FIX: Feature-based Image Transformations for Face Animation

Ali Arya, Babak Hamidzadeh

Dept. of Electrical & Computer Engineering, University of British Columbia,
2356 Main Mall, Vancouver, BC, Canada V6T 1Z4, Phone: (604)822-9181, Fax: (604)822-5949
Email: {alia,babak}@ece.ubc.ca

Abstract

This paper proposes a simple yet effective 2D image transformation method for face animation. Instead of using complicated 3D models or a large database of 2D images, a set of transformations are learned to create different visual effects in a given image, including talking, changing facial expressions, and head movement. This approach enables creation of realistic images with minimum input data and efficient computation. Scaling and mapping these transformations into different head orientations is discussed. The ShowFace system, a modular streaming framework for face animation, is also introduced which uses the proposed technique.

Keywords – Face Animation, Image Transformation, Talking Head, Multimedia, Streaming

1. Introduction

Face animation is a challenging area of computer graphics and multimedia systems research. Realistic and personalized face animation can be used in many application including video conferencing, online training and customer service, visual effects in movies, and interactive games. The ability to generate new and realistic multimedia data for a specific character is of particular importance in such cases where pre-recorded footage may not be available, hard/expensive to generate, or simply too limited due to interactive nature of the application.

Different methods have been proposed for computer face animation [4,8,9,10] that although have their own advantages, usually need a complicated 3D model of the head or a relatively large database of 2D images. Capability of being "personalized" to a character without recreating the model or database is also another missing feature in some the proposed approaches.

Recent development in multimedia systems and standards (like MPEG-4 [3] and SMIL [5]) have directly or indirectly imposed another requirement on face animation techniques (and systems in general). This new requirement is compatibility with new technologies and standards. Such compatibility issues can be seen at algorithm or system levels, and examples are MPEG-4 Face Animation Parameters (FAPs) and streaming structure.

Facial Action Coding System (FACS) [7] was an early effort to represent individual facial movements through use of a set of codes. MPEG-4 FAPs have formalized this approach and combined them with Face Description Parameters (FDPs) to represent feature points of the face and their respective movements for important facial actions. In this paper we discuss the Feature-based Image Transformations (FIX) method for creating face animation. This technique is based on the idea of learning a set of 2D image transformations that can map facial feature points and lines to new locations to create the effect of certain facial actions (as in FACS and FAPs). Proper interpolation then will be used to map other facial points. The main advantages of FIX are realistic output, personalization (based on the ability to apply the transformations to any image), and limited required input data (no complicated 3D model or extensive 2D image database).

Considering other requirements of a general face animation system, we analyze FIX in the context of ShowFace, a comprehensive face animation framework consisting of necessary modules for receiving animation description in a specifically designed language, creating audio and video data, and playing back the multimedia stream. In the next section, we review some related works in face animation. The FIX approach is discussed in Section 3. A brief review of ShowFace system and some concluding remarks are presented in Sections 4 and 5, respectively.

2. Related Work

3D head models have long been used for facial animation [10]. Such models provide a powerful means of head reconstruction in different views and situations, but they usually lack the realistic appearance and need expensive hardware and complicated algorithms that may not be suitable for applications like video phones. Recent approaches have shown successful results in creating 3D models from 2D images, e.g. a limited number of 2D photographs [10].

2D image-based methods are another alternative to face construction. Image morphing is an early mechanism for generating new images based on existing ones [8,9]. The most difficult task in morphing is finding control points, which is usually done manually. MikeTalk is an image-based system, which uses optical flow to solve the correspondence problem [8]. It performs automated morphing and creates visual speech based on pre-recorded visemes. The main issues are the limited ability in creating different facial images (e.g. moves and expressions), non-effectiveness of optical flow in detecting corresponding facial features specially in movements, and also required image database for each person.

Bregler, et al [4], combine a new image with parts of existing footage (mouth and jaw) to create new talking views. This method is also limited to a certain view where the recordings have been made. No transformation is proposed to make a talking view after some new movements of the head. In a more recent work, Graf, et al [9], propose recording of all

0-7803-7724-9/03/$17.00 © 2003 IEEE

visemes in a range of possible views, so after detecting the view (pose) proper visemes will be used. This way talking heads in different views can be animated but the method requires a considerably large database.

TalkingFace [1] combines optical flow and facial feature detection to overcome these issues. It can learn certain image transformations needed for talking (and to some degrees, expressions and head movements) and apply them to any given image. Tiddeman, et al [11], show how such image transformations can be extended to include even facial texture.

Different architectures are also proposed to perform facial animation, specially as an MPEG-4 decoder/player [2]. Although they try to use platform-independent and/or standard technologies (e.g. Java and VRML), they are usually limited to certain face models and lack a component-based and extensible structure, and do not propose any content description mechanism more than standard MPEG-4 parameters. Few content description and synchronization languages proposed recently, are usually limited macros [10] or not customized for face animation [6].

3. Facial Feature-based Image Transformations

3.1. Basic Concepts

View Morphing has been a useful technique for generating new images based on some existing ones and algorithms like optical flow can be used to automate the detection of control points required for morphing [8]. Arya and Hamidzadeh [1] have shown that using facial features can enhance the performance of optical flow-based view generation methods.

On the other hand, as experienced by FACS and MPEG-4 FAPs, knowing the movements of some important feature points and lines can be enough for generating new images (like talking and facial expressions) at some satisfactory level of quality. This means that instead of storing mapping information for all facial points, we can only learn and save such mapping vectors for limited features. When applied to an image, these vectors can be scaled according to size and orientation of the new image and then the mapping vector for other non-feature points can be interpolated. Making the method more flexible and decreasing the amount of required data. If I_1 and I_2 are images corresponding to two states of a face, the optical flow-based approach defines the translation function $\boldsymbol{T}_{12}$ as mapping vectors that take each point in I_1 to its best match in I_2:

$$I_2 = \boldsymbol{T}_{12}(I_1) \quad (1)$$

Here $\boldsymbol{T}_{12}$ has to be stored with all the mapping vectors for each pixel and due to its "blind" nature, can not be scaled or processed to handle a new image other than I_1 . The feature-based method, on the other hand, performs a manual or automated feature detection and forms a feature set $\boldsymbol{F}_i$ for each image I_i. The translation function will now be applied to these new data structures:

$$\boldsymbol{F}_2 = \boldsymbol{T}_{f,12}(\boldsymbol{F}_1) \quad (2)$$

With availability of geometrical information of the face, the Feature Translation Function $\boldsymbol{T}_f$ can now be processed to handle scaling, rotation, and even change of orientation, and it needs less data to be stored. Assuming that the related movements of head/face are very simple, the translation function can even be used for new characters. This idea is illustrated in Figure 1.

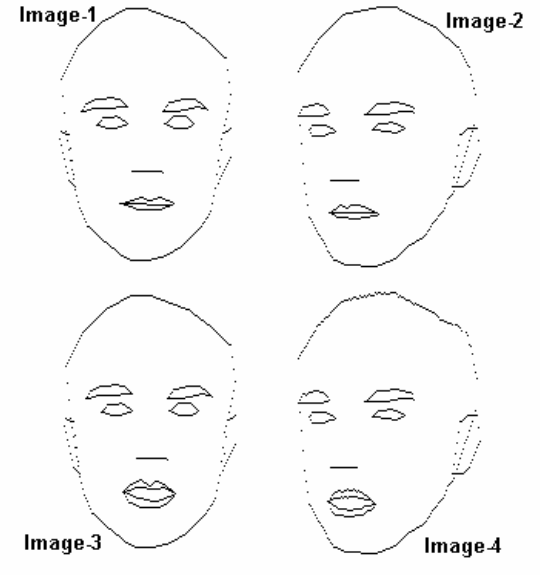


Figure 1. The Feature Translation Function learned between Images 1 and 3 is applied to features of Image 2 to create a feature set for Image 4.

The Feature-based Image Transformation for Face Animation mainly (FIX) consists of:

- Learning Feature Translation Function (FTF) between different facial states,
- Applying those function to the feature points of a source image
- Proper interpolation to find mapping vectors for non-feature points,
- View morphing to generate any number of intermediate images
- Filling newly appeared regions of face (after head movements).

3.2. Facial States and Features

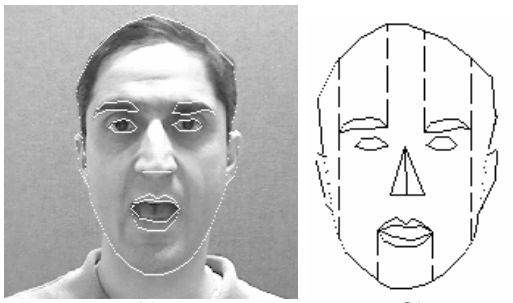
Facial activities are transitions between certain face "states" like a viseme or expression. In a training phase, a set of feature translation functions is learned by the system, which can map between these face states. Translation functions are found by tracking facial features when the model is performing the related transitions. A library of these functions is created based on following facial states:

- Visemes in full-view
- Facial expressions in full-view
- Head movements

For group 1 and 2, mappings for all the transitions between a non-talking neutral face and any group member are stored. In group 3, this is done for transitions between any two neighbouring states (30-degree steps from right profile to left).

Each transformation is defined in the form of $\boldsymbol{T}=(\boldsymbol{F},\boldsymbol{M})$ where $\boldsymbol{T}$ is the transformation, $\boldsymbol{F}$ is the feature set in the source image, and $\boldsymbol{M}$ is the mapping values for features. Source image

information is saved to enable scaling and calibration, explained later. The feature set for each image includes face boundary, eyes and eye-brows, nose, ears, and lips. These feature lines, and the facial regions created by them are shown in Figure 2.



(a) (b)

Figure 2. Facial Regions are defined as areas surrounded by two facial feature lines, e.g. inside eyes or between lower lip and jaw. Some face patches are removed from (b) for simplicity.

The solid lines are feature lines surrounding feature regions, while dashed lines define face *patches*. The patches are defined in order to allow different areas of the face to be treated differently. *Covisibility* is the main concern when defining these face patches. Points in each patch will be mapped to points in the corresponding patch of the target image, if visible.

3.3. Using Feature Translation Function

The transformations are done by first applying the FTF to the source feature points. This is shown in Figure 1. Simple transformations are those which have already been learned, e.g. $\boldsymbol{T}_{12}$ and $\boldsymbol{T}_{13}$ (assuming we only have Image-1) Combined transformations are necessary in cases when the target image is required to have the effect of two facial state transitions at the same time, e.g. creating Image-4 from Image-1 using $\boldsymbol{T}_{14}$.

Before applying any FTF to a new set of features, the mapping vectors have to be scaled based on the difference between source image in $\boldsymbol{T}=(\boldsymbol{F},\boldsymbol{M})$ and the new image features. Multiple transformations can then applied as linear combinations. Due to non-orthographic nature of some head movements, combined transformations involving 3D head rotation can not be considered a linear combination of some known transformations. Feature mapping vectors for talking and expressions (which are learned from frontal view images) need to be modified when applied to "moved" heads.

$$\boldsymbol{T}_{14} = a\,\boldsymbol{T}_{12} + b\,\boldsymbol{T}'_{13} \quad (3)$$

$$\boldsymbol{T}'_{13} = f_p(\boldsymbol{T}_{12}, , \boldsymbol{T}_{13}) = \boldsymbol{T}_{24} \quad (4)$$

where f_p is Perspective Calibration Function (PCF), and a and b are coefficients between 0 and 1 to control transition progress. PCF mainly involves moving $\boldsymbol{F}_1$ and $\boldsymbol{F}_3$ using $\boldsymbol{T}_{12}$, and then re-calculating the FTF.

When the input picture is something like Image-2 in Figure 1, i.e. the new image does not have the same orientation as the ones used in learning (Image-1 and Image-3), the required transformation (e.g. for talking) is $\boldsymbol{T}_{24}$ which still needs scaling/perspective calibration based on $\boldsymbol{T}_{12}$ and $\boldsymbol{T}_{13}$.

3.4. Facial Regions

The stored transformations only show the mapping vectors for feature lines. Non-feature points are mapped by interpolating the mapping values for the feature lines surrounding their regions. This is done based on the face region to which a point belongs.

Face regions are grouped into two different categories:

- Feature islands, surrounded by one or two "inner" feature lines
- Face patches, covering the rest of the face as shown in Figure 2-b.
- New Regions, which appear as the result of head movement or talking (inside mouth)

The mapping vector for each point inside a group-1 region is found using the following formula:

$$\boldsymbol{d}_{r,c} = (abs(u-r)\times\boldsymbol{d}_{u,c} + abs(r-l)\times\boldsymbol{d}_{l,c})/(u-l) \quad (5)$$

For the sake of simplicity, the mapping vector of each point is considered to be only a function (weighted average) of mapping vectors for two feature points directly above and below ($\boldsymbol{d}_{u,c}$ and $\boldsymbol{d}_{l,c}$), r and c are row and column in image for the given point, u and l are the row number for top and bottom feature points, and $\boldsymbol{d}$ is the mapping vector.

Face patches are defined based on *covisibility*, i.e. their points are most likely to be seen together. Defining the patches is necessary in order to preserve the geometric validity of the transformation. The mapping vector of each point in a patch is the weighted average of mapping of all the patch corners.

Filling the newly appeared regions is the only task that can not be done with a single input image. Here we use a second image (e.g. a profile) that has the required region, map it to the target state, and use the data for that specific region.

3.5. Experimental Results

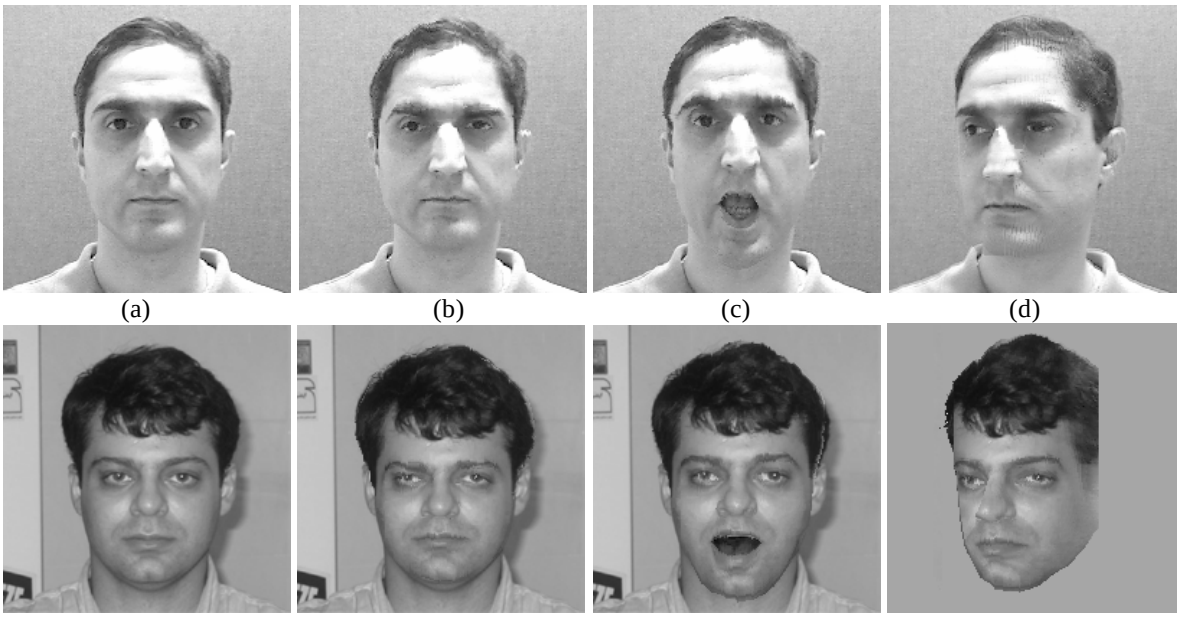
Figure 3 shows some sample images created by FIX method. Images in each row are generated using the transformations applied to the image at the left side. In case of second row, no profile (side) image was available so recreation of the side of the head has not been possible. In all talking images, a generic inside-mouth data is used to fill the newly appeared regions. This can be improved by using real data for the character in image if available (as in image c).

4. ShowFace System

4.1. Basic Structure

The basic structure of *ShowFace* system [2] is illustrated in Figure 4. Five major parts of this system are:

- Script Reader, to receive an FML script from a disk file, an Internet address, or any text stream provider.
- Script Parser, to interpret the FML script and create separate intermediate audio and video descriptions (e.g. words and viseme identifiers)
- Video Kernel, to generate the required image frames
- Audio Kernel, to generate the required speech
- Multimedia Mixer, to synchronize audio and video streams



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3. Transformed Faces, mapped from images at left (a,e): frown (b,f), viseme for sound "a" in "talk" (c,g), and rotate to right (d,h; image h does not have side information from a secondary image)

ShowFace relies on the underlying multimedia technology for audio and video display. The system components interact with each other using ShowFace Application Programming Interface, SF-API, a set of interfaces exposed by the components and utility functions provided by *ShowFace* run-time environment. User applications can access system components through SF-API, or use a wrapper object called *ShowFacePlayer*, which exposes the main functionality of the system, and hides programming details.

ShowFace system is designed and implemented with the concept of openness in mind. By that we mean the ability to use and connect to existing standard components and also independent upgrade of the system modules. To make most use of existing technologies, *ShowFace* components are implemented as Microsoft DirectShow filters. DirectShow will be installed as part of many application programs like browsers and games, and comes with a set of filters like audio and video decoders and renderers. This allows *ShowFace* objects to access these filters easily and rely on multimedia streaming services provided by DirectShow, e.g. receiving data from a URL reader or MPEG-4 decoder and sending data to a video player or file writer.

Video Kernel uses FIX technique to create the desired images. Audio Kernel uses a Text-To-Speech (TTS) engine and a set of pre-recorded diphones to generate the required audio data. The *ShowFacePlayer* wrapper object is implemented as an ActiveX control, which can be easily used in web pages and other client applications. An off-line tool, *ShowFace Studio*, is also developed to assist in detecting the features, creating the maps, and recording scripts.

4.2. Face Modeling Language

The main input format to the ShowFace system is content description using Face Modeling Language (FML). Based on extensible Markup Language (XML) and ideas from Synchronized Multimedia Integration Language (SMIL), FML is a description and modeling language specifically designed for face animation. FML allows a hierarchical description of animation from high level stories to low level head moves and MPEG-4 FAPs. A sample FML document is shown in Figure 5.

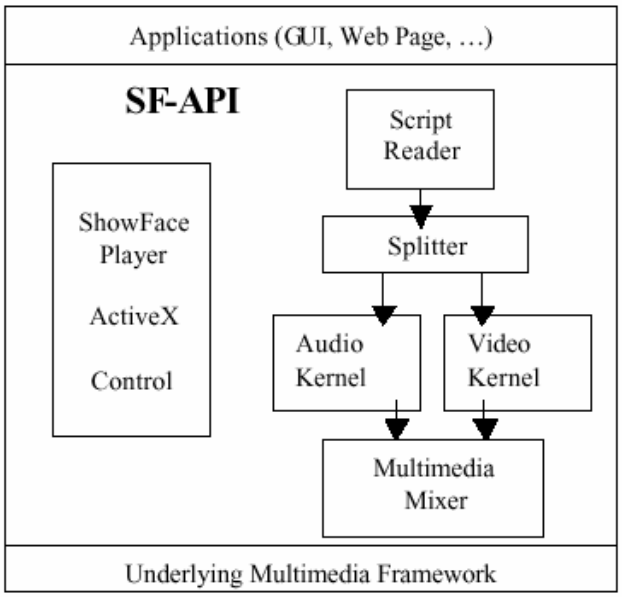


Figure 4. Component-based *ShowFace* System Structure

Each FML module has two main parts: **`model`** and **`story`**.

The former defines initialization, movement ranges, and behavioural **`templates`**. The latter describes the animation

sequence in a hierarchical way. Each **story** consists of **actions** (independent meaningful sets of moves). At the lower level there are time containers: **par**, **seq**, and **excl**. The first two construct parallel and sequential moves. **excl** is used for decision-making, i.e. taking different paths based on an external **event**. The external events used in decision-making can be set by an interactive user or another program through SF-API. The *ShowFacePlayer* object exposes a **SetEvent** method for other programs (specially scripts on a web page). The XML-based nature of this language allows the FML documents to be embedded in web pages. Normal XML parsers can extract data and use them as input to an FML-enabled player, through simple scripting. Such a script can also use XML Document Object Model (DOM) to modify the FML document, e.g. adding certain activities based on user input. This compatibility with web browsing environments, gives another level of interactivity and dynamic operation to FML-based system.

```
<fml>
  <model>  <!-- Model Info -->
    <img src="me.jpg" />
    <range dir="0" val="60" />
    <event id="user" val="-1" />
    <template name="hello" >
      <seq begin="0">
        <talk begin="0">Hello</talk>
        <hdmv begin="0" end="5"
           dir="0" val="30" />
      </seq>
    </template>
  </model>
  <story>  <!-- Story Timeline -->
    <action>
      <behavior template="hello" />
      <excl ev_id="user">
        <talk ev_val="0">Hi</talk>
        <talk ev_val="1">Bye</talk>
      </excl>
      <par begin="0">
        <talk begin="1">Hello
           World</talk>
        <exp begin="0" end="3"
           type="3" val="50" />
      </par>
    </action>
  </story>
</fml>
```

Figure 5. Sample FML Document

Another aspect of FML is its compatibility with MPEG-4 face definition/animation parameters. This has been achieved by:

- Translation of FML documents to MPEG-4 codes by the media player.
- Embedded MPEG-4 elements (**fap** element is considered to allow direct embedding of FAPs in FML document)
- Integrating FML into XMT framework by FML-to-MPEG converters.

5. Concluding Remarks

An approach to Feature-based Image Transformation for Face Animation (FIX) is proposed. The main objective of FIX is to use normal 2D images rather than complicated 3D models in order to create realistic personalized face animation. FIX also uses facial features and their translations for each face activity to minimize the amount of data required and make scaling and processing the transformation for new characters possible. Methods for combining simple transforms into more complicated ones are discussed.

The main advantage of FIX is its realism, simplicity compared to 3D geometric models, and minimum amount of input data compared to 2D image-based methods. The learnt transformations can be stored in a small database and easily applied to a given picture to create a personalized new image.

FIX is introduced in the context of ShowFace system which provides a comprehensive framework for face animation with streaming capability, structured input description (using FML), on-demand audio/video generation, and flexible open architecture.

Better feature detection is a main objective of our future work, since any error in detecting a feature point can directly result in a wrong transformation vector. This effect can be seen in cases like eyebrows where detecting the exact corresponding points between a pair of learning images is not easy. As a result, the learned transformation may include additive random errors which causes non-smooth eyebrow lines in transformed feature set and image.

References

[1] A. Arya and B. Hamidzadeh, "TalkingFace: Using Facial Feature Detection and Image Transformations for Visual Speech," *Proc Int Conf Image Processing (ICIP)*, 2001.

[2] Ali Arya and Babak Hamidzadeh, "Personalized Face Animation in ShowFace System," *Int. Journal of Image and Graphics*, Special Issue on Virtual Reality and Virtual Environments, World Scientific Publishing, 2003

[3] S. Battista, et al, "MPEG-4: A Multimedia Standard for the Third Millennium," *IEEE Multimedia*, October 1999.

[4] C. Bregler, et al, "Video Rewrite," *ACM Computer Graphics*, 1997

[5] D. Bulterman, "SMIL-2," *IEEE Multimedia*, October 2001.

[6] J. Cassell, et al, "BEAT: the Behavior Expression Animation Toolkit," *Proc ACM SIGGRAPH*, 2001.

[7] P. Ekman and W. V. Friesen, *Facial Action Coding System*, Consulting Psychologists Press Inc., 1978.

[8] T. Ezzat and T. Poggio, "MikeTalk: A Talking Facial Display Based on Morphing Visemes," *Proc IEEE Conf Computer Animation*, 1998.

[9] H. P. Graf, et al, "Face Analysis for the Synthesis of Photo-Realistic Talking Heads," *Proc IEEE Conf Automatic Face and Gesture Recognition*, 2000.

[10] W. S. Lee, et al, "MPEG-4 Compatible Faces from Orthogonal Photos," *Proc IEEE Conf Computer Animation*, 1999.

[11] B. Tiddeman, et al, "Prototyping and Transforming Facial Textures for Perception Research," *IEEE CG&A*, September 2001.