Tracking of Multiple Faces for Human-Computer Interfaces and Virtual Environments

Fu Jie Huang and Tsuhan Chen

Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{jhuangfu,tsuhan}@cmu.edu

ABSTRACT

In this paper, we describe a real-time face-tracking algorithm. We start with single face tracking based on statistical color modeling and the deformable template. We then expand the algorithm to track multiple faces, possibly with occlusion, by constraining the speed and size changes of the faces. We test the algorithm on sequences with different occlusion patterns, and analyze the tracking performance. We also present a tracking software library based on this algorithm. This library can be applied to human-computer interfaces, lipreading, and virtual environments.

1. INTRODUCTION

Image analysis for human faces has been an active research topic in image processing, computer vision and psychology. Specific research areas include face detection, face tracking, face recognition, face animation, etc. In this paper, we focus on face tracking that can serve as a front end to other facial image analysis tasks, such as face recognition, face expression analysis, gaze tracking and lip-reading.

Face tracking is different from face detection in that face tracking uses temporal correlation to locate human faces in a video sequence, instead of detecting them in each frame independently. With temporal information, we can narrow down the search range significantly and thus make real-time tracking possible.

Recently there have been a lot of research efforts in face tracking. Yang and Waibel [1] built a real-time face tracking system based on normalized color. Bradski [2] proposed the continuously adaptive mean shift algorithm. Colmenarez, et al. [3], DeCarlo and Metaxas [4] used a 3D face model in the tracking process. Malsburg [5] tracked specific feature points on the face to track the face. However, none of these algorithms deal with multiple faces, especially occlusion between faces, effectively. In this paper, we propose a multiple face tracking algorithm based on constraining the speed and size changes of the faces.

The outline of this paper is as follows. In section 2, we first review a single-face tracking algorithm [6] based on statistical color modeling and the deformable template, and in section 3 we extend it to a multiple face tracking algorithm based on certain constraints. In section 4, we will evaluate this approach by testing it on several video sequences, both synthetic and live video. We propose some extensions for the tracking algorithm and show its applications in section 5.

Work supported in part by Department of Defense and Intel Corporation.

2. SINGLE FACE TRACKING

For single face tracking, we use the method reported in [6], which is based on statistical color modeling and the deformable template. First, for each pixel in the current video frame, we calculate the probabilities of each pixel belonging to each of the two classes: the “face” class (the foreground class) and the “non-face” class (the background class). Then we use a deformable template to group the pixels more likely to belong to the face class. We deform the template so that the it includes as many face pixels as possible and at the same time includes as few background pixels as possible. The optimal deformation can be found by using logarithmic search.

For any given pixel, let ω_1 denote the hypothesis that the pixel belongs to the foreground, and ω_2 denote the hypothesis that the pixel belongs to the background. In order to derive $p(\omega_1 | x)$, the probability of a pixel with value x, a 3-dimensional vector representing (R, G, B) or (Y, U, V), belonging to the foreground, and $p(\omega_2 | x)$, the probability of the pixel belonging to the background, we first estimate the foreground color distribution $p(x | \omega_1)$ and the background color distribution $p(x | \omega_2)$. Once these are found, $p(\omega_1 | x)$ and $p(\omega_2 | x)$ can be derived from $p(x | \omega_1)$ and $p(x | \omega_2)$ using the Bayesian rule, $p(\omega_1 | x) = p(x | \omega_1) p(\omega_1) / p(x)$ and $p(\omega_2 | x) = p(x | \omega_2) p(\omega_2) / p(x)$.

We use a Gaussian mixture model to approximate the color distributions of the two classes. Specifically, we use a weighted sum of two Gaussian functions to model the face region since the skin color and the facial hair color (eyes, hair, beard) are the two dominant colors on a human face. For the background region, we also assume that there are two dominant colors. Thus, we can approximate the color distribution of each region as:

$$p(x | \omega_i) = \alpha_{i1} N(\mu_{i1}, C_{i1}) + \alpha_{i2} N(\mu_{i2}, C_{i2})$$

where $N(\mu, C)$ represents a Gaussian function, μ and C are the mean and covariance of the Gaussian function and α_{ij} is the weight of the Gaussian function in the mixture.

To track the face, we deform an elliptical template so that an energy function of the region $\Re$ inside the template, defined as

$$f(\Re) = \sum_{r \in \Re} \log\left(\frac{p(x_r \mid \omega_2)}{p(x_r \mid \omega_1)} \right)$$

is minimized, where r is a pixel in the region $\Re$ and x_r is the value of this pixel. Hence, the tracking problem becomes one that tries to minimize the energy function $f(\cdot)$ by adjusting the template.

3. MULTIPLE FACE TRACKING

In many applications, such as human-computer interfaces and video surveillance, we need to track multiple faces simultaneously. Tracking of multiple faces is different from single face tracking in that these faces may occlude each other.

Without loss of generality, we consider the occlusion between two faces. In the simple case that the faces in the image frame move in their own trajectories and do not occlude each other, we can track these faces separately with multiple color modeling and multiple deformable templates. Specifically, while training the color distribution for one specific face class, we treat other faces as part of the background class.

When there is occlusion, the difficulty of multiple face tracking depends on several factors, such as how similar these faces are, how long the occlusion lasts, and at what percentage one face is occluded by another.

If the skin colors of these two faces are identical, it is very difficult to track these faces correctly based on the color information. Figure 1 shows an example using synthetic video. When face A is partially occluded by another face B, the deformable template for face A will be expanded so that it includes the pixels in face B based on the optimization, as shown in Figure 1. The white ellipses show the tracking results for the two faces. The one that was tracking the face in the back is incorrectly expanded to cover both faces.

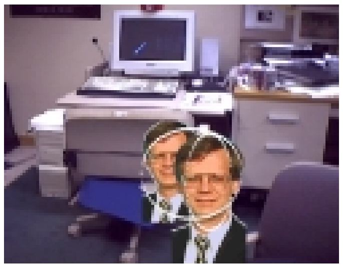

Figure 1 Occlusion between two faces with similar color distribution

If the color distribution of face A is different from B, then the template tracking face A will shrink to include the partial region not occluded by B. If the occlusion lasts for a considerable amount of time, then the template may not be able to follow the face anymore. If we consider cases when the occlusion is "transient," one way to deal with occlusion is to make use of the motion information of the face. In computer vision literature, Kalman filtering and its derivatives [7] are often used for tracking objects with noisy observation data, including occlusion [8]. Computation of Kalman filtering however is rather complicated.

Here we apply constraints on the speed and size changes of the face. We assume that during the occlusion, the speed and the size of the face being occluded does not change significantly, which can be formulated as:

$$|p' - p''| - \Delta_1 < |p - p'| < |p' - p''| + \Delta_1$$

where p is face position in the current frame, p' is the position in the previous frame, and p'' is in the frame before the previous frame. Δ_1 is a pre-fixed threshold, which decides the allowed speed change between two image frames.

The constraint for the shape change can be described as:

$$|w - w'| < \Delta_2$$

where w and w' are the width of the face in the current and the previous frame, and Δ_2 is the pre-fixed threshold. Δ_1 and Δ_2 are decided by experiments.

4. TRACKING RESULTS

To evaluate the tracking performance, we have created synthetic video sequences, so that ground truth data is available, with different occlusion patterns. Suppose there are two faces of different sizes (the larger one is of the person closer to the video camera), we let the smaller face move and be occluded by the larger face that remains still. We created these test sequences with several occlusion patterns as follows.

- Moving path: We simplify the moving paths into three cases as shown in Figure 2, where the arrows show possible moving paths:
    - The moving face keeps moving in the same direction.
    - The moving face changes its path during occlusion and then moves upwards.
    - The moving face changes its path during occlusion and then moves downwards.
- Face size change: We create sequences of the following three cases:
    - The moving face maintains a constant size.
    - The moving face's size decreases during occlusion.
    - The moving face's size increases during occlusion.

In our experiments, tracking of the larger face always succeeded. Therefore, here we focus on tracking results of the moving / occluded face, which is the smaller face.

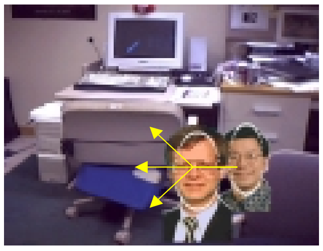

Figure 2 An example image from the test sequences.

We tested the algorithm with and without constraints on five different occlusion patterns as in Table 1. We calculate the distances between the face center positions from the ground truth data and the tracking results, and define the distance as the tracking error. Figure 3 shows an example tracking result. The occlusion in this example happens from frame 3 to frame 26, and peaks at frame 16 when the larger face occludes 90% region of the smaller face. The dotted curve shows the tracking error of the constraint-based algorithm, while the solid curve shows the tracking result of the algorithm without the constraints. We can see that the tracking error of the constraint-based algorithm goes back to about 10 pixels after the occlusion, which means the tracking is correct, while the tracking error of the algorithm without constraints keeps increasing, which means the template has lost the face.

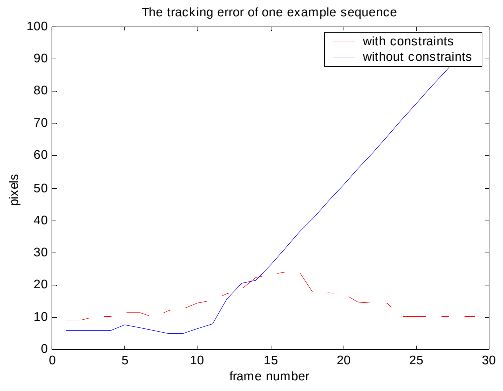

Figure 3 Tracking error of one example sequence

We tested the algorithm with and without constraints on sequences with different occlusion percentages for each occlusion pattern, and found the largest occlusion percentage with which the algorithms could still track the face correctly. The results are summarized in Table 1.

Table 1 Performance comparison

Occlusion pattern	With constraints	Without constraints
Same direction Constant size	88%	80%
Same direction Increase size	90%	90%
Same direction Decrease size	89%	78%
Move upwards Constant size	88%	80%
Move downwards Constant size	89%	87%

From Table 1, we can see that the algorithm with constraints on the speed change and the size change can handle a larger percentage of occlusion, therefore is more robust than the algorithm without constraints. In three out of the five occlusion patterns, the algorithm with constraints can handle around 10% larger occlusion than the algorithm without constraints. One exception is when the face moves in the same direction as its size increases, the algorithm without constraints performs as well as the algorithm with constraints. We conjecture that this is because when the face size increases, tracking becomes easier and the algorithm without constraints is already good enough.

In addition to synthetic sequences, we also tested the algorithm on live video. The result of one sequence is shown in Figure 4. We can see the system tracks the smaller face correctly throughout the occlusion.

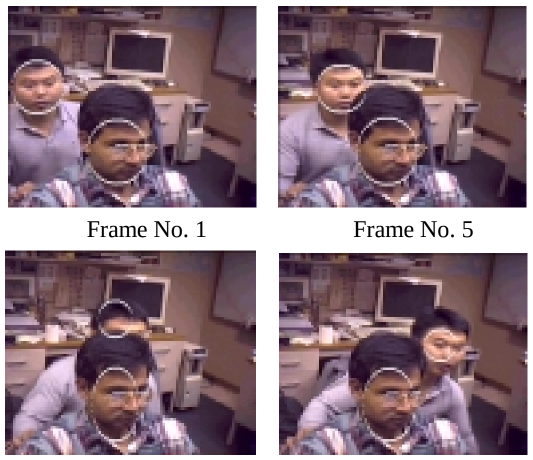

Figure 4 Example result with live video

5. APPLICATIONS AND EXTENSIONS

We have built a tracking software library [‡] for various applications. In one application, we integrate face tracking with a face detection module from Henry Rowley et al. [9] to initialize the system. The face detection module detects all the frontal view faces in the image frame. We then use the pixels in the detected face region to calculate the color distribution of the face and use the remaining pixels to calculate the color distribution for the background. Then we use the algorithm described in Section 2 to track the face. We use the size and aspect ratio of the tracked face to decide whether the tracking is correct or not. For example, if the aspect ratio (height over width) is larger than a pre-determined threshold, we assume that the face tracking has failed. Once the tracking fails, we reset the system to the detection module. The system diagram is shown in Figure 5.

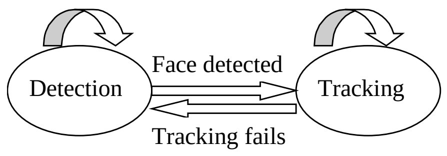

Figure 5 Block diagram of a real-time tracking system

We also developed an eye-tracking algorithm, which can be integrated with the face tracking system. To track the eyes, we start from the eye positions from the previous frame, and update them based on the information we obtain from the

[‡] Available at http://amp.ece.cmu.edu

current frame. First we position the search window around the initial position of each eye given by the previous frame. If the person is not moving too fast, the iris of the eye is still inside of the search window as shown in Figure 6.

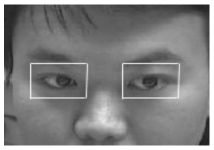

Figure 6 The search window

In the search window, we compare the normalized intensity of a pixel with a given threshold, and then calculate the center point of the pixels whose intensity is below the threshold. The center point is considered as the current position of the iris. We can use this position as the center of the search window for the next frame, and repeat this procedure again. Figure 7 shows the cropped search window regions, the binary image after thresholding the image intensity, and the resulting eye centers.

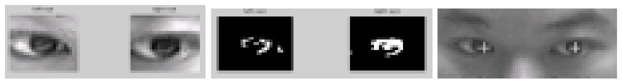

Figure 7 From left to right: the search window regions, the binary images, and the resulting eye centers

Based on the relative positions of the eyes and the face, we can decide the pose of the face, as shown in Figure 8, which is a human head seen from above.

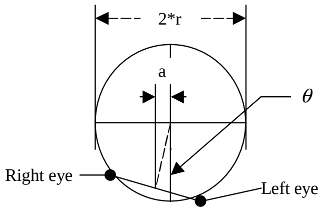

Figure 8 Pose estimation

With a being the distance between the projection of the mid-point of two eyes and the center of the face, and r being the radius of the head, we can estimate the pose θ as follows:

$$\theta = \arcsin(a/r)$$

We have used this pose estimator as part of a virtual conference system we have been building [10]. This system can estimate the pose of the user and render the corresponding avatar with the correct pose. We have also used the pose estimator as the front-end of a pose-invariant face recognition system [11]. We feed the cropped face region together with the pose information into the recognition module to obtain better recognition.

6. CONCLUSIONS AND FUTURE WORK

We described a face tracking algorithm based on the stochastic color model and the deformable template, and extended the algorithm to track multiple faces with some constraints on speed and size changes of the face during occlusion. We tested the algorithm on video sequences with different occlusion patterns and show that our algorithm works well for most of the test sequence.

Still, there are a lot of research work which can be done to improve this algorithm. For example, noticing that after occlusion the tracking error becomes very small, we should be able to reduce the tracking error during the occlusion by performing inverse tracking from the tracking results after the occlusion.

Acknowledgements

The authors would like to thank Dr. Tom Drayer, Department of Defense, and Dr. Ram Rao, Intel Corporation, for fruitful discussions.

7. REFERENCE

[1] J. Yang and A. Waibel, “A Real-Time Face Tracker,” *Proceedings of WACV’96*, pp. 142-147.
[2] G. Bradski, “Computer Vision Face Tracking for Use in a Perceptual User Interface,” http://developer.intel.com /technology/itj/q21998/articles/art_2.htm
[3] A. Colmenarez, R. Lopez and T. Huang, “3D Model-Based Head Tracking,” *Visual Communication and Image Processing*, San Jose, CA, 1997
[4] D. DeCarlo and D. Metaxas, “Deformable Model-Based Face Shape and Motion Estimation,” *IEEE Proc. of ICFG*, 1996.
[5] T. Maurer and C. Malsburg, “Tracking and Learning Graphs and Pose on Image Sequences of Faces,” *IEEE Proc. of ICFG*, pp. 176-181, 1995.
[6] R. Rao and R. Mersereau, “On Merging Hidden Markov Models with Deformable Templates,” *Proceedings of the Int'l Conf. on Image Processing*, Washington D.C., 1995.
[7] G. Welch and G. Bishop, “An Introduction to the Kalman Filter,” *Technical Report TR 95-041*, Computer Science, UNC Chapel Hill, 1995
[8] R. Rosales and S. Sclaroff, “Improved Tracking of Multiple Humans with Trajectory Prediction and Occlusion Modeling,” *IEEE Conf. On CVPR*, CA, 1998.
[9] H. Rowley, S. Baluja and T. Kanade, “Neural Network based Face Detection,” *IEEE Trans. On PAMI*, pp. 23-38, 1998
[10] H. Leung and T. Chen, “Networked Collaborative Environment with Animated 3D Avatar,” *IEEE workshop on Multimedia Signal Processing*, 1998.
[11] F.J. Huang, Z. Zhou, H-J Zhang and T. Chen, “Pose Invariant Face Recognition,” *IEEE ICFG 2000, Grenoble, France*.