Image-Based Animation of Facial Expressions*

Gideon Moiza† Ayellet Tal† Ilan Shimshoni‡ David Barnett† Yael Moses§

Abstract

We present a novel technique for creating realistic facial animations, given a small number of real images and a few parameters for the in-between images. This scheme can also be used for reconstructing facial movies, where the parameters can be automatically extracted from the images. The in-between images are produced without ever generating a three-dimensional model of the face. Since facial motion due to expressions are not well defined mathematically our approach is based on utilizing image patterns in facial motion. These patterns were revealed by an empirical study which analyzes and compares image motion patterns in facial expressions. The major contribution of this work is showing how parameterized “ideal” motion templates can generate facial movies for different people and different expressions, where the parameters are extracted automatically from the image sequence. To test the quality of the algorithm, image sequences one of which was taken from a TV news broadcast were reconstructed, yielding hardly distinguishable movies from the originals.

Keywords: facial animation, facial expression reconstruction, image morphing.

1 Introduction

The human face is one of the most complex and interesting objects that we come across on a regular basis. The face, and the myriad expressions and gestures that it is capable of making, are a key component of human interaction and communication. People are extremely adept at recognizing faces. This attribute presents both an advantage and a challenge to any system that manipulates facial images. The viewer is likely to be able to instantly spot any defects or shortcomings in the image. If the image is not a perfect rendition of an actual face, both in appearance and in motion, the user will notice the discrepancies. Therefore, any facial application needs to be highly accurate if it is to be successful.

In this paper we explore the issue of constructing images of facial expressions. Our method uses a small number of full frames. In addition, for each in-between frame, a small number of points on contours in the image are sufficient to describe the shape and the location of facial features, and to generate these frames. We will show that our method is capable of generating an image sequence in a faithful and complete manner.

*Work has been supported in part by the Israeli Ministry of Industry and Trade, The MOST Consortium
†Department of Electrical Engineering, Technion - IIT, Haifa, 32000, Israel, {ayellet, gideon}@ee.technion.ac.il
‡Department of Industrial Engineering, Technion - IIT, Haifa, 32000 Israel, ilan@ie.technion.ac.il
§Department of Computer Science, Interdisciplinary Center, Herzliya Israel, yael@idc.ac.il
This method can be used for producing computer graphics animations, for intelligent compression of video-conferencing systems and other low-bandwidth video applications, for intelligent man-machine interfaces, and for video databases of facial images or animations. Though the goal is to produce animations, one plausible way to test the quality of the technique is to sample a real facial movie, reconstruct it with our algorithm, and compare the two movies. This suggests that the same method can be used for compression of facial movies. In this case, the control parameters can be extracted automatically from the original movies using state-of-the-art tracking systems (e.g., [6, 21]).

Most of the techniques for facial animations utilized in computer graphics are based on modeling the three-dimensional structure of a human face and rendering it using some reflectance properties (see [23] for a survey). These techniques are the state-of-the-art of facial animations, and generate extremely compelling results. Geometric interpolation between the facial models is used in [22], where the models are digitized by hand. Measurements of real actors are used in [2, 13, 32]. The system described in [16] captures the facial expressions in three dimensions and can replay a three-dimensional polygonal face model with a changing texture map. The process begins with a video of a live actor’s face. Meshes representing the face are used jointly with models of the skin and the muscles in [31, 28, 19, 20]. The system of [9] is designed to automatically animate a conversation between human-like agents. Generating new face geometries automatically, depending on a mathematical description of possible face geometries is proposed in [11]. It has been shown in [24] how two-dimensional morphing techniques can be combined with three-dimensional transformations of a geometric model to automatically produce three-dimensional facial expressions.

We propose a different approach which avoids modeling and rendering in three-dimensions. It is thus less expensive. Instead, we use as a basis a set of real images of the face in question, and create the animations in the image domain, by producing the in-between images.

In this regard, our work is more related to the image morphing approach, such as [33, 1, 18], which have been proven to be very effective. One problem with most of this work however, is that the end-user is required to specify dozens of carefully chosen parameters. Moreover, these methods are more appropriate for morphing one object into another, where the object can be a person. Since the in-between person is not known, errors are more tolerable. Our goal, however, is different. We want to generate a movie of a specific person, given a few frames from a movie of this person.

In [8], a related though different problem is discussed. The mouth regions are morphed in order to lip-synch existing video to a novel sound-track. This method is different from ours since it combines footage from two video sequences, while we generate the sequence artificially. In [15], an animated talking head system is described, where morphing techniques are used to combine visemes taken from an existing corpus. This method is different from ours since we generate the visemes artificially.

Related work has also been done in the field of computer vision, where image-based approaches are utilized, bypassing three-dimensional models. In [10] a face is described by a large number of parameters describing its shape and its appearance. Given the parameters of a new image, it can be reconstructed. This approach differs from ours as we use information from neighboring images in the movie in order to reconstruct the in-between images. As a
result, not only much fewer parameters are needed, but also the resulting images look sharper as we avoid warping. In [3], optical flow of a given image to a base image is utilized. Both texture vectors and shape vectors, which form separate linear vector spaces, are used. The advantage of this approach is that optical flow is done automatically and is well explored. In addition, it is general and is not model-based. A consequence of this generality is that the resulting images are not as sharp as the approach we are pursuing, since optical flow is not always accurate. One way to look at our approach is as a model-based optical flow, where the model of the optical flow has been discovered empirically and is fixed for each expression.

Generally, the animation of faces requires handling the effects of the viewing position, the illumination conditions and the facial expressions. There has been a lot of work considering the effects of viewing position and illumination, mostly in computer vision (see [29, 27, 26, 30, 25] for a few examples). Unlike the changes in the viewing position which can be described by rigid transformations, the non-rigid transformations determining human expressions are not well defined mathematically. One way to handle these transformations is to learn from examples the set of transformations of a face while talking or changing expressions.

In this paper we take this avenue while focusing on these non-rigid transformations. We have been empirically studying image patterns in facial expressions. Though skin deformation can result from complex muscle actions and bone motion, our experimentations revealed that patterns do exist in images, both with respect to the same person on different occasions, and also between different people performing the same expression. Moreover, these patterns can be expressed in a rather simple way. In addition, our experimentations showed that there exists a strong correlation between the motion of regions in a face to the motion of a small number of contours bordering them.

Thus, the major contribution of this work is showing, based on the empirical results, how parameterized "ideal" motion templates can generate facial movies. We show that given very few images and a few image parameters that can be automatically extracted for the in-between images, the full sequence can be reconstructed. Our technique is combined with existing techniques dealing with changes in viewing position, yielding a system that deals both with head motion and facial expressions in a faithful manner.

We ran our algorithm on several movies of people performing various expressions, including a movie of a TV broadcaster. In order to evaluate the quality of our algorithm, we compared our artificial movies to the original movies. The movies were hardly distinguishable. On average, one full frame in fifteen is need. If we take into account the other information we use the compression rate is 1:13. When we combine our animation algorithm with standard compression schemes we get very good compression rates. For instance when we use the JPEG compression algorithm whose compression rate is about 1:30–1:40 we get that using our algorithm achieves a compression rate of 1:390–1:520. Using our algorithm above MPEG2 reduces the size of the movie by a factor of 4 with respect to MPEG2.

The rest of this paper is organized as follows. In Section 2 we describe the empirical experiments we conducted and draw conclusions. The goal of the experiments was to find image motion patterns that occur during facial expressions. In Section 3 we describe our algorithm for animating facial expressions, using the empirical analysis. In Section 4 we present the experimental results produced by our animation algorithm. We conclude in Section 5.
2 Empirical Analysis of Facial Expressions

The animation system, which will be described in the next section, must determine the correspondence between a pixel in the in-between frame and the matching pixels in the key-frames. To do so, it is necessary to know the patterns in which a facial point is moving when an expression is evolved. Since these patterns are unknown mathematically, we study them in this section experimentally. The result of this study will allow us to define a correspondence function that determines for each pixel in the in-between frame its corresponding pixels in the two key-frames.

The experiments were run on five subjects. Each subject performed five times each of the following six different face movement: a smile, a frown, a simple open-close of the mouth, the person making the sounds ‘oh’, ‘oo’, and ‘ee’. Note that this list consists of both speech and non-speech movements of the face. We will see (Section 4) that this list is sufficient for our application.

In order to track the motion, small uniformly colored stickers were attached to the subjects’ faces. The stickers were applied in such a way as to maximize the resolution in the areas of the face that were expected to move the most. The application pattern can be seen in Figure 1.

We first analyzed the image pattern of a given movement in a given run. For each marked point on the face, the direction in which each dot moved during the evolving of the expression was computed. This direction (eigendirection) was computed using Singular Value Decomposition (SVD) [12].

The vector data was used to create one graph per run, per person, per expression. In addition to the individual graphs, graphs displaying the mean and standard deviation of the motion per person for each expression were generated. See for example, the graphs generated for the open-close expression for a single subject in Figure 2. Finally, the data from all the runs of each expression were averaged, resulting in six graphs displaying the overall results across subjects, as shown in Figure 3. The mean is drawn in blue and the standard deviation in red. This allowed us to see if different people perform the same expression the same way.

Figure 2: Vector graphs for the open-close expression for a single subject

2.1 Results

The results can best be described by dealing separately with the various expressions, and the characteristics observed in the way the different subjects performed them. The graphs provide a fairly intuitive way to understand how the points on a face moved while performing the various expressions. The vector associated with each sticker is positioned so that its midpoint corresponds to the center of a bounding box drawn around the maximum extents of the point’s motion. It should be noted that the eigendirection vectors describe the overall dot movement, but do not necessarily provide a complete picture of the dot’s trajectory over time.

The most important result is that clear patterns of motion emerge for the different expressions. However, variations in the motion were observed, both between different runs by the same person, as well as between different subjects. In addition we found that the motion of a point is often non-linear. The points follow curved trajectories, or else exhibit hysteresis
(i.e. the point does not follow the same path when closing the mouth as it did when opened). Finally, we found that most of the time faces do not move symmetrically. As result we must define a separate correspondence function for each side of the face.

Another interesting result is how little the upper part of the face moves during most of the expressions. This implies that we can cut communication and processing costs by concentrating on the part of the face that does move, and using a much coarser algorithm for the upper face. This savings could be achieved by using fewer bits to represent the data, or making updates less frequently. If we are only interested in handling normal speech, then the region of interest can be constrained to the lower part of the face. This would be sufficient for applications such as lip reading or low bandwidth video-conferencing, possibly allowing a higher resolution image to be used with lower communication requirements.

2.2 Correspondence function

The goal of the experimental part was to generate a correspondence function between a pixel in the in-between frame and the matching pixels in the key-frames. In this section we describe this function.

The result of each run of the experiment is a function \( F \) that determines the location of a point from the key-frame in the in-between frame at time \( t \). This function clearly depends on the expression evolved by the face. Furthermore, variations in the motion were observed, both between different runs by the same person, as well as between different subjects. Therefore, we found that these variations can be parameterized. Finally, we found that this function is very complex to define as one entity. Therefore, the face is partitioned into regions with a common typical motion, as illustrated in Figure 10. These regions depend only upon the given contours and are extracted from them without user-intervention. The motion of the pixels in the region can be induced from a few parameters. As a result we obtain the function \( F(p, t, expr, region, varar) \), where \( p \) is the point location in one of the key-frames, \( t \) is the time along the expression evolving, \( expr \) is the expression under consideration (e.g., open-close of the mouth), and \( varar \) are parameters that can be measured from the image which determine the variation of the point location in the specific sequence of images. The function \( F \) should be continuous and differential. Special care should be taken to ensure these properties hold on the boundaries between the regions.

Let the two given key-frames be \( Im_0 \) and \( Im_1 \), the in-between image we wish to construct be \( Im \), and a point \( p \in Im \). We are seeking displacement functions, \( f_1(p) \) and \( f_2(p) \), of the point \( p \) from the in-between frame to the key-frame \( Im_0 \) and \( Im_1 \), respectively. These functions should be continuous and differential. If we can compute the parameters of the function \( F \), from the in-between frame, then the function \( f \) can be computed by \( f_i(p) = F^{-1}(p, t, expr, region, varar) \).

2.3 Handling the Various Regions

In this section we show practically how the expression, the region, and other variation parameters can be computed directly from the key-frames and the in-between frame. By using the function \( F \) that we learned in this experimental part, together with these parameters we show how the functions \( f_1(p) \) and \( f_2 \) can be computed.

Given a pixel \( p \in Im \) the goal is to find the corresponding pixels \( p_0 \in Im_0 \) and \( p_1 \in Im_1 \). We will now show how they are found for each region of the face.
We made an effort to find motion patterns which could be used in as many facial expressions as possible. In most cases we were able to find a single algorithm that fits all the expressions. The exception is the region above the mouth where we use two different algorithms, one for Smile, Open-Close and Ee and the other for Oo and Oh. The reason for that is that the upper lip gets more contracted in the Oo and Oh expressions than in the other expressions.

The chin region: The algorithm for the chin region is also used for the regions above and below the eyes. The algorithm has the following steps. We first draw a vertical line passing through \( p \in Im \) and find the intersection points \( c_1 \) and \( c_2 \) of this line with the contours of the chin and the lower lip. For each of these points we find the arc-length parameter \( t_i \) on the corresponding curve (see Figure 4(a)) and use these values to compute the corresponding points \( c_{ij} \) on the other two images (see Figure 4(b)). We also find \( u \) the ratio of the distance from \( p \) to \( c_2 \) with respect to the distance from \( c_1 \) to \( c_2 \).

We then find the corresponding points \( p_i \in Im_i \) corresponding to \( p \in Im \) utilizing the equation

\[
p_i = uc_{i1} + (1-u)c_{i2}.
\]

Note that the case where the motion is not completely vertical but somewhat tilted is also handled.

The cheek region: A cheek region is characterized by zero motion near the ear which gets larger for points closer to the chin. The initial angle of the motion vector is between \( 45^\circ - 75^\circ \) which approaches \( 90^\circ \) as we approach the chin. Given a point \( p \in Im \) we would like to find the angle and length of the motion vector. We denote by \( \theta_{\text{min}} \), the minimal angle of the motion vector near the ear and the first three points on the chin contour as \( c_i, i = \{0, 1, 2\} \). \( \theta_{\text{min}} \) is estimated as the average tangent direction at these first three points. We also find the horizontal ratio \( u \) of the distance from \( c_0 \) to \( p \) with respect to the distance from \( c \) to the beginning of the chin region (see Figure 5(a)).

Next we find \( L_{\text{max}} \) which is the maximal length of the motion vector between points on \( Im_0 \) and \( Im_1 \). This length is recovered from points on chin curve on the border of the chin area.

To find \( \theta \) and \( L \) in \( Im \) we use the following equation (see also Figure 5(b)).

\[
\theta = \theta_{\text{min}} + (90 - \theta_{\text{min}})u
\]
\[
L = L_{\text{max}}u
\]

Finally, we find \( p_0 \in Im_0 \) and \( p_1 \in Im_1 \) using the following equations.

\[
p_0 = p - kL(\cos \theta, \sin \theta)
\]
\[
p_1 = p + (1-k)L(\cos \theta, \sin \theta)
\]

where \( k \) denotes the closeness of \( Im \) to \( Im_0 \) as described above.
The region above the mouth: For this region two different algorithms are utilized. One for the expressions Smile, Ee and Open-Close. And the other for the expressions Oo and Oh. We will first describe the former algorithm. Given \( p \in Im \) we find the point \( c_1 \) which is closest to \( p \) on the upper lip contour and its arc-length parameter on the contour \( t_1 \). We also find the distance from \( p \) to \( c_1 \) denoted as \( L \) (See Figure 6(a)). Using \( t_1 \) we find the point \( c_{i1} \) on the lip contours of \( Im_i \). We draw a normal to the curve of length \( L \) from \( c_{i1} \) to find \( p_i \) in \( Im_i \) as illustrated in Figure 6(b)).

This algorithm is not appropriate for the Oo and Oh expressions because in these expressions the upper lip shrinks considerably. Thus we use an algorithm similar to the one described for the cheeks. Rather than elaborating on this algorithm we illustrate it in Figure 7.

The inner mouth region: One of the most challenging problems is to construct a model of the inner mouth containing the tongue and the teeth. In this paper we assume we have the image of the inner mouth whose size is on average 0.5% of the total size of the face.

The forehead region: Since the forehead hardly deforms, given a pixel \( p \in Im \) we choose \( p_i \in Im_i \) such that \( p_i = p \). This identity transformation is obviously applied after the rigid transformation which has been applied to the whole face.

The eye region: The eyes exhibit quite complex motions which have to be compensated for by the algorithm. Eyes can open and close, pupils can dilate and contract and the eye itself can move within its socket.

We assume that for each image in the sequence we are given a few points on the contours of the eyelids, the location of the center of the pupil and the radii of the pupil and the iris (a tracking system can supply this information). For each such sequence of images we construct a canonical image for each eye. Then, when we want to create the eye in image \( Im \) all is needed is a small number of parameters, and the canonical image. We will now describe how the canonical eye image is built and how the eye image of \( Im \) is reconstructed.

The canonical image of the eye is comprised of three sub-images. The white region of the eye is created from the union of all the white regions of the eye in the image sequence. This is required because in each image we see different parts of the white region because of the motion of the iris and the opening and closing of the eyelids. Similarly we compute the union of the iris images and finally we extract the image of the biggest pupil. See Fig. 8 for an illustration of this process. See Fig. 9 for the results of this algorithm which was run on an image sequence.

Suppose now that we are given the canonical image and we wish to construct the eye of \( Im \). We need to obtain a few points on the eye contours, the center of the pupil \( c_0 \), the radius of the pupil \( R_1 \) and the radius of the iris \( R_2 \). Given a point \( p \in Im \) in the eye we wish to reconstruct, we determine to which of the three regions it belongs and copy the color value from the respective eye component image.

3 Facial Expression Animation

The goal of the animation system is to construct a full sequence of face images while moving the head, talking and changing expressions. The input to the animation system is a few
images of a given face, and a set of image parameters that describe the location of a few facial features for each of the in-between frames. These image parameters can be determined either manually when a new movie is generated or automatically by a tracking system (e.g., [21, 4]) when used for video compression. The image parameters contain a few control points (between 10 and 20) on the lip, chin, and eye contours, and the pixels that build up the mouth interior.

The animation algorithm consists of two main parts: at first, a rigid transformation is found to compensate for head movements and then the non-rigid transformation is recovered to deal with facial expressions. Extensive work has been done in the area of recovery of rigid transformations [4, 7, 5]. We will discuss it and describe the method we use later. Our main contribution however, is in recovering the non-rigid transformations which do not have simple mathematical descriptions. This method will be presented now.

### 3.1 Non-Rigid Transformation Recovery

To generate an in-between frame we must determine the grey-level (color) of each pixel in the in-between frame. We first must determine the correspondence between a pixel in the in-between frame and the matching pixels in the key-frames. To do so we use the experimental part of this work. In the experimental part we found that the correspondence can be determined by a correspondence function. The parameters of the correspondence function are the expression, the face region, and some parameters that can be computed from the shape and location of special face contours (e.g., the lips). Our method therefore consists of four stages: contour construction, contour correspondence, pixel correspondence, and color interpolation. We elaborate on each of these stages below.

1. **Contour construction**: We are given a few points on the lips, chin, and eyes contours (5–10 points on each contour). These points can be either specified by the end-user or can be recovered automatically by a tracking system (e.g., [21]). Our algorithm constructs B-splines which follow the contours using the control points. These contours are extracted both for the in-between frame and for the key-frames.

2. **Contour correspondence**: At this stage a correspondence between points on the contours is established. The correspondence is found between the various contours on the three images – the two key-frames and the in-between frame that needs to be constructed. Our correspondence strategy is based on arc-length parameterization. Arc-length parameterization can effectively handle stretches of the contours. We require that each point on the contour in the in-between-frame will be mapped by the correspondence function to points on the key-frames. Formally, let $C, C_1, C_2$ be the corresponding contours on the in-between and the two key-frames. Let $c \in C$, then $f_i(c) + c \in C_i$.

3. **Pixel correspondence**: Given the location of a pixel $p \in Im$, the objective is to find two corresponding pixels, $p_0 \in Im_0$ and $p_1 \in Im_1$, such that $p_0 = f_0(p) + p$ and $p_1 = f_1(p) + p$. This is done for every pixel location in image $Im$. This correspondence is based both upon the templates found for the various expressions (as described in Section 2) and upon the contour correspondence previously computed.

We now use $D_j \subset Im$ to denote the regions of the face. Similarly, we denote the regions of the face in the input images as $D_{ji} \subset Im_i$, $1 \leq j \leq \text{no regions}$, $i = 0, 1$. We require that
the following equation holds for the contours bordering the regions $D_j$:

$$f_i(\partial D_j) + \partial D_j = \partial D_{ji}.$$  

The above conditions under-constrain the required function. This is actually the situation in which optical-flow based systems operate. The optical flow of points on the contours can be computed quite accurately. However, in the regions between the contours the optical-flow is estimated very poorly and therefore the resulting optical-flow is an interpolation of the optical-flow values which were found on the boundaries.

Therefore we use the empirical knowledge in order to add enough constraints and to guarantee the correct reconstruction. We have several types of functions $F_j$ (and therefore, several types of functions $f_j$). The function $F_j$ is very simple for the rigid regions and more complicated for the non-rigid regions (of course a rigid function is a simple special case of a non-rigid function).

Generally, we proceed in three stages: Finding the locations of the contour points in $Im$ related to the point $p$, finding the corresponding contour points in $Im_0$ and $Im_1$, and finally, finding the corresponding pixels $p_0$ and $p_1$.

Note that the pixel correspondence function has to be continuous both within the regions and on the borders between the various regions. In Section 2.3 we will discuss the function for each region of the face in detail.

4. Color Interpolation: During the previous stage, two corresponding pixels $p_0 \in Im_0$ and $p_1 \in Im_1$ were found for a given pixel $p$ in the output frame. The goal of the current stage is to compute the value (intensity or color) of $p$. We found that a linear interpolation between the two corresponding pixels is sufficient. We compute the distances between the mouth curve in the in-between image $Im$ to the mouth curves in the static images $Im_0$ and $Im_1$. The coefficients used for the linear interpolation are the relative distances between the mouth curves on the various images. Let $intensity_0$ (resp. $intensity_1$) be the value of $p_0$ (resp. $p_1$). Let $k$ be the relative distance between the mouth curve in $Im$ to the mouth curve in $Im_0$. The resulting intensity of the pixel is thus

$$intensity = (1 - k) \times intensity_0 + k \times intensity_1$$

There are various ways to find distances between contours. We use a very simple scheme which finds the average distance between the corresponding points on the curves, using the correspondence found in stage 2. Other methods for finding distances between curves can be used as well (e.g., see [6, 8]).

3.2 Rigid Transformations

Before we start compensating for the motion due to expressions we have to compensate for total head movement. We would like to recover the 3D motion of the head. But as we do not have a 3D model we are looking for a transformation which will compensate as much as possible for this movement. We thus assume that the face is a planar surface and estimate the projective transformation that this plane undergoes from image to image.

There are two ways to recover this transformation, one which uses point correspondences to compute the transformation and the other which tries to minimize an error function of the difference in color values in the two images.
We tried to use the first method by choosing four corresponding points in each image. To get the best estimate we chose the points to be nearly co-planar. This method yielded very inaccurate results due to this method’s non-robustness to small uncertainties in point positions.

We therefore turned to the second method which tries to find a transformation which minimizes the difference between the transformed image and the other image as follows:

A projective transformation of a planar point \((x, y)\) in that plane moves it to \((x', y') = (x + u(x, y), y + v(x, y))\) where \(u\) and \(v\) have the following functions.

\[
\begin{align*}
  u(x, y) &= a_0 + a_1x + a_2y + a_6x^2 + a_7xy \\
  v(x, y) &= a_3 + a_4x + a_5y + a_6xy + a_7y^2
\end{align*}
\]

When we are given the parameters \(a_i\) the transformation can simply be applied. When however, we are given images whose rigid transformation needs to be determined, our mission is to find \(a_0 \cdots a_7\) such that the transformed image and the other image are most similar, i.e.

\[
E = \sum_{(x, y) \in \mathbb{R}^2} \rho(I_2(x + u, y + v) - I_1(x, y))
\]

where \(I_1(x, y)\) and \(I_2(x, y)\) are the image intensities at pixels \((x, y)\) in the two images, \(E\) is the error function, and \(\rho\) is a function applied to the error at a pixel. The standard choice for \(\rho(w)\) is \(\rho(w) = w^2\) which is the least squares error estimate. This choice is not appropriate when the model does not explain totally the motion as is the case with faces where various other motions are involved [17]. We therefore have to choose a more robust function which is not drastically affected by large errors in small parts of the face. We therefore chose to use the Geman & McClure function:

\[
\rho(w, \sigma) = \frac{w^2}{\sigma + w^2}
\]

where \(\sigma\) is the control parameter of \(\rho\) which determines the tolerance for large errors. The larger \(\sigma\) is, \(\rho\) is tolerant for larger errors.

We minimize the error using the Simultaneous Over-Relaxation method (SOR) [5]. This method starts with an initial guess for the parameters \(a_i\) and using a gradient descent method converges to a local minimum of the error function. However, to converge to the global minimum an initial guess which is close to that minimum should be found. The biggest problem is to find an initial guess for the 2D translation component which might be quite large where as the rotation components are usually quite small and can be estimated as zero. We therefore use a pyramid based method to estimate it. At first the algorithm is run on a small image which is a scaled down version of the original image. The result is then used as the initial guess for an image twice as big in both coordinates. This process continues until the algorithm is run on the original image. During the iterative optimization procedure the parameter \(\sigma\) of the function \(\rho\) is reduced causing the optimized function to be less tolerant to errors (Graduated Non-Convexity [7]).

4 Results

To test our algorithm, we recorded movies of several people performing the various expressions. (This group of people is different from the group of people whose expressions were
analyzed in the empirical experiments.) Reconstructing in-between images of the movie is the best way to test the quality of the algorithm both because we can extract life-like image parameters from the real images and because we can compare the reconstructed images to the real images from the sequence to evaluate the quality of the results.

We first tested the algorithm on image sequences in which actors were asked to perform a single expression. The final experiment was on a image sequence recorded from a television news broadcast. The full sequences can be viewed in http://www.ee.technion.ac.il/~ayellet/movies.html.

Our algorithm was given the first and the last frame of each expression and the image parameters of the in-between frames. Some of the results are demonstrated below. Figures ??-?? show snapshots from three of the movies that were generated by our algorithm. Figure ?? shows the snapshots from a movie that generated an open-close expression. Figure ?? shows the snapshots from a movie that generated an oh expression. Figure ?? shows the snapshots from a movie that generated a smile expression.

To test the quality of our algorithm, we compared the actual in-between images to the images synthetically generated by our system. Some of the results are demonstrated below. For each expression, we show the two given images, a couple of in-between images as generated by our system, the actual in-between images from the real movie, and images that compare the two (i.e., the inverse of the picture generated by subtracting the colors of the pixels of the real and synthesized images). Figures ??-?? show a few such results. An “OH” expression is illustrated in Figure ?? An “OO” expression is shown in Figure ?? A smile is demonstrated in Figure ?? An “EE” expression is illustrated in Figure ?? Finally, the open-close expression is shown in Figure ??.

In the final experiment we recorded a movie of a news broadcaster saying a sentence. The sequence is 72 frames long (eight words). The sequence is divided into expressions and the algorithm is applied. Only five full frames are used to reconstruct the full sequence and the results are very good as can be seen in Figures ??-??.

As the results are hardly distinguishable from the original image sequence, our method can also be used for video compression. In this case, it is important to know how much storage is required with respect to other compression techniques. On average, we transfer one full frame in fifteen. For each of the in-between frames we transfer a few parameters describing the contours (approximately 20 pixel locations), which can be neglected. Moreover, for each of the in-between frames we transfer the pixels on the inner mouth, which might vary between 0 pixels (mouth is closed) to 50 × 50 pixels (mouth is open) for an image of a size of 768 × 576 pixels. In addition we transfer one image of the eyes (for the whole sequence) of approximate size 100 × 200. Thus on average, we get a storage reduction in the order of 1:13.

It is important to mention that the full images that need to be stored or transmitted, as well as the images of the inner mouth of the in-between frames, can be compressed by any of the well-known compression schemes, on top of our technique. When this is done, we get a much better storage reduction. For instance, if standard JPEG is used on top of our proposed technique, assuming a compression by a factor of 1:30–1:40, we can get a compression by a factor of about 1:390–1:520 on average.

Finally we compared the compression rate of our algorithm to that of MPEG2. MPEG2 should work wonders on this frame sequence since the background does not change and the motions are very small which is exactly what MPEG2 exploits in its compression scheme. We compressed the original sequence using MPEG2 and compared it to our key frames compressed using MPEG2. The resulting movie is four times smaller than the original
MPEG2 movie. Our compression rate is only 4 and not 13 because the difference between the key-frames is larger than the difference between consecutive frames in the original movie. We got this major improvement over MPEG2 because we exploit our knowledge of the typical motion of the face while undergoing expressions.

5 Conclusion

The human face is one of the most important and interesting objects encountered in our daily lives. To be able to animate the human face is an intriguing challenge. Realistic facial animations can have numerous applications, among which are generation of new scenes in movies, dubbing, video compression, video-conferencing, and intelligent man-machine interfaces.

In this paper we have explored the issue of generating movies of facial expressions given only a small number of frames from a sequence, and very few image parameters for the in-between frames. Since the change of expressions cannot be described mathematically, we have been empirically studying image patterns in facial motion. Our experimentations revealed that patterns do exist.

The empirical results served as a guide in the creation of a facial animation system. A major contribution of this paper is in showing how an “ideal” motion stored in a template can be used to generate animations of real faces, using a few parameters.

The results we achieved are very good. The animations created are highly realistic. In fact, the comparisons between the synthesized images created by our system and the original in-between images show that the differences are negligible. On average, one full image in fifteen is needed, and we get a compression rate of 1:13. Combining our scheme with MPEG2 yields results which are four times better than using only MPEG2.

There are several possible future directions. First, the empirical results can be applicable to a wide range of uses, most notably – expression recognition. Another issue we are exploring is how to build a model of the inner mouth. The goal is to make it possible to further reduce the storage and the bandwidth requirements. Third, incorporating illumination models into the system seems an intriguing problem. Currently, drastic changes in illuminations are handled by sending more images as a key-frames. Finally, we are planning to use our algorithm in various applications such as distance-learning systems and dubbing systems.

References


Vector graph for average Open-Close  Vector graph for average Oo  Vector graph for average Oh
Vector graph for average Ee  Vector graph for average Smile  Vector graph for average Frown

Figure 3: Vector graphs

(a) In the destination image $I_m$  (b) In the source images $I_{m_i}$, $i = 0, 1$

Figure 4: The chin region

(a) In the destination image $I_m$  (b) In the source images $I_{m_i}$, $i = 0, 1$

Figure 5: The cheek region

(a) In the destination image $I_m$  (b) In the source images $I_{m_i}$, $i = 0, 1$

Figure 6: The region above the mouth in the smile, ee and open-close expressions

Figure 7: The region above the mouth in the oo and oh expressions

Figure 8: Extracting the eye parameters

Figure 9: The result of the extraction

(a) a diagram  (b) the different regions of the news broadcaster’s face

Figure 10: An illustration of the different regions into which the face is divided. These regions are computed automatically from the given lip and chin contours.