FEATURE-BASED FACE TRACKING USING EXTENDED KALMAN FILTERING

Nils Ingemars and Jörgen Ahlberg

Image Coding Group
Dept. of Electrical Engineering
Linköping University

ABSTRACT

This work examines the possiblity to, with the computational power of today's consumer hardware, employ techniques previously developed for 3D tracking of rigid objects, and use them for tracking of deformable objects. Our target objects are human faces in a video conversation pose, and our purpose is to create a *deformable face* tracker based on a *head* tracker operating in real-time on consumer hardware. We also investigate how to combine *model-based* and *image based* tracking in order to get precise tracking and avoid drift.

Index Terms— Face tracking, extended Kalman filtering, model-based coding

1. INTRODUCTION

Tracking of faces and facial features are motivated by applications such as motion capture for entertainment (games, caricatures, movies, ...), low bitrate video communication (model-based coding), face recognition (access control, surveillance), and facial gesture/expression recognition (human-machine interfaces). Thus, a considerable amount of research has been devoted to this ares, and the technology is now mature enough to enter commercial products.

Face tracking methods can be divided in two major groups, *appearance-based* tracking and *feature-based* tracking. Appearance-based methods typically use a *generative model* that parameterizes the image of the face. The model is fitted to an image by finding the parameters that minimizes the difference between the input image and the generated image. Since the optmimization procedure is greedy and have many local optima, a good starting point is needed. In a tracking situation, a good starting point for each frame is given by the parameters estimated in the previous frame.

Feature-based methods, on the other hand, track small features, typically image patches. Since the feature tracking is done in 2D, some method must be employed to solve the 2D-to-3D problem. One such method is extended Kalman filtering (EKF), which will be used here. In order to estimate the global parameters of a rigid object (six degrees fo freedom), at least three features need to be tracked (two degrees of freedom per feature). However, in practice, more than ten are needed to achieve stable tracking. This has earlier been a severe limitation, prohibiting feature-based trackers of this kind to track deformations (e.g., facial expressions).

Feature-based trackers van also be *image-based* or *model-based*. An image based-tracker extracts patches from the current image and matches them in the following image, whereas a model-based tracker synthesizes the patches from a model of the face. The drawback with the image-based approach is drift (error accumulation), and the drawback with the model-based approach is the tracking instability due to model failures, i.e., when the model is not able to correctly model the object.

In this paper, we will describe a feaure-based face and facial feature tracker (i.e., tracking deformable objects), using and EKF to solve the 2D-to-3D problem. The tracker combines image and model-based tracking and operates in real-time on consumer hardware.

This work is inspired by and based on the work of Ström [5]. The main contributions of this work are

1. The extension to facial feature tracking (using a deformable model) in contrast to the rigid head tracking performed by Ström's tracker.
2. The combination of an image-based and model-based tracker having the advatanges of both to the cost of some added computational complexity.

The paper is outlined as follows. Section 2 describes the face model, the model parameters to be estimated and tracked, and the task of the tracker. Section 3 describes the tracking process in some more detail. Section 4 describes our implementation and shows some results, and, finally, our conclusions are given in Section 5.

2. FACE MODEL AND PARAMETERIZATION

The face model used is the good ol' Candide model [4, 1], see Fig. 1, which is a polygon model of some hundred vertices and polygons. The model is parameterized in the following way:

$$\mathbf{m} = \bar{\mathbf{m}} + \mathbf{A}\alpha + \mathbf{S}\sigma, \tag{1}$$

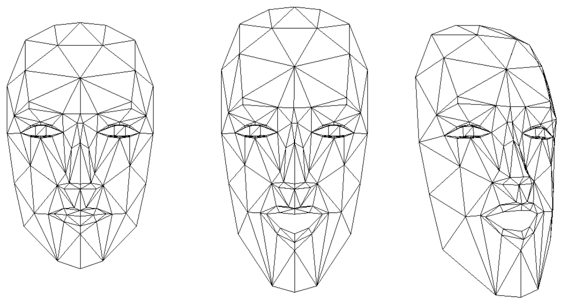

Fig. 1. The Candide model.

where $\bar{\mathbf{m}} = [x_1\, y_1\, z_1 \ldots x_N\, y_N\, z_N]$ is the standard shape of the model (the 3D coordinates of all the N vertices), the columns of $\mathbf{A}$ and $\mathbf{S}$ contain the *action* and *shape* deformations respectively, and the parameter vectors α and σ contain action and shape *parameters*. The conceptual difference between shape and action parameters is that shape parameters describe the deformations from a standard shape to a certain individual and the action parameters describe the deformations due to dynamic changes, such as lip motion. Shape parameters are thus fixed during a tracking session.

The deformed model is subject to global motion, i.e., rotation and translation, and thus

$$\mathbf{g} = \mathbf{R}\mathbf{m} + \mathbf{T}, \tag{2}$$

where $\mathbf{R}$ is a rotation matrix determined by the three Euler angles (rotation parameters) r_x, r_y, and r_z, and $\mathbf{T}$ is a translation vector determined by the three translation parameters t_x, t_y, and t_z. We assume that the rotation angles are small enough for gimbal lock not to occur, and that we thus can parameterize directly on Euler angles instead of quaterions. Considering, for example, a video conversation scenario, this assumption is non-controversial.

After deformation and global motion of the model, the vertices are projected onto the image plane using a perspective projection, i.e., for the ith vertex

$$\mathbf{u}_i = \frac{f}{z_i} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{3}$$

The vector $\mathbf{z}$ of all vertex image coordinates is thus a non-linear function

$$\mathbf{z} = h(\mathbf{x}) = h(r_x, r_y, r_z, t_x, t_y, t_z, \alpha_1, \ldots, \alpha_K) \tag{4}$$

of the model parameters. The task of the tracker is

1. 2D tracking: To track certain vertices, that is, estimate the image coordinates $\mathbf{z}_j$ from each image in a sequence (where j is the time index).

2. 2D-to-3D esimation: To estimate the model parameters $\mathbf{x}_j$ from $\mathbf{z}_j$.

This tracking process is deescribed in the following section.

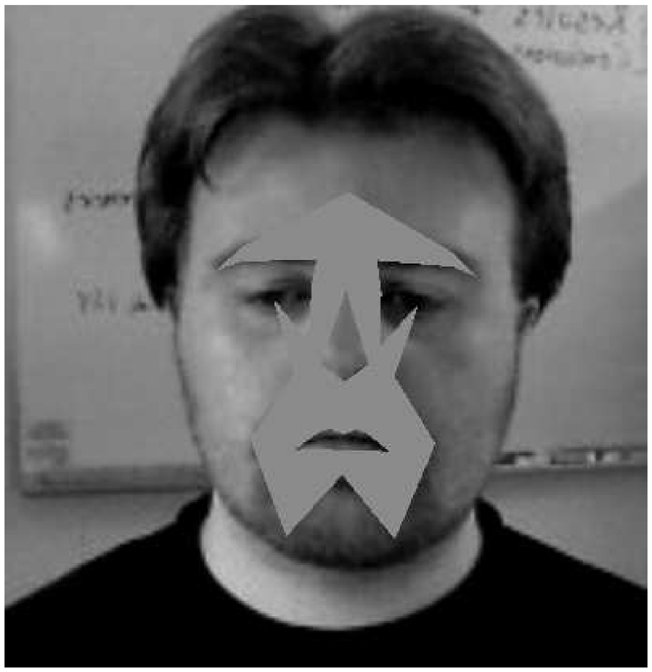

Fig. 2. The mask showing from which polygons features are selected.

3. THE TRACKING PROCESS

3.1. Feature selection and tracking

In the first frame of the image sequence to be analyzed, the model is manually adapted to the subject's face. Then, suitable features (image patches) to track are selected. Suitable features are features with some special visual properties that should be easy to track, for example corner-like points. They should also be in the front of the face as to be visible most fo the time (less self-occlusion) and to be on a surface more or less parallell to the image plane, we chose to limit the search for good features to a few polygons as shown in Fig. 2. Moreover, they should be spread out over the face, for two reasons. First, there should be enough feature points near vertices that are affected by the model deformation to be estimated. Second, spread-out features will give less noise in the estimation of rotation parameters.

When the feature points are found as projected points in the first frame, each of them is checked with respect to which projected model surface triangle they fall into. They are then linearly interpolated from the triangle corners into the face model as new vertices with their own values in $\bar{\mathbf{g}}$, $\mathbf{S}$ and $\mathbf{A}$. These values are interpolated from the corresponding values belonging to the corner vertices by using a weight for each corner. The weights sum to 1 and define the linear combination of the corner vertices coordinates to get the feature point vertex coordinates.

The feature points are not actually part of the model, but more like virtual vertices that are different from one tracking session to another.

The features are tracked using normalized zero-mean cross-correlation, matching patches from the previous frame to the current one. The size of the patches depends on the distance between the face and the camera, however, never less than 3×3 pixels. Subpixel precision is achieved by computing the center of gravity of the matching results in the 3×3 pixel neighbourhood around the best found match

3.2. State model

The dynamic state model is expressed as

$$\begin{aligned} \mathbf{x}_{j+1} &= \mathbf{F}\mathbf{x}_j + \mathbf{w}_j \quad (5) \\ \mathbf{z}_j &= h(\mathbf{x}_j) + \mathbf{v}_j \quad (6) \end{aligned}$$

where, as above, the observation vector $\mathbf{z}_j$ contains the image coordinates of the tracked vertices and $\mathbf{x}_j$ is the vector of model parameters. $\mathbf{w}_j$ is the process noise, i.e., the zero-mean Gaussian vector representing the frame-to-frame update of the model parameters, and $\mathbf{v}_j$ is the measurement noise. The corresponding covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ are assumed to be diagonal, $\mathbf{Q}$ (process noise) being fixed and $\mathbf{R}$ (measurement noise) depending on the matching results in the 2D tracking. See [2] for details.

The estimate $\hat{\mathbf{x}}_j$ is computed using an extended Kalman filter.

3.3. Avoiding drift by two-stage tracking

The image-based tracking scheme described so far suffers from the long sequence tracking problem, or *drift*. Any small error in the tracking accumulates, eventually leading to tracking failure. In practice, this happens quite quickly – quickly enough for the tracker to be of any intresting use.

Ström [5] tackle the problem by creating a model-based tracker, where the patches used in the 2D tracking are not extracted from the previous image but from a renderred image of the predicted face model. Mapping a reference frame (typically the first frame of the sequence) onto the model, the model parameters in the current frame are predicted by the extended Kalman filter. The face model can thus be rendered at the predicted pose, and patches be extracted from that rendered image.

In our experiments, it turned out that sudden tracking failures where much more common using the model-based approach. In order to combine the advantages of image and model-based tracking, we propose the following scheme:

1. Run the image-based tracking and estimate the model parameters.

2. Render the model using the estimated parameters and the texture from the reference frame.

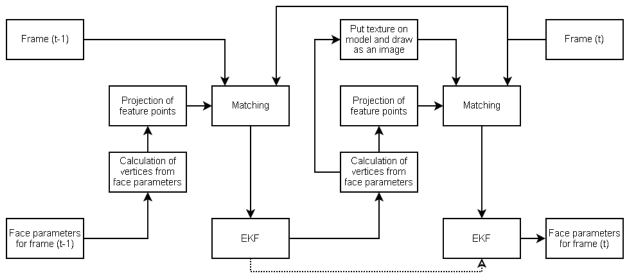
Frame (t-1)
Put texture on model and draw as an image
Frame (t)
Projection of feature points
Matching
Projection of feature points
Matching
Calculation of vertices from face parameters
Calculation of vertices from face parameters
Face parameters for frame (t-1)
EKF
EKF
Face parameters for frame (t)

Fig. 3. The tracking process.

3. Extract new patches, perform the 2D tracking once more with smaller search windows, and run the extended Kalman filter again.

An illustration of the tracking scheme is shown in Fig. 3

4. IMPLEMENTATION AND RESULTS

Since we aim at real-time performance, implementation issues are important. Another drivgin factor has been the time limit of the project, which means the existin libraries should be used as much as possible. We have used the the OpenCV library [3] for extended Kalman filtering and 2D tracking, and OpenGL for the rendering.

The performance of the tracker was tested with different values on parameters such as size of the searh range in the template mathcing, number of features to track, etc.. For example, tracking 50 features in a video stream of 320×240 pixels using search windows of 19×19 pixels resulted in a tracking frame rate of 21 frames per second. These results were obtained on computer with a 2GHz AMD Athlon 64 processor and an integrated ATI Radeon Xpress 200 graphics card.

An example of feature point selection is shown in Fig. 4 and some tracking results are shown in Fig. 5.

5. CONCLUSION

With the availability of much more computational power in consumer hardware than in 2002, the tracking scheme proposed by Ström can now be extended to include tracking of deformable models, which was quite unrealistic at the time. Faster computers does thus now only result in faster solution to the same problems, but new problems can be solved as well. Additionally, we propose a combined image and model-based tracker to get a robust tracker without drift.

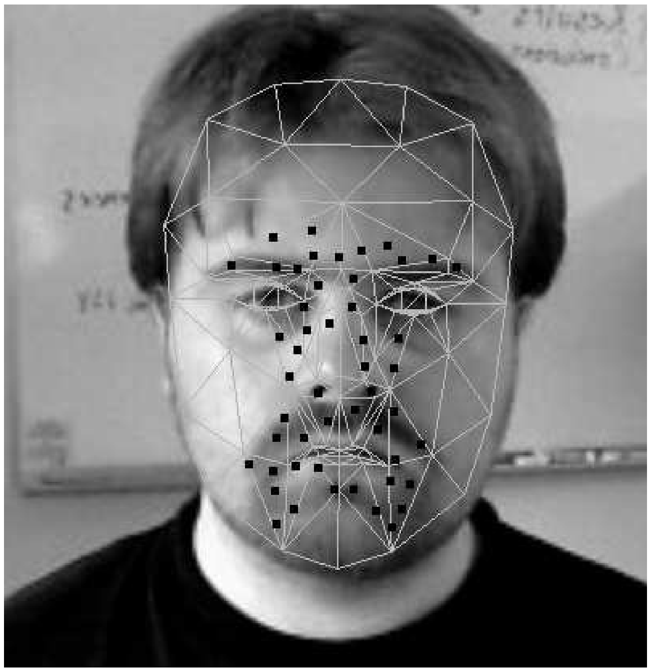

Fig. 4. 50 automatically selected feature points.

6. REFERENCES

[1] J. Ahlberg, *Candide3 – an updated parameterized face*, Tech. Rep. LiTH-ISY-R-2326, Linköping University, Sweden, 2001.

[2] N. Ingemars, *A feature-based face tracker using extended Kalman filtering*, M.Sc thesis, Linköping University, Sweden, 2007.

[3] http://www.intel.com/technology/computing/opencv/

[4] M. Rydfalk, *Candide, a parameterized face*, Tech. Rep. LiTH-ISY-I-866, Linköping University, Sweden, 1987.

[5] J. Ström, "Model-based real-time head tracking," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 10, pp. 1039–1052, 2002.

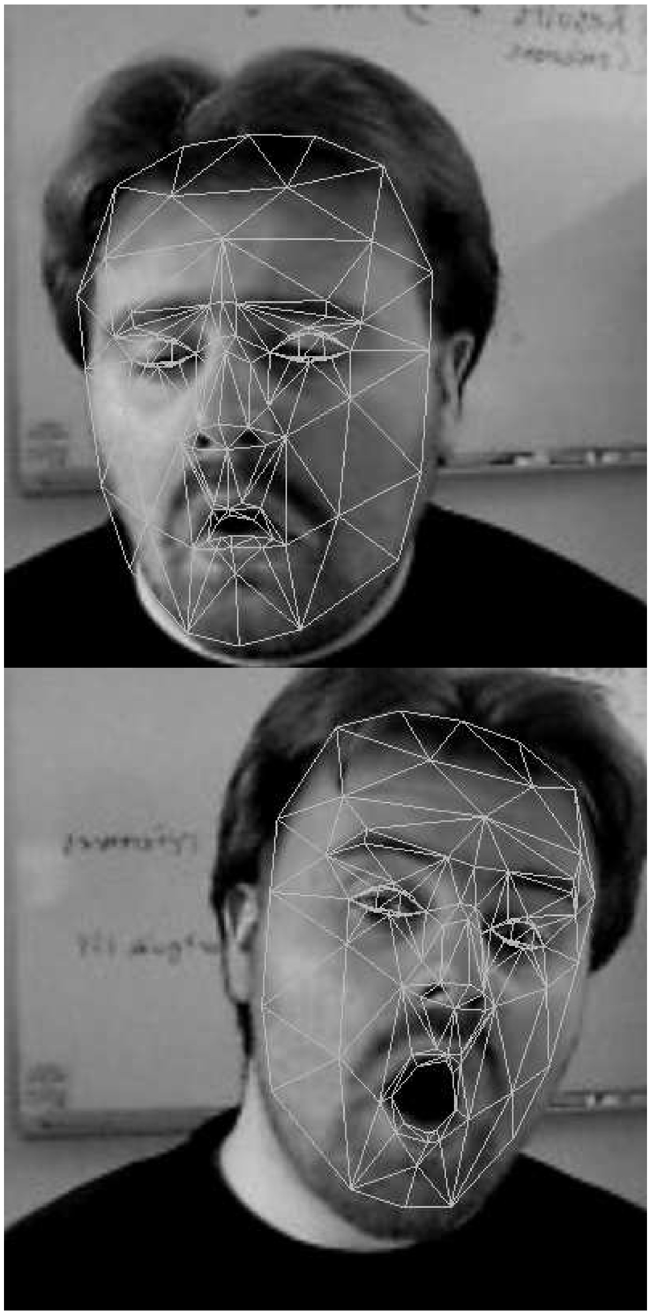

Fig. 5. Tracking examples.