Face Animation: A Case Study for Multimedia Modeling and Specification Languages

Ali Arya and Babak Hamidzadeh

Dept. of Electrical & Computer Eng., Univ. 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 chapter will discuss the multimedia modeling and specification methods, especially in the context of face animation. Personalized Face Animation is and/or can be a major User Interface component in modern multimedia systems. After reviewing the related works in this area, we present the ShowFace streaming structure. This structure is based on most widely accepted industry standards in multimedia presentation like MPEG-4 and SMIL, and extends them by providing a higher level Face Modeling Language (FML) for modeling and control purposes, and also by defining image transformations required for certain facial movements. ShowFace establishes a comprehensive framework for face animation consisting of components for parsing the input script, generating and splitting the audio and video "behaviors", creating the required images and sounds, and eventually displaying or writing the data to files. This component-based design and scripted behavior make the framework suitable for many purposes including web-based applications.

Keywords: Face Animation, Talking Head, Multimedia, Modeling, Language, Structured Content Description, XML, FML, MPEG-4, SMIL, FML.

INTRODUCTION

Specifying the components of a multimedia presentation and their spatial/temporal relations are among basic tasks in multimedia systems. They are necessary when a client asks for a certain presentation to be designed, when a media player receives input to play, and even when a search is done to retrieve an existing multimedia file. In all these cases, the description of the contents 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. For instance a textual description of a scene can be a very effective "encoded" version of a multimedia presentation that will be "decoded" by the media player when it recreates the scene.

Face Animation, as a special type of multimedia presentation, has been a challenging subject for many researchers. Advances in computer hardware and software, and also new web-based applications, have helped intensify these research activities, recently. Video conferencing and online services provided by human characters are good examples of the applications using face animation. Personalized Face Animation

includes all the information and activities required to create a multimedia presentation resembling a specific person. The input to such 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, and finally an architecture to put different components together in a flexible way.

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 (e.g. facial actions) a very effective coding/compression mechanism, provided the visual effects of these actions can be recreated with a minimum acceptable quality. Based on this idea, in face animation, some researches have been done to translate certain facial actions into a predefined set of "codes". Facial Action Coding System (Ekman and Friesen, 1978) is probably the first successful attempt in this regard. More recently, MPEG-4 standard (Battista, et al, 1999) has defined Face Animation Parameters to encode low level facial actions like jaw-down, and higher level, more complicated ones like smile.

Efficient use of bandwidth is not the only advantage of multimedia content specifications like 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 whole new idea of representing the spatial and temporal relation of the facial actions. In a generalized view, 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.

Having a powerful description and specification mechanism, also is obviously powerful in search applications that currently suffer when looking for multimedia content. MPEG-7 standard (Nack and Lindsay, 1999) is the newest arrival in the group of research projects aiming at a better multimedia retrieval mechanism.

Considering three major issues of Content Delivery, Content Creation, and Content Description, following features can be assumed as important requirements in a multimedia presentation system (Arya and Hamidzadeh, 2002):

1- Streaming, i.e. continuously receiving/displaying data
2- Structured Content Description, i.e. a hierarchical way to provide information about the required content from high level scene description to low level moves, images, and sounds
3- Content Creation (Generalized Decoding), i.e. creating the displayable content based on the input. This can be decoding a compressed image or making new content based on the provided textual description.
4- Component-based Architecture, i.e. the flexibility to rearrange the system components, and use new ones as long as a certain interface is supported.

5- Compatibility, i.e. the ability to use and work with widely accepted industry standards in multimedia systems.
6- Minimized Database of audio/visual footage.

The technological advances in multimedia systems, speech/image processing, and computer graphics, and also new applications specially 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 later, is a step toward such a framework. It is based on a modular structure that allows multimedia streaming using existing technologies and standards like MPEG-4, Windows Media and DirectX/DirectShow (http://www.microsoft.com/windows/directx), and XML (http://www.w3.org/XML). The components independently read and parse a textual input, create audio and video data, and mix them together. Each component can be replaced and upgraded as long as it conforms to the ShowFace Application Programming Interface (SF-API). SF-API also allows other programs like web browsers to connect to ShowFace components directly or through a wrapper object called *ShowFacePlayer*. The system uses a language specifically designed for face animation applications. Face Modeling Language (FML) is an XML-based structured content description language that describes a face animation in a hierarchical way (from high level stories to low level moves), giving maximum flexibility to the content designers. Receiving FML scripts as input, *ShowFace* generates the required frames based on a limited number of images and a set of pre-learned transformations. This minimizes the image database and also computational complexity which are issues in existing approaches, as reviewed later.

In Section 2, some of the related works will be briefly reviewed. This includes different approaches to multimedia modeling and specification (content description in general), multimedia systems architectures to support those specification mechanisms, and eventually, content creation methods used in related multimedia systems. The basic concepts and structure of *ShowFace* system will be discussed in Section 3 to 5. This includes the proposed Face Modeling Language (FML) and the system structure and components for parsing the input and creating the animation. Some experimental results and conclusions will be the topics of Sections 6 and 7, respectively.

RELATED WORK

Multimedia Content Description

The diverse set of works in multimedia content description involves methods for describing the components of a multimedia presentation and their spatial and temporal relations. Historically, one of the first technical achievements in this regards were related to video editing where temporal positioning of video elements is necessary. The SMPTE (Society of Motion Picture and Television Engineers) Time Coding (Ankeney, 1995, Little, 1994) that precisely specifies the location of audio/video events down to the frame level is base for EDL (Edit Decision List) (Ankeney, 1995,

Little, 1994) that relates pieces of recorded audio/video for editing. Electronic Program Guides (EPGs) are another example of content description for movies in form of textual information added to the multimedia stream.

More recent efforts by SMPTE are focused on Metadata Dictionary that targets the definition of metadata description of the content (see http://www.smpte-ra.org/mdd). These metadata can include items from title to subject and components. The concept of metadata description is base for other similar researches like Dublin Core (http://dublincore.org), EBU P/Meta (http://www.ebu.ch/pmc_meta.html), and TV Anytime (http://www.tv-anytime.org). Motion Picture Expert Group is also another major player in standards for multimedia content description and delivery. MPEG-4 standard that comes after MPEG-1 and MPEG-2, is one of the first comprehensive attempts to define the multimedia stream in terms of its forming components (objects like audio, foreground figure, and background image). Users of MPEG-4 systems can use Object Content Information (OCI) to send textual information about these objects.

A more promising approach in content description is MPEG-7 standard. MPEG-7 is mainly motivated by the need for a better more powerful search mechanism for multimedia content over the Internet but can be used in a variety of other applications including multimedia authoring. The standard extends OCI and consists of a set of Descriptors for multimedia features (similar to metadata in other works), Schemes that show the structure of the descriptors, and an XML-based Description/Schema Definition Language.

Most of these methods are not aimed at and customized for a certain type of multimedia stream or object. This may result in a wider range of application but limits the capabilities for some frequently used subjects like human face. To address this issue MPEG-4 includes Face Definition Parameters (FDPs) and Face Animation Parameters (FAPs). FDPs define a face by giving measures for its major parts, as shown in Figure 1. FAPs on the other hand, encode the movements of these facial features. Together they allow a receiver system to create a face (using any graphic method) and animate based on low level commands in FAPs. The concept of FAP can be considered a practical extension of Facial Action Coding System (FACS) used earlier to code different movements of facial features for certain expressions and actions.

After a series of efforts to model temporal events in multimedia streams (Hirzalla, et al, 1995), an important progress in multimedia content description is Synchronized Multimedia Integration Language (SMIL) (Bulterman, 2001), and XML-based language designed to specify temporal relation of the components of a multimedia presentation, specially in web applications. SMIL can be used quite suitably with MPEG-4 object-based streams.

There have also been different languages in the fields of Virtual Reality and computer graphics for modeling computer-generated scenes. Examples are Virtual Reality Modeling Language (VRML, http://www.vrml.org) and programming libraries like OpenGL (http://www.opengl.org).

Another important group of related works are behavioural modeling languages and tools for virtual agent. BEAT (Cassell, et al, 2001) is an 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. Other scripting and behavioural modeling languages for virtual humans are considered by other researchers as well (Funge, et al, 1999, Kallmann, and Thalmann, 1999, Lee, et al, 1999). These languages are usually simple macros for simplifying the animation, or new languages which are not using existing multimedia technologies. Most of the times, they are not specifically designed for face animation.

Multimedia Content Creation

In addition to traditional recording, mixing, and editing techniques in film industry, computer graphics research has been long involved in multimedia and specially image/video creation. Two main categories can be seen in these works: 3D geometrical models (Blanz and Vetter, 1999, Lee, et al, 1999, Pighin, et al, 1998) and 2D image-based methods (Arya and Hamidzadeh, 2002, Bregler, et al, 1997, Ezzat and Poggio, 1998, Graf, et al, 2000).

The former group involves describing the scene using 3D data (as in VRML and OpenGL) and then rendering the scene (or sequence of frames) from any point of view. These techniques need usually complicated 3d models and data and also computation, but are very powerful in creating any view provided the model/data is inclusive enough. Due to the way images are generated, and inability to include all the details, most of these methods do not have a very realistic output. In case of face animation, 3D techniques are used by many researchers (Blanz and Vetter, 1999, Lee, et al, 1999, Pighin, et al, 1998). To reduce the size of required data for model generation, some approaches are proposed to create 3D model based on few (two orthogonal) 2D pictures (Lee, et al, 1999).

It is shown that any view of a 3D scene can be generated from combination of a set of 2D views of that scene or by applying some transformations on them (Arya and Hamidzadeh, 2002, Ezzat and Poggio, 1998). This fact is base for the latter group of techniques i.e. 2D image-based. In a Talking Head application, Ezzat, et al (Ezzat and Poggio, 1998) use view morphing between pre-recorded visemes (facial views when pronouncing different phonemes) to create a video corresponding to any speech. Optical flow computation is used to find corresponding points in two images, solving the correspondence problem for morphing. Bregler, et al, (Bregler, et al, 1997) combine a new image with parts of existing footage (mouth and jaw) to create new talking views. Both these approaches are 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 based on MikeTalk, (Graf, et al, 2000), recording of all visemes in a range of possible views is proposed, 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.

Defining general image transformations for each facial action and using facial feature points to control the mapping, seem to be helpful in image-based techniques.

TalkingFace (Arya and Hamidzadeh, 2002) combines optical flow and facial feature detection to overcome these issues. It can learn certain image transformations needed for talking (and potentially expressions and head movements) and apply them to any given image. Tiddeman, et al, (Tiddeman, et al, 2001) show how such image transformations can be extended to include even facial texture.

Multimedia Systems Architectures

Different architectures are proposed for multimedia systems. They try to address different aspects of multimedia, mostly streaming and playback. The main streaming systems, aimed at web-based transmission and playback, are Microsoft Windows Media, Apple QuickTime, and Real Networks RealVideo (Lawton, 2000).

Different architectures are also proposed to perform facial animation, especially as an MPEG-4 decoder/player (Pandzic, 2001). 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.

STRUCTURED CONTENT DESCRIPTION IN *SHOWFACE*

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, and even when a search is done to retrieve an existing multimedia file. 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. For instance a textual description of a scene can be a very effective "encoded" version of a multimedia presentation that will be "decoded" by the media player when it recreates the scene.

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 whole new idea of representing the spatial and temporal relation of the facial actions. In a generalized view, 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.

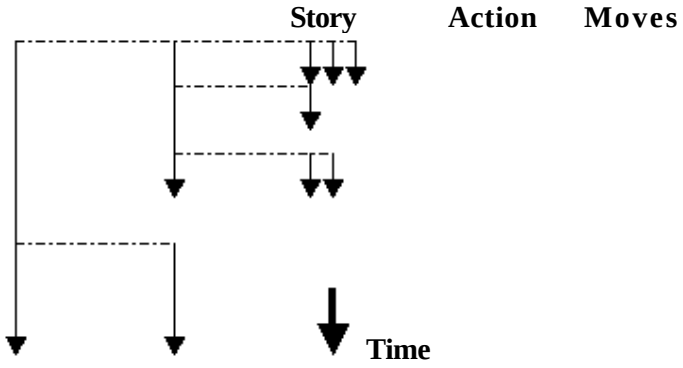


Fig. 1. FML Timeline and Temporal Relation of Face Activities

Face Modeling Language (FML) 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 behaviour templates
- Compatibility with MPEG-4 FAPs
- 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

The boundaries between these levels are not rigid and well defined. Due to complicated and highly expressive nature of facial activities, a single move can make a simple yet meaningful story (e.g. an expression). The levels are basically required by content designer in order to:

- Organize the content
- Define temporal relation between activities

- Develop behavioural templates, based on his/her presentation purposes and structure.

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/determined by a user. So another functionality of FML is to allow *user interaction* and in general *event handling* (Notice that user input can be considered a special case of *external event*.). This event handling may be in form of:

- Decision Making; choosing to go through one of possible paths in the story
- Dynamic Generation; creating a new set of actions to follow

A major concern in designing FML is compatibility with existing standards and languages. 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.

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.

```
<fml>
    <model>        <!-- Model Info -->
         <model-info />
    </model>
    <story>        <!-- Story Timeline -->
         <action>
              <time-container>
                   <move-set>
                        < . . . >
                   <move-set>
                   < . . . >
              </time-container>
              < . . . >
         </action>
         < . . . >
    </story>
</fml>
```

Fig. 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. It is possible to have more than one of each element but due to possible sequential execution of animation in streaming applications, a **model** element affect only those parts of document coming after it.

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

Fig. 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. This combination can be done using nested containers, as shown in Figure 4.

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

Fig. 4. Nested Time Container

FML also provides the means for creating a behavioral model for the face animation. At this time, it is limited to initialization data such as range of possible movements and image/sound database, and simple behavioral templates (subroutines). But it can be extended to include behavioral rules and knowledge bases, specially for interactive applications. A typical **model** element is illustrated in Figure 5, defining a behavioral template used later in **story**.

Event Handling and Decision Making

Dynamic 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. 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.

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

Fig. 5. FML Model and Templates

```
<event id="user" val="-1" />
<excl ev_id="user">
    <talk ev_val="0">Hello</talk>
    <talk ev_val="1">Bye</talk>
</excl>
```

Fig. 6. FML Decision Making and Event Handling

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, as illustrated in Section 5.

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)

Case Studies

1. Static Document

The first case is a simple FML document without any need for user interaction. There is one unique path the animation follows. The interesting point in this basic example is the use of loop structures, using **repeat** attribute included in any activity.

The **event** element specifies any external entity whose value can change. The default value for **repeat** is 1. If there is a numerical value, it will be used. Otherwise, it must be an **event** id, in which case the value of that **event** at the time of execution of related activity will be used. An FML-compatible player should provide means of setting external events values. *ShowFacePlayer* has a method called *SetFaceEvent*, which can be called by the owner of player object to simulate external events.

```
<event id="select" val="2" />
< . . . >
<seq repeat="select">
    <talk begin="0">Hello World</talk>
<exp begin="0" end="3" type="3" val="50" />
</seq>
```

Fig. 7. Repeated Activity. Using event is not necessary

2. Event Handling

The second example shows how to define an external event, wait for a change in its value, and perform certain activities based on the value. An external event corresponding to an interactive user selection is defined, first. It is initialized to -1 that specifies an invalid value. Then, an **excl** time container, including required activities for possible user selections, is associated with the event. The **excl** element will wait for a valid value of the event. This is equivalent to a pause in face animation until a user selection is done.

It should be noted that an FML-based system usually consists of three parts:

- FML Document
- FML-compatible Player
- Owner Application

In a simple example like this, it could be easier to simply implement the "story" in the owner application and send simpler commands to a player just to create the specified content (e.g. face saying Hello). But in more complicated cases, the owner application may be unaware of desired stories, or unable to implement them. In those cases, e.g. interactive environments, the owner only simulates the external parameters.

3. Dynamic Content Generation

The last FML example to be presented illustrates the use of XML Document Object Model (DOM) to dynamically modify the FML document and generate new animation activities.

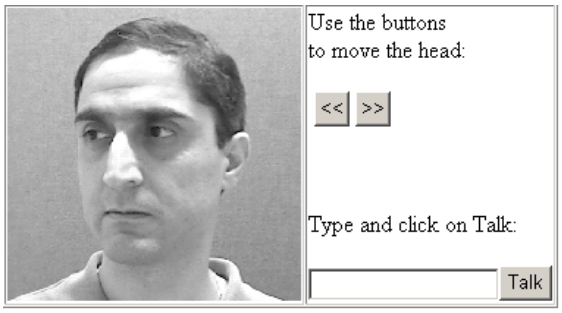


Fig. 8. Dynamic FML Generation

```
function onLoad()
{
    facePlayer.ReadFML("test.fml");
    facePlayer.Run();
}

function onHelloButton()
{
    facePlayer.SetFaceEvent(
        "user", 0);
}

function onByeButton()
{
    facePlayer.SetFaceEvent(
        "user", 1);
}
```

Fig. 9. JavaScript Code for FML Event shown in Figure 6

The simplified and partial JavaScript code for the web page shown in Figure 8 looks like this:

```
function onRight()
{
     var fml = fmldoc.documentElement;
     var new = fmldoc.createElement("hdmv");
     new.setAttribute("dir","0");
     new.setAttribute("val","30");
     fml.appendChild(new);
}
```

More complicated scenarios can be considered, using this dynamic FML generation, for instance, having a form-based web page and asking for user input on desired behavior, and using templates in **model** section of FML.

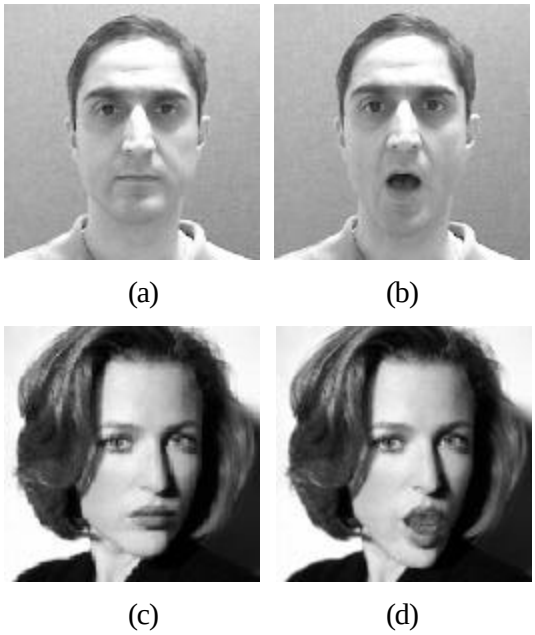
CONTENT CREATION IN *SHOWFACE*

Feature -based Image Transformation (FIX)

FML parser component of *ShowFace* system determines the visual activities required in the face. These activities are transitions between certain face *states* like a viseme or expression. In a training phase, a set of image-based transformations is learned by the system, which can map between these face states. Transformations are found by tracking facial features when the model is performing the related transitions, and then applied to a given image, as illustrated in Figure 10. A library of transformations is created based on following facial states:

- Visemes in full-view
- Facia l 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 (Figure 11).



(a) (b)

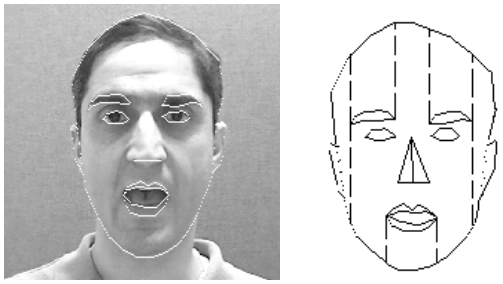
(c) (d)

Fig. 10. Facial Features and Image Transformations, (a) model state 1, (b) model state 2, (c) target character in state 1, (d) target character in state 2

Each transformation is defined in form of $T=(F,M)$ where T is the transformation, F is the feature set in the source image, and 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 12.

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.

Fig. 11. Moving Head States. The same three states exist for rotating to left, in addition to a full-view image, at the center.



(a) (b)

Fig. 12. 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 transformations are done by first applying the mapping vectors for the feature points. This is shown in Figure 13. Simple transformations are those which have already been learned, e.g. T_{12} and 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. T_{14}.

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.

$$T_{14} = a\ T_{12} + b\ T'_{13}$$
$$T'_{13} = f_p(T_{12}, , T_{13}) = T_{24}$$

where f_p is Perspective Calibration Function, and a and b are coefficients between 0 and 1 to control transition progress. T_{13} will also be scaled based on face dimensions in source/target image s.

When the *Image-2* is given, i.e. the new image does not have the same orientation as the one used in learning, the required transformation is T_{24} which still needs scaling/perspective calibration based on T_{13} and T_{12}.

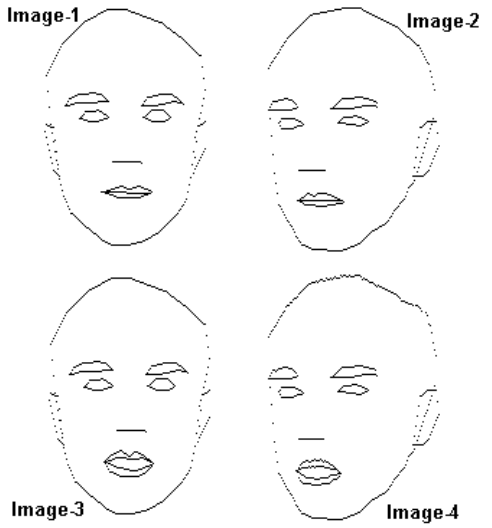


Fig. 13. Feature Transformation. T_{ij} is the transformation between Image-i and Image-j.

Facial Region Transformation

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 4-b.

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

$$\boldsymbol{d}_{r,c} = w(\boldsymbol{d}_{u,c}, \boldsymbol{d}_{l,c})$$

where the function w is a weighted average with distance as the weights, 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. Extra checking is performed to make sure a point inside a patch will be mapped to another point in corresponding patch of target image.

Sample Images
Figure 14 shows some sample outputs of the image transformations.

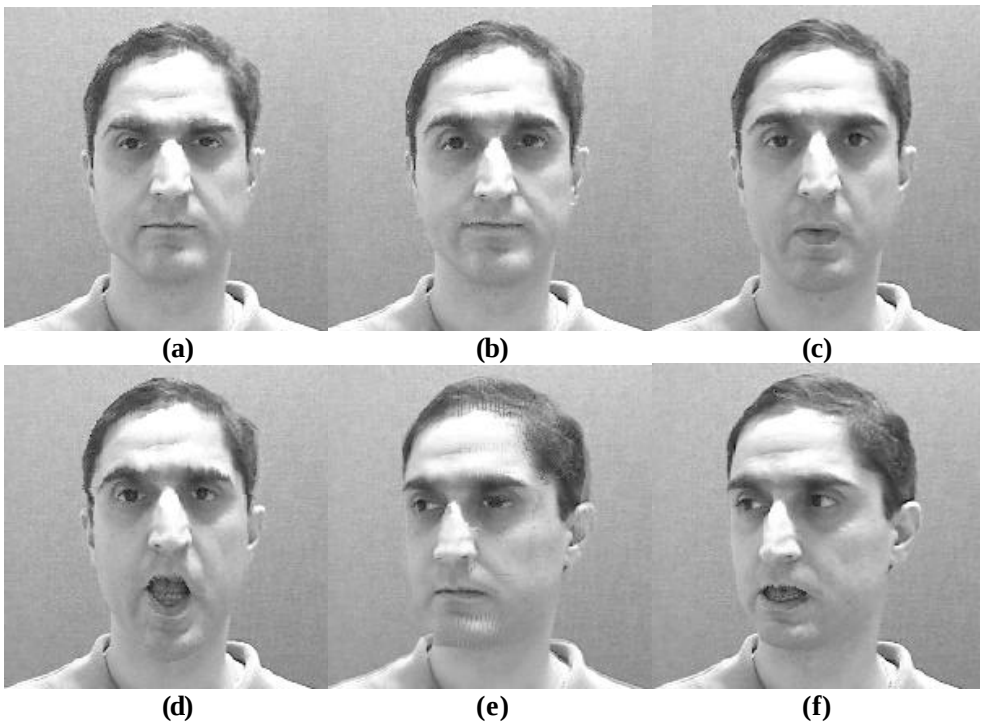


Fig. 14. Transformed Faces, mapped from 7-a. frown (a), smile (b), visemes for sounds "oo" and "a" in "root" and "talk" (c and d), rotate to right (e), and non-frontal talking (f)

Speech Synthesis

To achieve the best quality with minimum database requirements, *ShowFace* uses a concatenative approach to speech synthesis. Diphones (the transitions between the steady-state of a phoneme to the steady-state of another) are used as the basis of this approach. An off-line diphone-extraction tool is designed to create a database of diphones from existing audio footage. This database is normalized for power and pitch to provide a smooth initial set. The diphones are then dynamically connected based on the phoneme list of a given text to be spoken.

An FFT-based comparison of diphones finds the best connection point for two diphones at run time. This results in a dynamic time length calculation for diphones and words which will then be used to find the necessary duration of the corresponding visual transitions and the number of frames to be generated, in order to achieve a lip-synchronized audio-visual stream.

SHOWFACE FRAMEWORK

System Architecture

The basic structure of *ShowFace* system is illustrated in Figure 15. 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

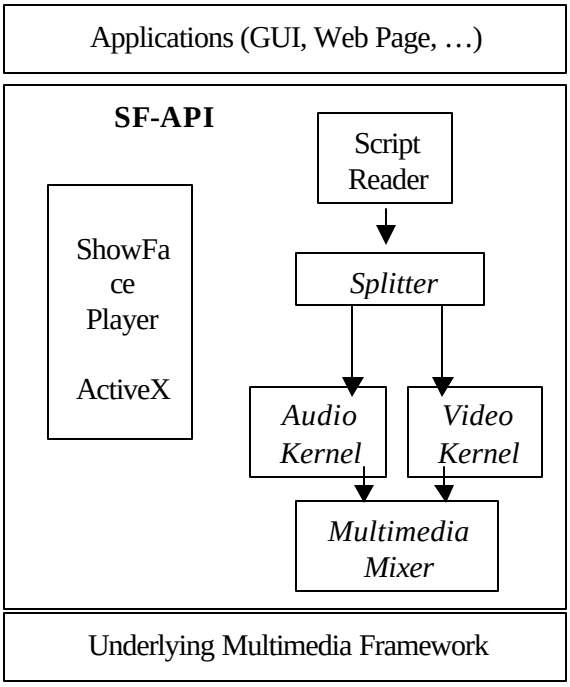


Fig. 15. Component-based *ShowFace* System Structure

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 is a multimedia streaming framework, which allows different media processing to be done by independent objects called *filters*, which can be connected using standard Component Object Model (COM) interfaces. 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.

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 the FML scripts. Some samples of transformed faces are shown in Figure 4.

CONCLUDING REMARKS

Reviewing the most important works in the area of multimedia specification and presentation, it's been shown that a comprehensive framework for face animation is a requirement which has not been met. Such a framework needs to provide:

- a structured content description mechanism,
- an open modular architecture covering aspects from getting input in standard forms to generating audio/video data on demand,
- efficient algorithms for generating multimedia with minimum use of existing footage and computational complexity.

An approach to such a framework, ShowFace System, is proposed. Unlike most of existing systems, ShowFace is not limited to an off-line application or a media player object, but provides a complete and flexible architecture. The component-based architecture uses standard interfaces to interact internally and also with other objects provided by underlying platform, making maximum use of existing technologies like MPEG-4, XML, and DirectX. These components can be used separately, or in a combination controlled by the animation application.

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. Main contributions of FML are its hierarchical structure, flexibility for static and dynamic scenarios, and dedication to face animation. Compatibility with MPEG-4 and use XML as a base are also among the important features in the language. 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 input images. Unlike 3D approaches, these transformations do not need complicated

modeling and computations. On the other hand, compared to usual 2D approaches, they do not use a huge database of 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. Ankeney, J. (1995). "Non-linear Editing Comes of Age", TV Technology, May 1995.

2. Arya, A., and Hamidzadeh, B. (2002). "ShowFace MPEG-4 Compatible Face Animation Framework", Int. Conf Computer Graphics and Image Processing (CGIP), Hawaii, 2002.

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

4. Blanz, V., and Vetter, T. (1999). "A Morphable Model For The Synthesis Of 3D Faces", ACM SIGGRAPH, 1999.

5. Bregler, C., et al (1997). "Video Rewrite: Driving Visual Speech with Audio", ACM Computer Graphics, 1997.

6. Bulterman, D. (2001). "SMIL-2," IEEE Multimedia, October 2001.

7. Cassell, J., et al (2001). "BEAT: the Behavior Expression Animation Toolkit", ACM SIGGRAPH, 2001.

8. Ekman, P., and Friesen, W.V. (1978). Facial Action Coding System, Consulting Psychologists Press Inc., 1978.

9. Ezzat, T., and Poggio, T. (1998). "MikeTalk: A Talking Facial Display Based on Morphing Visemes", IEEE Conf Computer Animation, 1998.

10. Funge, J., et al (1999). "Cognitive Modeling: Knowledge, Reasoning, and Planning for Intelligent Characters", ACM SIGGRAPH, 1999.

11. Graf, H.P., et al (2000). "Face Analysis for the Synthesis of Photo-Realistic Talking Heads", IEEE Conf Automatic Face and Gesture Recognition, 2000.

12. Hirzalla, N., et al (1995). "A Temporal Model for Interactive Multimedia Scenarios", IEEE Multimedia, Fall 1995.

13. Kallmann, M., and Thalmann, D. (1999). "A Behavioral Interface to Simulate Agent-Object Interactions in Real Time", IEEE Conf Computer Animation, 1999.

14. Lawton, G. (2000). "Video Streaming", IEEE Computer, July 2000.

15. Lee, W. S., et al (1999). "MPEG-4 Compatible Faces from Orthogonal Photos", IEEE Conf Computer Animation, 1999.

16. Little, T.D.C. (1994). "Time-based Media Representation and Delivery," in Multimedia Systems, J.F. Koegel Buford (ed)., ACM Press, 1994.

17. Nack, F., and Lindsay, A.T. (1999). "Everything You Wanted To Know About MPEG-7", IEEE Multimedia, July 1999.

18. Pandzic, I.S. (2001). "A Web-based MPEG-4 Facial Animation System", Int Conf Augmented Virtual Reality& 3D Imaging, 2001.

19. Pighin, F., et al (1998). "Synthesizing Realistic Facial Expressions from Photographs", ACM SIGGRAPH, 1998.

20. Tiddeman, B., et al (2001). "Prototyping and Transforming Facial Textures for Perception Research", IEEE CG&A, September 2001.

Photo and Biography
Ali Arya

Ali Arya received a B.S. degree in Electrical Engineering from Tehran Polytechnic, Iran, in 1990, and currently is a Ph.D. Candidate at the Department of Electrical and Computer Engineering, University of British Columbia, Canada. He has worked as research engineer, system analyst, and project manager in different research centers and industry-related companies, including Tehran Cybernetic Arm Project, Iran, and Honeywell-Measurex, Canada. His research interests include multimedia, computer graphics, real-time systems, and web-based applications. He is also an instructor for "Systems Software Engineering" and "Software Project Management" in UBC.

Photo and Biography
Babak Hamidzadeh

Babak Hamidzadeh received M.S. and Ph.D. degrees in Computer Science and Engineering from The University of Minnesota in Minneapolis, in 1989 and 1993, respectively. In that period, he also worked as a research associate at The Systems and Research Center of Honeywell Inc., and as a research scientist at The Research and Technology Center of Alliant Techsystems Inc. for over 3 years. From 1993 to 1996 he was an Assistant Professor of Computer Science and Computer Engineering at The Hong Kong University of Science and Technology. Currently, he is an Associate Professor of Electrical and Computer Engineering at The University of British Columbia. He is also a member of IEEE Computer Society, a Fellow of the BC Advanced Systems Institute and the holder of a Canada Research Chair in Information Technology. His areas of research include resource management and scheduling, real-time computing, parallel and distributed processing, database systems, multimedia, and communication networks.