M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 506
Appears in: Proceedings of the Modelling People Workshop at ICCV'99

Real Time Tracking and Modeling of Faces: An EKF-based Analysis by Synthesis Approach

J. Ström*, T. Jebara, S. Basu and A. Pentland

email: {jacob, jebara, sbasu, sandy}@media.mit.edu

Abstract

A real-time system for tracking and modeling of faces using an analysis-by-synthesis approach is presented. A 3D face model is texture-mapped with a head-on view of the face. Feature points in the face-texture are then selected based on image Hessians. The selected points of the rendered image are tracked in the incoming video using normalized correlation. The result is fed into an extended Kalman filter to recover camera geometry, head pose, and structure from motion. This information is used to rigidly move the face model to render the next image needed for tracking. Every point is tracked from the Kalman filter's estimated position. The variance of each measurement is estimated using a number of factors, including the residual error and the angle between the surface normal and the camera. The estimated head pose can be used to warp the face in the incoming video back to frontal position, and parts of the image can then be subject to eigenspace coding for efficient transmission. The mouth texture is transmitted in this way using 50 bits per frame plus overhead from the person specific eigenspace. The face tracking system runs at 30 Hz, coding the mouth texture slows it down to 12 Hz.

key words: face tracking, modeling, real-time, EKF, analysis by synthesis

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 goal of this paper is to describe a real-time system for simultaneous tracking and 3D modeling. The core idea is to select a dense set of feature points (essentially, optical flow at all the most information bearing points), thus extending the foundations of Azarbayejani and Pentland [3] and Jebara and Pentland [8] to achieve robust performance. Figure 1 illustrates how the system works. Patches around the featurepoints taken from the rendered 3D model (lower left corner) are matched against the incoming video, and the 2D trajectories of these featurepoints are then fed through a structure from motion (SfM) algorithm to update the pose information of the 3D model. In addition, the estimated structure serve as a starting point for the tracking. The 3D model helps in several ways. First, it compensates for rotation and scale, making it possible to use fast 2D block-matching to track the feature points. Second, it helps in gauging how reliably a certain feature point can be tracked. For instance, a feature point close to the model's edge is hard to track and a feature point on the back side of the model cannot be tracked at all. The estimate of the reliability of a point can then be forwarded to the SfM Kalman filter as a variance of the 2D block-matching measurement. Third, the model can be used to normalize the pose of the incoming image, making it possible to use an eigen-representation on the parts of the image. An important fourth aspect is the possibility to incrementally update the 3D structure and

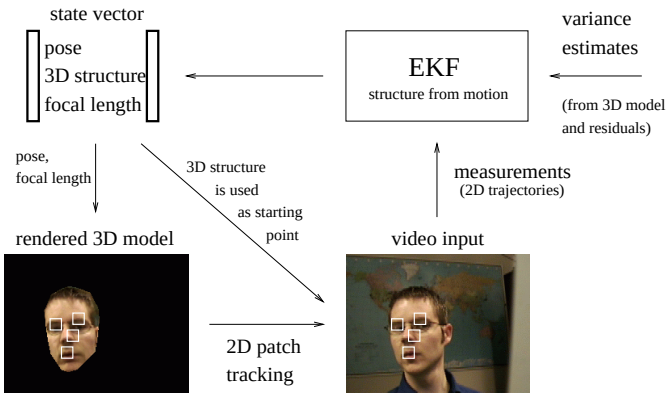

Figure 1: Patches from the rendered image (lower left corner) are matched with the incoming video. The 2D feature point trajectories are fed through the SfM extended Kalman filter that estimates the pose information needed to render the next model view. For clarity, only four of the twelve patches are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

*J. Ström is currently with the Dept. of Electrical Engineering at Linköping University, SE-581 LINKÖPING, Sweden.

the texture of the model until a full 3D textured head model is recovered. Note that in this formulation, the model is dynamically varying, not static. By tracking features on the side of the head during large out-of-plane rotations, this can potentially make the system a lot more robust.

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. In this work, the 3D structure estimate is used as a starting point for the 2D block-matching. However, the idea is that it should be used to regress the 3D model to better match the individual head structure.

1.1 Previous Work

The SfM algorithm used in this work is taken from the work of Azarbayejani and Pentland [3]. One of the applications presented therein is a head tracker that recovers the focal length of the camera as well as the motion and the structure of six feature points.

Our work is closely related to the real-time system developed by Jebara and Pentland [8], where 2D correlation trackers are used to feed the SfM filter. As in this paper, the estimated 3D structure provides a starting point for the 2D-correlation trackers, and the residual error is used to estimate the variances of the measurements. Jebara and Pentland's work differs from the one presented here in two important ways. First, the system in [8] tries to find the same points to track in each face (namely the eyes and the corners of the mouth and nose) whereas our system tries to pick the best points to track for each face. Second, Jebara and Pentland's system works by tracking 2D patches in 3D by deforming them affinely. Occlusion are treated as a high-noise measurement, since no information about geometric occlusion exists. In contrast, our work relies on a 3D model that is directly compared to the input image.

Many authors in the model-based coding area estimate the motion parameters of a 3D polygon head model using the optical flow between the synthesized image and the input video [2][15][19]. A simliar approach is presented in [4] where differential optical flow is used to rigidly move an ellipsoidal head model. However, the above-mentioned methods need a calibrated focal length in order to work. Another drawback is that they, in order to gain robustness, gather measurements from a large number of points. This makes it hard to reach real-time performance. Also, since optical flow can be measured accurately only in regions which contain edges, it seems wasteful to include optical flow from edge-free regions into the calculation. This has inspired us to carefully select only the points where we believe there is edge information enough to allow tracking.

1.2 Overview

The next section will discuss the initialization of the system, and how the feature points are selected. In Section 3, the 2D correlation tracking is described, and the structure from motion filter is briefly discussed in Section 4. The coding of the mouth region is described in Section 5. Results are presented in Section 6, and Section 7 discusses how we plan to make full use of this framework in the immediate future.

2 Initialization

The system is initialized from a frontal position as seen in Figure 2. A generic polygonal face model[1] [16] (left) is aligned to match with a head-on view of the face in the video sequence (middle). The pixels from

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.

the video are then texture-mapped onto the model (right). Jebara and Pentland [8] present a fully automatic scheme to align a 3D model with a face in the video input. Our system uses part of their scheme to permit a semi-automatic alignment; the biggest skin-colored blob in the image is found and the 3D model is aligned to it. The user has to make sure that she/he is in a frontal position before initiating tracking.

After alignment has been performed, the system selects which feature points to use. The part of the video input containing the face is cropped out (first image of Figure 3) and further processed. The reason why

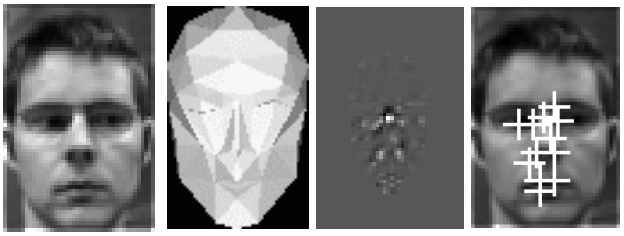

Figure 3: From left: The lowpass filtered incoming video, the weighting (the cosine of the angle between the surface normal and the camera direction), the final rating and the extracted feature points.

the video data is used and not the rendered 3D-model is that the discontinuities at the edges of the model will complicate the calculation of the image gradients. The cropped image is lowpass filtered and subsampled once to avoid locking on to features that are too vague

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

to be reliably tracked. The determinant of the Hessian $\frac{1}{4}\begin{vmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{vmatrix}$ is calculated in every point. To avoid selecting points on parts of the face surface perpendicular to the camera, the determinant is weighted with the cosine of the angle between the surface normal and the camera direction. These values can easily be obtained from the computer graphics hardware by rendering a grayshade version of the 3D model with lighting from the camera direction (Figure 3, second image). The resulting rating of each pixel is shown in the third image in Figure 3, where brighter pixels indicate a higher score. The twelve points with the highest ratings are then selected with a minimum-distance constraint between points. In detail, a sorted linked list with the 200 highest-rating pixels are created. The highest scoring pixel is chosen as a feature point, and all elements in the list closer to this point than the minimal distance are removed. This process is repeated twelve times. The fourth image in Figure 3 shows the twelve points selected. Each point is given a 3D position on the surface of the 3D model. Again, the computer graphics hardware can be used, this time reading out the value of the depthbuffer of the rendered image in the corresponding pixel location, to calculate the depth of the feature point.

3 Tracking

As shown in Figure 1, the tracking is carried out between the rendered frame and the video input. Both images (or rather, the relevant parts of them) 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 normalized correlation. More specifically, if $\mathbf{a}$ and $\mathbf{b}$ are the vectors obtained by raster-scanning the patch in the rendered image and the video input respectively, then the $\hat{\mathbf{b}}$ that maximizes

$$\rho = \cos(\theta) = \frac{\hat{\mathbf{a}} \cdot \hat{\mathbf{b}}}{||\hat{\mathbf{a}}||||\hat{\mathbf{b}}||} \qquad (1)$$

is selected, where $\hat{\mathbf{a}} = \mathbf{a} - \mu_{\mathbf{a}}$, $\hat{\mathbf{b}} = \mathbf{b} - \mu_{\mathbf{b}}$, and $\|\cdot\|$ represents the norm. The search window is centered around the position that is estimated from the structure from motion algorithm. An exhaustive search is carried out in the search window and the candidate with the lowest error is selected.

3.1 Subpixel Refinement

Since the two images were subsampled before matching, the accuracy of the tracking is only ± 1 pixel. However, since an exhaustive search was performed over the search window, the error is known in the adjacent positions. By approximating error derivatives with central differences, the error surface is approximated by a second degree Taylor polynomial. The (sub-pixel) location of the minimum of the resulting paraboloid is then used as the feature location. If the error surface is very irregular however, the minimum of the paraboloid can be outside the 2×2 pixel area. In this case the Taylor polynomial is a bad approximation of the error surface and the centroid of the 2×2 pixel area is used instead. Since the rough error surface might indicate that the point is off track, the variance for such a point is multiplied by 2 in the implementation.

3.2 Variance Estimation

The variance of a point can be made dependent on a number of different cues. For instance, if a feature point is occluded in the rendered image, we want the Kalman filter to discard that measurement. This can be done by setting the variance for that feature point to a high value in that time step. In the implementation occlusion is detected and corresponding feature points get a variance of 10000.

In addition, the variance is a function of the angle φ between the surface normal of the feature point and the camera direction. A small angle corresponds to an easily measurable head-on shot and should generate a low variance, whereas larger angles should map to higher variance. We scale the variance with $20([1 - \cos(\varphi)]^2 + 0.2)$.

Moreover, following [8], the variance is also influenced by the error from the normalized correlation. The idea is that a large error (small ρ) is produced when a feature point is far off the correct position, and thus the variance should increase. There are potential problems with this approach, e.g., an almost flat patch in a similar surrounding can move substantially without the error increasing much. Conversely, a rugged patch can produce a large error even if only moved slightly. However, since the feature points were selected on the basis of large Hessians, there is reason to believe that these types of problems should not occur. In order to estimate the mapping from ρ to the variance, a number of test images depicting faces were used. Feature points were extracted the same way as described in Section 2. Patches around these feature points were then translated and the normalized correlation coefficient ρ was measured. For each distance, the mean ρ was then calculated, resulting in the solid curve in Figure 4. In the implementation, the variance is made proportional to $-4\rho + 4.2$ which is the dashed curve in the same graph.

Finally, it is possible to use the shape of the Taylor paraboloid mentioned in Section 3.1 to estimate the covariances between the x and y measurements. For instance, a vertical line segment (such as the mouth) yields a very accurate measurement in the y direction, but a poor measurement in the x direction. The corresponding Taylor paraboloid $\frac{1}{2}\mathbf{x}^T H\mathbf{x} + C\mathbf{x} + D$ of the error surface will then be elongated in the x direction. On this error paraboloid, the error will be constant on certain ellipses. By choosing the covariance matrix $\Sigma = H^{-1}$ these ellipses will be iso-probability contours for a Gaussian. In the implementation we estimate the final covariance matrix as $I\sigma^2 + \Sigma$ where σ^2 is the variance estimated using the above-mentioned methods.

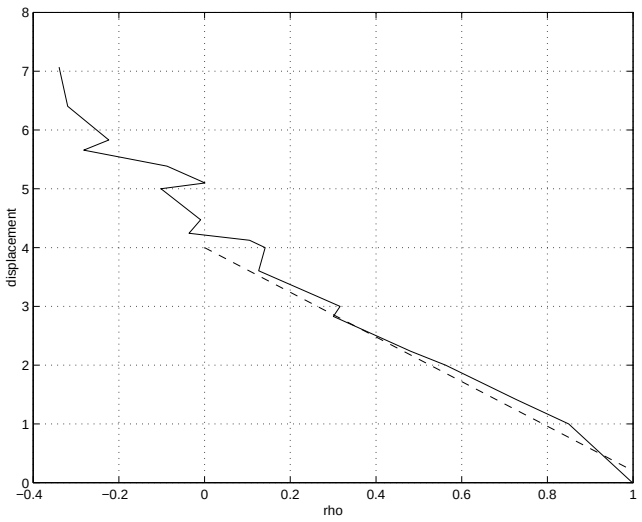

Figure 4: The solid curve is the mean ρ for a certain perturbation distance over a set of testing feature points. The dotted line shows the linear function used in the implementation to map ρ to the variance.

4 Structure from Motion

Recently, structure from motion has been reformulated into a stable recursive estimation problem and been shown to converge reliably [3]. By remapping the data into a new parameterized representation, what was essentially an under-constrained problem becomes uniquely solvable with no numerical "ill-conditioning".

4.1 Stable Representation for Recursive Estimation

The objective of SfM is to recover 3D structure, motion and camera geometry. These form the "internal state vector," $\mathbf{x}$, of the system under observation. These internal states are to be recovered by observation measurements of the system. For a thorough justification of the internal state vector representation, consult [3]. One internal state parameter is the camera geometry. Instead of trying to estimate focal length to describe the camera, we estimate $\beta = \frac{1}{f}$. The structure of points on the 3D object is represented with one parameter per point instead of an XYZ spatial location. The mapping from this 3 Cartesian form to one parameter is described in Equation 2 where α is the new representation of structure and u and v are the coordinates of the point in the image plane when tracking is initialized.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (1+\alpha\beta)u \\ (1+\alpha\beta)v \\ \alpha \end{bmatrix} \tag{2}$$

The relation is illustrated in Figure 5. In addition, we define translation as $(t_X, t_Y, t_Z\beta)$. Rotation is defined in terms of $(\omega_X, \omega_Y, \omega_Z)$ which are the incremental Euler angles for the interframe rotation. This representation of rotation overcomes the normality constraints of the quaternion representation by linearizing

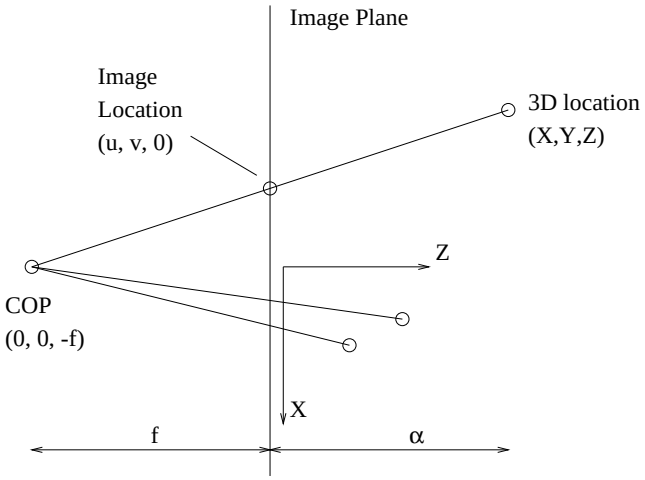

Figure 5: Model of central projection; the coordinate center is placed at the image plane rather than at the center of projection.

with a tangent hyper-plane on the unit hyper-sphere formed by the quaternion representation.

The final representation of the internal state vector has a total of $7+N$ parameters where N is the number of feature points being tracked (each of which requires a scalar depth value to determine 3D structure):

$$\mathbf{x} = (t_X, t_Y, t_Z\beta, \omega_X, \omega_Y, \omega_Z, \beta, \alpha_1, \alpha_2, \ldots, \alpha_N) \tag{3}$$

At each time step, we also have a measurement or observation vector $\mathbf{y}$ of size $2N$ with the following form:

$$\mathbf{y} = (X_1, Y_1, X_2, Y_2, \ldots, X_N, Y_N) \tag{4}$$

where (X_i, Y_i) are the positions of a feature point currently being tracked in the image. Unlike other formulations which are underdetermined at every time step, the above parametrization of the SfM problem is well-posed when $2N \geq 7 + N$ or when $N \geq 7$. Thus, if 7 or more feature points are being tracked in 2D simultaneously, a unique, well-constrained solution can be found for the internal state and a recursive filter can be employed.

4.2 Entering the Variance Estimates

Following [8], recall that Kalman filtering uses a noise covariance matrix to describe the expected noise on input measurements. Traditionally, the noise covariance matrix is denoted R and is $n \times n$ where n is the number of measurements in the observation vector $\mathbf{y}$. The role of R in the computation of the Kalman gain matrix is described by Equation 5. Adaptive Kalman filtering [6] proposes the use of a dynamically varying R matrix that changes with the arrival of new observation vectors to model the confidence of the new data. By changing R using the techniques described in Section 3.2, we can assign a weight on the observations and end up with a more robust overall estimate of internal state. The update equation for the Kalman gain is:

$$K = P^{-} H^T [H P^{-} H^T + R]^{-1} \tag{5}$$

5 Coding

Principal Component Analysis (PCA), or eigenspace analysis, has proved to provide a powerful tool for analysis and representation of face- as well as lip images [17] [10] [18] [12] [5]. Moghaddam and Pentland has demonstrated a model-based coding scheme for head-on images [11].

In our implementation we have chosen to include a "zeroth-order image coder", that reconstructs the tracked face using the head pose information, the texture of the first image plus the real-time texture around the mouth region. The idea is to show a low bit rate, model-based coder operating in real time (12 Hz).

Using the 3D head pose information from the tracker, it is possible to warp back the face to a frontal position. Figure 6 shows an example of this: The original image (a) is tracked (b), and it's texture is then

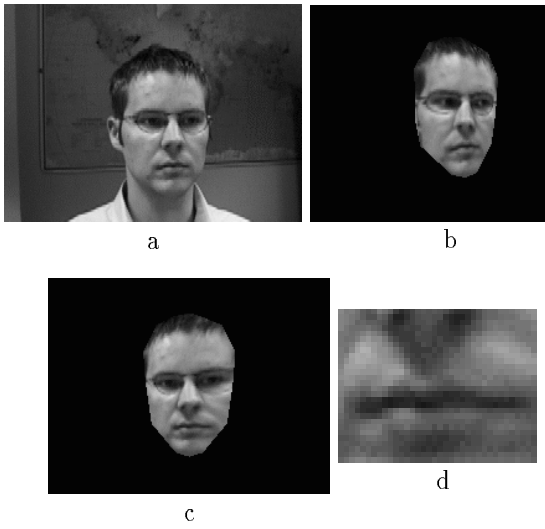

Figure 6: The original image (a) is tracked (b), and the pose information is used to warp the image to a head-on shot (c). From this image, a 36×28 pixel mouth-chip is cropped out (d).

warped back to a frontal pose (c). From this pose-normalized image it is reasonable to do projection onto an eigenspace.

A 36×28 pixel mouth chip is cropped out from the warped-back incoming video (Figure 6d). Since the human vision system is more sensitive to differences in luminance than in chrominance, the mouth chip is converted from RGB to YCrCb; the chrominance components Cr and Cb are subsampled by a factor of two, thus diminishing their impact on the eigenspace analysis. The mouth chip is then encoded using an eigenspace constructed from mouth-chip images obtained the same way. The ensemble mean (first image in Figure 7) is subtracted from the image and the residual is projected onto the eigenvectors (the first six are shown next to the mean image in Figure 7).

Figure 7: The mean image (extreme left) followed by the first six eigenimages.

Each coefficient is then quantized by dividing it by its standard deviation (the square-root of the corresponding eigenvalue) and encoded using a Lloyd-Max quantizer for a Gaussian source. The number of bits used to quantize a coefficient increases with the size of the corresponding eigenvalue. Twelve coefficients are used, and a total of 50 bits is used in the quantization.

The eigenspace is trained on images from the same person, but from a different sequence. The eigenvectors and eigenvalues must thus be transmitted to the decoder in order for it to be able to reconstruct the images. By quantizing the eigenvectors to 8 bits and compressing them losslessly with gzip, this data can be sent at around 17000 bytes. The first texture must also be sent (e.g. using JPEG) adding another 7000 bytes to the startup cost of a call. Each additional frame requires a mere 98 bits of information; 50 bits for the eigencoefficients and 48 bits of motion information (each degree of freedom quantized to 8 bits). A five minute conversation at 12 Hz would thus result in a data rate of about 1.8 kbit/s. Figure 8 shows examples of the mouth-chip coded this way.

Figure 8: Top row: original images. Bottom row: coded with 50 bits per mouth.

6 Results

The tracking system was implemented in real time on a SGI O2 workstation. The feature point finding algorithm (executed once at the start of the tracking process) took about 100 ms, and the rest of the tracking was running at 30 Hz. In Figure 9 a few frames from a longer sequence (`track.mpg`) are shown. The two side views (first and third from the top) show approximately how much out-of-plane rotation the system can currently cope with without loosing track (about $\pm 20^o$). The time duration of the sequence is about 12 seconds.

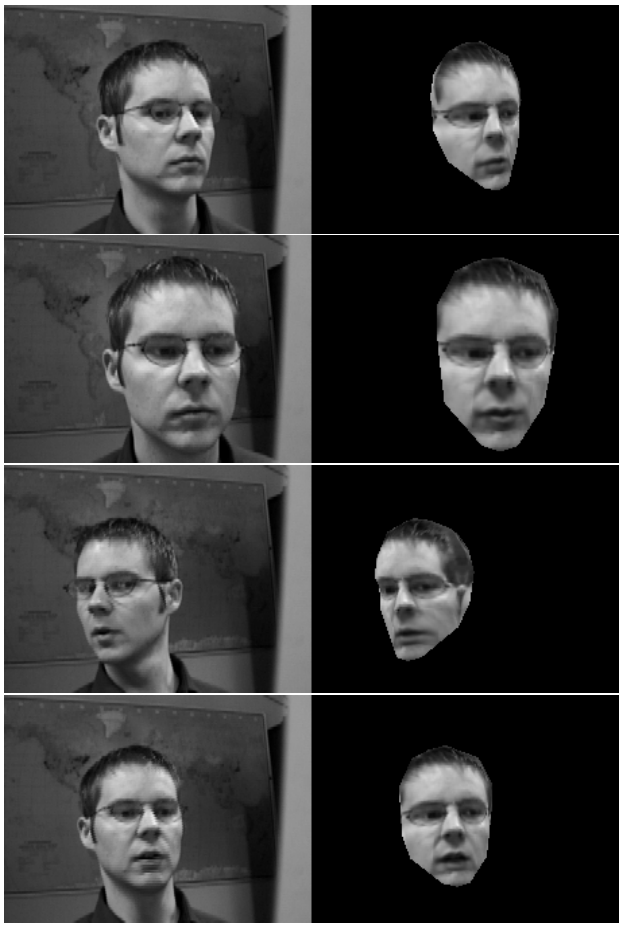

Figure 9: Left column: frame number 35, 121, 164 and 184 of a sequence. Right column: tracked and coded result.

The system is robust to occlusion of a small number of the feature points by, e.g., a finger. These points will have a much large error and will thus be disregarded by the extended Kalman filter.

One problem was that the lighting of the mouth patch (taken from the live video and then coded) differed from the lighting in the rest of the face (always represented by the static texture taken from the first head-on image) when the head was turned. This was solved by sampling the training mouth-patches from head-on views only. Thus the eigenspace could not span different lighting variations and the mouth-patch would always be consistent with the rest of the face. A more elegant solution to this would be to let the mouth-chip keep its correct lighting and change the lighting of the static texture according to some lighting model.

The resulting bit rate of 1.8 kbit/s (1.2 kbit/s if disregarding the initial transmission of the database and the first image) is comparable to the MPEG-4 facial animation standard bit stream (0.5 - 1 kbit/s) [1] [7] [13] [14]. However, some of the parameter compression schemes used in the MPEG-4 standard could be applied here as well, for instance arithmetic or temporal DCT coding of the motion parameters.

7 Future Directions

The work presented in this paper is work in progress and the list of things to add to the system is long. However, the most important additions lie on the modeling side. A first step would be to apply 3D-regression to the 3D structure estimates, thus obtaining a dense, 360^o 3D model. Jebara and Pentland estimated structure from texture using an eigenspace built from 3D cyberware scans [9], and the same methodology can be used here to estimate overall structure from pointwise structure. This generic 3D model can then be updated on-line and allow for more accurate tracking, and allow a dynamic estimate of the structure.

The next big step is to adaptively acquire texture from other points of view than head-on, and extracting new feature points in these areas. This can lead to significantly improved performance for large out-of-plane rotations.

8 Acknowledgements

Thanks to Fulbright Commission, Hans Werthen Stipendium, Telefonaktiebolaget LM Ericssons stiftelse för främjande av Elektroteknisk forskning, Sixten Gemzeus Stipendium, Stiftelsen Blanceflor Boncompagni-Ludovisi, född Bildt, Ernhold Lundströms stiftelse and Ångpanneföreningens Forskningsstiftelse.

References

[1] J. Ahlberg, H. Li, "Representing and Compressing Facial Animation Parameters Using Facial Action Basis Functions," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 93, no. 3, pp. 405-410, April 1999.

[2] K. Aizawa, H. Harashima, and T. Saito, "Model-based synthesis image coding (MBASIC) system for a person's face," *Signal Processing: Image Communication* vol. 1. no. 2, pp. 139-152, Oct. 1989.

[3] 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.

[4] S. Basu, I. Essa and Alex Pentland, "Motion Regularization for Model-Based Head Tracking," *Proceedings of 13th Int'l. Conf. on Pattern Recognition*, vol. C, pp. 611-616, Aug. 1996.

[5] C. Bregler and Y. Konig, "Eigenlips for Robust Speech Recognition," *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Adelaide, Australia, 1994.

[6] A. Gelb, *Applied Optimal Estimation,* M.I.T. Press, 1996.

[7] P. Eisert and B. Girod, `http://www-nt.e-technik.uni-erlangen.de/~eisert/publications.html`

[8] 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)

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

[10] K. Mase and A. Pentland, "LIP READING: Automatic Visual Recognition of Spoken Words," Proc. Image Understanding and Machine Vision, Optical Society of America, June 1989.

[11] B. Moghaddam and A. Pentland, "An automatic system for model-based coding of faces," *IEEE Data Compression Conference*, Snowbird, Utah, March 1995.

[12] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696 - 710, July 1997.

[13] MPEG Working Group on Visual *Coding of Audio-Visual Objects: Final Draft of International Standard, Part 2 (Visual)* ISO 14496-1, doc. no. N2501 of JTC1/SC29/WG11 Atlantic City meeting, Oct. 1998.

[14] MPEG Working Group on Visual *Visual Working Draft, Version 2 Rev 5.0* ISO 14496-2, doc. no. N2473 of JTC1/SC29/WG11 Atlantic City meeting, Oct. 1998.

[15] P. Roivainen, "Motion estimation in model-based coding of human faces," Licentiate Thesis LIU-TEK-LIC-1990:25, ISY, Linköping Univ., Sweden, 1990.

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

[17] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," J. Opt. Soc. Am., vol. 4, no. 3, pp. 519-524, March 1987.

[18] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, MIT 1991.

[19] W.J. Welsh, *Model-based coding of images*, Ph. D. Dissertation, University of Essex, U.K., Jan. 1991.