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.

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

[*]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, ilans@ie.technion.ac.il

[§]Department of Computer Science, Interdisciplinary Center, Herzeliya Israel, yael@idc.ac.il

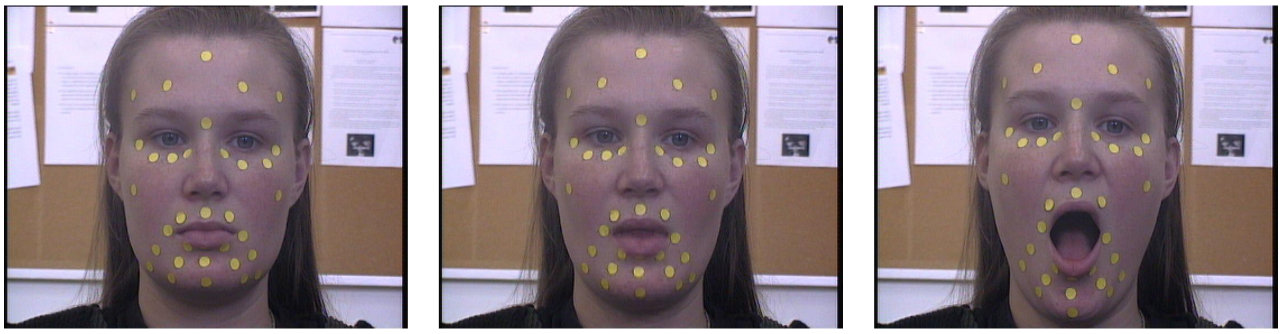


Figure 1: Sample frames showing dot placement and movement

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. In addition, in [15] the inter-viseme video sequence is generated by optical flow while we use model-based optical flow as discussed below.

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 patters 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 motion templates can generate facial movies. These parameters are extracted automatically from the image sequence. We show that given very few images and a few image parameters, 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. It is possible to combine our animation algorithm with standard compression schemes to get higher compression rates. Using our algorithm above MPEG2 reduces the size of the movie by a factor of 4 with respect to MPEG2. When the algorithm is used for compression, an accurate tracking system is required.

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

In this section we describe our experiments for finding image patterns in facial expressions. The goal of the empirical research was to find parameters that can characterize the various expressions and which can be used to produce animations. Thus, our empirical research was guided by two main objectives. The first was to see how much variance there was in the way a person performed the same expression on different runs. Then, we wanted to determine if there was a correlation between the way different people perform the same expression.

In order to provide a broad enough sample base, five subjects were selected, and each expression was repeated five times. Expressions representative of the types encountered in normal conversational speech were selected. First, a simple mouth open-close operation was performed, followed by the sounds ‘oh’, ‘oo’, and ‘ee’. A smile and a frown were then performed, as examples of non-speech expressions that commonly occur during a normal conversation. In between the execution of subsequent expressions, the subject returned to a neutral position, with mouth closed.

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.

Once the data were collected, several stages of analysis were performed in order to chart the motion of the dots over time. First, the data points associated with each sticker were grouped together. Singular Value Decomposition (SVD) was performed on each set of points, in order to determine the *eigendirection* in which each dot moved. The *singular value decomposition* (SVD) of a matrix X yields three matrices, such that

$$X = U\Sigma V^{\mathrm{T}}.$$

Σ is a diagonal matrix holding the *singular values* of X. The first column of U holds the normalized x and y components of the directional vector which is the principle axis of an ellipsoid drawn to best fit the data points [12]. The SVD was used to obtain a normalized vector indicating the direction in which the maximum variance of the data points was found. The lengths of the vectors were determined by projecting the data points along the directional eigenvector.

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.

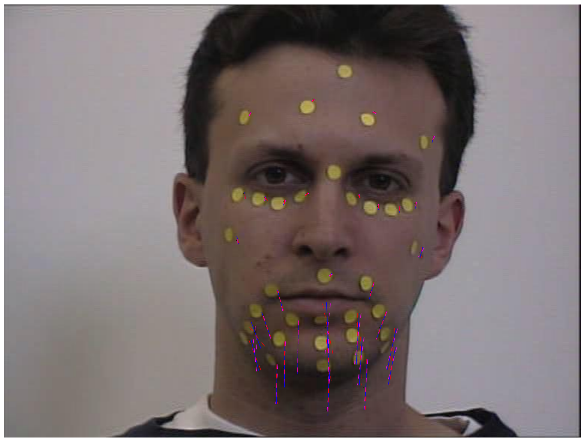


Figure 2: Vector graphs for the open-close expression for a single subject

After examining the data, several important observations can be made. The most important result is that clear patterns of motion emerge for the different expressions. The uniformity means that it is safe to make the assumption that at least within certain bounds, people perform the same expression the same way. Therefore, algorithms can be developed based on these results, and should be applicable to the majority of people performing these expressions.

Variations in the motion were also observed, both between different runs by the same person, as well as between different subjects. These variations can be attributed to several sources. First, the subjects were human, and not robotic automata, and therefore exhibited a certain degree of individuality. For this same reason, they did not always perform the same expression the same way. Variations were largest for those expressions which allowed the greatest freedom in the way they could be performed. For example, although there is only one way to open and close the mouth, the 'oo' sound can be made with mouth open or closed, lips pursed or relaxed.

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 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.

For the graphs involving data averaged over several runs, three vectors were drawn. The same length, namely the mean, was used for all three, with the direction changing to illustrate the mean $\pm$ standard deviation. All three vectors are drawn with their centers located at the average midpoint of the various runs. We summarize our findings for the six expressions below.

- *Open-Close:* Of the expressions, Open-Close was the easiest to recognize from the graphs, because of the large vertical displacement of the points on the chin. (This motion, however, also made it difficult to track, because the locations of the points on the chin overlapped as the mouth was opened.)

- *OO:* Generally, the 'oo' expression was characterized by small motions. The corners of the mouth move in, and the mouth opens slightly. Some of the subjects barely moved their mouth's at all, others clearly pursed their lips.

- *OH:* The 'oh' was a cross between the Open-Close and 'oo'. The mouth opened, and the edges of the mouth came slightly together.

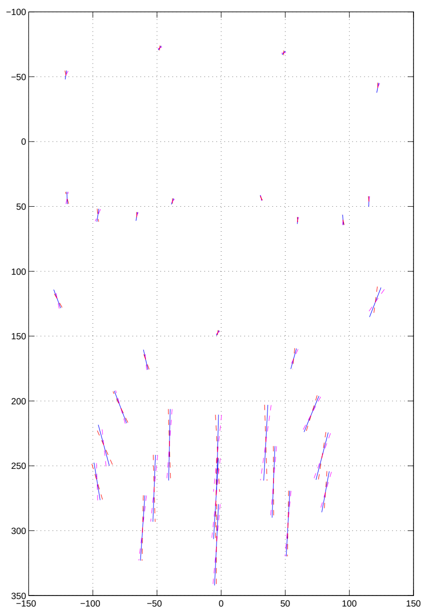
- *EE:* The 'ee' can be considered to be the "opposite" of the 'oo'. The corners of the mouth move out as opposed to in. Again, the mouth opens slightly.

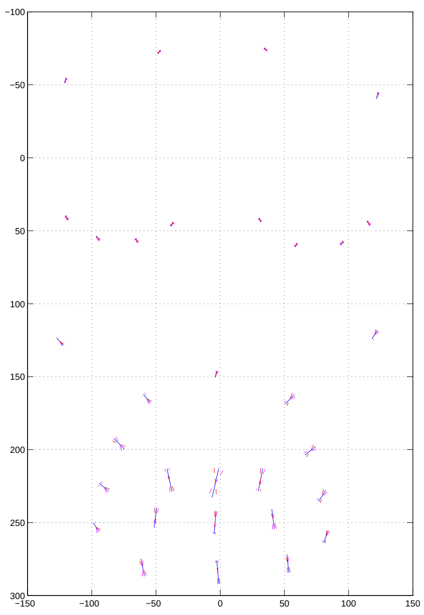
- *Smile:* In the smile expression, the corners of the mouth move up and out. This was the one expression where the points under the eyes moved a significant amount, almost straight up and down.

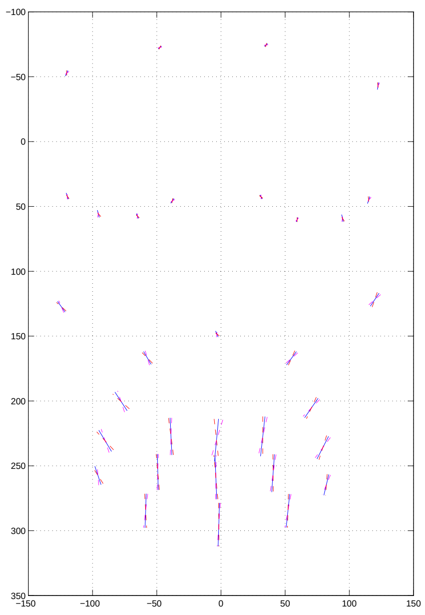
- *Frown:* The frown was the expression that the subjects had the most difficulty with. Different subjects performed completely different motions under the guise of 'frown'. There was also a variance between runs of the same person. In addition, it was the most difficult expression to analyze. Different points tended to move in different directions, even points in close proximity to each other.

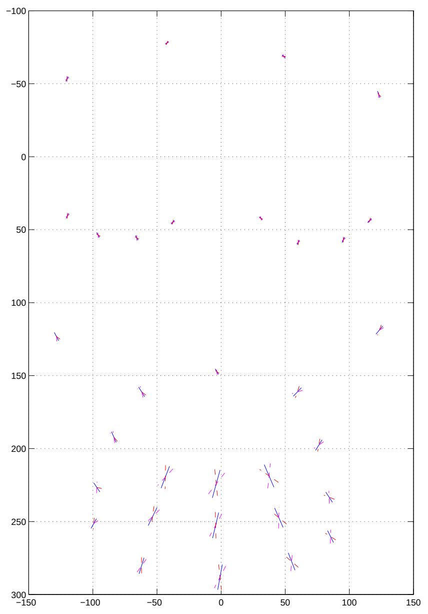
Ideally, each expression would be performed in a unique manner, the motion of the points would be large, linear, and smooth, and the motion would be symmetrical between the left and right sides of the face. In practice, however, there were variations between the way the different subjects performed the expressions. In general, the subjects exhibited various aspects of the following categories. There was no single person who completely fit a single profile. Nevertheless, such a classification is useful, as it allows us to focus on the types of difficulties encountered when video of real people is used as input.

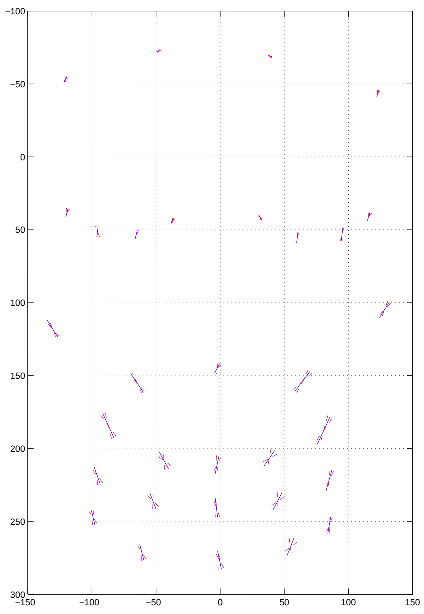
- *Non-Linear:* The first group that deviated from the ideal were those who exhibited non-linear motion. The points follow curved trajectories, or else exhibit hysteresis (i.e. the point does not follow the same path when closing the mouth as it did when opened).


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

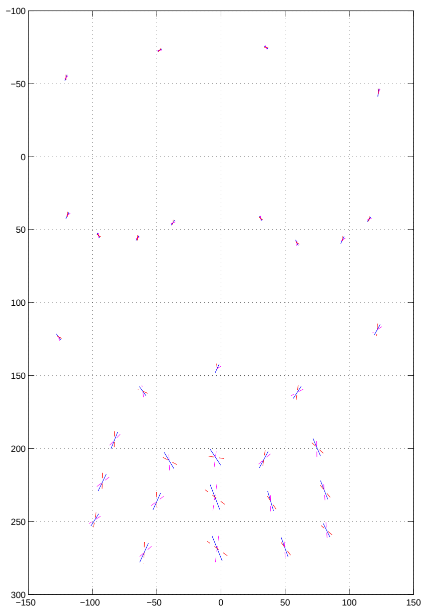

Vector graph for average Frown

Figure 3: Vector graphs

- *Non-Symmetric:* Many tasks would be made easier if everyone moved their faces symmetrically. If that were the case, then we could compute the motions from one side of the face, and merely reflect them for the other side. Unfortunately, this was not always the case

- *Inconsistent:* All of the subjects were inconsistent to a certain degree between runs of the same expression. For instance, sometimes the jaw would shift to the left, other times to the right. In some cases, the mouth was opened for the 'oo', other times not. Of the expressions, the frown produced the least consistent results, with sometimes wild fluctuations from run to run.

One of the most interesting results is how little the upper part of the face moves during most of the expressions. (There are other expressions, however, in which the eyebrows move as well.) 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.

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, a set of image parameters that describe the location of a few facial features for each of the in-between frames, and the type of expression. 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 output of the animation system is the sequence of the in-between frames and the movie built out of it.

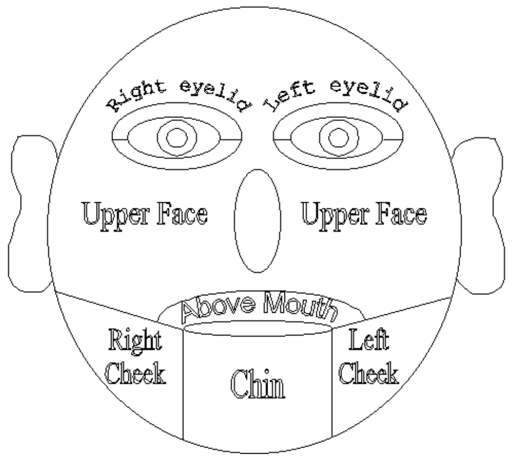
The study of pixel movements presented in the previous section yields a set of reconstruction algorithms for each expression and each facial region. We view each such algorithm as a motion template parameterized according to the specific face at hand. These parameters are extracted from the image sequence as described below. In other words, for each expression we have a function $\mathcal{F}(p_0, t)$ which gives for each point p_0, $p_0 \in$ template, at time t, $t \in [0,1]$ during the expression the "ideal" motion vector of the frame at time t. The vector is represented by its length $l = ||\mathcal{F}(p_0,t)||$ and its direction $\vec{N}$, such that $\mathcal{F}(p_0,t)/l = \vec{N}$.

In practice, however, 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 4. 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. We denote the regions of the face as $D_j \subseteq$ template, for $1 \leq j \leq$ no_regions. For each region we denote by $\mathcal{F}_j(p_0,t)$ the function defined on that region, $1 \leq j \leq$ no_regions, $p_0 \in D_j$, $t \in [0,1]$, such that

$$\mathcal{F}(p_0,t) = \bigcup_j \mathcal{F}_j(p_0,t).$$

The function $\mathcal{F}$ 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 static images 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 functions $f_i(p)$, $i = 0,1$, over the vector field such that



(a) a diagram

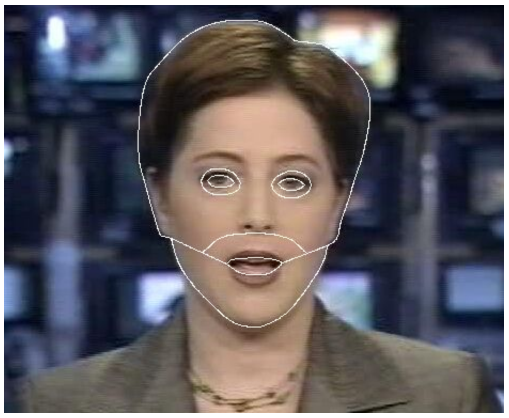

(b) the different regions of the news broadcaster's face

Figure 4: An illustration of the different regions into which the face is divided. These regions are computed automatically from the given lip and chin contours.

$f_i(p)$ is continuous, differential and

$$f_i(p) = v_i = p_i - p$$

where $p_i \in Im_i$, $i = 0, 1$. That is to say that v_i describes the motion of a point p to its position in Im_i. Had the image sequence been of an "ideal" subject, we could have derived these functions using $\mathcal{F}^{-1}(p, t_{Im})$ where t_{Im} is the time of Im in the sequence. However, as "ideal" subjects are hard to find, $\mathcal{F}^{-1}(p, t_{Im})$ has to be modified to fit the subject. This is not difficult to do because $\mathcal{F}$ is parameterized by a small number of motion vectors for each region while the motion vectors for all the pixels can be inferred from them. These motion vectors are computed from the given control points on the contours, and the time t_{Im} is computed from the relative distances between the contours of the three images as will be described below.

The facial expression 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

Our method for recovering the non-rigid transformation 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 the control points. These contours are extracted both for the in-between image Im and for the static images Im_i, $i = 0, 1$.

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 static frames Im_i, $i = 0, 1$, and the in-between image Im that needs to be constructed.

Let a contour $C_i \subseteq Im_i$, $i = 0, 1$, and the matching contour $C \subseteq Im$. We require that the following equation holds for every point $c \in C$ (on the contour): $f_i(c) + c \in C_i$. Our correspondence strategy is based on arc-length parameterization. Arc-length parameterization can effectively handle stretches of the contours.

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 $D_j \subseteq Im$ to denote the regions of the face. Similarly, we denote the regions of the face in the input images as $D_{ji} \subseteq Im_i$, $1 \leq j \leq$ 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 $\mathcal{F}_j$ (and therefore, several types of functions f_j). The function $\mathcal{F}_j$ 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 3.2 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

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

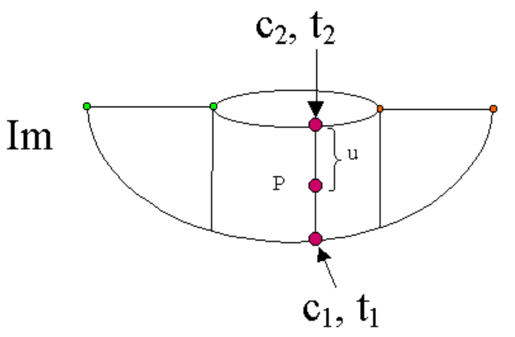
There are various ways to find distances between contours. We use a very simple scheme which finds the average distance between the corresponding points on the curves, using the correspondence found in stage 2. Other methods for finding distances between curves can be used as well (e.g., see [6, 8])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

3.2 Handling the Various Regions

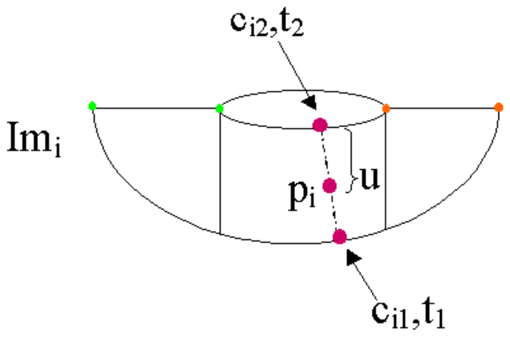
Given a pixel $p \in Im$ the goal is to find the corresponding pixels $p_0 \in Im_0$ and $p_1 \in Im_1$. We will now show how they are found for each region of the face.

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 5(a)) and use these values to compute the corresponding points c_{ij} on the other two images (see Figure 5(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.



(a) In the destination image Im



(b) In the source images Im_i, $i = 0, 1$

Figure 5: The chin region

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 $45° - 75°$ which approaches $90°$ 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 θ_{min} the minimal angle of the motion vector near the ear and the first three points on the chin contour as $c_i, i = \{0, 1, 2\}$. θ_{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 6(a)).

Next we find L_{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.

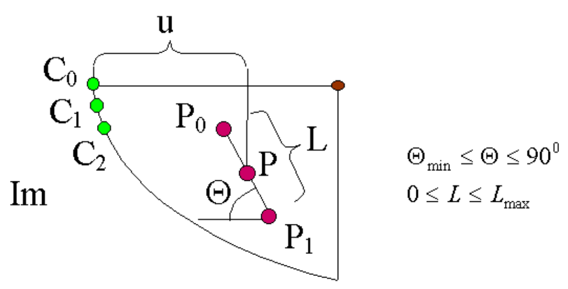
To find θ and L in Im we use the following equation (see also Figure 6(b)).

$$\theta = \theta_{min} + (90 - \theta_{min})u$$

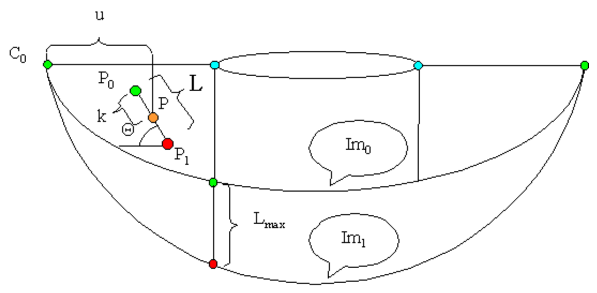
$$L = L_{max}u$$

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

$$p_0 = p - kL(\cos\theta, \sin\theta)$$



(a) In the destination image Im



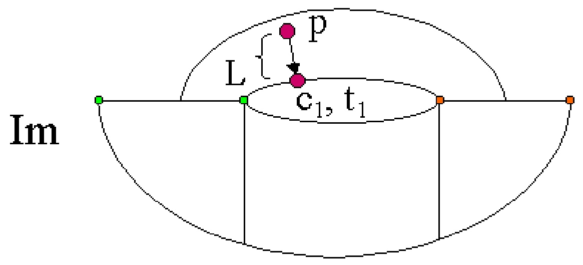
(b) In the source images Im_i, $i = 0, 1$

Figure 6: The cheek region

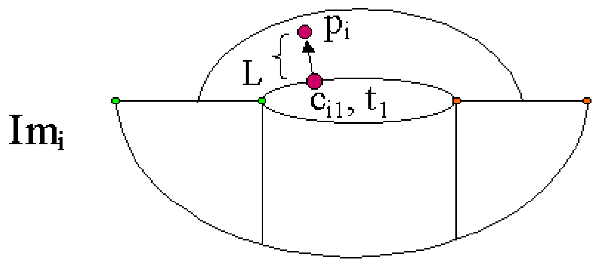
$$p_1 = p + (1 - k)L(\cos\theta, \sin\theta)$$

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 7(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 7(b)).



(a) In the destination image Im



(b) In the source images Im_i, $i = 0, 1$

Figure 7: The region above the mouth in the smile, ee and open-close expressions

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 8.

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.

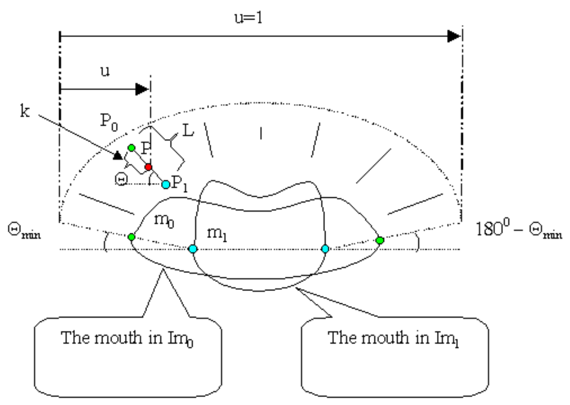


Figure 8: The region above the mouth in the oo and oh expressions

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. 9 for an illustration of this process. See Fig. 10 for the results of this algorithm which was run on an image sequence.

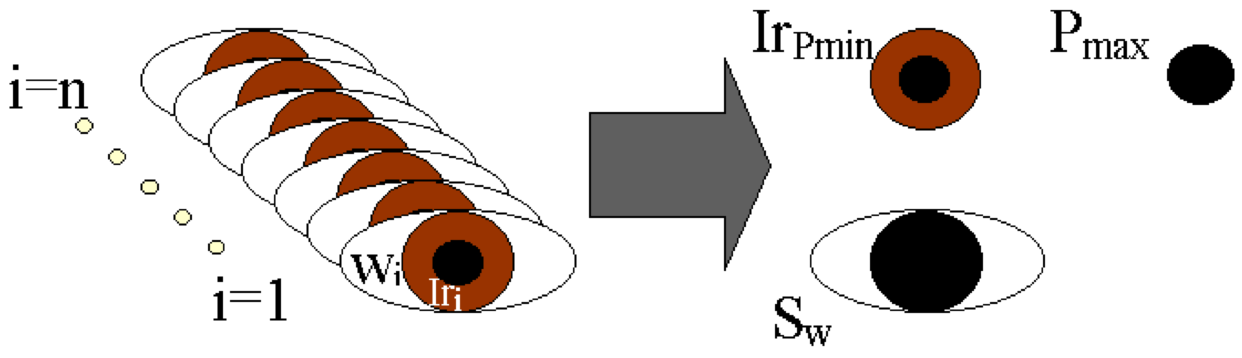


Figure 9: Extracting the eye parameters

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.

3.3 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

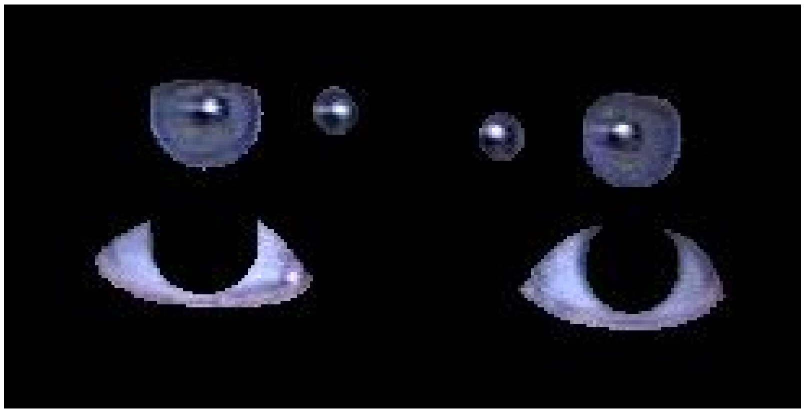


Figure 10: The result of the extraction process: For each of the two eyes three regions are shown: the white of the eye, the iris, and the pupil

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.

$$u(x,y) = a_0 + a_1x + a_2y + a_6x^2 + a_7xy$$
$$v(x,y) = a_3 + a_4x + a_5y + a_6xy + a_7y^2$$

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.

$$E = \sum_{(x,y)\in\Re^2} \rho(I_2(x+u, y+v) - I_1(x,y))$$

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 ρ 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 σ is the control parameter of ρ which determines the tolerance for large errors. The larger σ is, ρ 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 σ of the function ρ is reduced causing the optimized function to be less tolerant to errors (Graduated Non-Convexity [7]).

The above procedure is used to extract the rigid motion parameters $a_0 \cdots a_7$ between two images. We apply this procedure twice: once between the input image Im_0 and the image we are constructing Im, and the second time between Im_1 and Im. Using the extracted parameters we transform Im_0 and Im_1 and their respective contours to the head position of Im. The non-rigid transformation described in Section 3.1 is applied to the transformed images.

4 Experimental 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/facial-reconstruction.html.

Our algorithm was given the first and the last frame of each expression and the image parameters of the in-between frames. The parameters of the rigid transformation were extracted automatically. The points used for computing the facial contours were marked manually ($10 - 20$ pixel locations).

Some of the results are demonstrated below. Figures 11–13 show snapshots from three of the movies that were generated by our algorithm. Figure 11 shows the snapshots from a movie that generated an open-close expression. Figure 12 shows the snapshots from a movie that generated an oh expression. Figure 13 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 14–18 show a few such results. An "OH" expression is illustrated in Figure 14. An "OO" expression is shown in Figure 15. A smile is demonstrated in Figure 16. An "EE" expression is illustrated in Figure 17. Finally, the open-close expression is shown in Figure 18.

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 manually 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 19-22.

As the results are hardly distinguishable from the original image sequence, our method can also be used for video compression, in case a tracking system extracts the parameters accurately. 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. When the method is used for compression, one full image in fifteen is needed on average, and we get a compression rate of 1:13. In this case an accurate tracking system is required. 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 illumination 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

[1] T. Beier and S. Neely. (1992) Feature–based image metamorphosis. In *Proceedings of SIGGRAPH '92*, Vol 26(2), pp. 35-42, June 1992.

[2] Bergeron P. and Lachapelle P. (1985) Controlling facial expressions and body movements in the computer-animated short "Tony De Peltrie" In *ACM SIGGRAPH Advanced Computer Animation Seminar Notes*

[3] Beymer D. and Poggio T. (1996) Image representation for visual learning *Science* 272:1905-1909.

[4] Black, M.J and Yacoob, Y. (1995). Tracking and recognizing rigid and non-rigid facial motions using parametric models of image motion. In *Proc. Fifth International Conference on Computer Vision, ICCV'95*, pages 374–381.

[5] Black, M.J and Anandan, P. (1996). The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. Computer vision and image understanding, Vol 63, No 1, January 1996, pages 75–104.

[6] Blake, A. and Isard, M. (1994). 3D position, attitude and shape input using video tracking of hands and lips. In *Proc. ACM SIGGRAPH Conference*, pages 185–192.

[7] Blake, A. and Zisserman, A. (1987). Visual reconstruction. The MIT Press, Cambridge, MA.

[8] Bregler C., Covewll M. and Slaney M. (1997). Video rewrite: Driving visual speech with audio.. In *Proc. ACM SIGGRAPH Conference*, pages 353-360.

[9] Cassell J., Pelachaud C., Badler N., Steedman M., Achorn M., Becket T., Douville B., Prevost S. and Stone M. (1994) Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Proceedings of SIGGRAPH '94*, 28(2) pp. 413-420.

[10] Cootes T.F, Edwards G.J and Taylor C.J. (1998) Active appearance models In *Computer Vision – ECCV'98* Vol II Springer LNCS 1407, 484–498.

[11] DeCarlo D., Metaxas D. and Stone M.. (1998). An anthropometric face models using variational techniques. In *Proc. ACM SIGGRAPH Conference*, pages 67-74.

[12] Duda, R.M., Hart, P.E. *Pattern classification and scene analysis*, Wiley, 1973.

[13] Essa I., Basu S., Darrell T. and Pentland A. (1996) Modeling, tracking and interactive animation of faces and heads using input from video. In *Computer animation conference*, pp. 68-79, 1996.

[14] Essa I. and Pentland A. (1997) Coding analysis, interpretation and recognition of facial expression. In *IEEE PAMI*, Vol 19(7) pp. 757-763.

[15] Ezzat T. and Poggio T. (1998). MikeTalk: A Talking Facial Display Based on Morphing Visemes. In *Proc. Computer Animation Conference.*

[16] Guenter B., Grimm C., Wood D., Malvar H. and Pighin F. (1998). Making faces. In *Proc. ACM SIGGRAPH Conference*, pages 55–66.

[17] Hampel F.R., Renchetti E.M., Rousseeuw P.J, and Stahel W.A. (1986). Robust statistics: The approach based on influence functions. *Wiley, New York.*

[18] Lee S-Y., Chwa K-Y., Shin S. Y. and Wolberg G. (1995). Image metamorphosis using snakes and free-form deformations, *SIGGRAPH '95*, pp 439–448, 1995.

[19] Lee Y., Terzopoulos D. and Waters K. (1993). Constructing physics-based facial models of individuals. In *Proceedings of Graphics Interface*, pp. 1-8.

[20] Lee Y., Terzopoulos D. and Waters K. (1995) Realistic modeling for facial animation. In *ACM SIGGRAPH*, pp. 55-62.

[21] Moses, Y., Reynard, D., and Blake, A. (1995). Determining facial expressions in real time. In *Proc ICCV-95*, pages 296–301. IEEE Computer Society Press.

[22] Parke F.I. (1972). Computer generated animation of faces. In *Proceedings ACM annual conference.*

[23] Parke F.I. and Waters K. (1996). *Computer facial animation.* A K Peters, Wellesley, MA.

[24] Pighin F., Hecker J., Lischinski D., Szeliski R. and salesin D.H. (1998). Synthesizing realistic expressions from photographs. In *Proceedings of SIGGRAPH '98*, pp. 75-84.

[25] Sali, E. and Ullman, S. (1998). Recognizing novel 3-D objects under new illumination and viewing position using a small number of examples views or even a single view. In *Proceedings of International Conference on Computer Vision.*

[26] S. M. Seitz and C. R. Dyer (1996). View morphing: Synthesizing 3D metamorphosis using image transforms In *Proceedings of SIGGRAPH '96*, pp 21–30, 1996.

[27] Shashua, A. (1992). Illumination and view position in 3D visual recognition. In Moody, J., Hanson, J. E., and Lippman, R., editors, *Advances in Neural Information Processing Systems 4*, pages 68–74. Morgan Kaufman.

[28] Terzopoulos D. and Waters K. (1990) Physically-based facial modeling, analysis, and animation. *Journal of Visualization and Computer animation*, 1(4):73-80.

[29] Ullman, S. and Basri, R. (1991). Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:992–1005.

[30] Vetter, T. and Poggio, T. (1997). Linear object classes and image synthesis from a single example image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:733–742.

[31] Waters K. (1987) A muscle model for animating three-dimensional facial expression. IN *ACM SIGGRAPH conference proceedings*, vol 21, pp. 17-24.

[32] Williams L. (1990) Performance-driven facial animation In *ACM SIGGRAPH conference proceedings*, vol 24, pp.235-242.

[33] Wolberg G. Digital image warping. *IEEE Computer Society Press*, 1990.

[34] Yacoob, Y., Davis, L.S., (1996) Recognizing human facial expressions from long image sequences using optical flow. *IEEE PAMI* Vol 18(6), pp. 636-642.

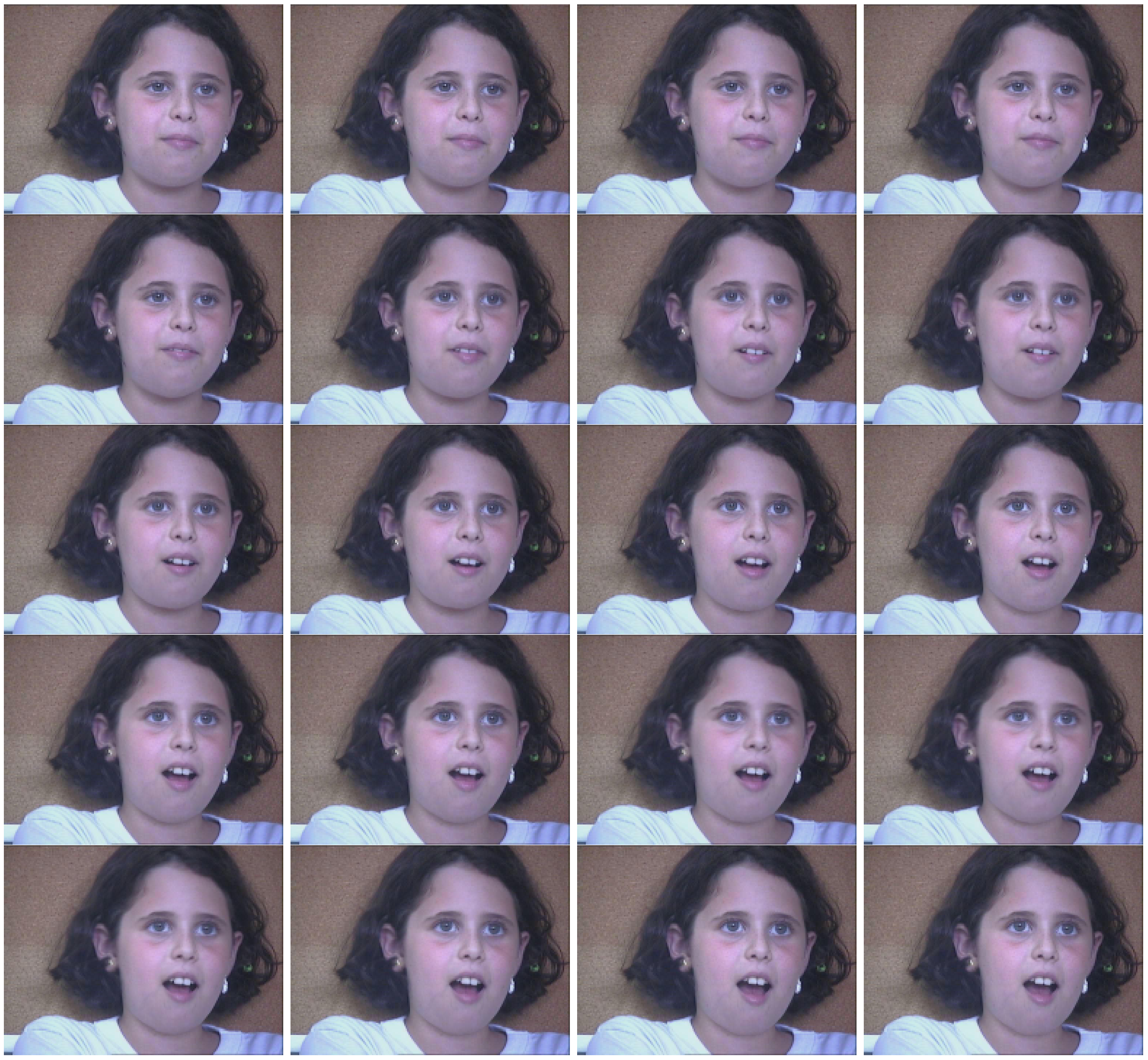


Figure 11: The open-close movie snapshots

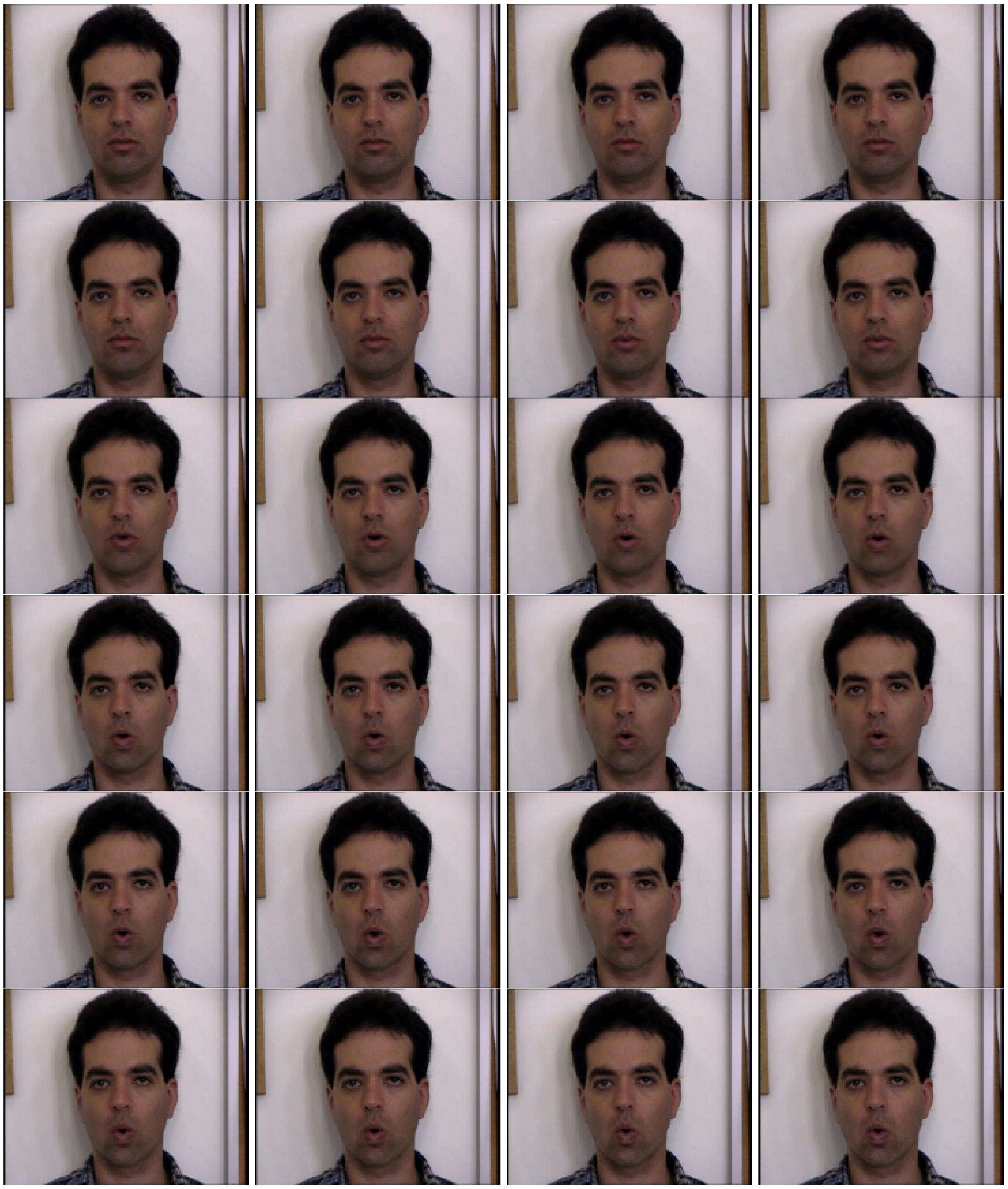


Figure 12: The OH movie snapshots

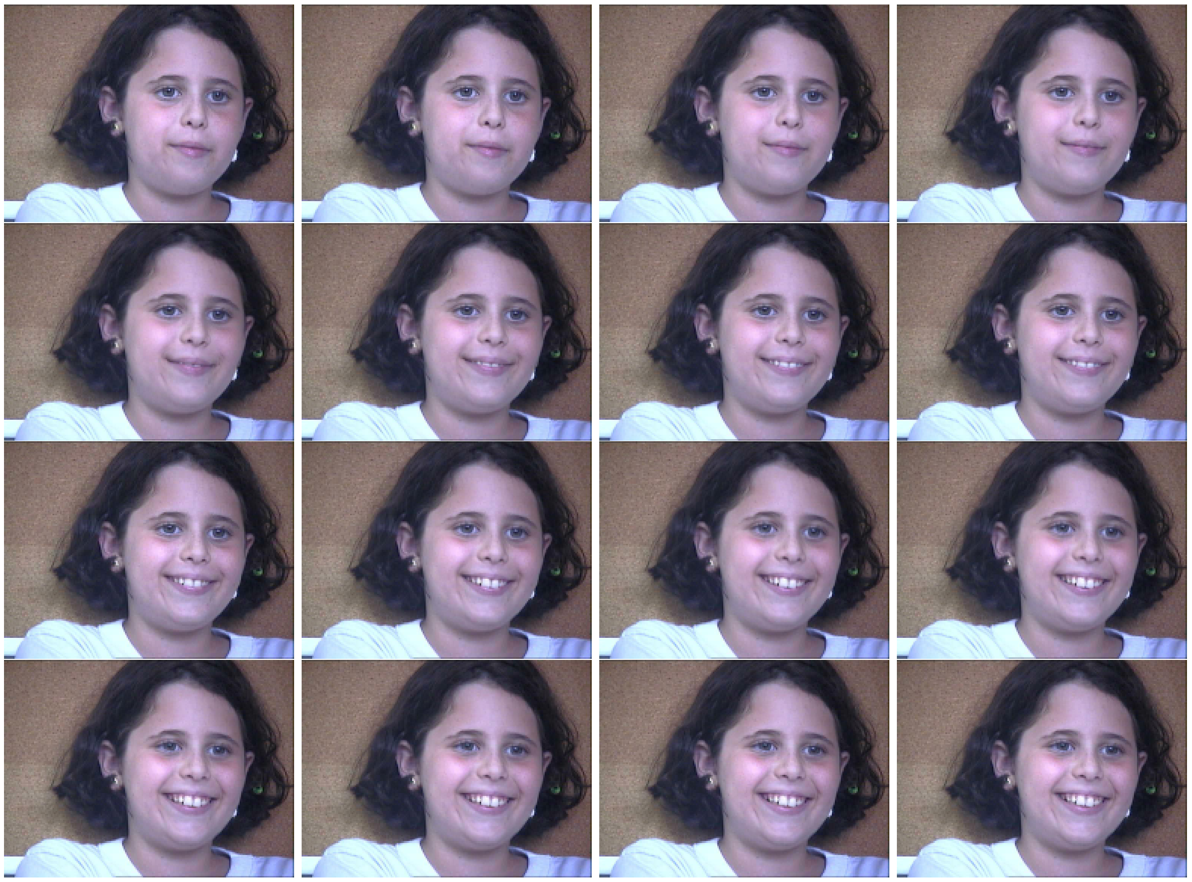
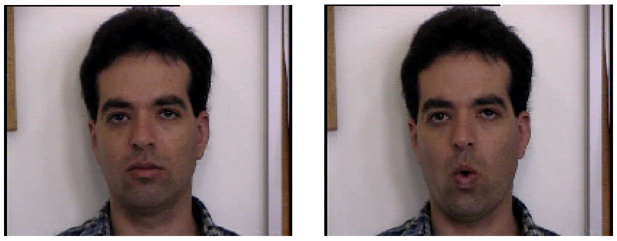
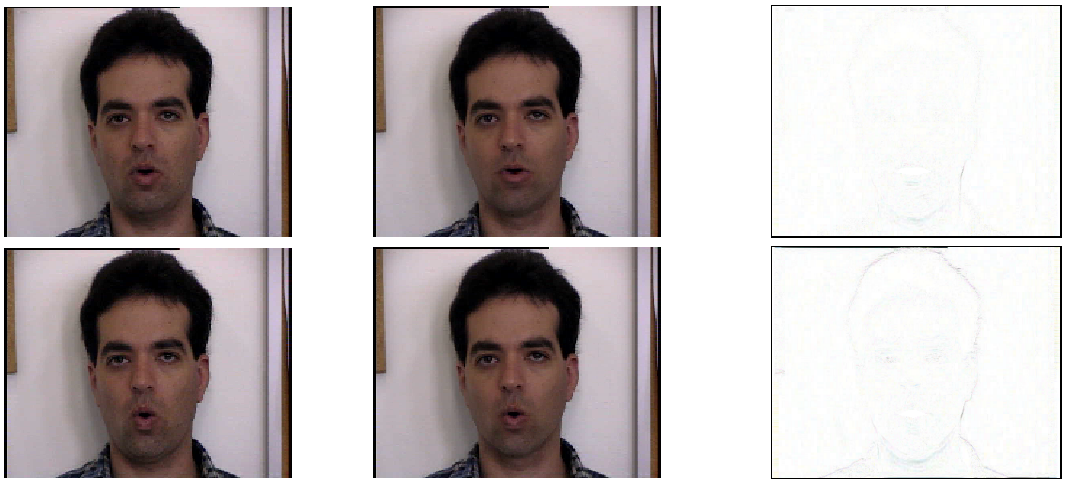


Figure 13: The smile movie snapshots

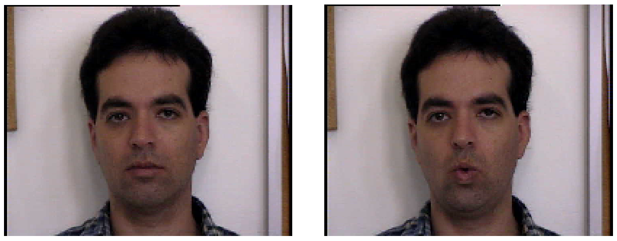


The original images

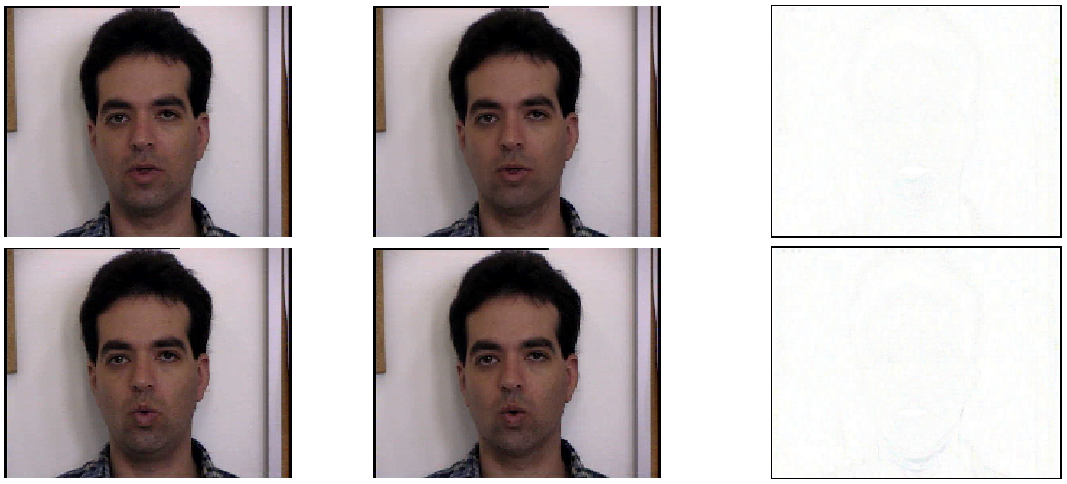


The original in-betweens The synthesized in-betweens The reverse difference images

Figure 14: The OH expression

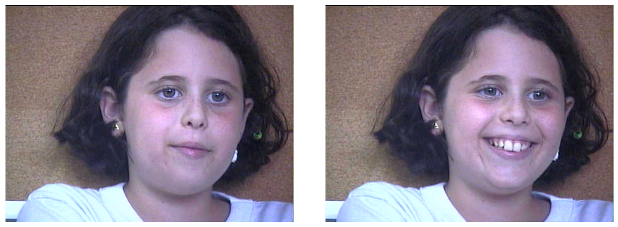


The original images

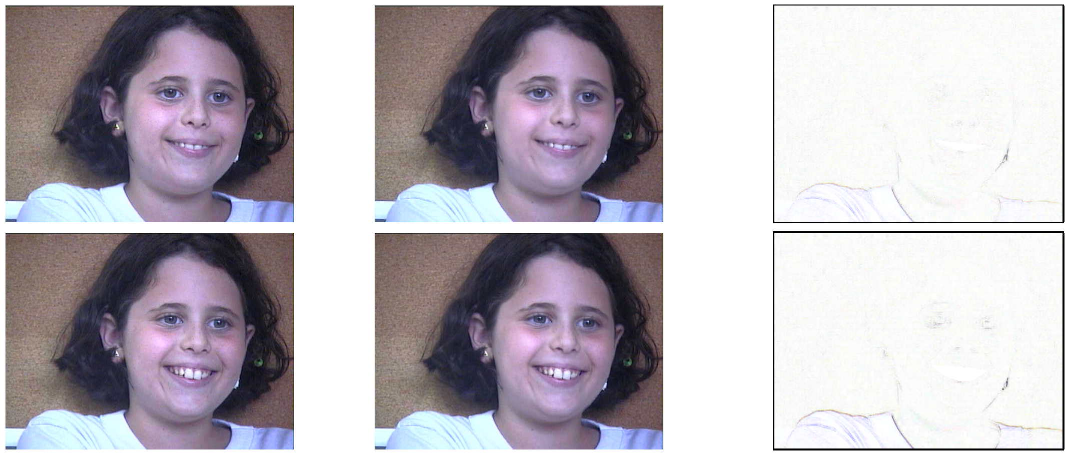


The original in-betweens The synthesized in-betweens The reverse difference images

Figure 15: The OO expression

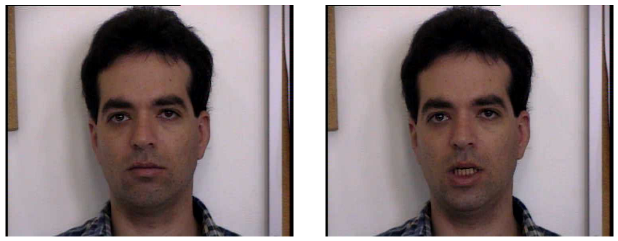


The original images

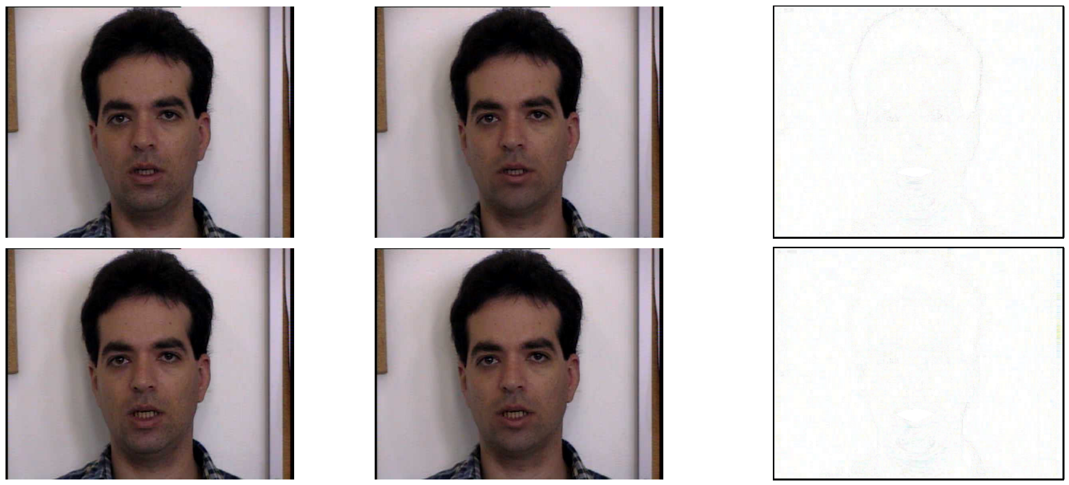


The original in-betweens The synthesized in-betweens The reverse difference images

Figure 16: The smile expression



The original images



The original in-betweens The synthesized in-betweens The reverse difference images

Figure 17: The EE expression

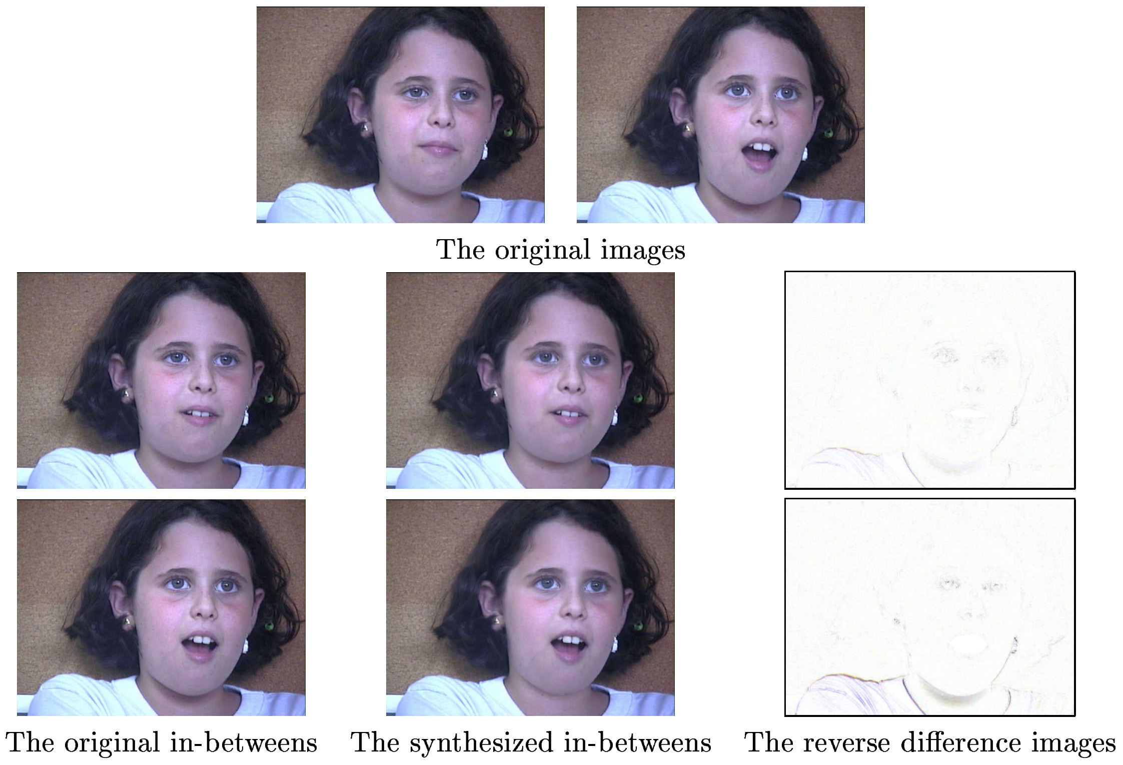


Figure 18: The open-close expression

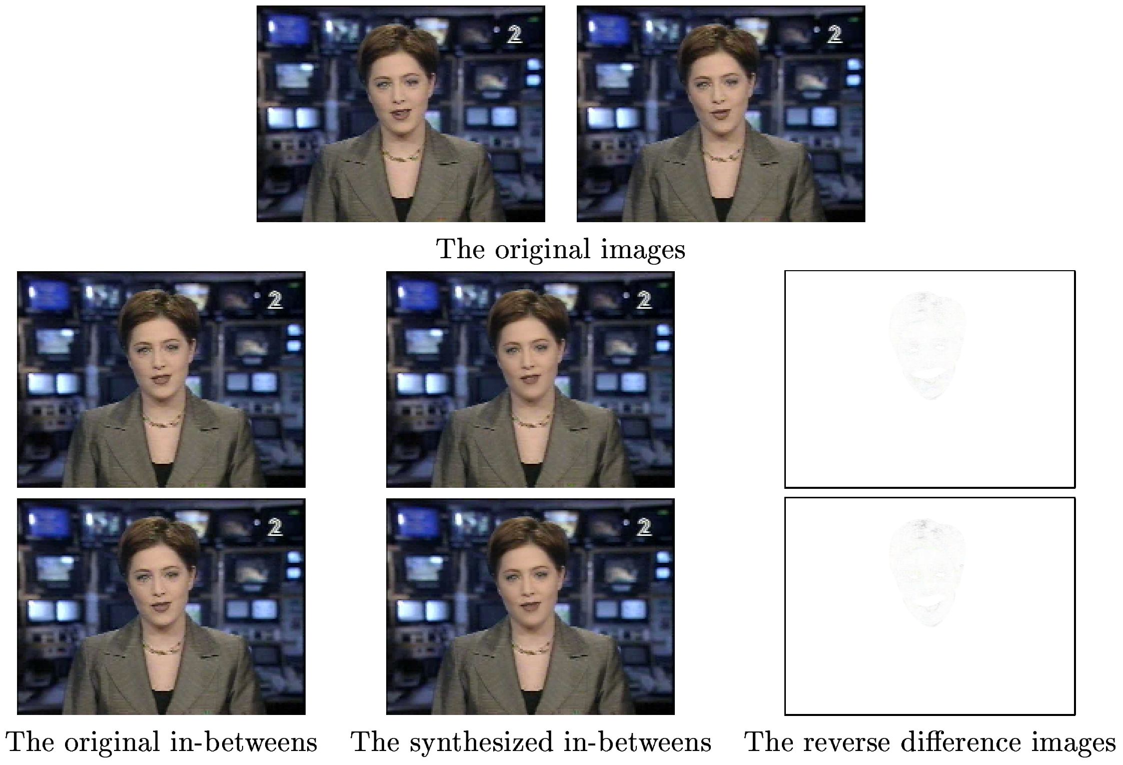
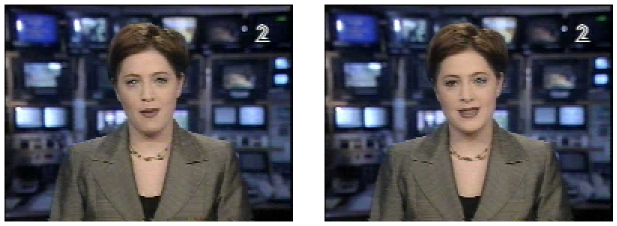
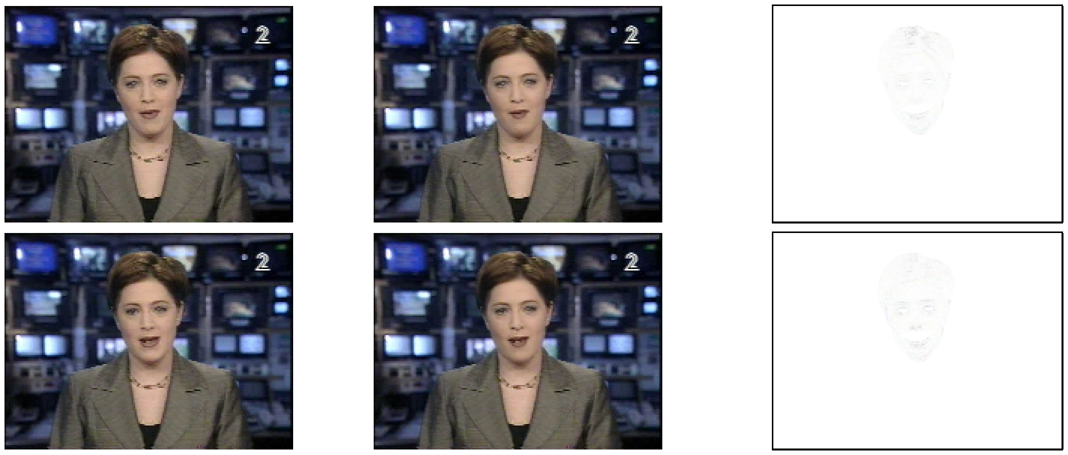


Figure 19: The OO expression of the news broadcaster

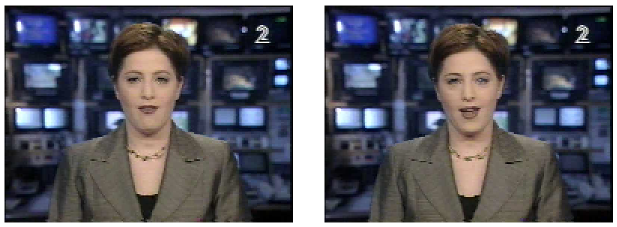


The original images

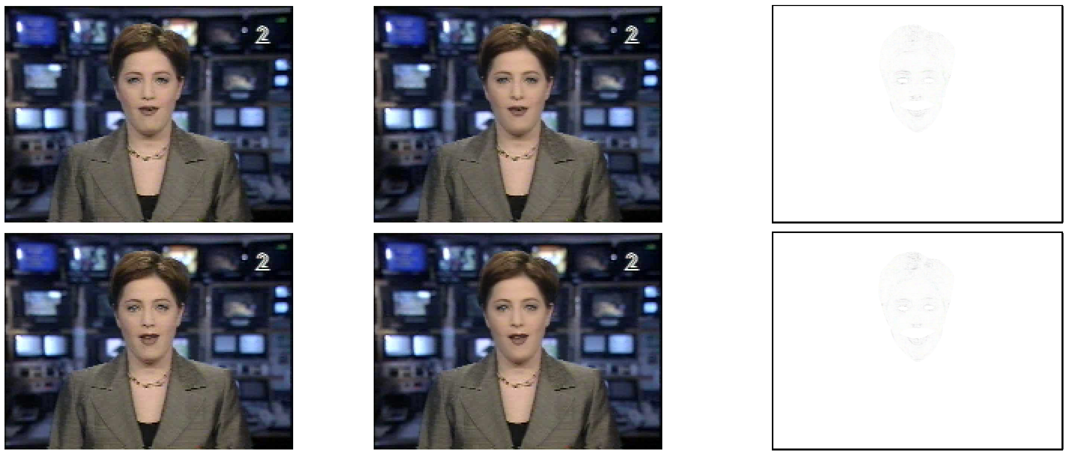


The original in-betweens The synthesized in-betweens The reverse difference images

Figure 20: The EE expression of the news broadcaster



The original images



The original in-betweens The synthesized in-betweens The reverse difference images

Figure 21: The open-close expression of the news broadcaster

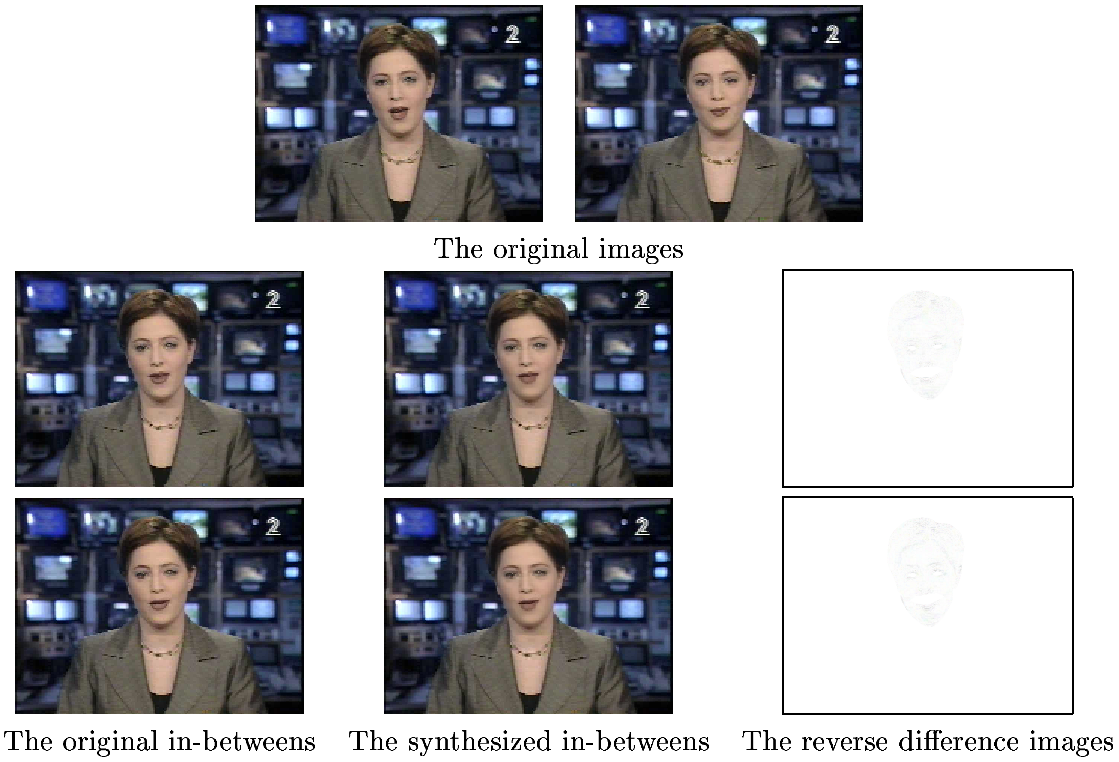


Figure 22: The smile expression of the news broadcaster