IEEE Computer Graphics & Applications, pp. 70-78, September 1998.

Analyzing Facial Expressions for Virtual Conferencing

Peter Eisert and Bernd Girod

Telecommunications Laboratory, University of Erlangen,
Cauerstrasse 7, D-91058 Erlangen, Germany
Email: {eisert, girod}@nt.e-technik.uni-erlangen.de

Keywords: model-based video coding, 3-D head model, MPEG-4, facial expression analysis, virtual conferencing.

Abstract

In this paper we present a method for the estimation of three-dimensional motion from 2-D image sequences showing head and shoulder scenes typical for video telephone and tele-conferencing applications. We use a 3-D model that specifies the color and shape of the person in the video. Additionally, the model constrains the motion and deformation in the face to a set of facial expressions which are represented by the facial animation parameters (FAPs) defined by the MPEG-4 standard. Using this model, we obtain a description of both global and local 3-D head motion as a function of the unknown facial parameters. Combining the 3-D information with the optical flow constraint leads to a robust and linear algorithm that estimates the facial animation parameters from two successive frames with low computational complexity. To overcome the restriction of small object motion, which is common to optical flow based approaches, we use a multi-resolution framework. Experimental results on synthetic and real data confirm the applicability of the presented technique and show that image sequences of head and shoulder scenes can be encoded at bit-rates below 0.6 kbit/s.

Introduction

Video conferencing, tele-presence and tele-teaching are video coding applications that have gained considerable interest in the past years. The transmission of image sequences typically imposes high demands on storage, processing power, and bandwidth of networks. Common networks like POTS (plain old telephone system), ISDN, and many computer networks provide only low bit-rates and cannot handle the amount of data arising when transmitting video uncompressed. Therefore, encoding of the sequences is essential. Highly efficient coding schemes like H.263 and MPEG-1 or 2 have been developed, achieving compression factors of about 1:100 with acceptable quality.

The use of model-based coding techniques further improves compression while at the same time providing us with the possibility of interactive manipulation of the scene content [1, 2]. The main idea of model-based coding is to create a virtual 3-D world with distinct 3-D objects that are described by their shape and texture. In this world we only have to specify how the objects are moving and deforming. The shape of the objects has to be transmitted only once, and we can then describe the scene in all following frames with few parameters specifying the position and deformation of the objects. The 2-D image sequence can finally be reconstructed by rendering the virtual world using computer graphics techniques.

Model-Based Coding of Head and Shoulder Scenes

In model-based coding of image sequences, we need 3-D models for all objects in the scene. Therefore, we have to put some restrictions on the scene content. In this paper we focus on head and shoulder scenes, which are typical for video telephone and video conferencing applications. A person is sitting in front of the system and a single camera records video frames from the speaking person. The encoder analyzes the incoming frames and estimates the 3-D motion and the facial expressions of the person. We obtain a set of facial expression parameters that describe (together with the 3-D model) the current appearance of the person. Only a few parameters have to be encoded and transmitted, leading to very low data-rates, typically less than 1 kbit/s. At the decoder we use the parameters to deform our head model according to the person's facial expressions and approximate the original camera frame by simply rendering our 3-D model. The principal structure of the model-based coder is depicted in Figure 1.

Besides the very low data-rate that is required to transmit the video we have the advantage of easy manipulation and interaction with the scene content because of the segmentation into single 3-D objects. For example, the eye contact problem that occurs in conventional video telephone systems can easily be solved by positioning the virtual camera inside the monitor that displays the image of the person you are talk-

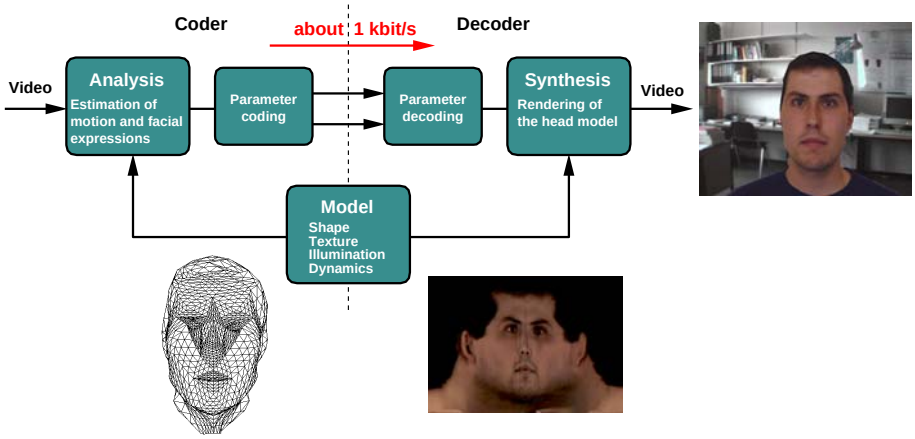

Figure 1: Basic structure of a model-based coder.

ing with. Or we are able to look around objects, if we move the synthetic camera according to the eye trajectory that is known by the coder. Another application for model-based coding is virtual conferencing [3, 4], which is an extension of the video telephone framework. Several people are connected to a network talking to each other. For each person there exists one 3-D model specifying his individual shape, color and motion characteristics. Facial animation parameters are estimated from the video frames and distributed to all participants. With the head models and their corresponding motion parameters, we are able to put all people at a table in a virtual conference room and to generate individual views for them.

In this paper we present a method for the estimation of facial animation parameters of a speaking person from two successive video frames. This algorithm serves as the basis for the applications mentioned above. First, the generic head model used for the description of the person's appearance is shown. This 3-D model also restricts the possible motions in the face simplifying the facial parameter estimation. We then propose our gradient-based linear motion estimator that works in a hierarchical way with low computational complexity. Finally, we validate the algorithm with experimental results for synthetic and real image sequences.

3-D Head Model

For the motion estimation we can use a simple virtual scene that consists of a head model and a camera model. If we want to generate synthetic views of a virtual conference, we have to combine several head models with other 3-D models of objects in the room during rendering, but for the estimation of facial animation parameters (FAPs), it is sufficient to model just one person.

The head model in the virtual world specifies the 3-D shape and color of a speaking person but also constrains the facial motion caused by the expressions. Like other well-known facial models [5, 6], our proposed model consists of a triangular mesh onto which texture is mapped to obtain a photo-realistic appearance. The shape of the surface is defined by second-order triangular B-splines [7] to facilitate the modeling of facial expressions.

Our face model is a generic 3-D model consisting of 101 triangular B-spline patches for the face and neck area and 36 rigid patches on the back of the head not influenced by facial expressions. Teeth and the interior of the mouth are separate 3-D objects. The patches' topology (Figure 2a) which is based on the Candide model [6], and the definition of the facial animation parameter units remain fixed for all persons. To adjust the model to an individual person only the texture and the position of the control points are changed using information from 3-D laser scans. To simplify the rendering, we approximate the spline surface of the adjusted model by a number of planar triangles, which results in a triangular mesh. An example of an individually adapted and subdivided mesh appears in Figure 2b.

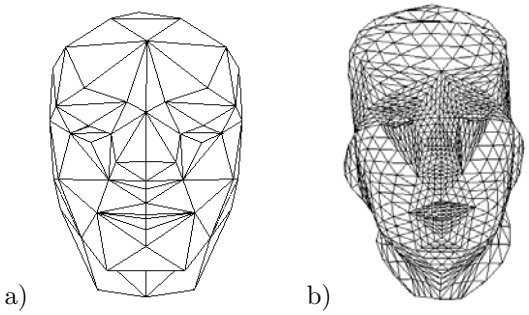

Figure 2: a) Topology of the triangular B-patches, b) Individually adapted triangular mesh used for rendering and motion estimation.

Triangular B-Spline-based Surface Modeling

The set of vertices that define the shape of our model has a very large number of degrees of freedom. Therefore, very complex objects can be modeled with sharp edges and discontinuities. The face of a person, however, has a smooth surface and facial expressions result in smooth movements of surface points due to the anatomical properties of tissue and muscles. These restrictions on curvature and motion can be modeled by splines which satisfy specific continuity constraints. It has been shown that B-splines are well-suited for the modeling of facial skin [8]. This type of spline exhibits some interesting properties useful for the implementation of the head model:

- Smoothness:
  A B-spline of order n is C^{n-1}-continuous. For our model we use second order splines leading to C^1-continuity of the surface.
- Local control:
  Movement of a single control point influences the surface just in a local neighborhood, which simplifies the modeling of facial expressions.
- Affine invariance:
  An affine transformation of the surface can be obtained by applying the same transformation on the control points. Facial movements are now defined by the transformation of a small number of control points instead of applying transformations on each vertex, which reduces the computational complexity.

Conventional B-splines or NURBS, which are commonly used in computer graphics, suffer from one well-known drawback. They are defined on a rectangular topology and therefore do not allow a local refinement in areas that are more curved. To overcome this restriction, we use *triangular B-splines* [7]. This spline scheme is based on triangular patches (shown in Figure 2a), which can easily be refined while still preserving the above mentioned properties of normal B-splines. For rendering we approximate the smooth spline and subdivide each patch into a number of planar triangles, leading to a mesh as illustrated in Figure 2b. The number of triangles can be varied to get either a good approximation or higher frame rate during rendering. The resulting triangular mesh is defined by a discrete set of vertices on the mathematically defined spline surface. We have to compute the B-spline basis functions [7] only at the discrete vertex positions on the surface. This can be done off-line. Once the basis functions are determined the position of the vertices $\mathbf{v}_j$ can be calculated by

$$\mathbf{v}_j = \sum_{i \in I_j} N_{ji} \mathbf{c}_i \qquad (1)$$

with N_{ji} being the precalculated basis functions of vertex j with

$$\sum_{i \in I_j} N_{ji} = 1, \qquad (2)$$

I_j the index set of vertex j and $\mathbf{c}_i$ the i-th control point. The index set I_j usually contains 3 to 6 indices for the control points that influence the position of the vertex.

Figure 3: Texture and depth information from the 3-D laser scanner.

To individualize the generic model, we use a 3-D laser scan of the person with a neutral expression. The scan provides us with information about color and depth as shown in Figure 3. The texture is mapped on the 3-D model, and the position of the control points is optimized to adapt the spline surface to the measured data. After the optimization, shape and texture coincide with the appearance of the person, and the model can now be deformed to show other expressions, rather than only the neutral one of the scan.

Modeling of Facial Expressions

For the estimation of facial animation parameters (FAPs) we must be able to animate our model to create different facial expressions. This task is simplified by the use of splines because they already constrain the motion of neighboring vertices. For the parameterization of the facial expressions the proposal of the MPEG-4 SNHC group [9] was adapted. According to that scheme, every facial expression can be generated by a superposition of 68 action units. These include both global motion, like head rotation, and local motion, like eye or mouth movement. 46 of these 68 FAPs are currently implemented.

To model the local movements, our generic face model contains a table describing how the control points of the mesh are translated or rotated for each FAP. This is done only for the small number of control points that are visible in Figure 4. For a given set of facial animation parameters that specifies an expression of a person, all positions of the control points are updated, and then the vertex locations are computed according to (1) by a linear combination of the control points. Examples of rendered expressions for different persons are depicted in Figure 5.

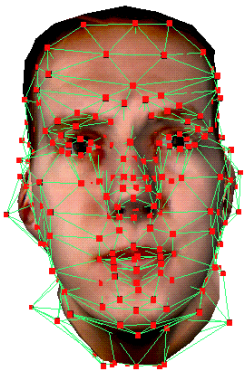

Figure 4: Control points of the shaped spline surface.

Figure 5: Different synthesized expressions.

Camera Model

The camera model describes the relation between our virtual 3-D world and the 2-D video frames and is used for both rendering and parameter estimation.

We use the perspective projection as shown in Figure 6, where the 3-D coordinates of an object point $[x\ y\ z]^T$ are projected into the image plane according to

$$\begin{aligned} X &= X_0 - f_x \frac{x}{z} \\ Y &= Y_0 - f_y \frac{y}{z}. \end{aligned} \tag{3}$$

Here, f_x and f_y denote the focal length multiplied by scaling factors in the x- and y-directions, respectively. These scaling factors transform the image coordinates into pixel coordinates X and Y. In addition, they allow the use of non-square pixel geometries. The two parameters X_0 and Y_0 describe the image center and its translation from the optical axis due to inaccurate placement of the CCD-sensor in the camera. For the

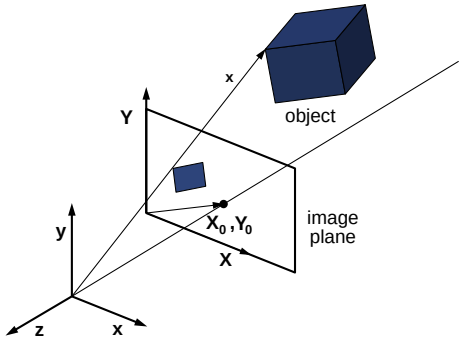

Figure 6: Camera model and its associated coordinate systems.

estimation of the FAPs, we have to model the real camera used for recording the video sequence in our synthetic world. The four parameters f_x, f_y, X_0 and Y_0 are therefore obtained from an initial camera calibration using Tsai's algorithm [10]. At the decoder we can then use arbitrary camera parameter settings if we do not want to reconstruct the original image exactly. This enables us to zoom into the scene if desired.

Motion Estimation and Facial Expression Analysis

Our motion estimation algorithm analyzes facial expressions of a speaking person by estimating changes of facial animation parameters (FAPs). We use the whole face image for the estimation, in contrast to feature-based approaches that exploit the information of discrete feature point correspondences. Estimating 3-D motion vectors from 2-D images is an ill-posed problem, and the errors of the estimates become very large if the position of the small number of features is not determined exactly. Therefore, we set up equations at each pixel location of the image leading to a large number of correspondences that are utilized for parameter estimation. In order to keep the computational complexity low we have developed a linear algorithm for the estimation that solves the large set of equations in a least-squares sense. The small errors that arise due to linearization are corrected using a feedback structure in a hierarchical framework.

We first describe the feedback loop to compensate the errors caused by linearization and to avoid error accumulation. The parametric 3-D motion description that constrains the possible motions of the face is shown in the next section. We then explain how we combine this spline-based motion constraint with the optical flow to achieve a robust facial parameter estimator. Because we obtain a highly over-determined system of equations, we are able to remove possible outliers. Finally, we describe the hierarchical structure of the algorithm that makes it possible to esti-

mate large changes in the facial animation parameters with low complexity.

Feedback Loop

The motion estimation algorithm presented in this paper estimates the facial animation parameters from two successive frames using the shape and texture information from the 3-D model. An easy way of doing this is to estimate the motion between two camera frames and move the 3-D head model according to these estimates in the virtual world. However, small errors in the estimation result in a wrong placement of the 3-D model whose shape information is necessary for the motion computation. In the next frame the facial parameters become even worse because the shape is not recovered exactly from the model. To avoid error accumulation in the long-term parameter estimation, a feedback loop [11] is used as depicted in Figure 7. The model of the head is moved accord-

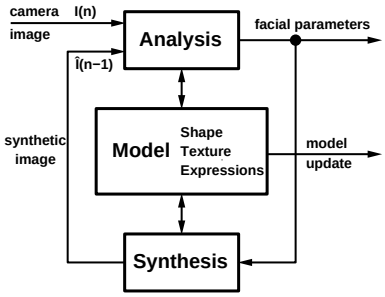

Figure 7: Feedback structure of the coder.

ing to the parameters estimated from two successive frames, and a synthetic image is generated by rendering the model with modified shape and position. The estimation is then performed between the actual camera frame $I(n)$ and the synthetic image of the previous frame $\hat{I}(n-1)$, which assures that no misalignment of the model occurs. An error in the estimated parameters again leads to a wrong placement of the model, but the error is removed during the estimation in the next frame because there is no mismatch between the 3-D model and the synthetically rendered image $\hat{I}(n-1)$.

3-D Motion Equation

Estimating 3-D motion of flexible bodies from 2-D images is a difficult task. However, the deformation of most objects can be constrained to simplify the estimation process. In our algorithm we use a parameterized head model whose local deformations are a function of the 68 unknown facial animation parameters. For each 3-D object point of the surface, we can set up one linear 3-D motion equation that determines how the point moves in the 3-D space if the facial animation parameters are changed. Instead of the 6 parameters specifying a rigid body motion, we now have 68 unknowns. Due to the large number of equations that we use for the parameter estimation we still have an over-determined linear system of equations that can be solved easily.

To set up the 3-D motion equation, we have to combine several transformations shown in Figure 8 to take into account the dependencies between the FAPs and the 3-D object points of the model. First, the control

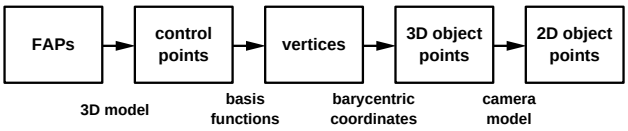

Figure 8: Transformation from FAPs to image points.

points of the surface are moved according to the given FAPs. Using the basis functions of the splines, the algorithm calculates the position of the vertices from the control points. Three vertices form a triangle and the 3-D motion of all object points inside this triangle specified by their barycentric coordinates is determined. The 2-D image point is finally obtained by projecting the 3-D point into the image plane. These transformations, that are all linear except for the projection, are incorporated into our parameter estimation algorithm.

The new control point position $\mathbf{c}_\mathbf{i}'$ can be determined from the position $\mathbf{c}_\mathbf{i}$ in the previous frame by

$$\mathbf{c}_i' = \mathbf{c}_i + \sum_k \Delta FAP_k \mathbf{d}_{ik} \tag{4}$$

where

$$\Delta FAP_k = FAP_k' - FAP_k \tag{5}$$

is the change of the facial animation parameter k between the two frames and $\mathbf{d}_{ik}$ the 3-D direction vector of the corresponding movement.

Strictly speaking, (4) is only valid for translations. If a number of control points are rotated around given axes by some action units, the description for the motion of control points becomes much more complicated due to the combination of rotation (defined by rotation matrices) and translation. The order of these operations can no longer be exchanged and the use of matrix multiplication results in a set of equations that is nonlinear in the parameters that have to be estimated. However, we can use the linear description (4) also for rotation, if we assume that the rotation angles between two successive frames are small. Then, the trajectory of a control point i that is rotating around the center O can be approximated by its tangent $\mathbf{d}_{ik}$ as shown in Figure 9. This tangent differs for all object points, but we have to set up equation (4) for all points individually anyhow because of local deformations in the surface.

For a rotation by the angle α_k

$$\alpha_k = \Delta FAP_k \cdot s_k, \tag{6}$$

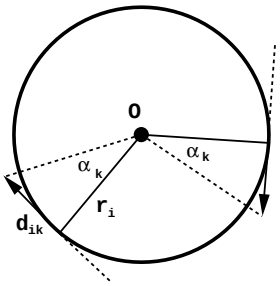

Figure 9: Approximation of rotations.

defined by the facial animation parameter changes ΔFAP_k and the corresponding scaling factor s_k the length of the translation vector $\mathbf{d}_{ik}$ can be calculated by

$$|\mathbf{d_{ik}}| = \Delta FAP_k \cdot r_i \cdot s_k. \tag{7}$$

Here, r_i is the distance between the object point and the given rotation axis. With this assumption (the direction of $\mathbf{d}_{ik}$ is specified by the direction of the tangent), (4) can also be used for rotation, leading to a simple linear description for all FAPs. Additionally, we can estimate both global and local motion simultaneously. The small error caused by the approximation is compensated after some iterations in the feedback structure shown in Figure 7.

Having modeled the shift in control points, the motion of the vertices of the triangular mesh can be determined using (1), and the local motion of an object point $\mathbf{x}$ is calculated from that using

$$\mathbf{x} = \sum_{m=0}^{2} \lambda_m \mathbf{v}_m = \sum_{j \in J} (\sum_{m=0}^{2} \lambda_m N_{mj}) \mathbf{c}_j, \tag{8}$$

where λ_m are the barycentric coordinates of the object point in the triangle that encloses that point. The motion equation for a surface point can be represented as

$$\mathbf{x}' = \mathbf{x} + \sum_k \Delta FAP_k \mathbf{t}_k = \mathbf{x} + \mathbf{T} \cdot \mathbf{\Delta FAP}, \tag{9}$$

where $\mathbf{t}_k$'s are the new direction vectors to the corresponding facial animation parameter calculated from $\mathbf{d}_k$ by applying the linear transforms (1) and (8). $\mathbf{T}$ combines all the vectors in a single matrix and $\mathbf{\Delta FAP}$ is the vector of all FAP changes. The matrix $\mathbf{T}$ can be derived from the 3-D model, and equation (9) then describes the change of the 3-D point location $\mathbf{x}' - \mathbf{x}$ as a linear function of FAP changes $\mathbf{\Delta FAP}$. Due to the presence of local deformations, we cannot describe the transformation with the same matrix $\mathbf{T}$ for all object points (as in the case for rigid body motion) but have to set $\mathbf{T}$ up for each point independently.

Motion Estimation

For motion estimation the whole face image is used by setting up the optical flow constraint equation

$$I_X \cdot u + I_Y \cdot v + I_t = 0 \tag{10}$$

where $[I_X\ I_Y]$ is the gradient of the intensity at point $[X\ Y]$, u and v the velocity in x- and y-direction and I_t the intensity gradient in the temporal direction. For grey level images I represents the luminance – using color images leads to three independent equations for the three color components.

Since u and v can take on arbitrary values for each point $[X\ Y]$, this set of equations is underdetermined, and we need additional constraints to compute a unique solution. Instead of determining the optical flow field by using smoothness constraints and then extracting the motion parameter set from this flow field, we estimate the facial animation parameters from (10) together with the 3-D motion equations of the head's object points [12, 13]. This technique is very similar to the one described in [14]. One main difference is that we estimate the motion from synthetic frames and camera images using a feedback loop as shown in Figure 7. This allows the usage of a hierarchical framework that can handle larger motion vectors between two successive frames. Beyond that, edge forces to avoid an error accumulation as described in [14] are not necessary with the rendering-feedback loop. Another difference between the two approaches is that we use a textured 3-D model, which allows us to generate new synthetic views of our virtual scene after having estimated the motion.

Writing equation (9) in its single components leads to

$$x' = x(1 + \frac{1}{x}\mathbf{t}_x \cdot \mathbf{\Delta FAP}) \tag{11}$$

$$y' = y(1 + \frac{1}{y}\mathbf{t}_y \cdot \mathbf{\Delta FAP}) \tag{12}$$

$$z' = z(1 + \frac{1}{z}\mathbf{t}_z \cdot \mathbf{\Delta FAP}). \tag{13}$$

$\mathbf{t}_x$, $\mathbf{t}_y$ and $\mathbf{t}_z$ are the row vectors of matrix $\mathbf{T}$. Dividing (11) and (12) by (13), inserting the camera model (3) and using a first order approximation leads to

$$u = X' - X \approx -\frac{1}{z}(f_x \mathbf{t}_x + (X - X_0)\mathbf{t}_z)\mathbf{\Delta FAP} \tag{14}$$

$$v = Y' - Y \approx -\frac{1}{z}(f_y \mathbf{t}_y + (Y - Y_0)\mathbf{t}_z)\mathbf{\Delta FAP}. \tag{15}$$

This equation serves as the motion constraint in the 2-D image plane. Together with (10) a linear equation at each pixel position can be set up

$$\frac{1}{z}\left[I_X f_x \mathbf{t}_x + I_Y f_y \mathbf{t}_y + (I_X(X - X_0) + I_Y(Y - Y_0))\mathbf{t}_z\right]\mathbf{\Delta FAP} = I_t \tag{16}$$

with z being the depth information coming from the model. We obtain an over-determined system that can be solved in a least-squares sense with low computational complexity. The size of the system depends directly on the number of FAPs.

Outlier Removal

At each pixel position of the object we can set up (16) leading to a large number of equations. We need at least the same number of equations as the number of estimated FAPs but due to the large number of face pixels, each of which contributes one additional equation, we can discard some possible outliers. These outliers can be detected by analyzing the partial derivatives of the intensity and the motion model. The optical flow constraint equation (10) is only valid for small displacement vectors because of the linearization of the intensity values. If the estimate of the displacement vector length for the pixel at position $[X\ Y]$

$$\hat{d}(X,Y) = \sqrt{\frac{I_t^2}{I_X^2 + I_Y^2}} \tag{17}$$

is larger than a threshold G, we classify the pixel as an outlier and do not use it for motion estimation.

Hierarchical Motion Estimation

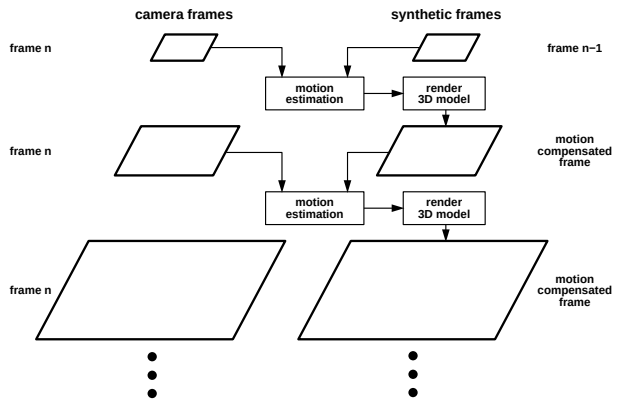

Figure 10: Image pyramid of the hierarchical motion estimation scheme.

The inherent linearization of the intensity in the optical flow constraint and the approximations used for obtaining a linear solution do not allow dealing with large displacement vectors between two successive video frames. To overcome this limitation, a hierarchical scheme is used for the motion estimation as shown in Figure 10. First, an approximation for the motion parameters between frame n and n-1 is computed from low-pass filtered and sub-sampled versions of the camera frame $I(n)$ and the synthetic frame $\hat{I}(n-1)$. Due to the use of low-resolution images, the linear intensity assumption is valid over a wider range. For sub-sampling, simple moving average filters are used to reduce aliasing. The resulting images are then further filtered by a Gauss filter to smooth the edges before estimating the motion. With the estimated parameter set a motion compensated image is generated by simply moving the 3-D model and rendering it at the new position. Due to the motion compensation, the differences between the new synthetic image and the camera frame decrease. Then, the procedure is repeated at higher resolutions, each time yielding a more and more accurate motion parameter set. Note, that the same camera frame is used in all levels, and that the synthetic image is iteratively changed from $\hat{I}(n-1)$ to $\hat{I}(n)$. In our current implementation we use three levels of resolution starting from 88 x 72 pixels. For each new level the resolution is doubled in both directions, leading to a final resolution of 352 x 288 pixels (CIF). At the same time the threshold G used to detect outliers for the motion estimation is reduced from 5 (first level) to 0.5 (highest resolution) which means that at higher levels more pixels are classified as outliers. Experiments with this hierarchical scheme showed that it is able to estimate displacements of up to 30 pixels between two frames.

Experimental Results

To validate the estimation algorithm, we perform experiments with synthetic and real video. The synthetic sequences are created by rendering the 3-D model with known facial animation parameters, which allows comparing the estimated FAPs with original ones. For real sequences recorded with a camera we have to individualize the generic 3-D model to the person in the video using a 3-D laser scan of that person. Unfortunately, we cannot determine the accuracy of the estimated FAP values for real image sequences, since we do not know their correct values. Therefore, we can only compare the reconstructed synthetic image with the original camera frame. The error measure that we use for that purpose is the PSNR (peak signal noise ratio) which is defined as

$$PSNR = 10 \log \left(\frac{255^2}{\frac{1}{N} \sum_{i=0}^{N-1} (I_{orig,i} - I_{synth,i})^2} \right). \tag{18}$$

In this equation $I_{orig,i}$ is the luminance component of pixel i of the original camera image with values from 0 to 255, and $I_{synth,i}$ is the corresponding value in the synthetic image. N denotes the number of pixels in the image.

Synthetic Sequence

In the first experiment we create a synthetic sequence (100 frames) with a resolution of 352 by 288 pixels (CIF resolution). This is done by rendering the 3-D model with well-defined facial animation parameters and a viewing angle of about 25^o. Ten facial animation parameters are estimated for every 5th frame using the proposed algorithm, and the results are compared to the correct values. Table 1 shows the relative error averaged over all estimates. The values in the table are measured relative to the maximum

of each value that corresponds to an extreme expression. In this experiment we used the facial animation parameters for global motion of the head (FAPs 48, 49, 50), opening of the jaw (FAP 3), movements of eyelids (FAPs 19, 20), eyebrows (FAPs 35, 36) and lip corners (FAPs 12, 13). The very accurate values

FAP	3	12	13	19	20
rel. error [%]	0.26	0.17	0.19	0.11	0.07
FAP	35	36	48	49	50
rel. error [%]	0.04	0.02	0.04	0.41	0.20

Table 1: Relative average error of the estimated FAPs in %.

for the estimated FAPs result also in an excellent reconstruction of the original frames. Because we use the same model for both creation of the sequence and for the estimation of the parameters, a nearly perfect reconstruction is achieved. This can be seen in Figure 11, where we plot the measured PSNR between the original and the synthetic image. Averaging the PSNR values that are computed only in the facial area and not for the background leads to value of about 70 dB. For comparison, the PSNR for the reconstructed sequence that has been generated using only global motion parameters (head translation and rotation) is also shown in the plot.

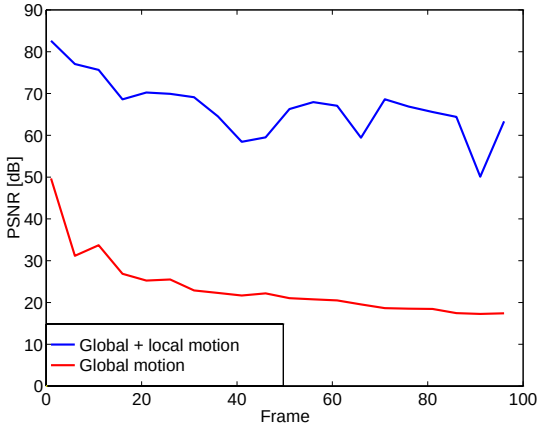

Figure 11: PSNR after global and global + local motion compensation.

Camera Sequences

In a second experiment a video of a talking person is recorded with a camera in CIF resolution and a frame rate of 12.5 Hz. The facial animation parameters are estimated for all 230 frames, and the corresponding synthetic sequence is rendered. We estimate 17 parameters including global head motion (6 parameters), movements of the eyebrows (4 parameters) and mouth motion (7 parameters). The total processing time for each frame is about 10.8 s on a 175 MHz O_2 SGI Workstation. 3.03 s of the time is spent for the motion estimation algorithm, including the setup of the equations and solving for the unknowns. The remaining time is mainly consumed by rendering the model and accessing files. Three frames of the camera sequence and the synthesized frames from the estimated parameters are shown in Figure 12.

Figure 12: Camera frames (left) and corresponding reconstructed synthetic frames (right).

Due to the use of real camera sequences, we have to measure the accuracy of the estimation by comparing the 2-D images from the camera and the renderer. The PSNR between original and synthetic images computed in the facial area is 31.6 dB in average. Figure 14 shows the PSNR for each frame in the case that 17 parameters are taken for the estimation (blue) or that only global motion estimation (6 parameters) is performed. Comparable results are also achieved for other sequences. In Figure 13 the original and synthetic views are given for a second sequence.

To estimate the bit-rate that is necessary to transmit a head and shoulder sequence over a network we use a Huffman coder to encode the facial animation parameters. The seventeen FAPs are predicted from the previous frames, the prediction error is quantized with 7 bits and then Huffman coded. This leads to an average bit-rate of about 0.58 kbit/s which corresponds to about 47 bits for each frame. In this calcu-

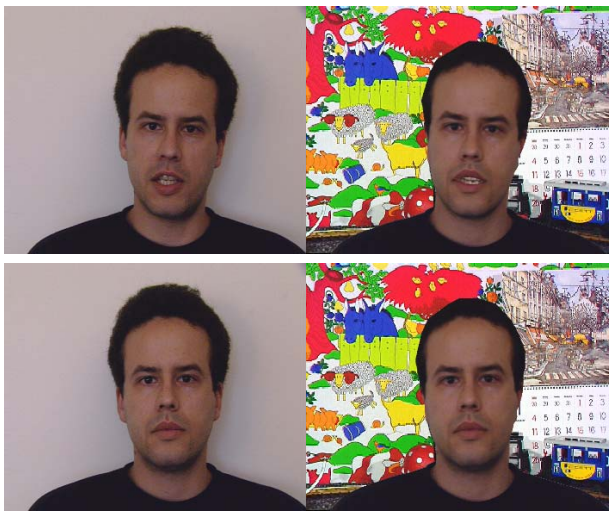

Figure 13: Camera frames (left) and corresponding reconstructed synthetic frames (right).

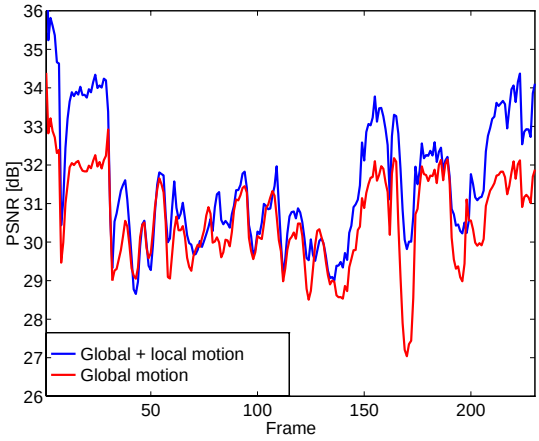

Figure 14: PSNR of a coded talking head sequence.

lation it is assumed, that the model is already known by the decoder.

Once we have estimated the facial animation parameters for our 3-D model, we can use them for the creation of virtual environments. Figure 15 shows an example for such a manipulation. Here, we have recorded a sequence of a speaking person (left side) and estimated the FAPs. These parameters were transmitted with an average bit-rate of about 0.57 kbit/s. The decoder then rendered the 3-D model together with a description of the background leading to images shown on the right side of Figure 15.

Conclusions

In this paper we have presented a method for the estimation of facial animation parameters from 2D image sequences. The combination of the optical flow constraint with 3D motion models lead to a robust

Figure 15: Camera frames (left) and corresponding reconstructed synthetic frames in a virtual environment(right).

and low complexity algorithm. Experiments with real video sequences showed that we can achieve bit-rates as low as 0.6 kbit/s at 31.6 dB PSNR.

Currently, we are investigating the influence of illumination variation that violates the optical flow constraint. To reduce this influence we add illumination models to our virtual scene and estimate both motion and illumination parameters like light direction and intensity [15]. Also, we are working on adding temporal 3D motion constraints to the estimation framework to further increase the robustness.

References

[1] D. E. Pearson, "Developments in model-based video coding," *Proc. IEEE*, 83(6), pp. 892-906, June 1995.

[2] W. J. Welsh, S. Searsby, and J. B. Waite, "Model-based image coding," *British Telecom Technology Journal*, 8(3), pp. 94-106, July 1990.

[3] K. Aizawa and T. S. Huang, "Model-Based Image Coding: Advanced Video Coding Techniques for Very Low Bit-Rate Applications," *Proc. IEEE*, 83(2), pp. 259-271, February 1995.

[4] I. S. Pandzic, T. K. Capin, N. Magnenat Thalmann, D. Thalmann, "Towards Natural Communication in Networked Collaborative Virtual Environments," *Proc. FIVE '96*, Pisa, 1996.

[5] F. I. Parke, "Parameterized Models for Facial Animation," *IEEE Computer Graphics and Applications*, pp. 61-68, 1982.

[6] M. Rydfalk, "CANDIDE: A Parameterized Face," *LiTH-ISY-I-0866 Linköping University*, October, 1987.

[7] G. Greiner, H.-P. Seidel, "Modeling with Triangular B-splines," *Proc. ACM/IEEE Solid Model-*

ing Symposium '93, pp.211-220, 1993.

[8] M. Hoch, G. Fleischmann and B. Girod, "Modeling and Animation of Facial Expressions based on B-Splines," *Visual Computer*, vol. 11, pp. 87-95, 1994.

[9] SNHC, "SNHC Systems Verification Model 4.0," *ISO/IEC JTC1/SC29/WG11 N1666*, Bristol, April 1997.

[10] R. Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, pp. 323-344, August 1987.

[11] R. Koch, "Dynamic 3-D Scene Analysis through Synthesis Feedback Control," *IEEE Trans. PAMI*, vol. 15, no. 6, pp. 556-568, June 1993.

[12] H. Li, P. Roivainen and R. Forchheimer, "3-D Motion Estimation in Model-Based Facial Image Coding," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 6, pp. 545-555, June 1993.

[13] J. Ostermann, "Analyse-Synthese-Codierung basierend auf dem Modell bewegter, dreidimensionaler Objekte," PhD thesis, VDI Reihe 10, Nr. 391, VDI-Verlag, 1995.

[14] D. DeCarlo and D. Metaxas, "The Integration of Optical Flow and Deformable Models with Applications to Human Face Shape and Motion Estimation," *Proceedings CVPR '96*, pp. 231-238, 1996.

[15] P. Eisert and B. Girod, "Model-based Coding of Facial Image Sequences at Varying Illumination Conditions," *Proc. 10th IMDSP Workshop 98*, pp. 119-122, Alpbach, Austria, July 1998.