Multicue HMM-UKF for Real-Time Contour Tracking

Yunqiang Chen [*], Yong Rui [†] and Thomas S. Huang [‡]

Abstract

We propose an HMM model for contour detection based on multiple visual cues in spatial domain and improve it by joint probabilistic matching to reduce background clutter. It is further integrated with unscented Kalman filter to exploit object dynamics in nonlinear systems for robust contour tracking.

Keywords: Parametric contour, HMM, unscented Kalman filters, joint probabilistic matching

1 Introduction

Reliable visual object tracking in complex environments is of great importance. Its applications include human computer interaction, teleconferencing, and visual surveillance, among many others. Unfortunately, after years of research, robust and efficient object tracking is still an open problem. Two of the major challenges are:

1. Various visual cues have been used to discriminate objects from background clutter (e.g., object contour [7] or color distribution [5]). However, none are robust individually. Multiple cues are combined recently (e.g. [3]). The major difficulty, however, is how to efficiently integrate multiple cues with spatial constraints and model the uncertainties of the detection in a principled way.

[*] Yunqiang Chen is with Siemens Corporate Research, Princeton, NJ 08540. yunqiang.chen@siemens.com
[†] Yong Rui is with Microsoft Research, Redmond, WA 98052. yongrui@microsoft.com
[‡] Thomas Huang is with University of Illinois at Urbana Champaign, Urbana, IL 61801. huang@ifp.uiuc.edu

2. Object dynamics is a useful constraint in temporal domain, but difficult to enforce when the states of the object are nonlinearly related to the observations. How to effectively integrate the dynamics with the uncertain contour detection result in a nonlinear dynamic system is critical for robust real-time object tracking.

The Hidden Markov Model (HMM) [11] provides a potential tool to solve the first difficulty. Traditional HMM usually works in 1D time domain. In this paper, we propose a new HMM in 2D spatial domain to detect object contour, where the contour is modeled by a parametric shape and the true contour points are searched along a set of normal lines along the shape. In the new HMM, multiple visual cues can be effectively integrated as well as various spatial constraints. Furthermore, the HMM provides probabilistic contour detection results with confidence measurement, based on which the more confidently detected contour segments contribute more and the cluttered contour segments contribute less in the final estimation of the object states. To reduce the background clutter, we further develop an efficient joint probabilistic matching (JPM) scheme to consider the contexture information when enforcing the contour smoothness. More accurate spatial constraints can be enforced by the proposed HMM.

To address the second difficulty, particle filter [7, 13] and extended Kalman filter (EKF) [2] have been proposed before. However, the particle filter requires random sampling in the state space and the computation grows exponentially with the state space dimension. The EKF is more efficient than the particle filter but only approximates the nonlinearity to the first order. The newly proposed unscented Kalman filter (UKF) can approximate the nonlinearity up to the second order without explicit calculation of the Jacobians. It provides a better alternative and is very efficient for real-time object tracking [8].

We then integrate the new HMM and the UKF seamlessly into a powerful parametric contour tracking framework in nonlinear dynamic systems. At each time frame t, the HMM provides probabilistic contour detection with confidence measurement, which are then passed to the UKF to estimate a more accurate object trajectory based on the nonlinear system dynamics. Furthermore, a robust online adaptation process is designed based on the probabilistic contour detection of the HMM to handle possible changes of object appearance or background for long-turn tracking in non-stationary environments.

The rest of the paper is organized as follows. In Section 2, we present our HMM for contour detection. In Section 3, a more accurate contour smoothness term is designed based on JPM in cluttered environments. In Section 4, parametric contour tracking is achieved by combining the UKF with the HMM. In Section 5, adaptive learning is further developed to handle appearance changes or dynamic environments. Experiments and concluding remarks are given in Section 6 and 7 respectively.

2 Contour Tracking Using HMM

Active contour methods (e.g. [1, 6]) have been proposed to track nonrigid objects by efficient dynamic programming. However, they provide only one 'optimal' solution with no alternatives and is not ideal for further refinement by object dynamics. HMM, on the other hand, can utilize the forward-backward algorithm ([11]) to provide probabilistic detection for each contour points, which allows further integration with UKF (in Section 4) and the online adaptation scheme (in Section 5) for better object tracking.

To reduce the 2D contour detection into a 1D HMM, we represent the contour by a parametric shape and restrict the contour searching to a set of normal lines along the predicted contour position as shown in Figure 1 (a). Let $\phi \in [1, M]$ be the index of the normal lines and $\lambda \in [-N, N]$ be the index of pixels along a normal line. $\rho_\phi(\lambda)$ denotes the color or intensity of pixel λ on ϕth normal line. Each normal line has $2N+1$ pixels. The center of each normal line is placed at the predicted contour position and indexed as 0. The 'true' contour can be detected if we can find all the contour points $s(\phi)$ based on various visual cues and spatial constraints. Note that instead of representing the contour by a set of points in 2D image, we now represent the contour by a 1D vector $s(\phi), \phi = 1, ..., M$.

Based on this 1D representation, HMM provides an efficient and principled way to find the contour based on various cues (e.g., edge and color) and prior constraints (e.g., contour smoothness constraint). The hidden states of the HMM are the location of the true contour points 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 the normal lines. An HMM is specified by the possible states (in our case, $[-N, N]$), the observation model $P(O_\phi|s_\phi)$, and the transition probabilities $p(s_\phi|s_{\phi-1})$ as illustrated in Figure 1 (b). We describe

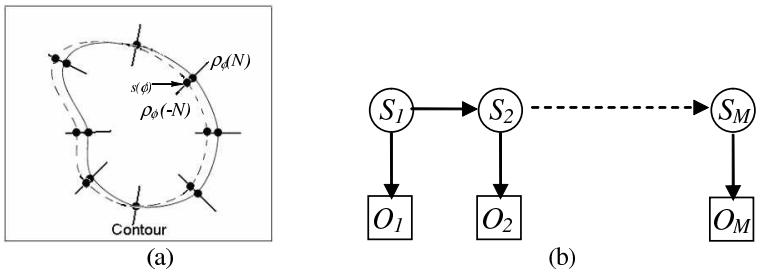

Figure 1. The contour model: *(a) The contour in 2D image: The solid curve is the predicted contour. The dashed curve is the true contour. We want to find the $s(\phi)$ which is the index of the true contour point on the ϕth normal line $\phi \in [1, M]$; (b) the Markovian assumption in our contour model. Note that the indices in not in time domain but indicate different normal lines on current frame.*

the multicue observation model and a simplified state transition model enforcing the simple contour smoothness in this Section. A more accurate smoothness constraint based on contexture information for better contour detection is deferred to Section 3. Unlike other contour models (e.g. [1, 6]) that resort to the dynamic programming to obtain the global optimal solution, the HMM offers a probabilistic contour detection scheme based on forward-backward algorithm.

The observation O_ϕ on line ϕ can include various cues, e.g. edge detection (i.e., $\mathbf{z}_\phi$) and pixel color $\rho_\phi(\lambda), \lambda \in [-N, N]$). The likelihood model of the edge detection ($\mathbf{z}_\phi$) can be derived similar to [7]. Because of noise and background clutter, there can be multiple edges along each normal line (i.e. $\mathbf{z}_\phi = (z_1, z_2, ..., z_J)$). Each z_j has equal probability of being the true contour. There is also probability q that the true contour is not detected by edge detector. With the assumption that the clutter is a Poisson process with spatial density γ and the edge localization error is Gaussian distributed with standard deviation σ_z, we obtain the edge likelihood model as follows (please refer to [7] for detailed derivation):

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

Color is a complementary cue to further reduce the background clutter and can be easily integrated into the observation likelihood. Let $p(v|FG)$ and $p(v|BG)$ represent the color histograms of the foreground (FG) and background (BG), respectively. If s_ϕ is the contour point on line ϕ, the segment $[-N, s_\phi]$

belongs to the foreground and the segment $[s_\phi + 1, N]$ belongs to the background. Combining the edge likelihood model and the color (i.e. v) of the foreground and background, we can have the following multicue observation likelihood model:

$$P(O_\phi|s_\phi) = p(\mathbf{z}_\phi|s_\phi) \cdot \prod_{i=-N}^{s_\phi} P(v = \rho_\phi(i)|FG) \cdot \prod_{i=s_\phi+1}^{N} P(v = \rho_\phi(i)|BG) \tag{2}$$

If more cues are available, they can be integrated similarly. As shown in Section 5, the HMM also allows us to update the color model (i.e. $p(v|FG)$ and $p(v|BG)$) in a probabilistic way to accommodate the lighting or object appearance changes and hence more robust in dynamic environments.

To complete the HMM model, the transition probability is discussed here to enforce the spatial constraint of the contour (e.g., contour smoothness). We defer a more complete smoothness constraint based on contextual information to Section 3. To exploit the contour smoothness in the HMM, it needs to be in a causal form. In Figure 1 (a), we can see that when the normal lines are dense (30 lines in our experiments) and the contour is smooth, the true contour points on adjacent normal lines tend to have similar amounts of displacement from the predicted contour position (center of each normal line). This correlation is causal and can be captured by transition probabilities $p(s_\phi|s_{\phi-1})$ defined as follows:

$$p(s_\phi|s_{\phi-1}) = c \cdot e^{-(s_\phi - s_{\phi-1})^2/\sigma_s^2} \tag{3}$$

where c is a normalization constant and σ_s regulates the strength of the smoothness constraint. This transition probability penalizes discontinuity of the contour and results in a smoother contour.

Given the observation $\mathbf{O} = \{O_\phi, \phi \in [1, M]\}$, the multicue based likelihood model and transition probabilities, we can obtain probabilistic contour detection results based on the efficient forward-backward algorithm. Let $\alpha_\phi(s) = p(O_1, ..., O_\phi, s_\phi = s)$ and $\beta_\phi(s) = p(O_{\phi+1}, ..., O_M, s_\phi = s)$ (please refer to [11] for details), we have:

$$P(s_\phi = s|\mathbf{O}) = \frac{\alpha_\phi(s)\beta_\phi(s)}{\sum_{u=-N}^{N} \alpha_\phi(u)\beta_\phi(u)}, \quad s \in [-N, N] \tag{4}$$

Instead of returning one single solution as traditional active contour methods, HMM provides probabilistic contour detection results, based on which we can estimate the confidence of the detected contour points on each normal line. This information can be effectively integrated with the UKF in Section 4, where the confidently detected contour points contribute more to the object state estimation. It is worth noting that the HMM cannot enforce the smoothness between the beginning and ending points of a closed contour. For strict smoothness constraint on closed contour, loopy belief propagation [9] can be used instead of forward-backward algorithm.

3 Improving Transition Probability

The transition probability in Eq. (3) only considers the contour points themselves with no contexture information. It is especially dangerous when strong background clutter is close to the tracked contour, e.g. the dark rectangle in Fig 2 (a). A more robust contour smoothness constraint should consider all pixels on the neighboring normal lines jointly similar to the joint data association technique in [12]. We propose joint probabilistic matching (JPM) of the neighboring normal lines for more accurate smoothness constraint and design an efficient dynamic programming technique to calculate the matching.

Since all the pixels along the normal lines could be the true contour (edge detector might miss the true contour), we have to track the transition between all the pixels. Let s_ϕ and $s_{\phi+1}$ be the true contour points on normal line ϕ and $\phi+1$, respectively. These two contour points segment the two normal lines into foreground (FG) and background (BG) segments. If the object is opaque, the edges on FG cannot transit to BG, which means the edges on FG/BG on line ϕ should match to the FG/BG on line $\phi+1$ respectively. A more accurate transition probability can be designed based on this constraint.

Let $E^F(i,j)$ and $E^B(i,j)$ be the matching errors between edge points on the neighboring foreground segments (i.e., segment $[-N,i]$ on line ϕ and $[-N,j]$ on line $\phi+1$) and background segments (i.e., segment $[i+1,N]$ on line ϕ and $[j+1,N]$ on line $\phi+1$), respectively. Let $\delta(i)=j$ be a mapping of

pixel i on line ϕ to pixel j on line $\phi + 1$. Then, the matching costs are defined as follows:

$$\begin{aligned} E^F(i,j) &= min_\delta \sum_{k\in[-N,i]} ||\rho_\phi(k) - \rho_{\phi+1}(\delta(k))||^2, \quad \delta(k) \in [-N, j] \\ E^B(i,j) &= min_\delta \sum_{k\in[i+1,N]} ||\rho_\phi(k) - \rho_{\phi+1}(\delta(k))||^2, \quad \delta(k) \in [j+1, N] \end{aligned} \tag{5}$$

A more accurate transition probability can then be defined as follows (compare with Eq. (3)):

$$-log(p(s_2|s_1)) = E^F(s_1, s_2) + E^B(s_1, s_2) + (s_2 - s_1)^2/\sigma_s^2 \tag{6}$$

The effect of this new transition probability can be illustrated by a synthesized image in Figure 2 (a). The dark rectangle on the background is very close to the tracked object (gray region). Along the two normal lines shown in Figure 2 (a), two edges can be detected, i.e., 'a' and 'b' on line 1 and 'c' and 'd' on line 2, which is distracting to the contour detection. The measurements on these two lines are shown in Figure 2 (b). Based on the simple contour smoothness in Eq. (3), the transition from 'a' to 'c' and from 'b' to 'c' have almost the same transition probabilities because $|a - c| \approx |b - c|$. However, if we consider all the edges together, we can see that 'ac' is not a good contour candidate because 'b' and 'd' are now on foreground and background respectively and they have no matching edges on the neighboring lines. The contour candidates 'ad' and 'bc' are better contour candidates. (The best choice between these two should be further decided based on all the normal lines.)

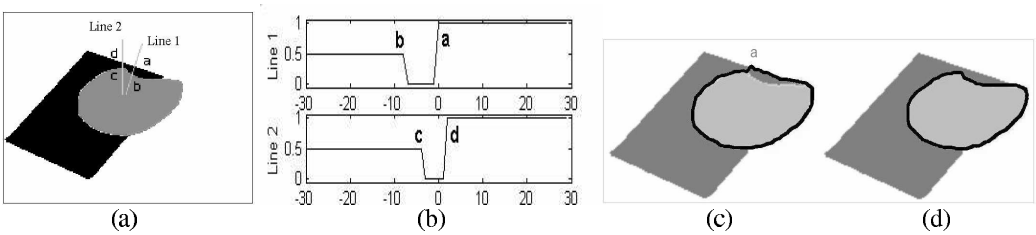

Figure 2. JPM based smoothness constraint: *(a) Synthesized image. (b) The intensity along line 1 and 2. 'ac' is a bad contour candidate and separates the two lines into unmatched FG and BG. (c) Without JPM, the contour is distracted by background edges. (d) With JPM, the contour is correct.*

The comparison between JPM based contour smoothness and the simple contour smoothness is shown in Figure 2 (c) and (d). Without JPM, the contour is distracted by the strong edges from the dark rectangle in Figure 2 (c). In Figure 2 (d), correct contour is obtained by utilizing the JPM term, which has large penalty for the contour to jump to background clutter and then jump back.

To use the JPM, we need to calculate $(2N+1)^2$ transition probabilities for one pair of neighboring normal lines and for each possible transition we should consider the deformable matching between two normal lines. The computation is very expensive. We design a dynamic programming method similar to the forward-backward algorithm by assuming the edges do not cross each other. The new method only costs $2 \cdot (2N+1)^2$ to calculate the transition probabilities for all $(2N+1)^2$ possible transitions. Given observation on lines 1 and 2, the joint matching can be calculated in the following recursive manner:

$$E^F(i,j) = \min(E^F(i-1,j)+d, E^F(i,j-1)+d, E^F(i-1,j-1)) + e(i,j) \tag{7}$$

where d is the penalty of shifting one pixel in the matching and $e(.,.)$ is the matching cost between two pixels. $E^F(i,j)$ is the minimal matching cost between segment $[-N,i]$ on line 1 and segment $[-N,j]$ on line 2. We start from $E^F(-N,j) = E^F(i,-N) = 0$, where $i,j \in [-N,N]$ and use the above recursion to obtain the matching cost $E^F(i,j)$ from $i=-N$ to N and $j=-N$ to N. A similar process is used to calculate $E^B(i,j)$. After obtaining all matching costs, the more comprehensive state transition probability can be easily computed as in Eq. (6) and plugged into the HMM in Section 2.

4 Parametric Contour Tracking by UKF

Previously, we have described the JPM-HMM model for detecting the contour using multiple cues and spatial constraints. In the temporal domain, there is also object dynamics, based on which we can combine the contour detection results on consecutive frames to achieve more robust tracking. Kalman filter is a typical method for linear systems [2]. But in visual tracking systems, the object states (e.g., head position and orientation) usually cannot be measured directly but are nonlinearly related to the measurements (e.g., contour points of the object). This problem can be formulated as follows.

Let $X_{0:t}$ and $Y_{0:t}$ represent the state trajectory and measurement history of a system from time 0 to t. With the assumption that the state sequence is a Markov chain and the measurements are independent given the states, tracking can be achieved by obtaining $p(X_t|X_{t-1}, Y_t)$. In our case, X_t is the parametric shape of the head, and Y_t is the detected contour on all normal lines. The two key components in the tracking system are object dynamics (i.e. $X_t = f(X_{t-1}, m_t)$) and observation model (i.e. $Y_t = h(X_t, n_t)$). The terms m_t and n_t are process noise and measurement noise, respectively. With the Gaussian distribution assumption, it has an analytical solution which is the well-known Kalman filter [2]. Unfortunately, the linearity is usually not satisfied in the real world.

For tracking human's head (see Section 6), we define the object state, system dynamics, and measurements in this section. We model human's head by an ellipse, $\Theta = [c_x, c_y, \alpha, \beta, \Phi]^T$, where (c_x, c_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. The object states at frame t include the shape parameters and the velocity: $X_t = [\Theta_t \quad \dot{\Theta}_t]^T$. We adopt the Langevin process [14] to model the head dynamics:

$$\bar{X}_{t|t-1} = f(X_{t-1}, m_t) = \begin{bmatrix} 1 & \tau \\ 0 & a \end{bmatrix} \begin{bmatrix} \Theta_{t-1} \\ \dot{\Theta}_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} m_t \tag{8}$$

where a and b are constants and m is Gaussian noise $N(0, Q)$. τ is the discretization time step. The object states cannot be measured directly but are nonlinearly related to the observed object contour. As shown in Figure 1, we restrict the contour detection along the normal lines of the predicted object position. Based on the probabilistic detection result in Eq. (4), the contour position on each normal line can be estimated by $s^*_\phi = \sum_{-N}^{N} s * P(s_\phi = s|O)$. Then, we have $Y_t = \mathbf{s}^* = [s^*_1, s^*_2, ..., s^*_M]^T$ as the measurement of the contour position. We can also estimate the uncertainty of the measurements based on the probabilistic detection results of the HMM so that the more confidently detected contour points contribute more to object state trajectory estimation in the UKF. Since the UKF assumes Gaussian distributed noise (i.e. n_t), the contour detection confidence is modeled by measurement variance $\sigma^2_\phi = \sum_{-N}^{N} s^2 * P(s_\phi = s|O)$.

The s_ϕ is the intersection of the ellipse $\Theta_t = [c_x, c_y, \alpha, \beta, \Phi]$ (i.e. the object states) and the normal

line ϕ (represented by its center point $[x_\phi, y_\phi]$ and its angle θ_ϕ). We first perform a coordinate translation and rotation to make the ellipse upright, centered at the origin. In the new coordinate, the normal line is represented by the new center ($[x'_\phi, y'_\phi]^T$) and the angle θ'_ϕ. s_ϕ can be calculated as follows:

$$\hat{s}_\phi = h(X_t, n_t) = \frac{\sqrt{\left[\frac{x'_\phi \cos\theta'_\phi}{\alpha^2} + \frac{y'_\phi \sin\theta'_\phi}{\beta^2}\right]^2 - \left[\frac{\cos^2\theta'_\phi}{\alpha^2} + \frac{\sin^2\theta'_\phi}{\beta^2}\right]\left[\frac{x'^2_\phi}{\alpha^2} + \frac{y'^2_\phi}{\beta^2} - 1\right]} - \left[\frac{x'_\phi \cos\theta'_\phi}{\alpha^2} + \frac{y'_\phi \sin\theta'_\phi}{\beta^2}\right]}{\left[\frac{\cos^2\theta'_\phi}{\alpha^2} + \frac{\sin^2\theta'_\phi}{\beta^2}\right]} + n_t \tag{9}$$

From the above equation, we can see that the relations between the measurements and the object states are highly nonlinear. The Unscented Kalman filter (UKF) provides a superior alternative [8] than EKF. Rather than linearizing the system, the UKF generates sigma points (i.e. hypotheses) in the object state space and applies the system dynamics and observation models to these sigma points. Then it uses the weighted sigma points to estimate the posterior mean and covariance of the predicted measurements. Compared with the EKF, the UKF does not need to explicitly calculate the Jacobians. It therefore not only outperforms the EKF in accuracy (second-order vs. first-order approximation), but also is computationally efficient. The UKF is implemented using the Unscented Transformation by expanding the state space to include the noise component: $x_t^a = [x_t^T m_t^T n_t^T]^T$. Let $N_a = N_x + N_m + N_n$ be the dimension of the expanded state space, where N_m and N_n are the dimensions of noise m_t and n_t, and let Q and R be the covariance of noise m_t and n_t. The UKF can be summarized as follows (refer to [8]):

1. Initialization: $\bar{x}_0^a = [\ \bar{X}_0^T \quad 0 \quad 0\ ]^T$ and $P_0^a = [\ P_0 \quad Q \quad R\ ]$
2. Iterate for each time instance t:
   (a) Generate $2N_a + 1$ scaled symmetric sigma points based on the state mean and variance;
   (b) Pass all the generated sigma points through the system dynamics in Eq. (8) and observation model in Eq. (9). Compute the mean and covariance of the predicted object states $\bar{X}_{t|t-1}$ and predicted contour position $\bar{Y}_{t|t-1}$ using the weighted sigma points;
   (c) Calculate the innovation $Y_t - \bar{Y}_{t|t-1}$ based on the measurement $Y_t = [s_1^*, s_2^*, ..., s_M^*]^T$ and the variance of the measurement on current frame (obtained by the JPM-HMM in Sections 2 and

3). Update the mean and covariance of the states (i.e. $P_{Y_tY_t}$, $P_{X_tY_t}$) and the Kalman gain $K_t = P_{X_tY_t}P_{Y_tY_t}^{-1}$ based on the sigma points and get the final estimation;

With respect to the computational cost, UKF is better than the EKF in our case. Remember that we use ellipse to model the object and Langevin process to model the dynamics and use 30 normal lines to detect the contour (i.e. $N_x = 2 \times 5 = 10$, $N_m = 5$ and $N_n = M = 30$). To calculate Jacobian matrix for EKF, we need to compute $\partial \hat{s}_\phi / \partial x_i$ for $\phi \in [1, M]$ and $i = 1, ..., N_x$ (i.e. $N_x \times M = 10 * 30$ terms). For UKF, it is very straight forward and only need to propagate $2 \times (10 + 5 + 30) + 1 = 91$ sigma points through the Eq (8) and Eqs (9), which is much easier than calculating $\partial \hat{s}_\phi / \partial x_i$. The UKF is also much more efficient than the particle filter because the number of the sigma points increases linearly with the number of dimension.

5 Online Adaptation

In a dynamic environment, both the objects and background may gradually change their appearances. An online adaptation is therefore necessary to update the observation likelihood models dynamically. However, adaptation might be adversely affected by tracking inaccuracy or partial occlusion. Fortunately, the probabilistic detection results from HMM (i.e. $P(s_\phi = s|\mathbf{O})$) in Eq. (4) allows us to update the observation models based on the tracking confidence on each normal line.

Based on the probabilistic detection results, the probability of pixel λ_ϕ belonging to the foreground object can be computed by integration: $P(\lambda_\phi \in FG) = \sum_{s=\lambda_\phi}^{s=N} p(s_\phi = s|\mathbf{O})$ (i.e. sum of all the probabilities that the true contour point on line ϕ is outside of pixel λ_ϕ). Based on this, we can weight the importance of each pixel when updating the color histograms of the foreground and background. In this way, the more confidently classified pixels will contribute more to the color model.

The adaptation makes the tracking system much more robust to the error caused by rough initialization or the changing lighting or appearance. Since the color close to the contour is more important than the color in the middle in helping us locate the contour, we use the pixels on the normal lines to update the color model. It is then plugged back into Eq. (2) during the contour searching on the next frame.

6 Experiments

To validate the robustness of the proposed framework, we apply it in various real-world video sequences captured by a pan/tilt/zoom video camera in a typical office environment. The sequences simulate various tracking conditions, including appearance changes, quick movement, out-of-plane rotation, shape deformation, background clutter, camera pan/tilt/zoom, and partial occlusion. In the experiments, we use the parametric ellipse $\Theta_t = [c_x, c_y, \alpha, \beta, \Phi]$ to model human heads and we adopt the Langevin process as the object dynamics, as discussed in Section 4. For all the testing sequences, we use the same algorithm configuration, e.g., object dynamics, state transition probability calculation. For each ellipse, 30 normal lines are used, i.e., $M = 30$. Each line has 21 observation locations, i.e., $N = 10$. The tracking algorithm is implemented in C++ on Windows XP platform. No attempt is made on code optimization and the current system can run at 60 frames/sec comfortably on a standard Dell PIII 1G machine. It is worth noting that we derive the HMM-UKF tracking framework based on ellipse. But it can be extended to other parametric contour if a corresponding observation model can be derived. Three set of comparisons are conducted to demonstrate the strength of the new method.

6.1 Multiple cues vs. single cue

First, we compare our tracker with the CAMSHIFT algorithm [4], which tracks ellipsoid objects based on color histogram. In both algorithms, H and S channels in the HSV color space are used and each color has 32 bins. Two sequences (i.e. Seq A and Seq B, captured at 30 frames/sec) are tested. While CAMSHIFT is one of the best single cue trackers, it loses track when the user rotates her head or when other objects of similar color (e.g. the door, the cardboard box or other faces) are close to the tracked face. The result of CAMSHIFT tracker is shown in the top row compared with our method on the bottom row in Fig. 3.

The CAMSHIFT tracker is quite robust when the object color remains the same and there is no similar color in the background. But it is easily distracted when patches of similar color appear in the background near the target because it ignores contour information and object dynamics. Also, there is

Figure 3. CAMSHIFT vs. JPM HMM

Figure 4. Plain HMM (top) vs. JPM-HMM (bottom)

no principled way to update the color histogram of the object. When the person turns her head and the face is not visible (Seq A, Frame 187), CAMSHIFT tracker loses track.

On the other hand, the proposed HMM probabilistically integrates multiple cues into the HMM model and successfully tracks the user through the sequence. It is worth noting that some other algorithms can also track the object in the sequence by fusing color and edge together, but require potentially more difficult initialization (e.g. a side view of the head including both skin and hair is needed to train a color model in [3]). In our algorithm, both foreground and background color histograms are probabilistically updated during tracking by the HMM. It only requires a rough initialization (e.g. an automatic face detector) and can effectively handles initialization error or object appearance change due to head rotation.

6.2 JPM-HMM vs. plain HMM

As we have described in Section 3, the traditional contour smoothness constraint only takes into account the contour points but ignores all other edge points. This can be dangerous in a cluttered environment. To show the strength of this JPM energy term, we compare the JPM-HMM (using the improved transition probability proposed in Eq. (6)) against the plain HMM model (using the traditional contour smoothness constraint in Eq. (3)).

The comparisons on sequence A and B are shown in Figure 4, where the top row is the result from the plain HMM and the bottom row is the result from the JPM-HMM. In the plain HMM, tracking is gradually distracted by the strong edges on the background clutter nearby. Because JPM tracks the contour points more accurately when estimating the contour smoothness by considering all edges jointly,

the JPM-HMM is much more robust against sharp edges on the background.

6.3 UKF vs. Kalman Filter

In this subsection, we evaluate HMM-UKF's ability to handle large distractions, e.g., partial occlusions, on Sequence C, which has 225 frames captured at 30 frames/second (see Figure 5). Note that the observation model in Eq. (9) of the tracking system is highly nonlinear. It is therefore quite difficult to calculate the Jacobians as required by the EKF. An ad hoc method to get around this situation is not to model the nonlinearity explicitly. For example, one can first obtain the best contour positions and then use least-mean-square (LMS) to fit a parametric ellipse followed by a Kalman filter. Although simple to implement, it does not take full advantage of the object dynamics. The UKF, on the other hand, not only exploits system dynamics in Eq. (8) and observation model in Eq. (9), but also avoids computing the complicated Jacobians. The comparison between the UKF and Kalman filter is shown in Figure 5.

As we can see in Figure 5, the person's head is occluded by his hand. Neither the color nor the edge detection can reliably obtain the true contour points. The LMS fitting plus Kalman filter fails at frame 150 when large distraction occurs, because it cannot integrate the unevenly distributed detection confidence (uncertain at the occluded part but more reliable on other parts of the contour) with the object dynamics. The UKF, on the other hand, successfully handles the partial occlusion.

Figure 5. Kalman filter (top row) vs. the UKF (bottom row)

The videos of above comparisons and some other tracking results with manual or automatic initialization can be found at http://www.ifp.uiuc.edu/ chenyq/pami/Tracking.htm.

7 Conclusion

In this paper, we propose a new HMM-UKF framework for real-time multicue object tracking. Several important features distinguish the proposed HMM-UKF from other approaches: (1) Integrating multiple visual cues in a principled probabilistic way; (2) Better contour smoothness constraint based on joint probabilistic matching; (3) Exploiting object dynamics in nonlinear systems; (4) Online adaptation allowing rough initialization and robustness to lighting/appearance changes. As further improvement, we are investigating multiple hypothesis tracking based on the proposed algorithm. Also, verification step based on the prior knowledge of the object is necessary to automatically evaluate the tracking performance.

References

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

[2] B. Anderson and J. Moore. *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[3] 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.

[4] G. R. Bradski. Computer video face tracking for use in a perceptual user interface. *Intel Technology Journal*, Q2, 1998.

[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] D. Geiger, A. Gupta, L. Costa, and J. Vlontzos. Dynamic-programming for detecting, tracking, and matching deformable contours. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(3):294–302, 1995.

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

[8] S. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceeding of the IEEE*, 92(3):401–422, 2004.

[9] M. K., W. Y., and J. M. Loopy belief propagation for approximate inference: an empirical study. In *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence*, 1999.

[10] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Int. Conf. on Machine Learning*, 2001.

[11] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE Trans. Acoust., Speech, Signal Processing*, 3(1):4–15, January 1986.

[12] 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.

[13] J. Sullivan, A. Blake, M. Isard, and J. MacCormick. Bayesian object localisation in images. *Int. J. Computer Vision*, 44(2):111–135, 2001.

[14] 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*, pages V:3021–3024, 2001.