents the object and the darker region represents the background clutter. There are two adjacent normal lines shown in the figure, i.e., line 1 and line 2. Points 'a' and 'b' are detected edge points on line 1. Similarly, points 'c' and 'd' are detected edge points on line 2. Our goal is to find where are the contour points on these two lines. The measurements on these two lines are shown in Figure 4. They are similar to each other except for some distortions. Based on the contour smoothness constraint only, the contour from 'a' to 'c' and the contour from 'b' to 'c' have almost the same transition probabilities because $|a - c| \approx |b - c|$. However, if we consider the region smoothness assumption as well, the possible contour can only be 'ad' or 'bc', but never be 'ac' or 'bd'. (The contour candidates 'ad' and 'bc' will be further discriminated by HMM based on all the observation lines.)

To verify this region smoothness concept, we have applied it on the synthesized image shown in Figure 5. When we use the region smoothness constraint, it has a large penalty for the contour to jump from $'a'$ to $'c'$, and the result is shown in Fig-

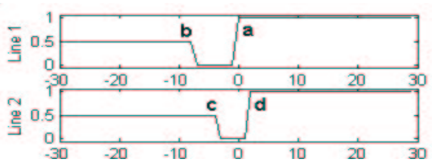

Figure 4. Observations on line 1 and line 2

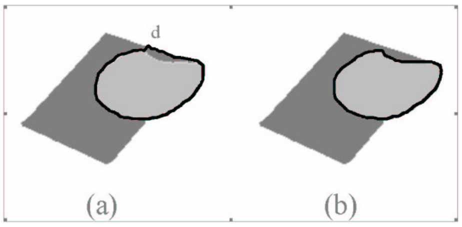

Figure 5. Test on synthesized image: *(a) Without the region smoothness constraint, the contour is distracted by a strong edge point (point d) from the background clutter. (b) With the region smoothness constrain, the contour is correctly detected.*

ure 5 (b). Without this constraint, the contour is distracted by the strong edge (point 'd') from the background clutters, as shown in Figure 5 (a).

Different from the uniform statistic region model in [3], our assumption here is more relaxed. The object can have various color regions (e.g. front or side view of human head with hair as shown in experiments) each of which is homogeneous *locally*. To illustrate this, another test is shown in Figure 6 which has different intensity regions in the foreground. We can see the difference between Figure 5 and Figure 6: the observations on line 2 are not the same. There is no segment 'cd'. No matter we match 'a' to 'c' or 'b' to 'c', the segment 'ab' does not have a matching part. Therefore, the region smoothness constraint penalties are the same for matching 'a' to 'c' or 'b' to 'c'. The algorithm favors the result in Figure 6 (b) because it is smoother.

3.2. Efficient association by dynamic programming

In the previous sub-section, we have shown that much more accurate transition probabilities between adjacent lines can be obtained by using the region smoothness constraint, which in turn results in better results (see Equation (14)). To ensure real-time tracking performance, in this sub-section we will explore an efficient algorithm for implementing the JPDAF.

The association probabilities between all the possible pairs of states ($(2N+1)^2$) should be calculated. Because we can not assume observations on two adjacent normal lines are exactly the same, correlation of the measurements on two lines can not be used. Windowed correlation technique could be used to estimate $p(s_\phi|s_{\phi-1})$ [22]. However, it has the following limitations: 1). it is only an approximation to the true $p(s_\phi|s_{\phi-1})$; 2). there is no principled way to determine the right window size; and 3). the computation is not efficient: it

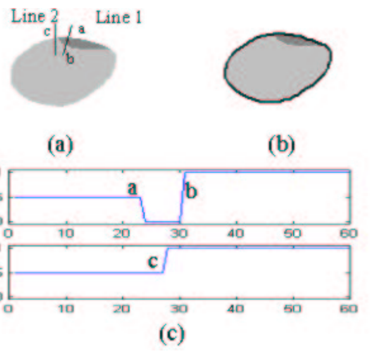

Figure 6. Foreground object with multiple regions: *(a) The synthesized image. (b) Contour tracking with regions smoothness constraint. (c) Observation on line 1 and line 2.*

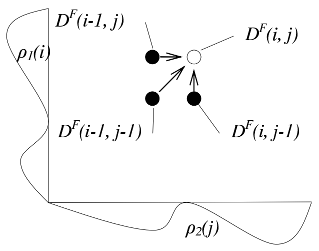

Figure 7. Dynamic programming: $D^F(i,j)$ *takes the min value of* $D^F(i-1,j)$, $D^F(i,j-1)$ *and* $D^F(i-1,j-1)$ *plus the matching distance of* $\rho_1(i)$ *and* $\rho_2(j)$.

costs $O(W\cdot(2N+1)^2)$ to compute all $(2N+1)^2$ transition probabilities, where W is the length of the window. To overcome these limitations, we develop a more efficient algorithm using dynamic programming. It costs $O((2N+1)^2)$ with a smaller coefficient than W to calculate the association probability for all $(2N+1)^2$ transition probabilities, and there is no window size to be determined. Given measurement lines 1 and 2, the calculation of the matching distance can be explained in the following recursive equation (see also Figure 7):

$$D^F(i,j) = min\begin{cases} D^F(i-1,j) + d(\rho_1(i),\rho_2(j)) \\ D^F(i,j-1) + d(\rho_1(i),\rho_2(j)) \\ D^F(i-1,j-1) + d(\rho_1(i),\rho_2(j)) \end{cases} \tag{15}$$

where $d(.,.)$ is the cost of matching two pixels. $D^F(i,j)$ is the best matching distance between segment $[-N,i]$ on line 1 and segment $[-N,j]$ on line 2. We start from $D^F(-N,j) = D^F(i,-N) = 0$, where $i,j \in [-N,N]$ and use the above recursion to obtain the matching distance $D^F(i,j)$ from $i = -N$ to N and $j = -N$ to N. Similar process should be gone through to calculate the $D^B(i,j)$, but start from $D^B(i,N) = D^B(N,j) = 0$ to $D^B(-N,-N)$. After obtaining all the matching distances, the state transition probabilities can be computed and contour tracking can be accomplished by the Viterbi algorithm described in Section 2.3.

4. Extending JPDAF-HMM to temporal domain

The previous two sections have described the JPDAF-HMM model for each individual frame. In this section, we will extend it to handle tracking in video sequences. Specifically, we will discuss how to probabilistically train the parameters for frame t based on the tracking results at frame $t-1$. We will then give a complete algorithm for the JPDAF-HMM tracking framework.

4.1. On-line learning of observation model

In dynamic environment, both object and background may gradually change their appearances. An on-line training is therefore necessary to adapt the observation likelihood models dynamically. A naive way is to completely trust the contour returned by the Viterbi algorithm at frame $t-1$, and average all the pixels inside and outside the contour to obtain the new foreground/background color model at frame t. However, if error occurs at frame $t-1$, this procedure may adapt the model in the wrong way. Fortunately, HMM allows us to train the observation models in a probabilistic way.

Instead of completely trusting the contour obtained at frame $t-1$, we can make a soft decision of how to update the observation models by using the forward-backward algorithm. The "forward probability distribution" is defined as follow:

$$\alpha_\phi(s) = p(O_1, O_2, ..., O_\phi, s_\phi = s) \tag{16}$$

which can be computed efficiently using the recursion,

$$\alpha_1(s) = p(s_1 = s)p(O_1|s_1 = s) \tag{17}$$

$$\alpha_{\phi+1}(s) = \left[\sum_u \alpha_\phi(u)a_{u,s}\right] p(O_{\phi+1}|s_{\phi+1} = s) \tag{18}$$

Similarly, the "backward probability distribution" is:

$$\beta_\phi(s) = p(O_{\phi+1}, O_{\phi+2}, ..., O_M, s_\phi = s) \tag{19}$$

which can be computed efficiently using the recursion,

$$\beta_M(s) = 1 \tag{20}$$

$$\beta_\phi(s) = \sum_u a_{s,u} p(O_{\phi+1}|s_{\phi+1} = u)\beta_{\phi+1}(u) \tag{21}$$

After computing the forward and backward probability, we can compute the probability of each state at line ϕ;

$$P(s_\phi = s|\mathbf{O}) = \frac{\alpha_\phi(s)\beta_\phi(s)}{\sum_u \alpha_\phi(u)\beta_\phi(u)}, \quad s \in [-N, N] \tag{22}$$

which represents the probability of having contour point at s on the measurement line ϕ.

Based on these probabilities, the probability of pixel λ_ϕ being on foreground (or background) can be computed by integrating $P(s_\phi = s|\mathbf{O})$ along the normal line.

$$P(\lambda_\phi \in BG) = 1 - P(\lambda_\phi \in FG) = \sum_{s=-N}^{s=\lambda_\phi} p(s_\phi = s|\mathbf{O}) \tag{23}$$

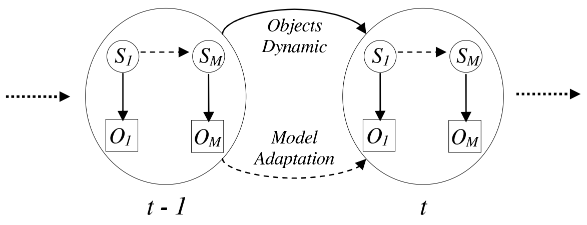

Figure 8. The diagram of the tracking

This probability gives us a robust way to weigh different pixels during the observation models adaptation. The more confidently classified pixels will contribute more to the color model while the less confidently classified pixels will contribute less:

$$\begin{aligned} p(v|BG) &= \frac{\sum_{s=-N}^{N} P(s \in BG) \cdot (O_\phi(s) == v)}{\sum_{s=-N}^{N} P(s \in BG)} \\ p(v|FG) &= \frac{\sum_{s=-N}^{N} P(s \in FG) \cdot (O_\phi(s) == v)}{\sum_{s=-N}^{N} P(s \in FG)} \end{aligned} \tag{24}$$

The new adapted models will reflect the changing color distributions during the tracking. It is then plugged back into Equation (6) during the contour searching in the next frame. To reduce the number of parameters, we are not training the transition probabilities, which is related to the density of the observation lines and remains relative constant during the tracking process.

4.2. Tracking diagram

The complete JDAPF-HMM tracking procedure is summarized as follows (also see Figure 8):

1. **Prediction:** Predict where the object will be in the current frame t based on the tracking results in previous frame $t-1$ and the object's dynamics. Observations are collected along a set of normal lines of the predicted contour. We adopt the Langevin process to model the human movement dynamics [23]:

$$\begin{bmatrix} \theta_\mathbf{t} \\ \dot{\theta}_\mathbf{t} \end{bmatrix} = \begin{bmatrix} 1 & \tau \\ 0 & a \end{bmatrix} \begin{bmatrix} \theta_\mathbf{t-1} \\ \dot{\theta}_\mathbf{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} m_t \tag{25}$$

   where $\theta = [x, y, \alpha, \beta, \Phi]$ is the parametric ellipse, $a = \exp(-\beta_\theta\tau)$, $b = \bar{v}\sqrt{1-a^2}$. β_θ is the rate constant, m is a thermal excitation process drawn from Gaussian distribution $N(0, Q)$, τ is the discretization time step and $\bar{v}$ is the steady-state root-mean-square velocity.

2. **Contour tracking:**
   (a) **Observation likelihood**: Evaluate the observation likelihood for every pixel on each normal line ϕ:

$$p(O_\phi|s_\phi = \lambda_\phi), \lambda_\phi \in [-N, N], \phi \in [1, M]$$

   based on edge detection and the color value of each pixel on the line by using Equation (6).

(b) **Transition probabilities**: Evaluate the state transition probabilities based on JPDAF as shown in Equation (14). Efficient dynamic programming solution to JPDAF is explained in Section 3.2.

(c) **Best contour**: With previously computed observation likelihood and the transition probability matrix, best contour w.r.t the given observations can be find by the Viterbi Algorithm described in Section 2.3.

(d) **Contour fitting**: Based on the detected contour, fit the best ellipse using Equation (13).

3. **Model adaptation:** Using forward-backward algorithm to estimate a soft classification of each pixel (to foreground and background) on the normal lines and update the color model of foreground and background based on Equation (24). Update the velocity of the objects (e.g., translation, rotation and scaling). Go to step 1 when new frame arrives.

To begin this tracking procedure, a separate initialization module is needed. This can be done either manually or by change detection [20]. We currently hand initialize on the first frame in each sequence.

5. Experiments

To validate the efficacy and robustness of the proposed JPDAF-HMM approach, we have applied it to various real-world test sequences. In the experiments, the human head is modeled by an ellipse $\theta = [x, y, \alpha, \beta, \Phi]$, as discussed in Section 2.3. For each ellipse, 30 normal lines are used, i.e., $M = 30$. Each line has 21 observation locations, i.e., $N = 10$. We use the Langevin process to model the human movement, as discussed in Section 4. The JPDAF-HMM tracking algorithm is implemented in C++ on Windows 2000 platform. No attempt is made on code optimization and the current system runs at 30 frames/sec comfortably on a standard Dell PIII 1G machine.

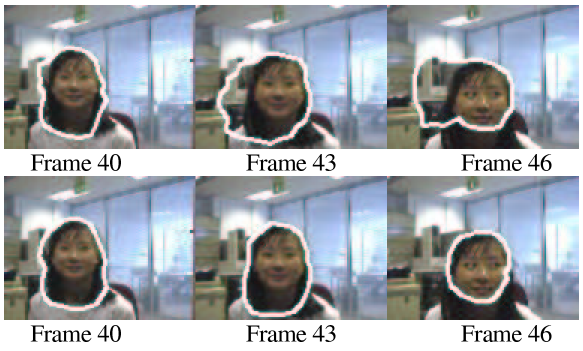

Figure 9. Sequence A: comparison of plain HMM with JPDAF-HMM: *Top row is the results of plain HMM without JPDAF. Bottom row is the results of JPDAF-HMM.*

To test the robustness of the proposed algorithm, we use video sequences captured by both a fixed camera and a pan/tilt /zoom camera. The sequences simulate various tracking conditions, including appearance changes, quick movement, out of plane rotation, shape deformation, camera zoom in and out, and partial occlusion. Sequence A, shown in Figure 9, is in a cluttered office environment with 400 frames (30 frames/second). Note that the background clutter (sharp edges and similar color) impose great challenges to any contour based visual tracking algorithms. The color of the door is very similar to that of human face and causes great difficulty to color-based tracking algorithms. For fine-level comparing purpose, we display the contour results returned from the Viterbi algorithm instead of the final ellipse. The comparison is shown in Figure 9. In the plain HMM, tracking is distracted by the strong edges on the background when the foreground boundary does not have high contrast and the error is gradually enlarged. The JPDAF-HMM ensures that the contour is not distracted by the sharp edges on the background because of its sophisticated transition probabilities (see Section 3).

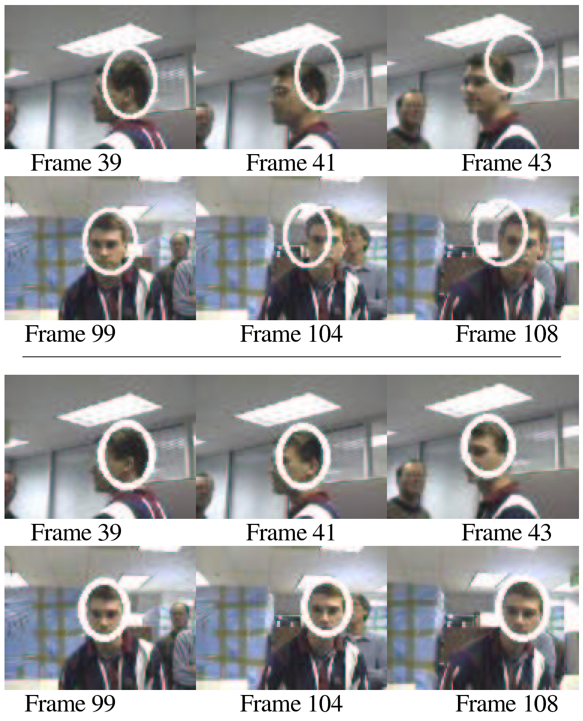

Figure 10. Sequence B: tracking results in cluttered environment with multiple people's presence: *The top two rows are the results of plain HMM without JPDAF. The bottom two rows are the results of the JPDAF-HMM.*

Figure 10 shows another comparison on Sequence B, a 200-frame sequence captured at 30 frames/sec. It is a very cluttered environment with multiple people's presence. There are both appearance and lighting changes of the person's head. The proposed JPDAF-HMM still successfully tracks the target through out the sequence while the plain HMM fails. To test JPDAF-HMM's ability to handle partial occlusion, we have conducted further experiments on Sequence C, which

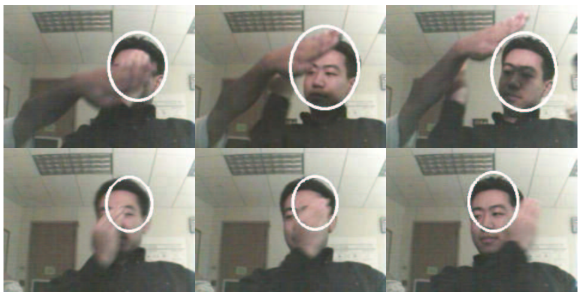

Figure 11. Tracking with partial occlusions

has 91 frames captured at 15 frames/sec (see Figure 11). Because of JPDAF-HMM's ability to globally model foreground/background's region properties, it successfully handles the partial occlusion. More tracking results are presented in the supplement material *1053.zip* submitted to the conference. To demonstrate the robustness of the proposed JPDAF-HMM approach, for all the testing sequences, we use the same algorithm configuration, i.e., $M = 30$, $N = 10$. We obtain promising tracking results in all the tested sequences.

6. Conclusion

In this paper, we have proposed a new HMM framework targeted at real-time contour-based object tracking. A joint probability data association filter (JPDAF) is used to compute the HMM's transition probability, taking into account the inter-correlated neighboring measurements. To ensure real-time performance, we have further developed an efficient method to calculate the data association probability via dynamic programming, which allows the proposed JPDAF-HMM to run comfortably at 30 frames/sec. This new tracking framework not only can easily incorporate various image cues (e.g., edge intensity, foreground color and background color), but also offers an on-line learning process to adapt to changes in the scene (see Section 4.1). We have reported promising tracking results on various real-world video sequences.

7. Acknowledgements

This work was supported in part by National Science Foundation Grant CDA 96-24396, EIA-99-75019 and in part by Gift Grant from Microsoft Research. Thanks Stan Birchfield for allowing us to use the test sequences.

References

[1] A. Amini, T. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 12(9):855–867, September 1990.

[2] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, pages 232–237, 1998.

[3] C. Chesnaud, P. Refregier, and V. Boulet. Statistical region snake-based segmentation adapted to different physical noise models. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 21(11):1145–1157, November 1999.

[4] A. Colmenzrez and T. Huang. Face detection with information-based maximum discrimination. In *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, pages 782–787, 1997.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, pages II 142–149, 2000.

[6] J. Denzler and H. Niemann. Active-rays: Polar-transformed active contours for real-time contour tracking. *Real-Time Imaging*, 5(3):203–13, June 1999.

[7] D. Geiger, A. Gupta, L. Costa, and J. Vlontzos. Dynamic-programming for detecting, tracking, and matching deformable contours. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 18(5):575, May 1996.

[8] R. Haralick. *Computer and Robot Vision*.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29(1):5–28, 1998.

[10] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. European Conf. on Computer Vision*, pages 767–781, 1998.

[11] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Computer Vision*, 1(4):321–331, 1988.

[12] S. Kuo and O. Agazzi. Keyword spotting in poorly printed documents using pseudo-2d hidden markov-models. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 16(8):842–848, August 1994.

[13] B. Li and R. Chellappa. Simultaneous tracking and verification via sequential posterior. In *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, pages 110–117, 2000.

[14] Q. Liu, Y. Rui, A. Gupta, and J. Cadiz. Automating camera management in a lecture room environment. In *ACM CHI*, 2000.

[15] A. Nefian and M. Hayes, III. Maximum likelihood training of the embedded hmm for face detection and recognition. In *Proc. IEEE Int'l Conf. on Image Processing*, 2000.

[16] N. Peterfreund. Robust tracking of position and velocity with kalman snakes. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 21(6):564–569, June 1999.

[17] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Mag.*, pages 4–15, January 1986.

[18] C. Rasmussen and G. Hager. Joint probabilistic techniques for tracking multi-part objects. In *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, pages 16–21, 1998.

[19] G. Rigoll, S. Muller, and B. Winterstein. Robust person tracking with non-stationary background using a combined pseudo-2d-hmm and kalman-filter approach. In *Proc. IEEE Int'l Conf. on Image Processing*, 1999.

[20] H. Tao, H. S. Sawhney, and R. Kumar. Dynamic layer representation and its applications to tracking. In *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, pages 134–141, June 2000.

[21] D. Terzopoulos and R. Szeliski. Tracking with kalman snakes. In *Active Vision*, pages 3–20, 1992.

[22] K. Toyama and G. Hager. Keeping your eye on the ball: Tracking occluding contours of unfamiliar objects without distraction. In *IEEE Inter. Conf. on Intelligent Robots and Systems*, volume 1, pages 354–359, 1995.

[23] J. Vermaak and A. Blake. Nonlinear filtering for speaker tracking in noisy and reverberant environments. In *Proc. IEEE Int'l Conf. Acoustic Speech Signal Processing*, 2000.

[24] Y. Wu and T. S. Huang. Color tracking by transductive learning. In *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, pages I 133–138, 2000.