ShowFace: A Framework for Personalized Face Animation

Ali Arya, Babak Hamidzadeh

Dept. of Electrical & Computer Engineering, University of British Columbia,

2356 Main Mall, Vancouver, BC, Canada V6T 1Z4,

Telephone: (+1-604)822-9181, Fax: (+1-604)822-5949

Email: {alia,babak}@ece.ubc.ca

Abstract

In this paper ShowFace streaming structure is proposed as a comprehensive framework for Personalized Face Animation. This structure is based on widely accepted industry standards in multimedia presentation like MPEG-4 and SMIL, and extends them by defining image transformations required for certain facial movements, and also by providing a higher level Face Modeling Language (FML). It defines a comprehensive framework for face animation consisting of components for content description, creation, and playback. The component-based design and scripted behaviour make the framework suitable for many purposes including web-based applications.

Keywords

Face Animation, Virtual and Augmented Reality, Interaction with Virtual Humans, User Interface, Behavioural Animation.

1. Introduction

Recent developments in multimedia-related areas like Virtual Environment, Video Conferencing, Computer Games, and Agent-based Online Applications have drawn a considerable attention to character animation. Replacing audio-visual data of "real people" with multimedia presentations based on "virtual agents" seem to be very beneficial, and in some cases necessary. Saving bandwidth in video conferencing by replacing video data with animation "commands" can be considered an example of the former cases, while creating new scenes with "unavailable" characters is an example of the latter.

Personalized Face Animation includes all the information and activities required to create a multimedia presentation resembling a specific person. The input to such a system can be a combination of audio-visual data and textual commands and descriptions. A successful face animation system needs to have efficient yet powerful solutions for providing and displaying the content, i.e. a content description format, decoding algorithms, creating required content, and finally an architecture to put different components together in a flexible way.

Considering three major issues of Content Delivery, Content Creation, and Content Description, the following features can be assumed as important requirements in a multimedia presentation system:

- Streaming, i.e. continuously receiving and displaying data
- Structured Content Description, i.e. a hierarchical way to provide information about the content from high level scene description to low level moves, images, and sounds

- Generalized Decoding, i.e. creating the displayable content based on the input. This can be decoding a compressed image or making a new image as requested.
- Component-based Architecture, i.e. the flexibility to rearrange the system components, and use new ones as long as a certain interface is supported
- Compatibility, i.e. the ability to use and work with widely accepted industry standards in multimedia systems
- Minimized Database of audio-visual footage and modeling

The technological advances in multimedia systems, speech/image processing, and computer graphics, and also new applications especially in computer-based games, telecommunication, and online services, have resulted in a rapidly growing number of publications regarding these issues. These research achievements, although very successful in their objectives, mostly address a limited subset of the above requirements. A comprehensive framework for face animation is still in conceptual stages. The ShowFace system, discussed in this paper, is a step toward such a framework. In Section 2, some of the related works are briefly reviewed. The basic concepts and structure of ShowFace system are discussed in Sections 3 to 5. Some experimental results and our evaluation criteria are the topics of Sections 6. Section 7 summarizes the proposed approach with some concluding remarks.

2. Related Work

2.1. Content Description

Multimedia Content description has been the subject of some research projects and industry standards. In case of face animation, Facial Action Coding System (FACS) [11] was one the first attempts to model the low level movements, which can happen on a face, by defining a set of coded actions. Although not defined for animation purposes but to study facial actions, FACS has been used in computer graphics and animation due the convenience the coding system provides. MPEG-4 [4] standard uses a similar idea (i.e. coded facial movements) by introducing Face Animation Parameters for facial features and their movements. These parameters can define low-level moves of features (e.g. jaw-down) and also higher-level set of moves to form a complete facial situation (e.g. visemes or expressions). The standard does not go further to define a dynamic description of the scene including facial activities and their temporal relation. Generating facial views based on these parameters and the underlying face model are out of the scope of MPEG-4 standard. MPEG-4 includes eXtensible MPEG-4 Textual format (XMT) framework to represent scene description in a textual format providing interoperability with and languages like SMIL and VRML.

Indirectly related is Synchronized Multimedia Integration Language, SMIL [8], an XML-based language for dynamic (temporal) description of the events in a general multimedia presentation. It defines time containers for sequential, parallel, and exclusive actions related to the presented objects, in order to synchronize the events. SMIL does not act as dynamic content description for facial animation or any other specific application.

BEAT [9] is another XML-based system, specifically designed for human animation purposes. It is a toolkit for automatically suggesting expressions and gestures, based on a given text to be spoken. BEAT uses a knowledge base and rule set, and provides synchronization data for facial activities, all in XML format. This enables the system to use standard XML parsing and scripting capabilities. Although BEAT is not a general content description tool, but it demonstrates some of the advantages of XML-based approaches.

Recent advances in developing and using Embodied Conversational Agents (ECAs), especially their web-based applications, and growing acceptance of XML as a data representation language have drawn attention to XML-based markup languages for virtual characters [1,10,17,20]. The basic idea is to define specific XML tags related to agents' actions such as moving and talking. Virtual Human Markup Language (VHML) [17] is an XML-based language for the representation of different aspects of "virtual humans", i.e. avatars, such as speech production, facial and body animation, and emotional representation. It comprises a number of special purpose languages, such as EML (Emotion Markup Language), FAML (Facial Animation Markup Language), and BAML (Body Animation Markup Language). In VHML, timing of animation elements, in relation to each other and in relation to the realisation of text, is achieved via the attributes `"duration"` and `"wait"`. Multimodal Presentation Markup Language (MPML) [20] is another XML-based markup language developed to enable the description of multimodal presentation on the WWW, based on animated characters. It offers functionalities for synchronizing media presentation (reusing parts of SMIL) and new XML elements such as `<listen>` (basic interactivity), `<test>` (decision making), `<speak>` (spoken by a TTS-system), `<move>` (to a certain point at the screen), and

`<emotion>` (for standard facial expressions). MPML addresses the interactivity and decision-making not directly covered by VHML, but both suffer from a lack of explicit compatibility with MPEG-4 (XMT, FAPs, etc).

Scripting and behavioural modeling languages for virtual humans are considered by other researchers as well [13,15,16]. These languages are usually simple macros for simplifying the animation, or new languages which are not using existing multimedia technologies. Few of them are specifically designed for face animation.

2.2. View Generation

3D head models have long been used for facial animation [5,16,19]. Such models provide a powerful means of head reconstruction in different views and situations, but they usually need expensive hardware and complicated modeling and computation, and lack the realistic appearance. Recent approaches have shown successful results in creating 3D models from 2D images, e.g. a limited number of 2D photographs [16,19].

2D image-based methods are another alternative to face construction. Image morphing is an early mechanism for generating new images based on existing ones [3,12,14]. The most difficult task in morphing is finding control points, which is usually done manually. MikeTalk [12] is an image-based system, which uses optical flow to solve the correspondence problem. 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, especially in movements, and also required image database for each person.

Bregler et al. [7] 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. Proper transformations are not proposed to make a talking view in a head orientation other than the recorded views. In a more recent work, Graf et al [14] propose recording of all visemes in a range of possible views, so after detecting the view (pose) proper visemes will be used. This way talking heads in dfferent views can be animated but the method requires a considerably large database and still is limited to one person.

TalkingFace [3] combines optical flow and facial feature detection to overcome some of 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 [21] show how such image transformations can be extended to include even facial texture.

2.3. Architectural Issues

Different architectures are also proposed to perform facial animation, especially as an MPEG-4 decoder/player [6,18]. 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.

3. Structured Content Description

3.1. Design Ideas

Describing the contents of a multimedia presentation is a basic task in multimedia systems. It is necessary when a client asks for a certain presentation to be designed, when a media player receives input to play back, and even when a search is done to retrieve an existing multimedia object. In all these cases, the description can include raw multimedia data (video, audio, etc) and textual commands and information. Such a description works as a Generalized Encoding, since it represents the multimedia content in a form not necessarily the same as the playback format, and is usually more efficient and compact.

Although new streaming technologies allow real-time download/playback of audio-video data, but bandwidth limitation and its efficient usage still are, and probably will be, major issues. This makes a textual description of multimedia presentation (in our case facial actions) a very effective coding/compression mechanism, provided the visual effects can be recreated with a minimum acceptable quality.

Efficient use of bandwidth is not the only advantage of facial action coding. In many cases, the "real" multimedia data does not exist at all, and has to be created based on a description of desired actions. This leads to the idea of representing the spatial and temporal relation of the facial actions. In general, such a description of facial presentation should provide a hierarchical structure with elements ranging from low level "images", to simple "moves", more complicated "actions", to complete "stories". We call this a Structured Content Description, which also requires means of defining capabilities, behavioural templates, dynamic contents, and event/user interaction. Needless to say, compatibility with existing multimedia and web technologies is another fundamental requirement, in this regard.

Face Modeling Language (FML, http://www.ece.ubc.ca/~alia/Multimedia/fml_1.html) [2] is a Structured Content Description mechanism based on eXtensible Markup Language. The main ideas behind FML are:

- Hierarchical representation of face animation
- Timeline definition of the relation between facial actions and external events
- Defining capabilities and behavioural templates
- Compatibility with MPEG-4 FAPs and XMT framework
- Compatibility with XML and related web technologies

FACS and MPEG-4 FAPs provide the means of describing low-level face actions but they do not cover temporal relations and higher-level structures. Languages like SMIL do this in a general purpose form for any multimedia presentation and are not customized for specific applications like face animation. A language bringing the best of these two together, customized for face animation, seems to be an important requirement. FML is designed to do so.

Fundamental to FML is the idea of Structured Content Description. It means a hierarchical view of face animation capable of representing simple individually-meaningless moves to complicated high level stories. This hierarchy can be thought of as consisting of the following levels (bottom-up):

- Frame, a single image showing a snapshot of the face (naturally, may not be accompanied by speech)
- Move, a set of frames representing linear transition between two frames (e.g. making a smile)
- Action, a "meaningful" combination of moves
- Story, a stand-alone piece of face animation

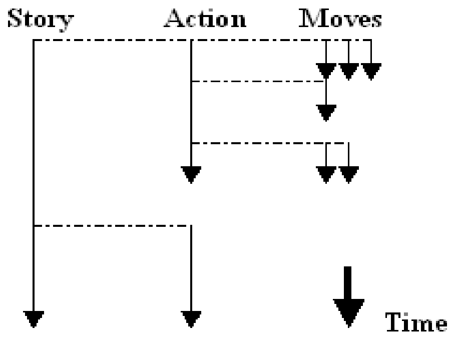


Figure 1: FML Timeline

FML defines a timeline of events (Figure 1) including head movements, speech, and facial expressions, and their combinations. Since a face animation might be used in an interactive environment, such a timeline may be altered and/or determined by a user. So another functionality of FML is to allow user interaction and in general event handling (e.g. user input or decision-making).

A major concern in designing FML is compatibility with existing standards and languages. The growing acceptance of MPEG-4 standard makes it necessary to design FML in a way it can be translated to/from a set of FAPs. Also due to similarity of concepts, it is desirable to use SMIL syntax and constructs, as much as possible. FML fits smoothly in MPEG-4 XMT framework for multimedia content description and authoring.

3.2. Primary Language Constructs

FML is an XML-based language. The choice of XML as the base for FML is based on its capabilities as a markup language, growing acceptance, and available system support in different platforms. Figure 2 shows typical structure of an FML document.

```
<fml>
    <model>     <!-- Model Info -->
         <model-item />
    </model>
    <story>          <!--StoryTimeline-->
         <action>
              <time-container>
                   <move-set />
              </time-container>
         </action>
    </story>
</fml>
```

Figure 2: FML Document Map; Time-container and move-set will be replaced by FML time container elements and sets of possible activities, respectively.

An FML document consists, at higher level, of two types of elements: **`model`** and **`story`**. A **`model`** element is used for defining face capabilities, parameters, and initial configuration. A **`story`** element, on the other hand, describes the timeline of events in face animation. Face animation timeline consists of facial activities and their temporal relations. These activities are themselves sets of simple Moves. The timeline is primarily created using two time container elements, **`seq`** and **`par`**, representing sequential and parallel move-sets (Figure 3). A story itself is a special case of sequential time container. The begin times of activities inside a **`seq`** and **`par`** are relative to previous activity and container begin time, respectively.

```
<seq begin="0">
    <talk  begin="0">Hi</talk>
    <hdmv  begin="0" end="5" type="0" val="30" />
</seq>
<par begin="0">
    <talk begin="1">Hi</talk>
    <exp begin="0" end="3" type="3" val="50" />
</par>
```

Figure 3: FML Primary Time Container

FML supports three basic face activities: talking, expressions, and 3D head movements. They can be a simple Move (like an expression) or more complicated (like a piece of speech). Combined in time containers, they create FML Actions.

Also supported in FML are behavioural templates as a primary means of behavioural modeling. Templates work similar to subroutines in normal programming languages (Figure 4).

3.3. Event Handling and Decision Making

Dynamic and interactive applications require the FML document to be able to make decisions, i.e. to follow different paths based on certain events. To accomplish this, **`excl`** time container and **`event`** element are added. An event represents any external data, e.g. the value of a user selection. As shown in Figure 5, the new time container associates with an event and allows waiting until the event has one of the given values, then it continues with action corresponding to that value. Iterations are possible through the use of **`repeat`** attribute in all time containers which can also be associated to an external event instead of a fixed number.

```
<model>
     <img src="me.jpg" />
     <range type="0" val="60" />
     <template name="hello" >
        <seq begin="0">
           <talk begin="0">Hi</talk>
           <hdmv begin="0" end="5" type="0" val="30" />
        </seq>
     </template>
</model>
<story>
     <behavior name="hello" />
</story>
```

Figure 4: FML Model and Templates

```
<event name="user" val="-1" />
<excl ev_name="user" repeat="5">
     <talk ev_val="0">Hi</talk>
     <talk ev_val="1">Bye</talk>
</excl>
```

Figure 5: FML Decision Making and Event Handling

3.4. Compatibility

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.

Another aspect of FML is its MPEG-4 compatibility, 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)
- Translation to MPEG-4 FAPs in XMT framework

4. Content Creation

4.1. Facial Feature-based Image Transformations

As experienced by FACS and MPEG-4 FAPs, knowing the mapping vectors related to 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 we can only learn and save a set of 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. 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:

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

$\boldsymbol{F}_2 = \boldsymbol{T}_{f,12}(\boldsymbol{F}_1)$ (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 or new head orientations of the same character. This idea is illustrated in Figure 6. The mapping between Images 1 and 3, and also 1 and 2 are pre-learned. In runtime, these mappings can be applied to a new character in the same head positions, or as illustrated in this figure, they can be combined to create a talking non-frontal image (Image-4).

Combining the mappings is meaningful when each belongs to a different group of facial actions (e.g. head movements and visemes) as explained in Section 4.2.

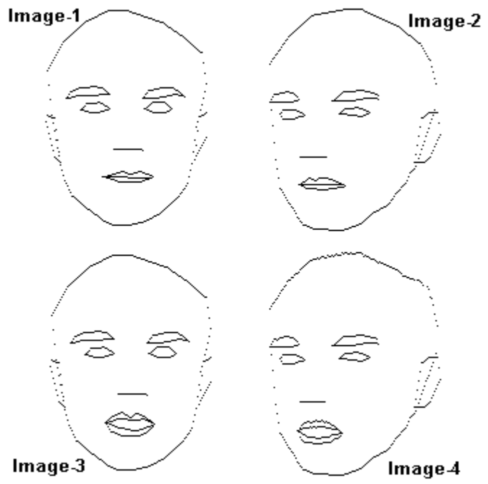


Figure 6: The Feature Translation Function learned between Images 1 and 3 is applied to features of Image 2 to create a feature set for Image 4.

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)
- Texture mapping (if texture/colour changes are also saved during the learning phase)

4.2. 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, shown in Table 1
- Facial expressions in full-view (Anger, Disgust, Fear, Joy, Sadness, and Surprise)

- Head movements

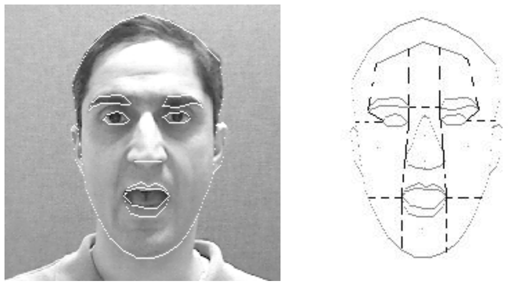
Table 1: Visemes List

Viseme	Example
/A/	soft
/a/	apple
/E/	see
/e/	set
/i/	sit
/m/	mother
/n/	no
/o/	go
/u/	foot
/v/	video
NULL	(silence)

For group 1 and 2, mappings for all the transitions between a non-talking neutral face and any group member are learned and stored. No transition from one group to another is necessary since they will be created by FIX. In group 3, this is done for transitions between any two neighbouring states (30-degree steps from right profile to left).

It should be noted that a more comprehensive set of visemes should be used to create a fully realistic visual speech. In this work, for the sake of simplicity and without a considerable loss of quality, single image is used for some visemes with a minimum similarity. Also coarticulation (the visual effect of surrounding phonemes on any viseme) is not considered and need to be included in future research.

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 7.



(a) (b)

Figure 7: Facial Regions are defined as areas surrounded by two facial feature lines, e.g. inside eyes or between lower lip and jaw.

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.

4.3. Using Feature Translation Function

The transformations are done by first applying the FTF to the source feature points. This is shown in Figure 6. 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 6, 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 $\boldsymbol{T}_{13}$.

4.4. 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 7-b.
- New Regions, which appear as the result of head movement or talking (inside mouth)

The mapping vector for each point inside a feature island is found using the following formula:

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

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 the features close to that patch.

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.

5. ShowFace System

The basic structure of *ShowFace* system is illustrated in Figure 8. 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

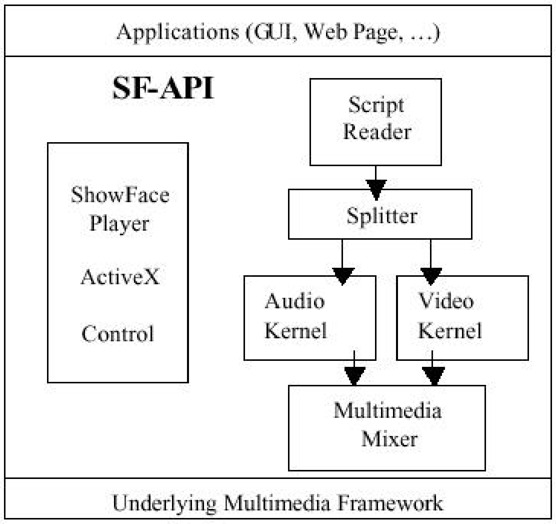

Applications (GUI, Web Page, ...)
SF-API
Script Reader
Splitter
ShowFace Player ActiveX Control
Audio Kernel
Video Kernel
Multimedia Mixer
Underlying Multimedia Framework

Figure 8: Component-based ShowFace System Structure

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.

6. Experimental Results and Evaluation

Figure 9 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). Case studies for using FML documents can be found in [2].

Based on the basic ideas presented in Section 1, following criteria are used to evaluate *ShowFace*:

- Realism
- Graphic capabilities
- Timeliness and streaming
- Simplicity and efficiency
- Descriptiveness
- Compatibility and openness

Considering these criteria, we realize that *ShowFace* is capable of creating graphic content in a variety of fundamental facial states with reasonable realism, while using minimum computation, model complexity, and image database. The software structure allows streaming applications and real-time operation (with some performance optimizations). The inclusion of FML is a major feature for content description, and overall system is compatible with important multimedia-related standards and technologies like MPEG-4 FAPs and XMT framework and efficiently uses XML tools and underlying multimedia environment.

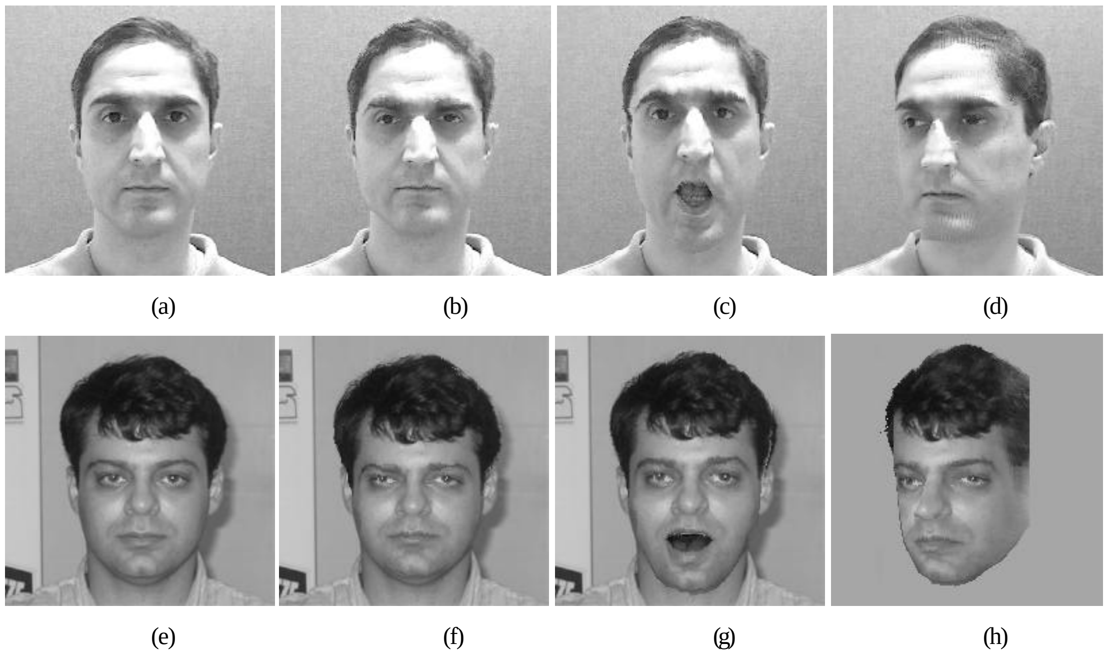


Figure 9: Transformed Faces, mapped from images at left (a,e): frown (b,f), viseme for sound "a" in "talk" (c,g), and rotate to right (d,h; image h does not have side information from a secondary image)

7. Concluding Remarks

An approach to a comprehensive framework for face animation is proposed. The main objectives of such a system are mentioned to be streaming capability, structured input description, on-demand audio/video generation, and flexible open architecture. A component-based architecture is designed and implemented, which uses standard interfaces to interact internally and also with other objects provided by underlying platform.

An XML-based Face Modeling Language (FML) is designed to describe the desired sequence of actions in form of a scenario. FML allows event handling, and also sequential or simultaneous combination of supported face states, and will be parsed to a set of MPEG-4 compatible face actions. Future extensions to FML can include more complicated behaviour modeling and better coupling with MPEG-4 streams.

The image-based transformations used in the video kernel are shown to be successful in creating a variety of facial states based on a minimum (sometimes only one) input images. They can be extended to include facial textures for better results, and the system allows even a complete change of image generation methods (e.g. using a 3D model), as long as the interfaces are supported.

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.

Combination of pre-learned transformations is used to create more complicated facial states. As discussed, due to perspective nature of head movements, this may not be a linear combination. Methods for shrinking/stretching the mapping vectors as a function of 3D head rotation are being studied and tested. Another approach can be defining the mapping vectors in term of relative position to other points rather than numeric values. These relational descriptions may be invariant with respect to rotations.

References

[1] Arafa, Y., et al. **Two approaches to Scripting Character Animation**, Proc First Int Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents, Bologna, Italy, July 2002.

[2] Arya, A., and B. Hamidzadeh, **An XML-Based Language for Face Modeling and Animation**, Proc IASTED Int. Conf. on Visualization, Imaging and Image Processing, 2002.

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

[4] Battista, S., et al. **MPEG-4: A Multimedia Standard for the Third Millennium**, IEEE Multimedia, 6(4), 1999.

[5] Blanz, V., and T. Vetter, **A Morphable Model for the Synthesis of 3D Faces**, Proc ACM SIGGRAPH, 1999.

[6] Bonamico, C., et al. **A Java-based MPEG-4 Facial Animation Player**, Proc Int Conf Augmented Virtual Reality& 3D Imaging, 2001.

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

[8] Bulterman, D. **SMIL-2**, IEEE Multimedia, 8(4), 2001.

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

[10] De Carolis, B., et al. **APML, a Markup Language for Believable Behaviour Generation**, Proc First Int Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents, Bologna, Italy, July 2002.

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

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

[13] Funge, J., et al. **Cognitive Modeling: Knowledge, Reasoning, and Planning for Intelligent Characters**, Proc ACM SIGGRAPH, 1999.

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

[15] Kallmann, M., and D. Thalmann, **A Behavioral Interface to Simulate Agent-Object Interactions in Real Time**, Proc IEEE Conf Computer Animation, 1999.

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

[17] Marriott, A., and J. Stallo, **VHML: Uncertainties and Problems. A discussion**, Proc First Int Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents, Bologna, Italy, July 2002.

[18] Pandzic, I.S. **A Web-based MPEG-4 Facial Animation System**, Proc Int Conf Augmented Virtual Reality & 3D Imaging, 2001.

[19] Pighin, F., et al. **Synthesizing Realistic Facial Expressions from Photographs**, Proc ACM SIGGRAPH, 1998.

[20] Prendinger, H., et al. **Scripting Affective Communication with Life-like Characters in Web-based Interaction Systems**, Applied Artificial Intelligence, vol.16, no.7-8, 2002.

[21] Tiddeman, B., et al. **Prototyping and Transforming Facial Textures for Perception Research**, IEEE CG&A, 21(5), 2001.