Model-based Face Tracking for Dense Motion Field Estimation

Timothy F. Gee
Image Science and Machine Vision Group
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6010
geetf@ornl.gov

Russell M. Mersereau
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
rmm@ece.gatech.edu

Presented at the IEEE 30th Applied Imagery and Pattern Recognition Workshop

Abstract

When estimating the dense motion field of a video sequence, if little is known or assumed about the content, a limited constraint approach such as optical flow must be used. Since optical flow algorithms generally use a small spatial area in the determination of each motion vector, the resulting motion field can be noisy, particularly if the input video sequence is noisy. If the moving subject is known to be a face, then we may use that constraint to improve the motion field results. This paper describes a method for deriving dense motion field data using a face tracking approach. A face model is manually initialized to fit a face at the beginning of the input sequence. Then a Kalman filtering approach is used to track the face movements and successively fit the face model to the face in each frame. The 2D displacement vectors are calculated from the projection of the facial model, which is allowed to move in 3D space and may have a 3D shape. We have experimented with planar, cylindrical, and Candide face models. The resulting motion field is used in multiple frame restoration of a face in noisy video.

1. Introduction

The motivation for this work is to perform video restoration in forensic applications. Often in surveillance video, it is necessary to remove noise from input video. This is particularly the case for video involving faces, because it is often necessary to improve the quality of an image of a perpetrator to determine suspects.

Frame averaging is a simple and effective tool for reducing noise in video; however, when motion is present it will cause blurring due to the misregistration of moving objects. Motion compensation of individual frames to register an object across frames greatly reduces blurring caused by frame averaging. Unfortunately, a major problem with motion compensation of noisy video is that the quality of the motion estimation is affected by the noisy input images.

In this paper, we are able to accurately and robustly calculate motion data of a moving face from noisy video by using a model-based approach. We present a method for motion estimation that uses facial feature tracking and a face surface model to robustly and accurately estimate the actual motion data. Once the motion data are obtained, the video frames are warped so that they are registered. Lastly, frame averaging can be performed to fuse frames to produce a single image with reduced noise.

2. Our Approach

2.1 Overview

This model-based approach takes advantage of the fact that we are interested solely in the motion estimation of a face. Five rectangular facial regions are tracked using block matching. These rectangular regions are shown in Figure 1. These regions are areas that contain a large amount of information. This makes the block matching very robust to noise since a large number of pixels are used for a small number of measurements, and the particular regions are very descriptive.

The spatial coordinates of the centers of the feature rectangles are used as the observations in an Extended Kalman Filter (EKF) [4] that tracks the position and orientation of the feature rectangles as a group in 3D space. The resulting position and orientation data can be used to control any wireframe model. An example is the Candide wireframe face model [1], which is shown in Figure 2. The wireframe model is used to generate a motion displacement vector for each pixel in the image. In this work, displacements are generated from one frame to its following frame. This al-

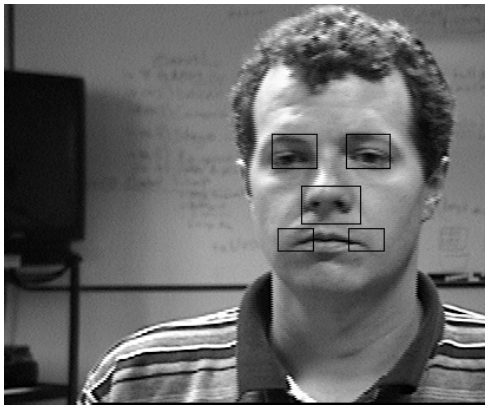

Figure 1. Facial feature rectangles

lows us to continually average frames as we move forward in time through the video.

The face tracking approach described here is not intended for continual automatic face tracking. This algorithm is aimed at processing a small number of frames (on the order of 10-20) with user interaction. Generally, forensic video restoration is performed offline, and only a small number of frames are available. Steps to make this algorithm more amenable to continuous tracking might include warping the rectangles with changes in pose and using the EKF and facial model to recover from block-matching errors.

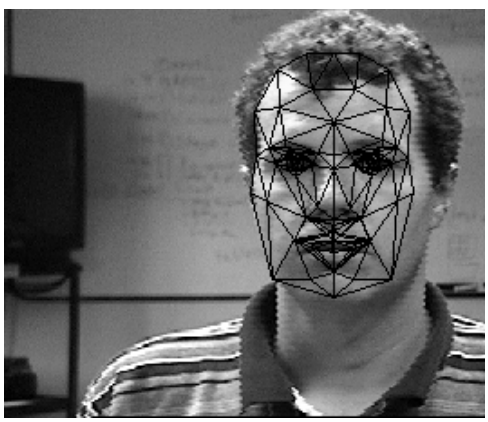

Figure 2. Candide model used

2.2 Facial feature block matching

A graphical user interface is used to initialize the facial feature blocks. The user manually alters the position, orientation, and scale of the feature rectangles so that they correctly cover the features of interest: both eyes, the nose, and both corners of the mouth.

Since the feature rectangles are allowed to move in 3D space, they will not necessarily project onto the image space as rectangles. We are assuming orthographic projection within the facial model, so the rectangles become parallelograms. We find the smallest rectangles in the the image space which completely contain these parallelograms. Those rectangles are used for the block matching within the images. Since for our purposes, the number of frames is likely to be small, and we are assuming a small change in pose, the grayscale information in the original rectangles serve as the references for matching in all frames. The miniumum Mean Absolute Difference (MAD) criterion [6] is used as a difference measure. That is we minimize

$$MAD(q,r) = \frac{1}{MN} \sum_{(m,n)\in\mathcal{B}} |f(m+q,n+r,k) - f(m,n,1)| \quad (1)$$

within each search window, where f are video frames indexed by the integer time index k, m and n are integer spatial indices, q and r are integer displacments, $\mathcal{B}$ is the rectangular region being considered as a match, and $f(m,n,1)$ is the first frame in the sequence. The search region for q and r is limited to a rectangular search region. The extent of that rectangular search region is a function of the preceding frame's block location, so the search region changes as the matched blocks move in space and time.

2.3 Extended Kalman Filter

The Extended Kalman Filter tracks the following state variables:

$$\mathbf{s} = \begin{bmatrix} x & y & q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T, \quad (2)$$

where x and y are the spatial coordinates of the facial model, and the remaining variables define the 3D orientation of the Candide model using quaternions [4]. Note that while the model is allowed to change orientation in 3D space, the z dimension is otherwise ignored. Also, scaling is fixed (i.e. we are assuming orthographic projection). Originally, scaling was considered, but it made the algorithm very unstable. Therefore, we set the scale during initialization, and assume that the face moves very little in the z direction during the sequence.

The equations used are [6]:
the state prediction equation,

$$\hat{\mathbf{s}}_b(k) = \Phi(k,k-1)\hat{\mathbf{s}}_a(k-1), \quad (3)$$

the error covariance prediction equation,

$$\mathrm{P}_b(k) = \Phi(k,k-1)\mathrm{P}_a(k-1)\Phi(k,k-1)^T + \mathrm{Q}(k), \quad (4)$$

the Kalman gain equation,

$$\mathrm{G}(k) = \mathrm{P}_b(k)\mathrm{H}^T(k)\{\mathrm{H}(k)\mathrm{P}_b(k)\mathrm{H}^T(k) + \mathrm{R}(k)\}^{-1}, \quad (5)$$

the state update equation,

$$\hat{\mathbf{s}}_a(k) = \hat{\mathbf{s}}_b(k) + \mathrm{G}(k)[\mathbf{w}(k) - \mathbf{h}(\hat{\mathbf{s}}_b(k))], \quad (6)$$

and the error covariance update equation,

$$\mathrm{P}_a(k) = \mathrm{P}_b(k) - \mathrm{G}(k)\mathrm{H}(k)\mathrm{P}_b(k), \quad (7)$$

where $\mathbf{s}$ is the state vector,
$\mathbf{w}$ is the observation vector,
$\mathbf{h}$ is the non-linear system observation function,
R is the observation noise covariance matrix,
Q is the state noise covariance matrix,
P is the state estimation error covariance matrix,
G is the Kalman gain matrix,
Φ is the state update matrix,
and H is the observation matrix of the linearized system obtained by

$$\mathrm{H}(k) = \left.\frac{\mathbf{h}(\mathbf{s}(h))}{\mathbf{s}(k)}\right|_{\mathbf{s}=\hat{\mathbf{s}}_b(k)}. \quad (8)$$

The subscripts "b" and "a" denote before and after the update respectively. Φ is set to the identity matrix to assume constant position. The initial value of $\mathbf{s}$ is set according to the manual initialization of the facial model. $\mathbf{w}$ is a ten-element vector containing the *x* and *y* coordinates of the facial feature tracking rectangles. That is

$$\mathbf{w} = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \cdots & x_5 & y_5 \end{bmatrix}^T, \quad (9)$$

where x_n and $y_n, n = 1, 2, \ldots$ are the face feature rectangle centers.

2.4 Generation of Dense Motion Field

The facial surface model is used to convert face tracking information into a dense motion field. This is the inverse of what was done in work by Basu et al. [2] in which they use an optical flow algorithm to calculate a dense motion field, and then they iterate to determine the head position and orientation that could enable a 3D facial model to produce the given motion field. The benefit to their approach is that face tracking can be achieved without dependence on the presence of all facial features. However, here we assume that the facial features will be present, and we are using them as a robust way to determine the position and orientation of the head. Thus we are able to create a dense motion field even though noise is present.

The generation of the dense motion field from the face surface model assumes an orthographic projection. It is reasonable to assume that within the limited depth change of the face that there is little variation in scaling. The motion field that we are seeking describes the displacement from one frame to the next. That is

$$f(m,n,k) = f(m - u(m,n,k), n - v(m,n,k), k-1), \quad (10)$$

where *u* and *v* are real values containing the horizontal and vertical motion displacements respectively. From the above equation, we see that for a given pixel, the motion vector describes the location from where that pixel originated in the previous frame. The benefit of this form will be described below.

The facial surface model can be any 3D shape represented by a wireframe mesh of triangular regions. For each frame, we calculate the coordinates of each of the vertices of the facial model wireframe. For each pixel of the given frame we determine where it projects onto the wireframe model using orthographic projection. If the given pixel does not fall on the model, then we assume the motion for that pixel is zero. If it projects onto a triangle in the mesh, we use that triangle's plane and the known *x* and *y* location of the pixels to determine the *z* or depth component. If the pixel projects to multiple triangles, as can happen with a 3D structure, the projection point closest to the camera is used. Once the *z* component is obtained, we find the 3D position of the point in the previous frame, and the difference is the 3D motion vector. Since we are only concerned with the 2D motion vector, we perform orthographic projection which is simply removing the *z* component of the 2D motion vector.

2.5 Motion-compensated frame averaging

The motion compensation of frames is achieved by the backward warping [3] of each pixel. Because of how the motion vectors are defined in (10), each pixel has a single motion vector whose head is at that pixel and whose tail corresponds to a location in the previous frame. The tail of the motion vector does not necessarily correspond to an integer pixel location; however, a source pixel is easily obtained by bilinear interpolation of the four nearest pixels.

At each frame, we calculate the average of the current frame and all preceding frames of interest. This is calculated recursively by continually warping the average image one frame forward in time. That is we recursively calculate

$$\hat{f}(m,n,k) = \frac{1}{k} f(m,n,k) + \frac{k-1}{k}\hat{f}(m - u(m,n,k), n - v(m,n,k), k-1), \quad (11)$$

where $\hat{f}$ is the averaged frame.

3. Experiments and Results

We tested our approach for motion-compensated frame averaging on a sequence of 16 images. We also used an implementation of an optical flow algorithm for creating motion estimation data. The algorithm was designed by Black [3], and it is a robust extension to the Horn-Schunck [5] optical flow algorithm. We will refer to it hereafter as the robust optical flow algorithm.

We artificially add Gaussian noise to the images to achieve a signal-to-noise ratio of 2.1 dB. The first image in the noise sequence is shown in Figure 3, and the last image is in Figure 4. Although the robust optical flow algorithm has a spatial smoothness constraint, it has trouble with the noise in the image sequence. The frame-averaged result obtained from using the robust optical flow algorithm is shown in Figure 5. As one can see, the noise in the video sequence has caused the motion field generated from the robust optical algorithm to be noisy. Thus the fused result gives the impression of a melted face.

Figure 3. First frame of original sequence with noise added

Figure 4. Last frame of original sequence with noise added

Figure 5. Frame fusion using robust optical flow algorithm

We used three different face surface models in our model-based approach. Those are the Candide model, a cylindrical model, and a planar model. The Candide and cylindrical models are plotted together in Figure 6. The frame-averaging results for the different models are shown in Figures 7, 8, and 9.

The model-based approach is able to perform well and is able to remove noise through motion-compensated frame averaging. It appears that the different models perform roughly equally. Remember that all pixels not on the model are not motion-compensated. Therefore, the shoulders are blurred. Also, the models have different extents. For example, the models vary in how much hair is within the model. Also note that the models leave a trail. This is not a problem, since we are only concerned with the facial region.

4. Conclusions

Here we have shown that a 3-D model can be used to add robustness to the optical flow calculations for a face in a video sequence. Additionally, the resulting optical flow can be used to fuse multiple frames to create a single frame restored face. The tracking is based on facial features that are easily identified and have significant edge information. The 3-D model enables us to convert the facial feature tracking into realistic optical flow data for the entire face. Experimental results have shown that in noisy video data, this approach is more accurate that using a less-constrained optical flow algorithm

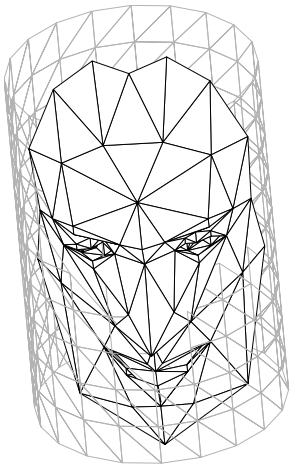

Figure 6. Relationship of cylindrical and Candide models used

Figure 7. Frame fusion using Candide 3D model

Figure 8. Frame fusion using cylindrical 3D model

Figure 9. Frame fusion using planar model

Acknowledgements

Special thanks to Dr. James Goddard of the Image Science and Machine Vision Group for providing assistance and source code for performing Extended Kalman Filtering with quaternions.

References

[1] J. Ahlberg. Candide-3 – an updated parameterised face. Technical Report LiTH-ISY-R-2326, Dept. of Electrical Engineering, Link'oping University, Sweden, 2001.

[2] S. Basu, I. Essa, and A. Pentland. Motion regularization for model-based head tracking. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition*, 1996.

[3] M. Black. *Robust Incremental Optical Flow*. PhD thesis, Yale University, 1992.

[4] J. Goddard and M. Abidi. Pose and motion estimation using dual quaternion-based extended kalman filtering. In *Int. Soc. for Optical Engineering Photonics West Symposium on Electronic Imaging*, 1998.

[5] B. Horn and B. Schunck. *Artificial Intelligence*, 17(1-3):185–203, aug 1981.

[6] A. Tekalp. *Digital Video Processing*. Prentice Hall, 1995.