Constructing a Videophone for the Hearing Impaired using MPEG-4 Tools

Nikolaos Sarris and Michael G. Strintzis
Aristotle University of Thessaloniki, Greece

Lip Telephone is a special videoconferencing system we built with tools provided by the MPEG-4 audio–visual standard. The videophone achieves high-fidelity representation in the area of the speaker's mouth, letting lip readers communicate over a standard telephone connection or the Internet.

The basic problem with digital-video transmission is that we must store and transmit a vast amount of data over communication lines. In recent years, many standards have emerged for compressing moving images, which have contributed to multimedia technological developments. In 1992 the Moving Picture Experts Group (MPEG) completed the International Organization for Standardization and International Electronics Commission (ISO/IEC) MPEG-1 video-coding standard and approved the MPEG-2 standard in 1994 (see http://www.cselt.it/mpeg). These standards made interactive video on CD-ROM and digital television possible. The International Telecommunications Union-Telecommunications (ITU-T) organization established the H.261 standard in 1990[1] and H.263 in 1995, which they especially targeted to videophone communications. These have achieved successful videophone image sequence transmission at rates of approximately 64 kilobits per second (Kbps).

The techniques employed in these standards are based on segmenting images in uniformly sized 8×8 rectangular blocks. We can code the contents of the blocks using the discrete cosine transform (DCT) and estimate their motion in consecutive frames so that only the differences in their positions must be transmitted. This exploits spatial and temporal correlation in local areas and achieves great compression rates. The side effects of this approximation, however, are visual errors on the borders of the blocks (a blocking effect) and regularly spaced dots on the reconstructed image (a mosquito effect). These effects are more perceptible when we need higher compression rates, as in videophone applications. In addition, these standards impose the same technique on the whole image, making it impossible to distinguish the background or other areas of limited interest, even when the image is still.

Users can easily tolerate these problems during a traditional videoconference session where sound remains the basic means of communication, but the system is useless to the hearing impaired who lip read. Even these users, however, can easily tolerate image degradation in some areas of the image. For example, the background doesn't need to be continuously refreshed, and we don't need to code most areas of the head, besides the mouth area, with extreme accuracy. Obviously then, multimedia technology can benefit from methods that use knowledge of the scene contents, detecting different objects in the scene and prioritizing their coding appropriately. We can use some innovative features in the emerging MPEG-4 standard to overcome the limitations of traditional videophones. We can exploit the capabilities of MPEG-4 in videoconferencing systems to improve the quality of the reconstructed multimedia stream even in situations where the communication channel bandwidth is limited.

In our case study, we introduced some of the capabilities provided by the MPEG-4 audio–visual standard to content-based video coding. MPEG-4 has standardized the format of content-based coded data but has left most of the methods needed to perform the coding undefined so that researchers can propose different implementations. (See the sidebar "MPEG-4" for more information.) In this article, we design a videophone system, Lip Telephone, based on such methods. Our system lets the hearing impaired communicate over a standard telephone line because it significantly reduces the required bandwidth by using content-based coding techniques while maintaining high image quality in the area of the user's lips, which is necessary for lip reading.

General system description

Existing videophone systems lack the image resolution and accuracy necessary to visually

1070-986X/01/$10.00 © 2001 IEEE

MPEG-4

In 1993 MPEG launched the MPEG-4 standard, approving version 1 in October 1998 and version 2 in December 1999. MPEG-4 is the first audio–visual representation standard to model a scene as a composition of objects with specific characteristics and behavior.[1] Real and synthetic objects can be qualities coded by different techniques leading to different reconstruction qualities. A subgroup of MPEG-4 called Synthetic/Natural Hybrid Coding (SNHC) developed the framework for combining synthetic and natural content in the same scene.

In addition, the standard provides a detailed framework for representing the human face. This is accomplished with a 3D face object, which is mainly associated with two sets of attributes: facial definition parameters (FDPs) (see Figure A) and facial animation parameters (FAPs). Table A gives a representative subset. The first attribute describes a set of characteristic feature points on the face, and the second provides a set of facial deformation parameters, which can adequately describe almost any facial expression.

Both sets of parameters are based on the physiology of the human face. In particular, FAPs are based on the study of minimal facial actions and closely relate to muscle actions. The units of measurement for the FAP values are relative to the face dimensions in Figure A. The nonrigid motion dictated by any particular FAP may be unidirectional or bidirectional on one 3D axis, as Table A shows.

Clearly, this standard doesn't provide the means of performing the necessary operations to detect the position of feature points. Having at its disposal their positions and the values of the animation parameters, MPEG-4 does provide a model-based coding scheme for their transmission over the communication channel. The MPEG-4 decoder on the receiver side demultiplexes the coded information and displays the reconstructed image. The gain we get by coding with this technique to transmit a human face image sequence is obvious because a system only has to transmit the 3D model at the beginning of the session and only send the values of each of the 68 FAPs (numbers) for every frame. Related research reported experimental coding methods based only on the 3D face object that delivers videoconferencing scenes of tolerable quality at bandwidths as low as 10 Kbps.[1]

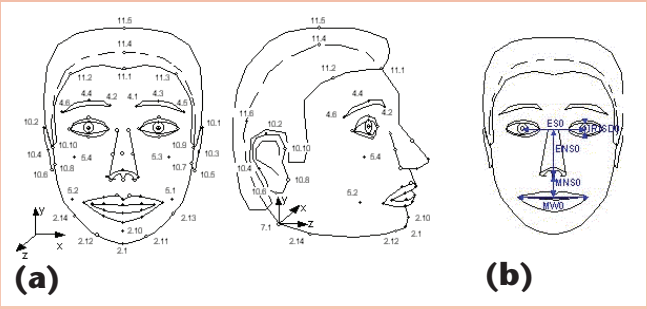


Figure A. The MPEG-4 facial definition parameters (a) and the facial animation parameter units (b).[2]

References

1. G.A. Abrantes and F. Pereira, "MPEG-4 Facial Animation Technology: Survey, Implementation and Results," *IEEE Trans. CSVT*, vol. 9, no. 2, Mar. 1999.
2. MPEG Video and SNHC, *Text of ISO/IEC FDIS 14496-3: Audio Doc.* ISO/MPEG N2503, Atlantic City MPEG Meeting, Oct. 1998.

Table A. Some of the FAPs MPEG-4 describes.[2]

Number	FAP Name	FAP Description	Units	Unidirectional or Bidirectional	POS Motion	FDP Group Number	FDP Subgroup Number
3	open_jaw	Vertical jaw displacement (doesn't affect mouth opening)	MNS	U	Down	2	1
4	lower_t_midlip	Vertical top middle inner lip displacement	MNS	B	Down	2	2
5	raise_b_midlip	Vertical bottom inner lip displacement	MNS	B	Up	2	3
6	stretch_l_cornerlip	Horizontal displacement of left inner lip corner	MW	B	Left	2	4
7	stretch_r_ornerlip	Horizontal displacement of right inner lip corner	MW	B	Right	2	5
8	lower_t_lip_lm	Vertical displacement of midpoint between left corner and middle of top inner lip	MNS	B	Down	2	6

understand speech. We can overcome this limitation with the help of MPEG-4 by transmitting the area of the speaker's mouth with high resolution and fidelity. We achieve a high compression rate with relatively poor quality for the remainder of the image, which isn't critical for lip reading. The

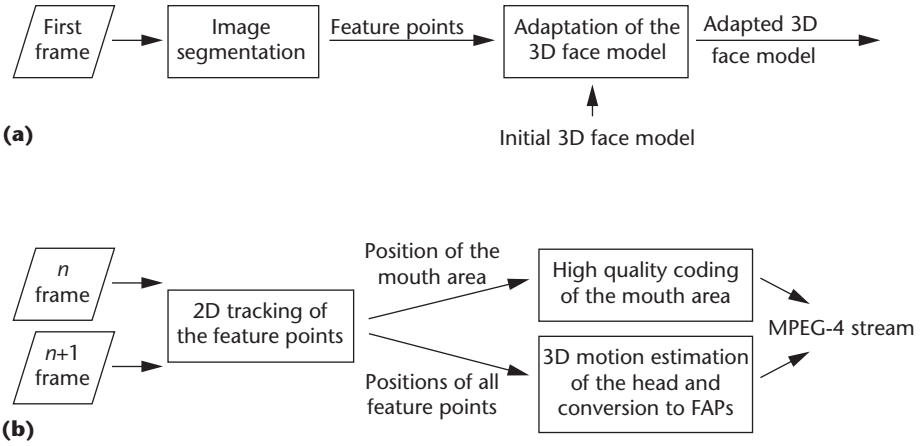


Figure 1. Lip Telephone block diagram: (a) initialization phase and (b) tracking phase.

MPEG-4 standard greatly assisted in this effort because it provides a framework that allows

- compression of a whole image or specially defined parts of the image (such as rectangular or irregular areas of the whole frame) by conventional methods and
- a special methodology for coding a human face, describing its position, shape, texture, and expression, with two sets of arithmetic parameters: facial definition parameters (FDPs) and facial animation parameters (FAPs).

When using the second methodology, given a 3D model of the speaker's head, the only data that we must transmit are the FAP values, which can adequately describe almost all facial expressions. The receiver will know the 3D head model because the system will transmit it at the beginning of the conference session. We use conventional coding methods to code the mouth area, which requires high quality in the decoded image stream. Our experimental results demonstrate that we can ensure excellent system performance with a simple, low-cost integrated services digital network line.

MPEG-4 provides the framework for developing model-based coding schemes, but it's open regarding which techniques that we can use to achieve the image analysis and modeling within these schemes. Thus, for the present system, we developed techniques for the following tasks:

- *Segmenting the videophone image and detecting the face's characteristics*. We detect the face and some of its characteristic points within the first image frame.
- *Adapting a 3D face model to the speaker's face*. We adapt a neutral face model to the particular speaker according to the characteristic points detected.
- *2D feature tracking*. We track the face's characteristic points in subsequent image frames.
- *Estimating the 3D motion of the speaker's head*. We estimate the 3D motion of the whole head based on the observed 2D motion of the feature points.
- *Developing the user interface and communicating channel handling issues*. We developed a window-based user interface and resolved the issues of transmitting the encoded stream over the communication channel.

Figure 1 highlights these procedures. We divided the system operation into two phases:

- In the initialization phase, we extract the feature points from the first image frame and adapt the neutral face model to these characteristics. This process takes place only at the beginning of a conference session.
- In the tracking phase, we track the feature points in subsequent image frames. From these, we detect and code the mouth area and estimate the 3D motion of the speaker's head.

We incorporated a generic face mesh as an initial, or neutral face model, labeling every node with a special tag showing the part of the face it belongs to—for example, a mouth or left eye. We developed this labeling for an interactive tool that highlights one node at a time and requests that the user identify the face part. Obviously, this is an offline process required every time we incorporate a new model into the system.

The high resolution coding of the detected mouth area and the creation of the MPEG-4 stream are issues the standard handles. MPEG-4 provides a rate controlled DCT-based coding algorithm, which is an improved version of the one in MPEG-2 with the added capability of defining areas in the image with specific coding characteristics. Therefore, having detected the mouth area's position in every frame, we code this area with the

provided algorithm so that when reconstructed it displays the mouth area with the desired quality and accuracy. Then, the system multiplexes the speaker's head and mouth objects in a stream according to the MPEG-4 format.

Segmenting the image and extracting facial features

Segmenting an image in general and detecting and extracting facial features in particular are common problems in many image-processing applications such as facial recognition, tracking, and modeling. A comprehensive review in Salembier and Marques[2] separates the known and followed strategies into those based on classification, transition, and homogeneity. In the special case of videoconferencing images, face detection becomes a simpler task because the programmer knows the scene content—one face approximately in the middle of the scene. Thus, the programmer can also use extra cues based on the knowledge of the face's geometrical structure. For example, Sirohey[3] gives a face segmentation and identification algorithm, using the elliptical structure of the human face. Sirohey fits an ellipse to a properly preprocessed edge map, marking the boundary between the head and the background regions. Eleftheriadis and Jacquin[4] use a similar approach, fitting a 2D ellipse onto binary edge data and postprocessing techniques to eliminate detection errors. However, such approaches only work well when the background is homogeneous or lightly to moderately cluttered.

Techniques based on neural networks,[5] principal component analysis (PCA),[6] or simpler template matching[7] suffice for face detection. However, we mostly use these techniques to localize the face in a larger rectangular area, and they can't locate the exact positions of features, which is necessary in our application. Researchers have also proposed systems based on the analysis of color distribution,[8] but they lack robustness when dealing with varying illuminations and when similarly colored objects appear in the background. Others have reported more complicated techniques based on the use of deformable contours, or deformable 3D models[9] to accurately detect a face's position and shape. These are based on an initial contour approximation of a face (either 2D or 3D), which is deformed by applying specific forces until the desired fitting to the given face has been achieved. Results of these methods are usually robust in drawing a rough estimate of the face contour but must be assisted by motion information, which will separate the face from the background. Several object-based coding schemes also exploit motion information in the image scene to segment homogeneously moving rigid objects.

Segmenting an image in general and detecting and extracting facial features in particular are common problems in many image-processing applications.

Possible methods for extracting the face region from the rest of the scene are based on either temporal or spatial homogeneity. Methods using temporal criteria may separate users from their homogeneous movement in the scene. However, such methods won't separate the face from the rest of the visible body and also won't work when the user is standing still. We can apply spatial criteria because the region should generally be similarly colored, and connected and have an ellipsoidal shape. However, difficulties arise in their implementation because dark areas, like eyes and facial hair, don't have the same color as the rest of the face. Moreover, the skin color may differ among users, and the proximity of the user's neck as well as objects in the background with color similar to the skin's can produce confusion.

To deal with these difficulties, we implemented a semiautomatic method to train a neural network to recognize image blocks contained in the facial region. We do this by positioning the user's face in a highlighted area. We then use the contents of this area as positive examples and the contents of the rest of the image scene (outside the highlighted area) as negative examples to train a feed-forward neural network.

To avoid dependencies on varying scene illumination, we only use the chrominance components Cb and Cr from the YCbCr color space to describe the scene's contents. Moreover, the feature vectors we use to train the neural network aren't built solely from the direct chrominance values because the resulting neural network then wouldn't be capable of separating skin pixels

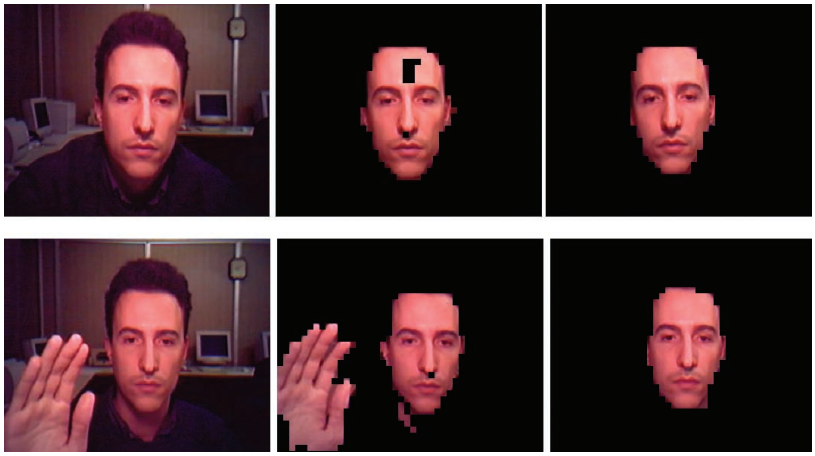


Figure 2. (a) Camera image, (b) the facial region the neural network identified, and (c) the corrected facial region after postprocessing.

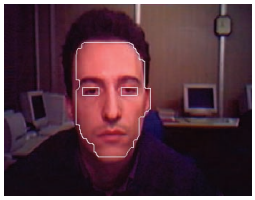


Figure 3. Contour refinement and blink detection.

Figure 4. The facial regions and features extracted by thresholding in the frontal and profile image views. In the profile view, the rectangles show the search areas for the particular features.

from similarly colored pixels in the scene's background. Rather, we break the facial and background areas into consecutive rectangular blocks from which the Cb and Cr histograms are quantized and concatenated to form the training vectors. We use the trained neural network to determine facial blocks in subsequent frames captured from the camera. We accomplish this by dividing every such frame in rectangular blocks and constructing a feature vector from each block in the same way as in the training process—that is, the Cb and Cr histograms are built, quantized, and concatenated. The system then consults the neural network with each feature vector to decide whether or not the particular block belongs to the user's face. Results are quite satisfactory (see Figure 2 for two samples) even though the neural network might misclassify a small number of blocks.

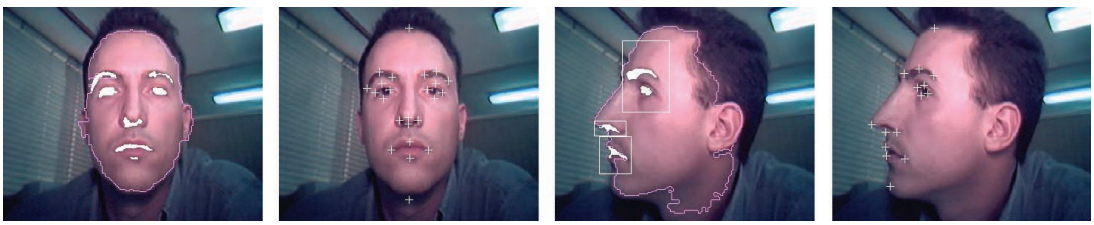
To compensate for these errors, we perform a number of postprocessing operations, similar to the ones proposed in Chai and Nghan,[8] on the neural network's output. Finally, we use a connected component algorithm to find all connected regions and accept only the largest area, based on the knowledge that only one face should be in the scene. Figure 2c shows typical results from these postprocessing operations.

Having detected the facial region in the image, we implement specific operators to locate the exact positions of the eyes, eyebrows, nose, and mouth. To locate the position of the eyes, we use a temporal criterion by detecting the blinking of the eyes. To detect the exact eye region within each window, we seek the connected regions within the window and use the largest region to represent the eye. Our proposed algorithm is robust in all situations where the user's eyes are clearly visible in the scene. Figure 3 shows a sample result.

Having reliably located the positions of the eyes, we employ an iterative adaptive thresholding algorithm on the image luminance Y within the facial area until we find the connected thresholded areas at the expected positions. We perform the thresholding operation adaptively according to the mean luminance value observed within the detected facial area. We exploit geometric knowledge of the face's structure to eliminate false features. Figure 4 shows sample results for a face's frontal and profile view. We use the detected areas' edge points as the exact positions of the feature points.

3D facial model adaptation

We propose a novel method for adapting a generic 3D face to a particular person's face viewed by a camera. The related literature has proposed several methods for deforming 3D face models, such as capturing the required geometry of the face with a 3D laser scanner.[10] Won-Sook et al.[11] use two orthogonal photos of the target face and deform the generic face model with a 3D geometric transformation, specifically a type of free-form deformation (FFD) usually termed the Dirichlet FFD.[12] Pighin et al.[13] estimate the required 3D positions of the facial features from a series of captured image frames of the target face and transform the generic model using an interpolation function based on radial basis functions. Zhang[14] uses information from one view of the target face to measure the face, eyes, and mouth dimensions, which he then uses to adapt a simple 3D face model by rigidly transforming the whole face and locally correcting the position and ori-



entation of the face, eyes, and mouth. The method requires geometric assumptions however, because we can't totally deduce the 3D characteristics of the features from only one view.

Our approach differs from all these methods because it treats the facial model as a collection of facial parts, which we deform according to separate affine transformations. This simple method proves effective for face characterization because the physiological differences in characteristics between faces are based on precisely such local variations. For example, a person may have a longer or shorter nose or narrower eyes.

We capture the required face geometry from two orthogonal views of the face acquired by the camera and segment them according to the method we described in the previous section. Having located the facial features, we define their edge points (such as left-most or right-most eyes) as the face's feature points. Knowing the positions of these feature points in 2D, standard geometrical calculations—assisted by least squares methods—provides the required positions of the feature points in the 3D space.

Next, we must deform the 3D facial model to displace its corresponding nodes toward these positions while maintaining the face's natural characteristics and symmetry. The first part of this adaptation process involves a rigid transformation of the model. Thus, we rotate and translate the model to match the real face's pose. We calculate the rotation and translation transformations by slightly modifying the spatial resection problem in photogrammetry. We perform a nonlinear minimization so that the rotation and translation transformations bring the model feature nodes as close as possible to their required positions.

The rigid transformation can align the generic model with the required face and scale it to meet the total face dimensions. However, we must also alter the face's local physiology in this way. Thus, in the second step of the adaptation process, we transform the model in a nonrigid way, further displacing the feature nodes and bringing them as close as possible to their exact calculated positions while the facial parts retain their natural characteristics. To perform this adaptation, we split the model into face parts (left eye, right eye, mouth, and so on) and perform a rigid adaptation on every part separately. This means that we rotate, translate, and stretch every 3D face part to minimize the distances of the feature points belonging to that face part from their required positions. We accomplish this with the following transformations:

1. *Centering at the origin.* The center of the face part is the 3D center of the feature nodes contained and the whole part is translated toward the origin so that this center is on the origin. We do the same with the set of required feature positions. That is, we find their center and translate them toward the origin in the same manner.

2. *Alignment.* We rotate the face part around the origin so that three lines connecting three pairs of feature nodes are aligned with the corresponding lines connecting the required feature positions. We do so by minimizing the differences in the gradients of these lines.

3. *Stretching.* We scale the face part around the origin with different scale factors for every axis to minimize the distances of the transformed nodes from their required positions.

4. *Translation.* After the stretching transformation, we translate the face part back toward its original position by adding the position vector of the face part center calculated and subtracted in step 1.

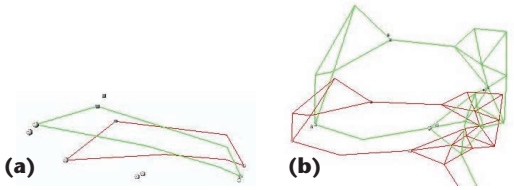


Figure 5. Local adaptation of the right eyebrow (a) and right eye (b).

Figure 5 shows the results of these steps for two face parts. Among all the face parts, we define one series of nodes as border nodes because they don't belong to any face part. We find the positions of these nodes after the deformation by linearly interpolating the neighboring nodes belonging to the face parts. We do this to ensure that we make a smooth transition from one facial part to the other and filter out possible discontinuities.

Thus, the final deformed model adapts to the particular characteristics implied by the feature points (for example, a bigger nose or smaller eyes), keeping the generic characteristics of a human face (such as smooth skin and a symmetrical face). Figure 6 shows the results of the rigid and nonrigid adaptation procedures.

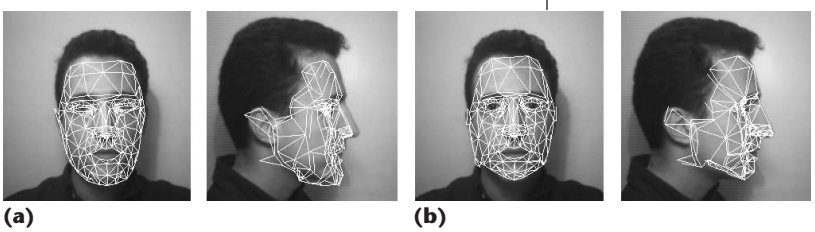


Figure 6. Rigid (a) and nonrigid (b) adaptation of the face model on the front and profile views.

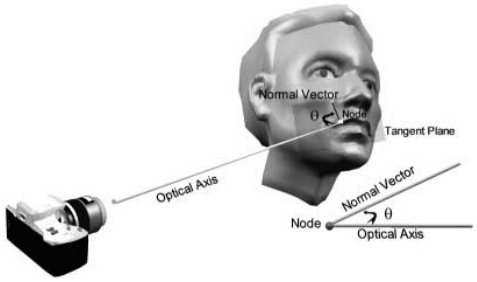


Figure 7. The tangent plane and normal vector of a node in relation to the optical axis.

Estimating 2D feature tracking and 3D motion

Among the techniques that researchers have proposed for estimating facial motion, most employ dense optical flow estimation algorithms and are too complex for real-time, reliable computation requirements. Researchers have generally avoided feature-based tracking methods because they tend to give unreliable results. Successful attempts to simplify this task by using artificial markers or heavy make-up on the tracked faces proved that feature-based tracking is suitable, but these techniques aren't applicable to real, unrestricted systems. Pighin, Szeliski, and Salesin[15] propose minimizing an error function over the set of facial expressions and face positions spanned by the 3D model by altering the values of facial position and expression parameters. Valente and Dugelay[16] propose a similar approach where a set of eigenvectors represent the facial expressions.

Although in theory these two methods can yield the required values of facial motion in the parametric space introduced, they're based on the assumptions that a realistic rendering of the model is available and that we can perfectly compensate lighting conditions. These methods introduce several parametric models of the face for interpreting facial motion and expressions. However, the related research literature hasn't sufficiently explored FAPs, the only standardized facial animation parameter space, recently introduced by MPEG-4. Only Eisert and Girod[17] have proposed extracting FAPs. Their method is based on a linearized dense optical-flow motion estimation method that uses a spline-based 3D facial model to extract FAPs. They can only generate this model, however, using specialized 3D scanning equipment, and the motion estimation method they propose suffers from the aforementioned complexity of dense optical-flow methods.

We propose a technique for improving the reliability of feature-based tracking methods for 3D facial motion estimation. We introduce a criterion for assessing the reliability of tracked features that correspond to 3D model nodes, which we combine with two other criteria for a feature tracking algorithm's accuracy, to provide a measure of confidence in the estimated position of every feature. Furthermore, we use the framework standardized by MPEG-4 to provide a nonrigid facial motion model, which guides the feature tracking algorithm. We integrate these techniques into a system, which estimates the 3D rigid and nonrigid motion of points of a human face and FAPs without using any face markers.

Facial-feature tracking

Researchers have proposed various methods for tracking motion in images based on dense optic flow, block matching, Fourier transforms, or pel recursive methods.[18] We prefer a feature-based approach because of its speed and its natural suitability to this knowledge-based node-point correspondence problem. Generally, the features we track can be corners, edges, or points, and we must select these features so we can easily and reliably track them.

Kanade, Lucas, and Tomasi propose a widely used feature-based algorithm known as the KLT algorithm.[19] The KLT algorithm is based on minimizing the sum of squared intensity differences between a past and a current feature window, which is performed using a Newton–Raphson minimization method. Although the KLT algorithm yields satisfactory results, it's important to assess the results of tracking in our system so that we use the optimum set of feature correspondences in the model adaptation stage. For this reason, we sort the tracked feature points according to two criteria introduced by and closely related to the KLT algorithm's operation and a third criterion related to the nature of the 3D model to be adapted. These criteria are defined as follows:

- *Trackability*. The ability of a 2D feature point to be tracked reliably, which is related to the roughness of the texture within its window.[19]
- *Dissimilarity*. The sum of squared intensity differences within the feature window W between the two consecutive image frames I and J. Dissimilarity indicates how well we've tracked the feature.[19]
- *Reliability*. Every tracked feature point corresponds to a feature node on the 3D face model, which we've adapted to the face located in the previous image frame. We define the reliabili-

ty metric of a node as cosθ, where θ is the angle formed by the optical axis and the normal vector to the surface at the node in question (see Figure 7).

We combine these criteria to provide a measure of confidence for each feature point we track. According to this measure, we choose the best-tracked feature points for use in the next step, which is estimating facial motion.

Estimating rigid 3D motion

Having found the set of the most reliably tracked positions of the feature points, we must move the head model so that its feature nodes project as close as possible to these positions. This means that we need to compute the 3D motion parameters (**R**, **T**), which will minimize the distance of these projections from their positions tracked in the image. The related literature has proposed many solutions to this problem,[20] but the solution's accuracy depends highly on the reliability of the given feature correspondences.

Once we've confirmed that the selected feature correspondences from the previous step are appropriate, we employ the method proposed in Tsai and T.S. Huang,[21] which we enhanced to include the focal length of our camera to estimate **R** and **T** up to a scaling factor. Furthermore, using the 3D model coordinates in the previous frame, we compute this scaling factor and determine an absolute solution for **R** and **T**. We further optimize this solution with an iterative minimization algorithm.

After determining **R** and **T**, we know the new rigidly transformed 3D positions for all the model nodes. As we mentioned earlier, we rank the tracked feature points according to their trackability, dissimilarity, and reliability and only use the ones with sufficiently high rank to estimate the 3D rigid motion. Thus, many features may not be accurately positioned by the tracking procedure in the current frame. We calculate the correct positions of these feature points by estimating **R** and **T** matrices. In this way, these points are relocated in the image (see Figure 8) so that we can track them in subsequent frames.

Obviously, a reliable relocation always requires a reliable 3D motion estimation. Moreover, because it's always possible that the real features aren't visible because of occlusions by head rotation, we recheck the sign and magnitude of the reliability metric. If it's negative, the tangent to the node surface is oriented toward the opposite direction of the camera and thus the feature point is invisible on the image frame. Therefore, we don't consider this feature point for tracking in the next frame. Even if the reliability metric is positive, its magnitude must be greater than a threshold because there's a high possibility that the reprojected feature will be close to the face's border. In that case, a small calculation error may cause the feature to be assigned to the background, which of course is undesirable.

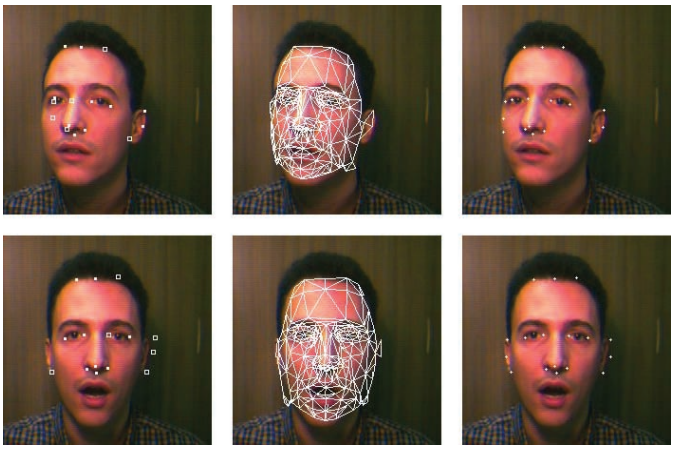


Figure 8. Example of the assessment of tracked features (left), the rigid fitting of the 3D model (middle) and the relocation of the mistracked (rejected) features by projection of the corresponding transformed model nodes (right). Rejected feature points are empty boxes, accepted feature points are full boxes, and the corrected positions are crosses.

This method has proven reliable even under extreme rotation conditions where most other methods fail. Figure 9 (next page) shows the tracking results for a moving head image sequence. It's evident that although the feature tracking module can incorrectly estimate the positions of some feature points, our method selects the most accurately tracked points and thus calculates the head's accurate 3D rigid motion.

Estimating nonrigid 3D facial motion

MPEG-4 has defined a set of 68 FAPs to fully describe the allowed rigid and nonrigid motions we can observe on a human face in a neutral pose. It uses three FAPs to describe the rotation of the head around the neck—FAP values 48 to 50 for head pitch, yaw, and roll respectively. After estimating the head's rigid motion, we can readily compute these FAPs from the Euler angles of the rotation matrix **R**. Most of the other FAPs affect only one FDP describing its nonrigid motion, but one FDP may be affected by more than one FAP. For each of these

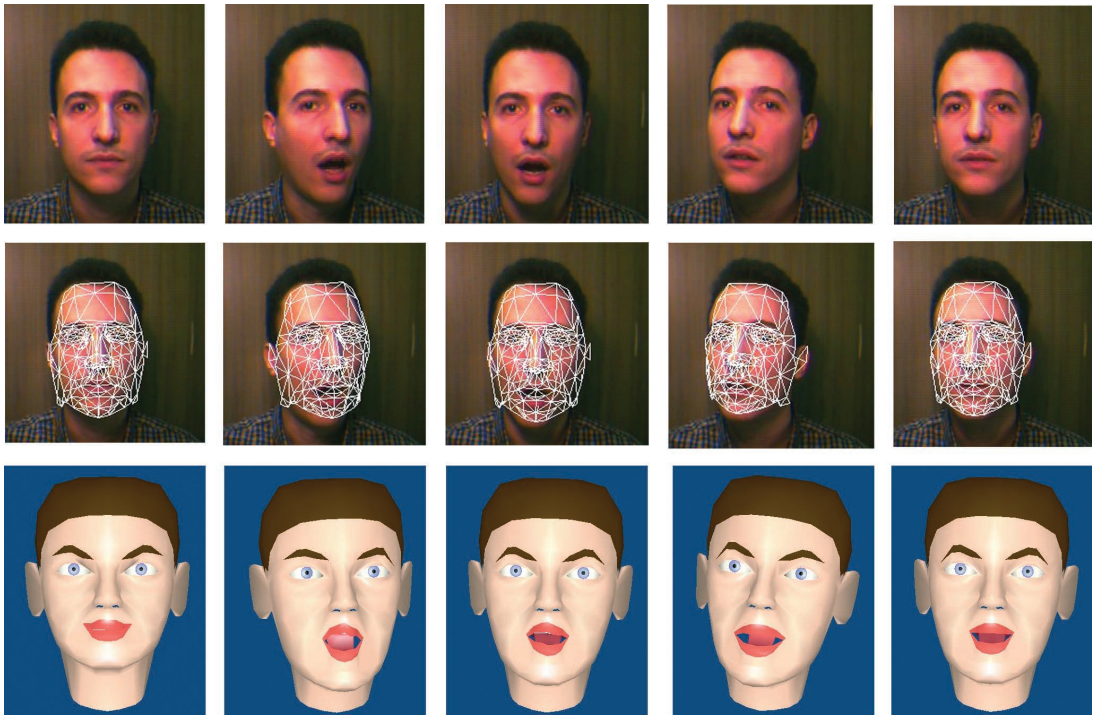


Figure 9. Results of simultaneous rigid and nonrigid tracking.

parameters, MPEG-4 specifies the movement's direction, while the FAP value describes the nonrigid motion's algebraic value. Once we successfully recover the rigid part of the motion using the proposed approach, we can use the local 2D motion estimates provided by the feature tracker to determine the corresponding feature point's nonrigid motion. Although only the 2D motion of each feature point is available, we can recover its 3D nonrigid motion because MPEG-4 specifies the possible axes of movement. We identified three cases for evaluating the possible number of degrees of freedom (DOF) for a feature point's nonrigid motion:

- The feature point can move in any of three possible directions. This happens only for the eyes, tongue, and jaw. However, we can handle the eyes as separate rigid objects and restrict the DOF of the tongue and jaw, as we only need three DOF to support exaggerated movement (for example, by cartoon characters). Thus, in our case of real human face image sequences, the possible number of DOF will always be smaller than three.

- The feature point can only move in two directions (two DOF). For example, we can translate FDP 2.1 along the X-axis using FAP 15 and along the Z-axis using FAP 14. If (x, y, z) are the initial values of the corresponding wireframe node in the neutral pose (for example, time 0), then $(x + F_{15}, y, z + F_{14})$ is the position of the same node at frame t if only nonrigid motion exists. Thus, under the presence of rigid motion the resulting position of the 3D node will be

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{R} \begin{bmatrix} x + F_{15} \\ y \\ z + F_{14} \end{bmatrix} + \mathbf{T} \tag{1}$$

The projection equations for two corresponding 3D points (x_i, y_i, z_i) and (x_i', y_i', z_i') are given by

$$X_i' = f \frac{x_i'}{z_i'}, Y_i' = f \frac{y_i'}{z_i'} \tag{2}$$

where (X_i, Y_i) and (X_i', Y_i'), $i = 1 \ldots n$, are the 2D feature correspondences on the image plane.

Using the projection equations and Equation 1, we obtain a system of two equations with two unknowns, which we can use to determine the unknown FAP values F_{15} and F_{14}.

- The feature point can only move in one direction (one DOF). We can deal with this case in a similar way, resulting in an over-determined system with two equations and one unknown. Note that the tracking procedure the KLT trans-

form uses has two DOF. Although this kind of tracking procedure suits FDP nodes affected by 2 FAPs, it isn't suitable for FDP nodes affected by only one FAP (one DOF). For such nodes, it would be better to design a 1D feature tracker, which determines the feature position along a 2D line. We determine this line using Equation 1 and the projection Equation 2 on the image plane. For this reason, we constrain the KLT tracking algorithm to optionally search along a single 2D line, which is the projection of the calculated 3D axis of permitted nonrigid motion. Figure 10 shows examples of nonrigid tracking along a line.

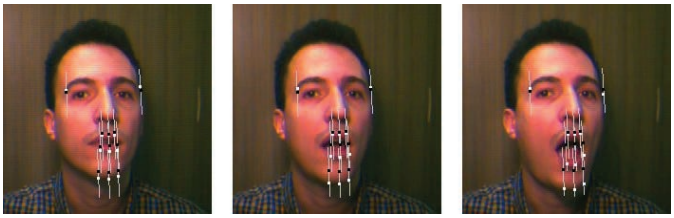


Figure 10. Examples of tracking along a line. Black boxes are rigid positions of feature points. White boxes are nonrigid positions of feature points.

We used a moving head image sequence illustrating both rigid and nonrigid motion to demonstrate the results of our method. The **R** and **T** matrices calculated by the rigid-motion estimation module transformed the head model adapted to the face top, left-most image in Figure 10, which is projected on every subsequent frame to illustrate the proposed technique's accuracy. The nonrigid motion estimation method produced an MPEG-4 compatible FAP file, which an MPEG-4 FAP player used to animate a synthetic face and illustrate the correspondence between the captured image sequence and the synthetically produced face animation. From the projection of the 3D model and the observation of the synthetic face's rotation, it's evident that both the head's rigid and nonrigid motion are estimated with adequate accuracy in every frame. Figure 9 gives the tracking results.

User interface and networking issues

We designed the system's interface to be user friendly and simple, providing the following operations:

- Users can dial a telephone number and establish a connection using a standard modem over a standard telephone line or provide an Internet protocol address if an Internet connection exists.
- The system can transmit and receive video images of the speakers in real time.
- The system can transmit and receive audio if a user has at least partial hearing. When this feature is disabled, the system uses the bandwidth gained to improve the video image.
- We provide a text window where users can exchange messages in real time.
- We provide a whiteboard window so users can draw or paste areas of their desktop. Users can explain many concepts by diagramming information or using a sketch.

Figure 11 shows an initial graphical user interface we developed to incorporate the menus and buttons required for these operations. The left-most button below the video image lets users place a call and establish a connection with another Lip Telephone user. The next three buttons to the right launch the helper application windows (whiteboard, chat, and file transfer), and the two right-most buttons enable or disable the video and audio. The File menu provides optional operations for capturing and saving still images or video to the local disk. The Options menu controls the settings for the video and audio. The Drivers menu

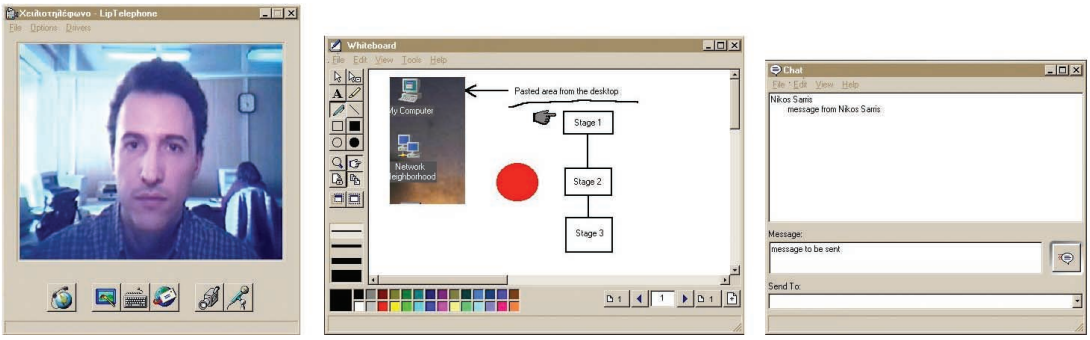


Figure 11. The graphical user interface main window and helper applications.

lets users select the capture card and camera they wish to use (if more than one is in use).

For data communication between the two parties, we selected the real-time protocol (RTP)[22] over an IP connection, which is established over a telephone line (standard or ISDN) or the Internet. RTP is the Internet standard protocol for transporting real-time data. We prefer it over transmission-control protocol (TCP) because it provides special functionality suited for carrying real-time content, such as time stamps and control mechanisms for synchronizing different streams with timing properties. RTP consists of data and control parts. The data part is a thin protocol providing support for applications with real-time properties such as continuous media, including timing reconstruction, loss detection, security, and content identification. The control part provides support for real-time conferencing of groups of any size over the Internet. This support includes source identification and support for gateways like audio and video bridges as well as multicast-to-unicast translators. It offers quality-of-service feedback from receivers to the multicast group and support for synchronizing different media streams.

Results

The results from our initial experiments with the system show satisfactory image quality for 10 fps (frames per second). Tests with hearing-impaired users prove that 10 fps are adequate for lip reading provided that the image is clear in the lip area. In a CIF-sized image (352×288 pixels), the bounding box of the lips is less than 100×50 pixels for the usual position of the speaker's face—that is, positioned so that the face covers almost the whole area of the image. We achieved adequate image quality (minimum 25 dB) in our experiments for the lip area at a coding rate of 0.5 bpp (bits per pixel). This means that the bandwidth required to transmit an image of the lip area is around 20 Kbps at 10 fps. This is the system's main bandwidth overhead because we can code the sound with 2 Kbps at reasonable quality (11 KHz sampling rate), and for the rest of the head, only six floating-point numbers (240 bps) must be transmitted for the description of the head's rigid motion, or 68 floating point numbers (2.7 Kbps) for the description of both rigid and nonrigid head motion. Furthermore, we can arithmetically code these numbers, as proposed in MPEG-4, providing a bitstream requiring a maximum bandwidth of 2 Kbps. This means that the whole system requires a bandwidth of less than 25 Kbps operating at 10 fps, which we can achieve over a standard telephone line or a fast Internet connection.

The system initialization, which involves locating the facial features and adapting the face model, requires 5 to 10 seconds, but the system can perform that in parallel as it establishes the connection, thus making the delay acceptable. During normal operation, the system can currently code frames at 5 to 7 fps on a standard Pentium II PC at 450 MHz with 256 Mbytes of RAM, but we're working on optimizing the system's speed, which we expect to reach 10 fps.

Conclusion

We've presented a method for combining traditional coding methods with object-based coding techniques to implement a videophone system with specific requirements. We can find such requirements in many imaging applications, such as when the background scene is less important than the foreground. The problem usually lies with effectively segmenting the image of higher importance and applying a model-based coding method to the object of less importance. We hope that the methods here will provide useful ideas for different ways of tackling such hybrid coding problems. **MM**

Acknowledgments

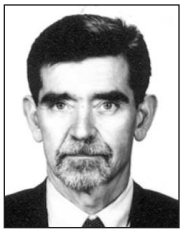
The Greek Secretariat of Research and Technology, Technologies for the Disabled project Lip Telephone: A Videophone for the Deaf, supported this work.

References

1. *Video Codec for Audiovisual Services at px64 kbit/s*, ITU-T Recommendation H.261, Int'l Telecommunications Union-Telecommunications Geneva, 1990.
2. P. Salembier and F. Marques, "Region-Based Representations of Image and Video: Segmentation Tools for Multimedia Services," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 8, Dec. 1999, pp. 1147-1169.
3. S. Sirohey, *Human Face Segmentation and Identification*, master's thesis, Computer Vision Laboratory, Univ. of Maryland, College Park, Md., 1993.
4. A. Eleftheriadis and A. Jacquin, "Automatic Face Location Detection for Model-Assisted Rate Control in H.261 Compatible Coding of Video," *Signal Processing: Image Communication*, vol. 7, no. 4-6, Nov. 1995, pp. 435-455.
5. S. Lawrence et al., "Face Recognition: A Convolutional Neural Network Approach," *IEEE Trans. Neural Networks*, special issue, vol. 8. no. 1, Jan. 1997, pp. 98-113.
6. I. Craw et al., "How Should We Represent Faces for

Automatic Recognition?," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, Aug. 1999, pp. 725-736.
7. Z. Liang, "Tracking A Face for Knowledge-Based Coding of Videophone Sequences," *Signal Processing: Image Communication*, vol. 10, no. 1-3, July 1997, pp. 93-114.
8. D. Chai and K.N. Nghan, "Face Segmentation using Skin Color Map in Videophone Applications," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 4, June 1999, pp. 551-564.
9. D. DeCarlo and D. Metaxas, "The Integration of Optical Flow and Deformable Models With Applications to Human Face Shape and Motion Estimation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 231-238.
10. L. Yuencheng, D. Terzopoulos, and K. Waters, "Realistic Modeling for Facial Animation," *Computer Graphics* (Proc. Siggraph 95), ACM Press, New York, 1995, pp. 55-62.
11. L. Won-Sook et al., "MPEG-4 Compatible Faces from Orthogonal Photos," *Proc. Int'l Conf. Computer Animation*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. 186-194.
12. L. Moccozet and N. Magnenat-Thalmann, "Dirichlet Free-Form Deformations and their Application to Hand Simulation," *Proc. Int'l Conf. Computer Animation*, IEEE CS Press, Los Alamitos, Calif., 1997, pp.93-102.
23. F. Pighin et al., *Realistic Facial Animation Using Image Based 3D Morphing*, tech. report UW-CSE-97-01-03, Computer Science Dept., Univ. of Washington, 1997.
14. L. Zhang, "Automatic Adaptation of a Face Model Using Action Units for Semantic Coding of Videophone Sequences," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, no 6, Oct. 1998, pp. 781-795.
15. F. Pighin, R. Szeliski, and D. Salesin, "Resynthesizing Facial Animation through 3D Model-Based Tracking," *Proc. Int'l Conf. Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. 143-150.
16. S. Valente and J.-L. Dugelay, "Face Tracking and Realistic Animations for Telecommunicant Clones," *IEEE MultiMedia*, vol. 7, no. 1, Jan.–Mar. 2000, pp. 34-43.
17. P. Eisert and B. Girod, "Analyzing Facial Expressions for Virtual Conferencing," *IEEE Computer Graphics and Applications*, vol. 18, no. 5, Sept./Oct. 1998, pp. 70-78.
18. F. Dufaux and F. Moscheni, "Motion Estimation Techniques for Digital TV: A Review and a New Contribution," *IEEE Proc.*, vol. 83, no. 6, IEEE Press, Picataway, N.J., 1995, pp. 858-876.
19. J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 593-600.
20. J.K. Aggarwal and N. Nandhakumar, "On the Computation of motion from Sequences of Images - A Review," *IEEE Proc.*, vol. 76, no. 8, IEEE Press, Picataway, N.J., 1988, pp. 917-935.
21. R.Y. Tsai and T.S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, Jan. 1984, pp. 13-26.
22. H. Schulzrinne et al., *RTP: A Transport Protocol for Real-Time Applications*, RFC1889, Audio–Video Transport Working Group, Jan. 1996

Nikolaos Sarris is a PhD candidate in the Electrical and Computer Engineering Department at the Aristotle University of Thessaloniki, Greece, where he is also a graduate research assistant. His research interests include image processing, computer vision, model-based 3D image-sequence analysis, synthesis and coding, and video coding standards. He has an MEng in computer systems engineering from the University of Manchester Institute of Science and Technology, United Kingdom. He is a member of the Technical Chamber of Greece.



Michael G. Strintzis is a professor of electrical and computer engineering and director of the Informatics and Telematics Research Institute at the University of Thessaloniki, Greece. His research interests include 2D and 3D image coding, image processing, biomedical signal and image processing, and DVD and Internet data authentication and copy protection. He has a diploma in electrical engineering from the National Technical University of Athens, Greece, and an MA and PhD in electrical engineering from Princeton University. He is an associate editor of *IEEE Transactions on Circuits and Systems for Video Technology* and was awarded one of the Centennial Medals of the IEEE.

Readers may contact Sarris at the Informatics and Telematics Inst., Center for Research and Technology Hellas, 1st Km Thermi-Panorama Rd., Thermi-Thessaloniki, GR-57001 PO Box 361, Greece, email nikos@dion.ee.auth.gr.