Reinitialization of a Model-Based Face Tracker

Jacob Ström*
Department of Electrical Engineering
Linköping University
SE-581 83 LINKÖPING, Sweden
jacobs@isy.liu.se

Abstract

This paper investigates how to reinitialize a model-based face tracker after tracking failure. The information gathered in the original initialization is used to obtain a quick an reliable reinitialization. The color information of the model texture map is used to obtain a skin color model that is per definition adapted to the lighting, camera and the specific skin tone of the user. The search for the largest skin colored blob is then further refined by template matching using a part of the face texture between the eyes. From this refined position the tracker is restarted. A method for automatic detection of tracking failure is also presented.

1 Introduction

Automatic tracking and modeling of human faces from image sequences is an important and challenging task in computer vision. Applications include face recognition, model-based coding for video conferencing, avatar control and computer graphics. The reinitialization of a tracker is an important system component; sooner or later the tracking will fail. Jebara and Pentland [3] detect tracking failure by normalizing the face texture back to frontal position and measuring the "Distance From Face Space" (DFFS). When the DFFS is larger than a threshold, a face detection process restarts the tracker from scratch, disregarding information gathered so far. The key point of this paper is that robustness and speed can be gained by having a special reinitialization procedure. The reason for this is that a lot of information that was gathered in the initialization stage can be reused to help the reinitialization. For example, the texture map of the face that was obtained during initialization can be used to build an accurate skin color model as well as providing templates for template matching.

Crowley and Berard [2] use three visual processes for tracking and continuous reinitialization; a skin-color blob model, a correlation tracker and a blink detector. While their work is 2D based, we have previously proposed a 3D real-time head tracking system using model-based techniques [6], and the work presented here is based on that system.

In this paper, the tracking accuracy is continuously measured, and when a tracking failure is declared, the reinitialization procedure starts. First, a skin color model is obtained from the original face texture. This color model will be very accurate, since neither the person, the camera nor the lighting conditions have changed. The image is searched for the biggest skin colored blob, which in turn serves as a search area for template matching. A region between the eyes is cropped out from the original texture and used as a template. The 3D model is then translated until the point between the eyes matches the result of template matching. After this, the model-based loop is restarted, and if the model is inside the system's zone of convergence, the tracker will lock on to the target.

The paper is organized as follows: The next section describes how the model-based face tracking system works. It is followed by a discussion on typical tracking failures in Section 3. Section 4 treats the proposed reinitialization algorithm and Section 5 contains conclusions and future work.

2 The Model-Based Tracker

The face tracker is based on the idea of selecting a dense set of feature points (essentially, optical flow at all the most information bearing points), thus extending the foundations of Azarbayejani and Pentland [1] and Jebara and Pentland [3] to achieve robust performance. Figure 1 illustrates how the system works. Patches around the feature points taken from the rendered 3D model (lower left corner) are matched against the incoming video, and the 2D trajectories of these feature points are then fed through a structure from motion (SfM) algorithm to update the pose in-

*Telephone number: +46 13 28 21 63 Fax number: +46 13 28 44 22 email: jacobs@isy.liu.se

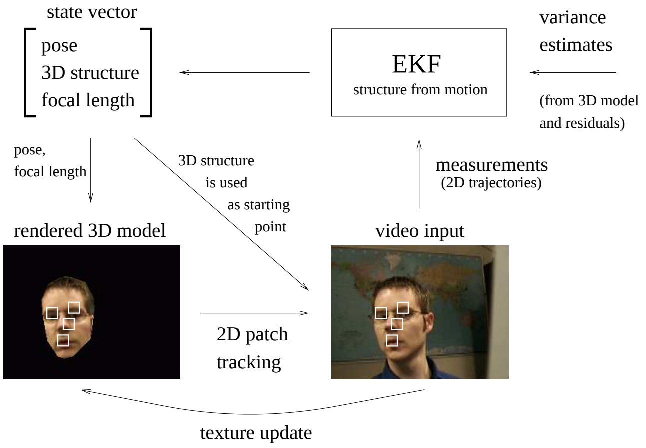

Figure 1: *Patches from the rendered image (lower left corner) are matched with the incoming video. The 2D feature point trajectories are fed through the SfM extended Kalman filter that estimates the pose information needed to render the next model view. For clarity, only four patches are shown.*

Figure 2: *A generic 3D polygonal head model is aligned with a head-on shot of the video sequence, and the corresponding pixels are texture mapped to the surface of the face model.*

formation (translation and rotation) of the 3D model. In addition, the estimated structure serve as a starting point for the tracking. The 3D model is incrementally updated with the texture of the sides of the head, until a full 3D textured head model is recovered. By tracking features on the side of the head, the system can cope with large out-of-plane rotations.

The extended Kalman filter provides a convenient mechanism for fusing all the information about the reliability of the measurements into an estimate of the 3D structure, the pose and the focal length of the camera.

2.1 Initialization

The system is initialized from a frontal position as seen in Figure 2. A generic polygonal face model[1] [5] (left) is aligned to match with a head-on view of the face in the video sequence (middle). The pixels from the video are then texture-mapped onto the model (right).

After alignment has been performed, the system selects which feature points to track. By selecting features where the determinant of the Hessian $\begin{vmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{vmatrix}$ is high, the system avoids locking on to features that are too vague to be reliably tracked. A total of 24 points are found for further tracking. Consult Ström et al. [6] for more details on this process.

[1] The 3D model is a modified version of CANDIDE

2.2 Tracking

As shown in Figure 1, the tracking is carried out between the rendered frame and the video input. Both images are subsampled in order to track larger and more robust features. Since the feature points are fixed with respect to the 3D model, their 2D coordinates in the rendered image are known. A 7×7 pixel patch around each feature point is cropped out. This patch is then matched with patches from the video input image in a 11×11 pixel search window using weighted normalized correlation. An exhaustive search is carried out in the search window and the candidate with the lowest error is selected. The 2D measurements are fed to the extended Kalman filter that extracts the rotation and translation parameters of the object. The tracker is running in excess of 25 Hz on an SGI O2 computer of 300 MHz. A more complete description of the tracker and the extended Kalman filter can be found in [6].

3 Tracking Failure

3.1 Convergence Zone

The convergence zone of a tracking system is defined to be the set of all poses (translation and rotation) of the model that will converge to the true pose, if the tracker is allowed to run. The tracker works by starting inside the convergence zone in the first frame, and as the pose changes in each frame (caused by the user's movements), the tracker will converge to the true pose. As long as the pose changes in each frame are inside the convergence zone, the tracking will work.

3.2 Typical Failures

There are three typical cases when a tracker fails. The first is if the user moves too quickly from one frame to the next. Then the pose of the model will be outside the convergence zone, and the tracking will fail. The second case is when the user's head is rotated far away from the original (frontal) position, such as in Figure 6c. Many of the easily trackable points (around the eye and nose) are then facing away from the camera, and errors in the 3D modeling are also becoming more visible. Both of these factors make the convergence zone shrink substantially, and tracking can be lost even at slow pose velocities. The third case is when an obstacle such as an arm completely occludes the face — the convergence zone is then virtually zero. A scenario containing all three of the typical cases is when the computer operator turns his head quickly 180 degrees towards someone standing behind the computer, shown in Figure 6d.

If the tracker output is used for controlling a Human Computer Interface, then a tracking failure in such a situation is not severe: When the computer operator is turned away from the screen, he/she does not need to control the computer. However, when the user returns to the screen, the tracker should start working again. Thus a good reinitialization procedure can be almost as good as continuous tracking for some applications.

4 Reinitialization

4.1 Failure Detection

The tracking procedure described in Section 2.2 will produce a correlation value, ρ, for each point. A ρ close to 1 indicates an accurate match, whereas a low ρ means that the measurement is insecure. If the tracking has failed, it is unlikely that any point will produce a high ρ. We declare a tracking error if the best ρ is smaller than a threshold value for the entire duration of ten frames. The resulting detector is not water proof, but since the ρ measurements are a bi-product of the tracking, it is virtually cost-free. It also works reasonably well in practice.

4.2 Skin Color Processing

Once a tracking failure is declared, the reinitialization process starts with building a skin color model. The texture map of the 3D model from the original initialization is used. Since the lighting, camera and the skin tone of the person can be assumed to be the same as when this texture was acquired, a simple gaussian in the YUV color space suffices to approximate the skin color histogram. Statistics is gathered from points in the original texture (Figure 3a), using the image shown in Figure 3b to mask all but the forehead and the cheeks. The input image (Figure 3c) is subsampled and the skin color probability is calculated and thresholded for each pixel. The biggest connected skin color blob is then found, as illustrated in Figure 3d.

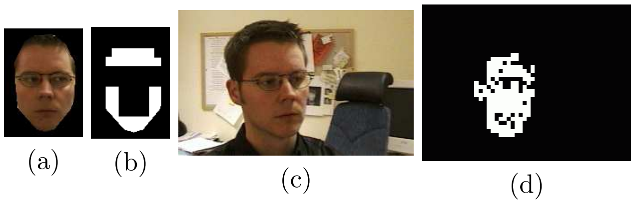

Figure 3: *(a) Original texture (b) mask (c) input image (d) color blob*

4.3 Template Matching Refinement

The color blob obtained in the previous section is a robust way to locate the face, but not exact enough. To refine this position, a 13×13 template (Figure 4a) from the area between the eyes is cropped out from a subsampled version of the original texture. The bounding box of the color blob is then used as a search window for the template. Exhaustive search using normalized correlation is used in the search window, and the location of the maximum of the error surface (shown in Figure 4b) is used. Figure 4c shows a result of the template matching. Empirically, the texture between the eyes is rather insensitive to fairly large changes in scale and rotation.

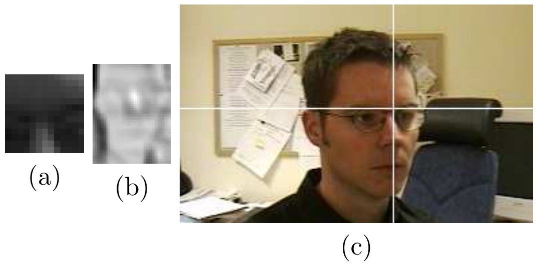

Figure 4: *(a) Between-eyes template, (b) error surface, (c) result*

4.4 Restarting the tracking

The X and Y image coordinates for the "between-eyes" patch obtained above will not suffice to resolve the unknown six degrees of freedom that constitute the head pose. However, if the model is placed within the convergence zone, the tracker will resolve the remaining unknowns. Thus it is important that the model is positioned in a pose that is as plausible as possible. We have chosen to translate the head in the x and y domain using the same z as in the original initialization. (This is the same as assuming that the head approximately as big on the screen as it was during the original initialization.) The rotation is zeroed so that the face is head on. Figure 5 shows an example of this. The left most image shows the video input at

Figure 5: *Tracker convergence at restart.*

the time of reinitialization. The tracker is restarted in the position shown in the second image. After five iterations (about 0.2 s) the tracker has converged to the correct pose.

4.5 Results

Figure 6 illustrates a tracking example. The tracking is initialized in (a), and continued in (b). The image (c) shows the maximum rotation that can be managed without losing track, whereas the track is lost in (d). In (e) the tracking is regained.

5 Conclusions and Future Work

This paper has presented how a model based tracker can be reinitialized after tracking failure. Information from the original initialization is used to make the process fast and robust. However, improvements are possible in a number of areas. The failure detection used is simple and cost free but (1) sometimes creates false alarm if the head is rotated too much and (2) sometimes fails to detect a tracking failure if the translation is correct but the rotation is wrong. Type (2) errors can be avoided by adding a DFFS type criterion as proposed by Jebara and Pentland [3], whereas type (1) errors are harder to handle. As shown in Figure 5, the system can reinitialize even if the user is not head-on. To be able to reinitialize during large out of plane rotations however, the system must be able to guess the rotation of the head before the tracker is restarted. This can be done by either by starting the tracker form a small set of rotated poses or by creating a small number of templates by rotating the 3D model. The templates can then be correlated with the video in the skin blob area and the best template will indicate the approximate pose of the head.

References

[1] A. Azarbayejani and A. Pentland, "Recursive Estimation of Motion, Structure, and Focal Length," *IEEE Pattern Analysis and Machine Intelligence*, vol. 17, no. 6 pp. 562-575, June 1995.

[2] J. L. Crowley and F. Berard, "Multi-Modal Tracking of Faces for Video Communications," In *Proceedings of IEEE Conference in Computer Vision and Pattern Recognition, CVPR '97*, San Juan, June 1997.

(a)

(b)

(c)

(d)

(e)

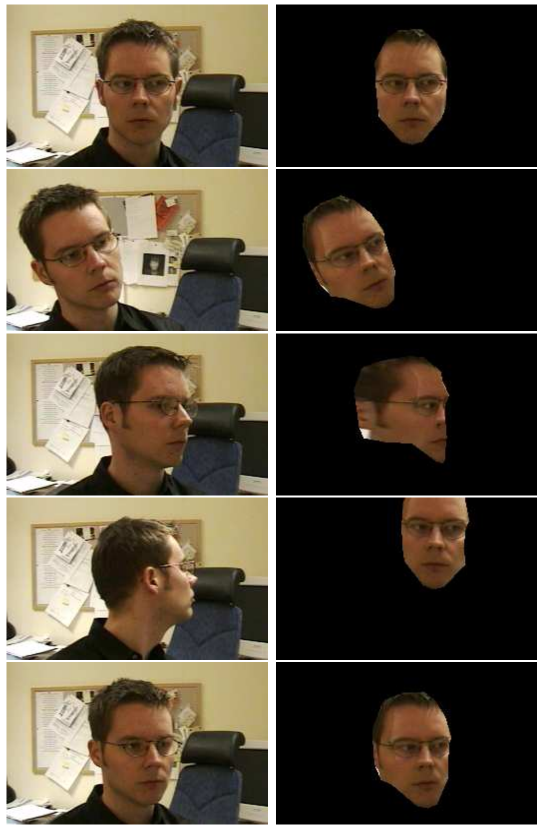

Figure 6: *Typical tracking sequence.*

[3] T. Jebara and A. Pentland, "Parametrized Structure from Motion for 3D Adaptive Feedback Tracking of Faces," *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR'97)

[4] T. Jebara and A. Pentland, "Mixtures of Eigenfeatures for Real-Time Structure from Texture," *Proceedings of ICCV'98*, Bombay, India, January 4-7, 1998.

[5] M. Rydfalk, "CANDIDE, a parameterized face," *Technical Report*, LiTH-ISY-I-0866, Linkoping University, 1987.

[6] J. Strom, T. Jebara, S. Basu, and A. Pentland, "Real Time Tracking and Modeling of Faces: An EKF-based Analysis by Synthesis Approach," *Proceedings of the Modelling People Workshop* at ICCV'99, August 1999