PERSONALIZED FACE ANIMATION FRAMEWORK FOR MULTIMEDIA SYSTEMS

by

Ali Arya
B.Sc., Tehran Polytechnic, Iran, 1989

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES
(Department of Electrical and Computer Engineering)

We accept this thesis as conforming
to the required standard.

....

THE UNIVERSITY OF BRITISH COLUMBIA

February 2004

Abstract

Advances in multimedia-related technologies are enabling new applications such as virtual agents, video conferencing, visual effects in movies, and virtual players in computer games. Such applications are, in turn, motivating much research in digital character and face animation. This thesis addresses an important area in this field, Personalized Face Animation which is concerned with creating multimedia data representing the facial actions of a certain character, such as talking, expressions, and head movements. Much success has been achieved for this purpose using 3D head models (general and customized to specific individuals) and also view morphing based on 2D images. The model acquisition and computational complexity of 3D models, and large image databases for 2D methods, however, are major drawbacks. The thesis addresses these issues along with other important ones, mainly realism, authoring tools, content description, and architecture of the whole face animation system.

We propose a comprehensive framework for personalized face animation which we call *ShowFace*. *ShowFace* integrates a component-based architecture, well-defined interfaces, helper objects and tools with a simple, yet effective, approach to content generation. These are paired with a language for describing face animation events. *ShowFace* is designed to satisfy the following basic requirements of face animation systems:

- Generalized decoding of short textual input into multimedia objects that minimizes the model complexity and database size
- Structured content description for face activities like talking, expressions, and head movement, their temporal relation, and hierarchical grouping into meaningful stories
- Streaming for continuously receiving and producing □frames□ of multimedia data
- Timeliness issues
- Compatibility with existing standards and technologies and
- Efficiency with regards to algorithms and required data

ShowFace achieves this objective by introducing:

- Feature-based image transformations along with a 2D image-based method for creating MPEG-4 compatible and realistic facial actions. This is accomplished without the need for a complicated 3D head model and/or large databases of 2D images
- A face modeling language which is an XML-based language. It is compatible with MPEG-4 standard and specifically designed for face animation It is also capable of describing spatial and temporal relations of facial actions, behavioural templates, and external event handling.
- A component-based structure for development of animation applications. This structure has a well-defined interface, independently usable components, and streaming capability and
- A comprehensive set of evaluation criteria for face animation systems

The thesis review basic concepts and related work in the area of face animation. Then the *ShowFace* system is introduced and its contributions are thoroughly discussed. A comparative evaluation of the system features and performance is also provided.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
API	Application Programming Interface
AU	Action Unit
AVI	Audio-Video Interleaved
BEAT	Behaviour Expression Animation Toolkit
CML	Cognitive Modeling Language
COM	Component Object Model
DOM	Document Object Model
ECA	Embodied Conversational Agent
EDL	Edit Decision List
EPG	Electronic Program Guide
FACS	Facial Action Coding System
FAP	Face Animation Parameters
FDP	Face Definition Parameter
FFT	Fast Fourier Transform
FIX	Feature-based Image Transformations
FML	Face Modeling Language
FTF	Feature Translation Function
MPEG	Motion Picture Experts Group
MPML	Multimodal Presentation Markup Language
OCI	Object Content Information
OF	Optical Flow
PCF	Perspective Calibration Function
SDK	Software Development Kit
SF-API	ShowFace Application Programming Interface
SMIL	Synchronized Multimedia Integration Language
SMPTE	Society of Motion Picture and Television Engineers

TTS	Text-To-Speech
VDP	Visible Difference Predictor
VHD	Virtual Human Director
VHML	Virtual Human Markup Langugae
VRML	Virtual Reality Modeling Language
XML	Extensible Markup Language
XMT	Extensible MPEG-4 Textual format

Acknowledgements

The work presented in this thesis is done under the supervision of Dr. Babak Hamidzadeh and Dr. Rabab Ward, at the Department of Electrical and Computer Engineering (ECE), University of British Columbia (UBC). Their effective guidance and insightful ideas, throughout the entire project, are greatly appreciated. This research has also benefited from invaluable helps by Dr. David Lowe of UBC Department of Computer Science (CS), to whom the author offers the most sincere gratitude. The author also wishes to thank the heads and professors at UBC departments of ECE and CS, and Institute for Computing, Information, and Cognitive Systems (ICICS), for the wonderful learning and research opportunity they have provided to the students.

Special thanks go to ECE, CS, and ICICS staff, in particular Doris Metcalf (ECE Graduate Program Clerk), Cathleen Holtvogt (ECE Administrative Clerk), and Kirsty Barclay (ICICS Technical Writing Advisor), for their administrative and technical support, and also other ECE and CS graduate students for the active, supportive, and friendly environment they have created.

Last but not least, the author would like to thank his dear friends, Mr. Kouros Goodarzi and his wife Parisa, for reviewing and proofreading the final manuscript, and their love, friendship, and patient support. The kindness of these and other friends and family members have made this research, and a lot more, possible. Thank you all !

To my mother and sister,

who have been the greatest sources of inspiration in all my life.

1. Introduction

I frequently look up reverently at the image of him (St. Ambrose) high on the church wall, which is said to be a very close resemblance, and almost alive and breathing in the stone. This is no small reward for having come here. I can not describe to you the authority in his face, the majesty of his brows, the serenity of his gaze. Were his voice not lacking, this would be the living Ambrose.

Francesco Petrarch – 1353 [68]

1.1. Background and Motivations

From the humble cavemen painting on the walls of their caves to the esteemed masters of the École des Beaux Arts in 19th century Paris, to the animators of recent motion pictures such as *Antz* and *The Matrix*, visual artists have long been creating views that represent ideas, events, and characters. Regardless of what they represent in their work, and how they choose to do so, these artists share one common role: visualization. In the absence of other means, their creative minds and hands were what they had to bring ideas into form. The invention of camera brought a new player into this game.

Cameras opened up a new approach to visualization, that of actually "recording" views instead of "creating" them [1], and gradually, they proved to be quite reliable and precise in doing it. Although the technology was very limited at the beginning, it was not hard to foresee its improvement over time, as it actually happened. The ability of the new devices to record and display scenes posed an important question to visual artists: if it is possible to *record* subjects, is it still significant to draw or paint them? Modern art has a definite, positive answer to this question. The key to this answer lies in realizing that visualization is not simply a mirroring of the "real" world. It is also about representing things that do not exist externally, or are hard to capture on camera, or representing existing subjects but in different ways (for instance, introducing the effect of the artist's viewpoint and impressions).

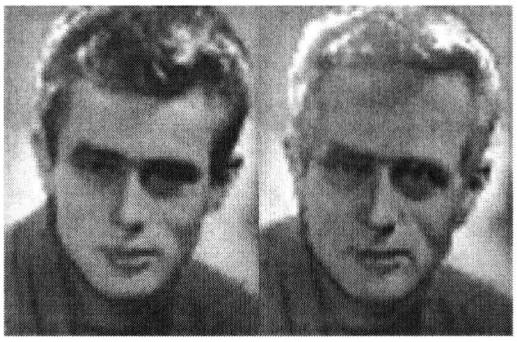
[1] In one sense, recording can be considered a special type of creating content, but here we use the term "creation" to mean generating "new" content, even if based on existing materials.

This realization led to a variety of styles that form what we now call modern art. In 1890, Vincent Van Gogh wrote [101]:

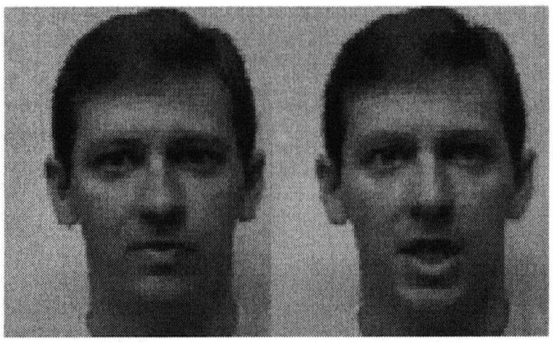
> *What excites me most in my profession, much, much more than anything else, is portraiture, modern portraiture. I endeavour to do this through colour ... you see, I should like ... to paint portraits which a hundred years from now will seem to the people of those days like apparitions. Thus I do not attempt to achieve this through photographic resemblance, but through our impassioned aspects, using our science and our modern taste for colour as a means of expression and of exaltation of character.*

The 20th century saw many advances in visual art and technology, and many interactions between them, and the ability of view creation and view recording to co-exist was proved even more strongly. The entertainment industry (and especially movies) provides good examples of such co-existence within traditional and computer-generated animation, visual effects, and other areas. Although pre-recorded content played a dominant role for decades, technological achievements have given artists and other content providers the means of "creating" desired multimedia content. As a result, we see more and more synthetically created, or modified, elements in traditional multimedia presentations such as movies, and also newer types such as computer games and virtual/augmented/virtualized environments [12,63]. In all these cases we still see the key concept of visualization. Even with all the achievements in "recording" multimedia data, "creating" is necessary when the subject does not exist, is not available, or needs non-real modifications. Character Face Animation is an important example in this area.

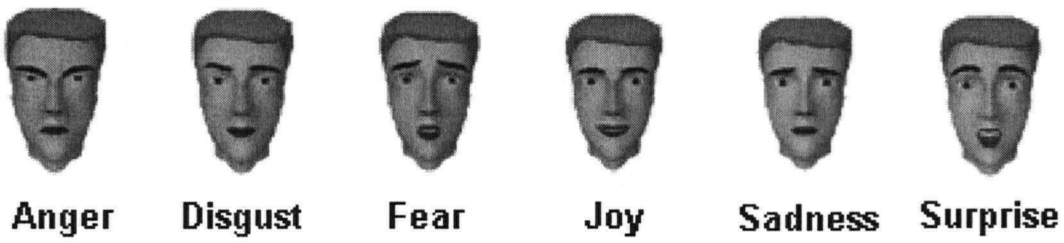
Recent developments in multimedia-related areas such as Virtual Environment, Video Conferencing, Computer Games, and Agent-based Online Applications have drawn considerable attention to character animation [73,76,77,92,94,95,96,98]. Replacing the audio-visual data of "real people" with "realistic" multimedia presentations based on "virtual software agents" seems to be very beneficial, and in some cases necessary. For example, it is beneficial to save bandwidth in video conferencing by replacing video data with animation "commands"; while creating new movie scenes with "unavailable" characters is a necessity. Figure 1-1 illustrates some exemplary cases.



(a)



(b)



(c)

Figure 1-1. Some Applications of Virtual Software Agents

(a) Aging James Dean by Perception Lab, University of St. Andrews, Scotland [97]
(b) Visual Speech Synthesis by MikeTalk [41]
(c) Synthetic Facial Expressions for Virtual Environments [78]

Different methods have been proposed for computer face animation [30,80,88,89,95]. While each of these methods has its own advantages, they usually require a complicated 3D model of the head, or a relatively large database of 2D images. The capability to "personalize" a character (i.e. animating a specific individual) without recreating the model or database is also another missing feature in some of the proposed approaches. Compatibility with multimedia technologies and standards, at algorithm or system levels, is also another requirement of face animation systems that is not addressed by all proposed methods. Examples of such compatibility are MPEG-4 [1] Face Animation Parameters (FAPs) [13,40] and streaming structure [13,59].

Besides the emergence of new techniques for creating animation content, modern multimedia presentations have also evolved in their styles and formats. They no longer a one-piece pre-recorded stream of audio-visual data but a combination of processed and/or

[1] Motion Picture Experts Group

synthesized components. These components include traditional recordings, hand-drawn animation, computer-generated sound and graphics, and other media objects required to be created and put together based on certain spatial and temporal relations. This makes content description one of the basic tasks in multimedia systems and a basic means of "interaction" with "virtual agents". Such a description can also lead to a "generalized encoding", since it represents the multimedia content in a form not necessarily the same as the playback format, and usually is 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. Also, having the content well described makes the search and retrieval of multimedia data easier.

Efficient encoding is not the only advantage of content specifications. In many situations, 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 novel idea of representing the desired spatial and temporal relation of multimedia objects. In a generalized view, such a description provides a hierarchical structure with elements ranging from low-level "images," to simple "moves," and more complicated "actions," to complete "stories". We call this a Structured Content Description, which also requires means for defining capabilities, behavioural templates, dynamic contents, and event/user interaction.

Finally, in addition to topics such as content creation, presentation format, and compression, the study of computer-generated character animation (like any other software problem) should include system-level consideration and architectural issues such as modularity, re-use, openness, and last but not least, performance evaluation criteria. Some questions to be answered in this regard are the following:

- What are the requirements of a face animation system?
- What architectural components are needed?
- How can we evaluate such a system?
- What do features like realism and interactivity really mean?

The work presented in this thesis tries to address the issues we have introduced so far, in the limited context of generating face animation for specific characters. In the following section of this chapter, we focus on the problem of Personalized Face Animation and clarify the objectives of our research in more detail.

1.2. Personalized Face Animation

1.2.1. Problem Model

Personalized Face Animation is the creation of multimedia presentation of specific individuals with a variety of facial activities, such as talking and expressions. It utilizes a set of input data (e.g. descriptions and commands) and some knowledge of the character to be animated (including but not limited to 3D head models, 2D images, and textual descriptions). Face animation is a challenging area of computer graphics and multimedia systems research [30,80,95]. Realistic and personalized face animation is the basis for Virtual Software Agents that can be used in many applications, including video conferencing, online training and customer service, visual effects in movies, and interactive games. Table 1-1 summarizes some of these applications for virtual software agents.

Table 1-1. Example Applications for Virtual Software Agents

Application	**Description**
Video Phone/Conference	Sending a text description of video to be created at the receiver side; this serves as a highly efficient compression form for use in low-bandwidth channels. Recreating lost or extra frames in a video stream
Online Services	Realistic interactive trainer, customer service representative, operator, etc (requires proper algorithms to control the agent and decide the actions)
Special Effects in Movies	Recreating "unavailable" characters in new situation, with new speeches, expressions, etc
Computer Games	Interactive characters

A software agent can play the role of a trainer, a corporate representative, a specific person in an interactive virtual world, and even a virtual actor. Using this technology, movie producers can create new scenes including people who are not physically available. Further, communication systems can represent a caller without any need to transmit high volume multimedia data over limited bandwidth lines. Adding intelligence to these agents makes them ideal for interactive applications such as on-line games and customer service. In general, the ability to generate new and realistic multimedia data for a specific character is of particular importance in cases where pre-recorded footage is unavailable, difficult or expensive to generate or transmit, or simply too limited due to the interactive nature of the application.

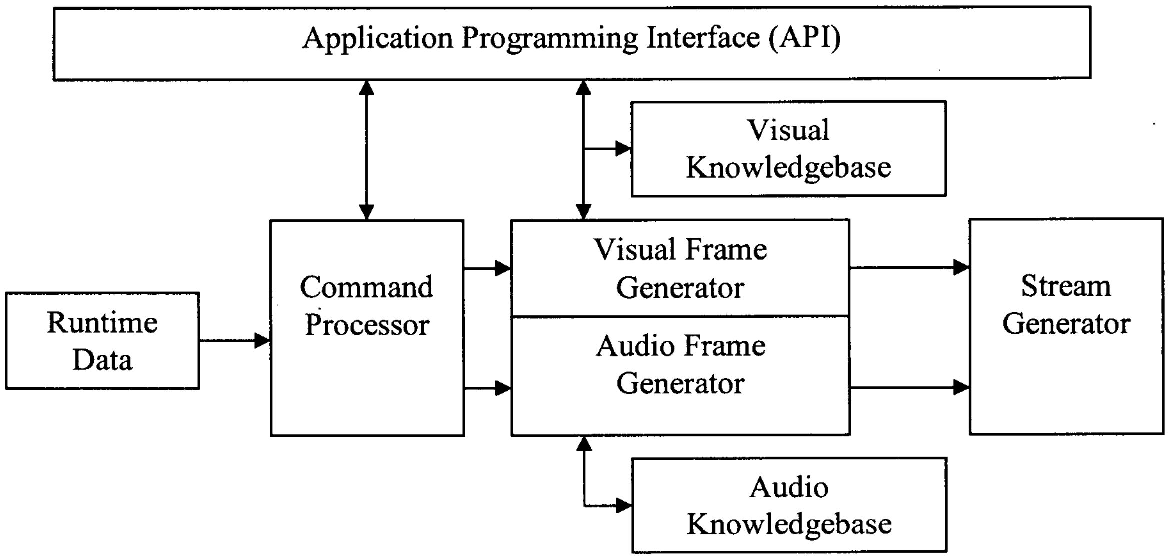


Figure 1-2. Personalized Face Animation System

Personalized Face Animation includes all algorithms, data, modules, and activities required to create a multimedia presentation resembling a specific person. Although most of research on face animation has been focused on parts of the complete system, personalized face animation requires a comprehensive framework that allows proper interaction by users and different applications. We envision the building blocks of this framework as shown in Figure 1-2. The input to such a system can be a combination of audio-visual data and text commands describing the requested animation. A successful face animation system needs to have efficient yet powerful solutions for the following functions:

- Processing the input in order to determine the required multimedia objects
- Generating the multimedia objects
- Streaming the output for the next stages (playback, file, remote systems, etc)

The above functions are implemented through the use of appropriate audio and visual knowledge basis. These may be libraries of images and speech segments; they may also be head or vocal tract models, or pre-learned transformations and behaviours. Complexity of the models and related computation, size and flexibility of databases, and quality of produced multimedia objects are major concerns, in this regard. Finally, client applications should be able to interact with the system components through a well-defined interface.

Due to its nature, personalized face animation has close relations to many other areas in Computer Science and Engineering, for example computer graphics, computer vision, human-machine interface, and speech synthesis, as illustrated in Figure 1-3.

Computer graphics techniques play a major role in creating virtual software agents. 3D and 2D methods are used to model and construct views of virtual characters. In relation to graphics techniques, image processing operations are used to apply filters and other transformations to images. An example of this is the generation of intermediate views between key frames using Metamorphosis (View Morphing) [15]. Optical flow (OF) is a method used in computer vision for motion detection [14]. It can be used by software agents for finding control points in morphing. Other computer vision techniques can also be useful for detecting heads, faces, and facial features in order to mimic the activities of a real person, or simply map the features to new states (e.g. moving lips of a real image to create visual speech).

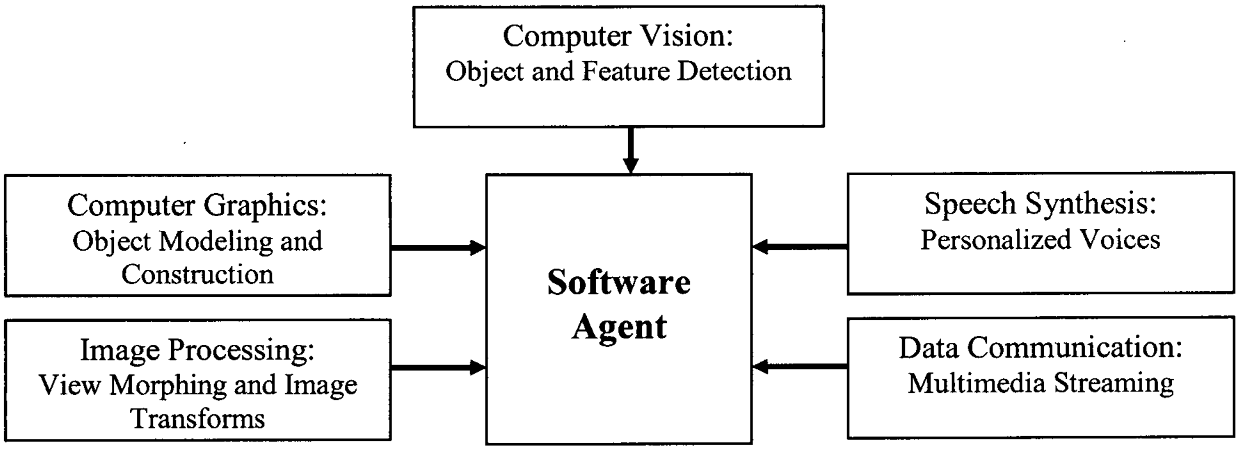

Computer Vision: Object and Feature Detection
Computer Graphics: Object Modeling and Construction
Image Processing: View Morphing and Image Transforms
Software Agent
Speech Synthesis: Personalized Voices
Data Communication: Multimedia Streaming

Figure 1-3. Personalized Face Animation and Related Domains

Due to the multimedia nature of software agents, speech synthesis also plays an important role. Different text-to-speech (TTS) systems [37] can be used in order to create proper audio output for an agent. These are discussed in the coming chapters, but due to a considerable amount of existing research, and the availability of commercial and free products, audio generation is not a main focus of this thesis.

Finally, software agents need to work with data communication systems (especially web-based ones) to provide capabilities such as streaming and efficient data transfer. Compatibility with different multimedia streaming and compression standards is an important issue in this regard.

1.2.2. Objectives

Considering the three major issues, content delivery, content creation, and content description, the following features can be assumed to be important requirements in a multimedia presentation system in general, and in face animation specifically:

- Generalized Decoding: creating the displayable content with acceptable quality based on some inputs. This may be decoding a compressed image or making a new image, as requested.
- Structured Content Description: a hierarchical way to provide information about content from high-level scene description to low-level moves, images, and sounds
- Streaming: structural and computational fitness for continuously receiving and displaying data
- Component-based Architecture: the flexibility to rearrange the system components, and use new ones, as long as a certain interface is supported
- Compatibility: the ability to use and work with widely accepted industry standards in multimedia systems
- Algorithm and Data Efficiency: a minimized database of audio-visual footage and modeling/input data, and simple efficient algorithms

Recently, technological advances have rapidly increased in multimedia systems, speech/image processing, and computer graphics, and also in new applications, especially in computer-based games, telecommunication, and online services. These have resulted in a rapidly growing number of related publications. These research achievements, although very successful in their objectives, mostly address a limited subset of the above requirements. A comprehensive framework for personalized face animation is still in the conceptual stages.

This thesis introduces the *ShowFace* system that we have developed as a step toward such a framework.

ShowFace uses a modular structure consisting of components for content description, creation, and delivery (streaming and playback). The components are related to each other through the *ShowFace* Application Programming Interface (SF-API) and are accompanied by tools for off-line design and wrapper objects to simplify application development. *ShowFace* introduces Face Modeling Language (FML) and Feature-based Image Transformations (FIX) as the main tools for content description and visual content creation.

Although the research in this area has clearly improved the quality and performance of face animation systems, a well-defined set of criteria to evaluate such systems has not been developed, yet. The words "Realistic," "Real-time," and "Compatible" are often used in the literature to describe the proposed methods and systems. These words however often lack clear definition in a specific application domain; some equally important issues are also omitted from many recent proposals. In this thesis, we compile the most important objectives considered in the literature, combine them with some general and application-specific requirements, give more precise definitions, and then, as much as possible, provide measures of evaluating a proposed system against those requirements.

1.3. Thesis Structure

This thesis introduces a system that we have developed and named *ShowFace*. *ShowFace* is a comprehensive face animation framework that is designed with the objectives presented in 1.2.2. The *ShowFace* system offers the following features as the main contributions of this thesis:

- A component-based architecture [2,5,7] with streaming capability for face animation with off-line authoring tools and web-ready objects
- A structured content description using Face Modeling Language (FML) [4,6], a language we have developed specifically for face animation, based on the Extensible Markup Language (XML, http://www.xml.org)

- An efficient personalized face animation with Feature-based Image Transformation (FIX) [3,5], a technique based on learning image transformation using only 2D images, and applying them to a given 2D image
- A comprehensive set of evaluation criteria for face animation systems based on high-level objectives

In Chapter 2, some of the existing work and approaches in face animation and the following related areas are briefly reviewed:

- Multimedia modeling and specification (content description in general)
- Visual and audio content creation
- Architectural and system-level issues
- Evaluation criteria

The basic concepts and structure of our Face Modeling Language, Feature-based Image Transformations, and the *ShowFace* system are discussed in Chapters 3, 4, and 5, respectively. Chapter 6 is devoted to defining the evaluation criteria and the experimental results of this project. Some concluding remarks are presented in Chapter 7.

2. Related Work

2.1. 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, some of the first technical achievements in this area were related to video editing where temporal positioning of video elements is necessary. The SMPTE (Society of Motion Picture and Television Engineers) Time Coding [10,65] that precisely specifies the location of audio/video events down to the frame level is base for EDL (Edit Decision List) [10,65] that relates pieces of recorded audio/video for editing. Electronic Program Guide (EPG) is another example of content description for movies in the 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 content (see http://www.smpte-ra.org/mdd). Metadata can include items from title to subject and components. The concept of metadata description is base for other similar research projects such as 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 (MPEG) is another major player in the field of standards for multimedia content description and delivery. MPEG-4 standard [13], introduced 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 such as 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 the MPEG-7 standard [75]. MPEG-7 is mainly motivated by the need for a better and more powerful search mechanism for multimedia content over the Internet. It can also 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.

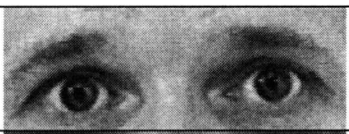
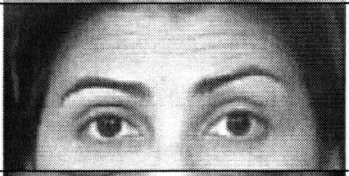
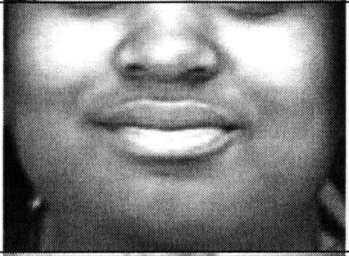
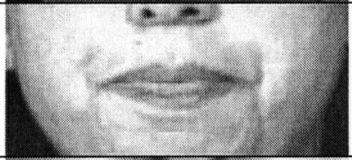
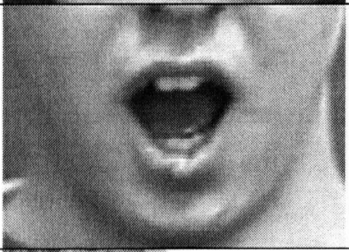
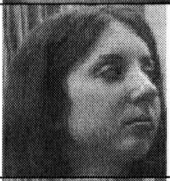
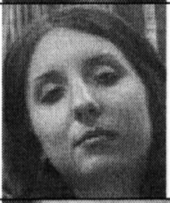
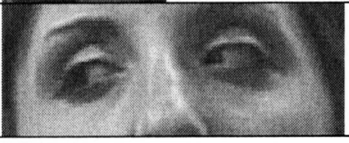
Most of these methods are not aimed at, and customized for, a certain type of multimedia stream or object. This may result in a wider range of applications but limits the capabilities for some frequently used subjects such as human face. Study of facial movements and expressions started from a biological point of view. After some older investigations, for example by John Bulwer [22] in late 1640s, Charles Darwin's book *The Expression of the Emotions in Men and Animals* [32] can be considered a major departure for modern research in behavioural biology.

More recently, one of the most important attempts to describe facial activities (movements) was Facial Action Coding System (FACS) [38]. Introduced by Ekman and Friesen in 1978, FACS defines 64 basic facial Action Units (AUs). A major group of these Action Units represent primitive movements of facial muscles in actions such as raising brows, winking, and talking. Eight AUs are for 3D head movements, i.e. turning and tilting left and right and going up, down, forward and backward. Table 2-1 illustrates some of these Action Units.

FACS has been successfully used for describing desired movements of synthetic faces and also in tracking facial activities [80]. It forms a comprehensive set of codes for most important facial actions needed in face animation and can be a good basis for animation control and authoring but it does not provide any higher level construct to describe the sequence of actions and spatial and temporal relations between facial activities. In other words, FACS is not designed to be a face animation description language.

Table 2-1. Facial Action Coding System, Partial List of Action Units

**Images and AU descriptions from Automated Face Analysis Project
Robotics Institute, Carnegie Mellon University
(http://www-2.cs.cmu.edu/afs/cs/project/face/www/facs.htm)**

AU	Description	Example Image
1	Inner Brow Raiser	
2	Outer Brow Raiser	
12	Lip Corner Puller	
24	Lip Pressor	
27	Mouth Stretch	
51	Head Turn Left	
56	Head Tilt Right	
61	Eye Turn Right	

In the early 1990's, some newer approaches to content description for face animation were introduced in the form of dedicated animation languages. Reeves [88] describes a language used in Pixar (http://www.pixar.com) that allows the face animation model to be treated as a program. This special language has built-in functions for creating graphical primitives and performing simple transformations on them. Many aspects such as realism and synchronization are not addressed. Kalra et al. [53] describe a face animation system based on layered abstractions with another special language that provides simple synchronizations. The layers correspond to low-level muscles, minimal perceptible actions, phonemes and expressions, words and emotions, and finally synchronization of activities. Following is an example of this language:

```
[ emotion HAPPY while
        [ say "Hi"]
]
```

Although the system lacks detailed synchronization (e.g. event handling) and does not use an open technology and standard language (which did not exist at that time) but the main concepts of layered description and synchronization of facial actions are major contributions.

MPEG-4 standard also uses an approach similar to FACS to integrate face animation into multimedia communication, including natural and synthetic audio, natural and synthetic video, as well as 3D graphics [13]. To do this, MPEG-4 includes Face Definition Parameters (FDPs) and Face Animation Parameters (FAPs). FDPs define a face by giving coordinates and other information for its major feature points such as eyes and lips corners. They allow personalization of a generic face model to a particular face, and are more suitable for synthetic faces. FAPs on the other hand, encode the movements of these facial features. There are more than 70 FAPs defined similar to FACS AUs. MPEG-4 FAPs can be grouped into:

- Visemes (visual representation of phonemes, i.e. the facial state when uttering a specific sound)
- Facial Expressions
- 3D Head Movements
- Other Facial Movements

It should be mentioned that Phonemes are possible sounds uttered when speaking a language. Phonemes can share the same visual representation due to their similarities, although they are different with regards to the vocal tract. MPEG-4 defines only 14 clearly distinguished visemes and 6 facial expressions (Figure 1-1-c). More details on FAPs and FDPs can be found online at the following URL:
http://www.research.att.com/projects/AnimatedHead/joern2.html

Together, FDPs and FAPs allow a receiver system to create or customize a face model (using any graphic method) and animate it based on low-level commands in FAPs. It should be noted that FAPs do not need to be used with a synthetic face and geometric models. They are independent of animation method and simply define the desired movements. They can be used to apply image transformations to a real 2D picture in order to create a visual effect such as talking, facial expression, or any facial movements.

Although MPEG-4 defines two sets of higher-level codes, i.e. visemes and expressions, compared to low-level FACS AUs, but it still has only a set of animation commands and not an animation language. After a series of efforts to model temporal events in multimedia streams [50,60,87], an important progress in multimedia content description is Synchronized Multimedia Integration Language (SMIL) [21], an 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), its XML-based version known as X3D, and programming libraries like OpenGL (http://www.opengl.org).

MPEG-4 standard includes Extensible MPEG-4 Textual format (XMT) framework [55] to represent scene description in a textual format providing interoperability with languages such as SMIL and VRML. It consists of two levels of textual formats. XMT-A is a low-level XML-based translation of MPEG-4 contents. XMT-Ω is a high-level abstraction of MPEG-4 features, allowing developers to create the scene description in languages such as SMIL and VRML. These descriptions can then be compiled to native MPEG-4 format to be played back by MPEG-4 systems. They can also be directly used by compatible players and browsers for each language, as shown in Figure 2-1.

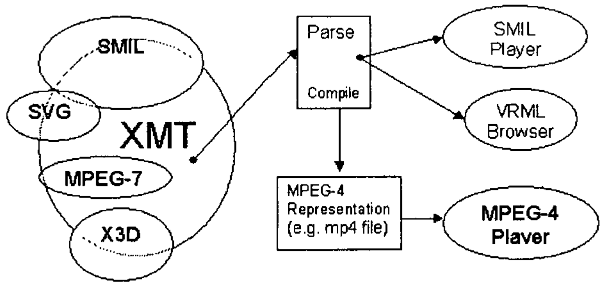


Figure 2-1. Interoperability in XMT

None of these languages are customized for face animation, and they do not provide any explicit support for it, either. The absence of a dedicated language for face animation, as an abstraction on top of FACS AUs or MPEG-4 FAPs, has been evident especially within XMT framework. 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 markup languages for virtual characters [11,33,71,84]. The basic idea is to define specific XML tags related to agents' actions such as moving and talking.

Virtual Human Markup Language (VHML) [71] 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, emotional representation, dialogue management, and hyper and multimedia information (http://www.vhml.org). 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". These take a time value in seconds or milliseconds and are defined for all elements in EML and FAML, i.e. for those parts of VHML concerned with animation. A simple VHML document looks like this:

```
<vhml>
      <person disposition="angry">
      <p>
            First I speak with an angry voice and
            look very angry,
            <surprised intensity="50">
```

```
                    but suddenly I change to look
                    more surprised.
                </surprised>
        </p>
        </person>
    </vhml>
```

Multimodal Presentation Markup Language (MPML) [84] is another XML-based markup language developed to enable the description of multimodal presentation on the WWW, based on animated characters (http://www.miv.t.u-tokyo.ac.jp/MPML/en). It offers functionalities for synchronizing media presentation (reusing parts of the Synchronized Multimedia Integration Language, 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).

Another important group of related works are behavioural modeling languages and tools for virtual agents [24,83]. Behaviour Expression Animation Toolkit (BEAT) [24] 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 a 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, it demonstrates some of the advantages of XML-based approaches together with the power of behavioural modeling.

Funge et al. [43] propose the concept of cognitive modeling for character animation. Their system is based on a set of geometric, behavioural, and cognitive models for the characters to be animated. In this approach not only the physical capabilities but also the behavioural and cognitive processes are defined and modeled. This provides the possibility of a more realistic animation, not only for appearances but also for the behaviours. A special Cognitive Modeling Language (CML) is also developed to support this system. CML does not provide any explicit support for face animation and (unlike BEAT) is not XML-based. Neither is Parameterized Action Representation (PAR) [1], another language proposed to describe and model actions of an agent based on interaction with environment and the personality and emotions. In PAR, the agent personality is defined in terms of parameters such as openness, agreeableness, and extraversion. Similar parameters are defined for other aspects affecting the behaviour.

Other scripting and behavioural modeling languages for virtual humans are considered by researchers [39,44,52,62,70,93,100]. These languages are usually simple macros to simplify the animation, or new languages that are not using existing multimedia technologies. Most of the times, they are not specifically designed for face animation. Lee et al. [62] have proposed the concept of a hierarchical presentation of facial animation but no comprehensive language for animation and modeling is proposed. The models proposed in face recognition and video analysis can also be considered with regards to content description [85,94,104].

2.2. Content Creation

2.2.1. Visual Content

Animation techniques can be grouped into the following general (and possibly overlapped) categories [80]:

- Interpolation; creating intermediate frames between given ones
- Performance-driven; tracking and following actions of a real person
- Direct parameterization; using a parameter set for head model
- Muscle-based and PsudoMuscle-based; simulating the body tissues

Creating the visual content for face animation is tightly related to the topic of object modeling, and in this case human head and face modeling. Object modeling is a fundamental issue in graphics and also vision (e.g. object recognition) for which two distinct approaches can be observed: 3D and 2D. Objects can be modeled using a 3D geometry or multiple 2D views [16,48]. Examples of 3D models are Wireframes, Constructive Solid Geometry, 3D Surfaces, and Volume Elements (Voxels). 2D object models are usually in the form of Aspect Graphs. Aspects are typical views of a certain object. Aspect Graphs represent these views and the relation between them. Other views of the object can be considered a function of these aspects (e.g. a linear combination or interpolation [90]). Figure 2-2 illustrates some of these approaches.

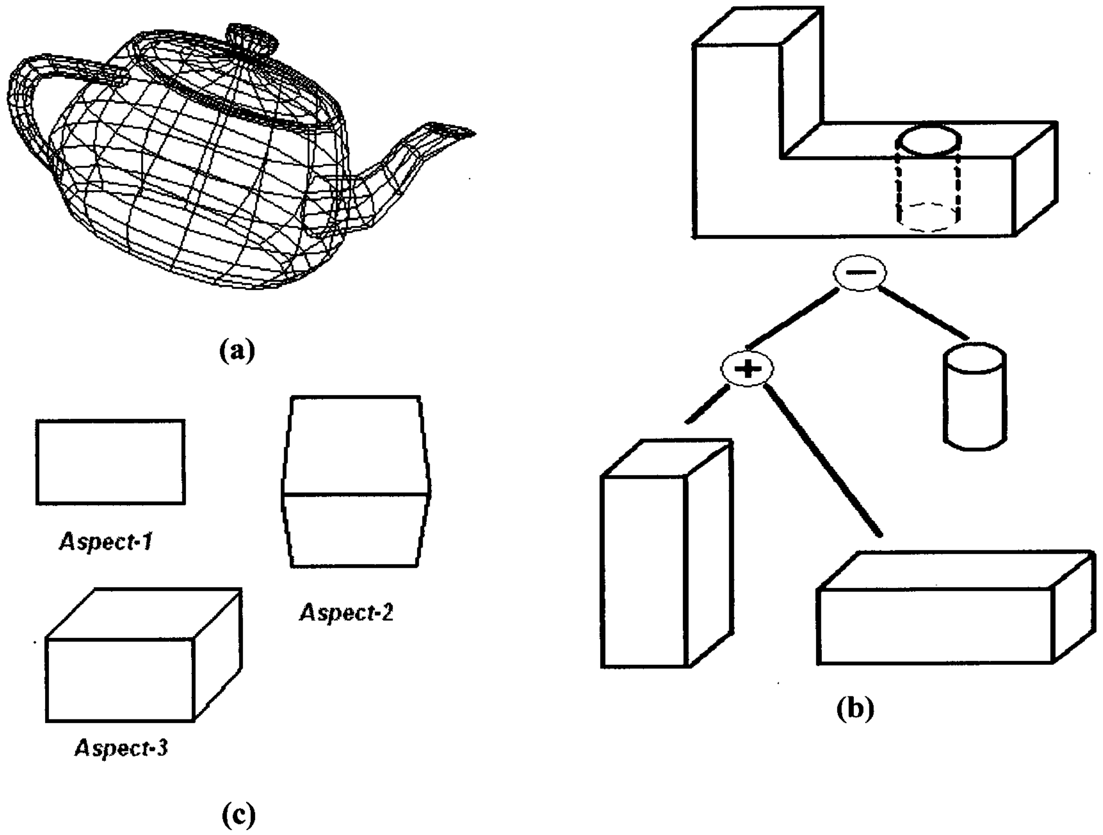
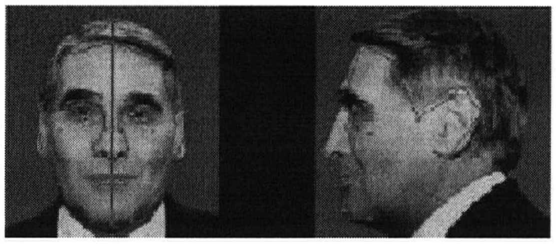


Figure 2-2. Some Approaches to Object Modeling

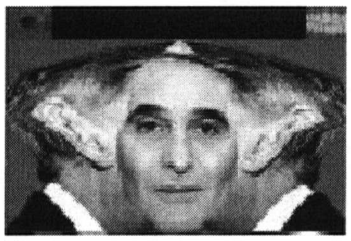
(a) Wireframe Models, (b) Constructive Solid Geometry, (c) 2D Aspect Model

3D human models have long been used for facial and character animation [17,34,35,36,40,47,51,57,58,62,78,82,99,102]. They are mainly based on volume (Constructive Solid Geometry, Voxels, Octrees) or surface (Splines and Polygons) modeling [80]. Such models provide a powerful means of body and head reconstruction in different views and situations, and have been used successfully for creating non-photo-realistic avatars and similar applications. More recent works have tried to use them for photo-realistic purposes, especially limited to facial animation rather than full body. But they usually lack the realistic appearance due to difficulty of providing 3D information for all the details (such as facial skin and hair). They may also need expensive hardware such as laser range finders, and complicated algorithms for reconstructing views (which requires more powerful hardware again). As a result, they may not be suitable for applications such as video phones where a small limited-power hardware has to perform all the processing, or cases when photo realism is a major concern.

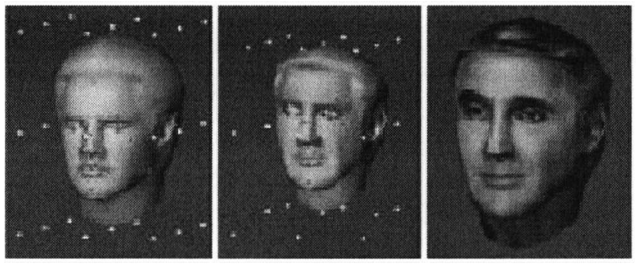
3D models of human head can be generated using 3D data or by customizing generic head models with a set of modeling parameters (e.g. MPEG-4 FDPs). The former approach uses 3D digitizers, laser range finders, and photogrammetry [80]. The latter approach results in less expensive modeling hardware but the generated models might be less realistic due to limited actual 3D information. Recent approaches have shown successful results in finding the 3D model definition parameters from 2D images, e.g. a limited number of 2D photographs [62,82]. After creating the model, computer graphics techniques can be applied to reconstruct the desired views of the 3D head. Texture mapping (again based on the 2D images) can then be used to give a more realistic appearance [62,80,82].



(a)



(b)



(c)

Figure 2-3. 3D Head Models from Photographs

(a) Orthogonal Photographs, (b) Texture Map, (c) Modified Generic Head Model. From Virtual Human Director by Lee et al. [62].

2D image-based methods are another alternative for face construction [19,27,29,41,46,49,64,91,97]. A new image can be created as a function (e.g. linear combination) of some existing ones. A video stream is a sequence of single frames. Given a source and a target frame, such a sequence can be generated by interpolating a group of intermediate frames. In traditional animation this is done by a technique referred to as Key Framing. Image Metamorphosis or Morphing [15,61,103] is the computer graphics version of key framing.

Mapping vectors are defined for source pixels that translate them to the target location, generating the effect of changing/moving images. This mapping process is called Image Warping [103] which can be performed in forward and backward ways. In Forward Warping, target locations for source pixels are found, so the output image is based on source image data (i.e. source pixels will be moved to perform the warp). Backward Warping involves finding locations in source image corresponding to pixels in target image. In this case the output image is based on the target image data [1]. View Interpolation is the process of creating intermediate images (synthetic views) by placing the pixels in the locations along the mapping vector [25,69,91,103]. Final morph is usually formed by blending forward and backward warps (e.g. a weighted average based on distance of the intermediate frame from source and destination).

Showing source and target images with I_0 and I_n, and forward and backward mapping vectors with M_f and M_b, the intermediate image I_j can be created using the following equations where w and b are warping and blending functions:

$$I_{jf} = w(I_0, \frac{j}{n} M_f) \tag{2.1}$$

$$I_{jb} = w(I_n, \frac{n-j}{n} M_b) \tag{2.2}$$

$$I_j = b(I_{jf}, I_{jb}) \cong \frac{(n-j)I_{jf} + (j)I_{jb}}{n} \tag{2.3}$$

[1] Forward and Backward Warping should not be confused with Forward and Inverse mapping done for each warp operation. In Forward Mapping, for each pixel in input image a new location is found in output image. This can result in unused locations in output ("holes") that have to be filled with interpolation. Inverse Mapping involves going through all locations (pixels) in output image and finding an input pixel corresponding to them. Inverse Mapping is used more frequently due to its computational efficiency [87].

As shown in equation 2.3, the blending of forward and backward warps is usually a straightforward operation. The critical issue in the warping/morphing procedure is to find the corresponding source and target pixels (Correspondence Problem). The correspondence problem can be solved manually by selecting a set of key points, or automatically using computer vision and feature/object detection techniques such as those applied in Stereo Vision [48], e.g. Optical Flow [14,45,104].

Manual or semi-manual methods for solving correspondence problem include identifying specific control points or lines in image pair, associating them with each other, and finding the other corresponding points based on the control points.

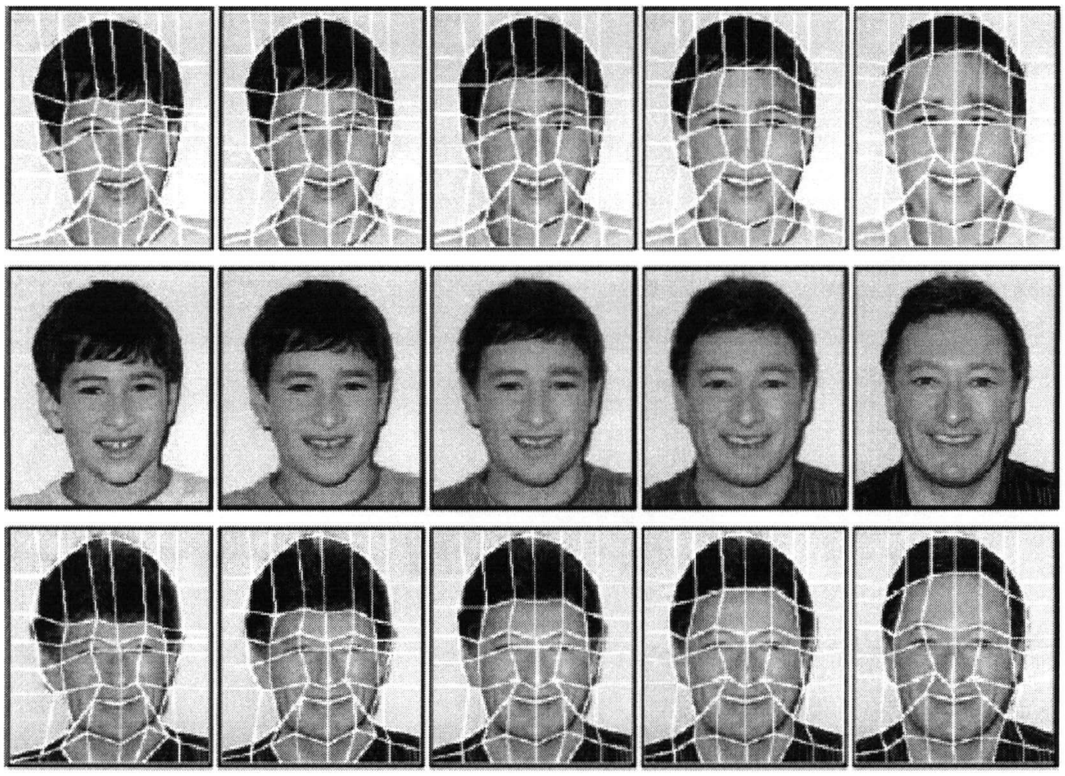


Figure 2-4. Mesh-warping for Face Animation

First and third rows are forward and backward warps applied to the source and target images, and the second row is the blend that creates the final morph [103]. Meshes are defined on the first image and moved to the target locations by the animator.

Mesh-warping is the most common method in this regard [103]. Mapping vector of each point inside a mesh is a function of mapping vectors of corner points which in turn are specified by the animator. Each mesh is defined according to some standard size and does not necessarily relate to actual regions in the image (e.g. face). Figure 2-4 shows an example of such mesh-based approach for face animation where meshes are defined roughly based on facial geometry.

Beier and Neely [15] proposed the Feature-based Morphing where the corresponding points in image pair are found based on a set of corresponding features (in their case, straight lines). The algorithm uses the distance from feature line and the projection of the point onto that line as parameters to find the corresponding point in the second image, knowing the corresponding line. This algorithm is illustrated in Figure 2-5. Although more expressive than mesh-based methods, the algorithm does not seem to be suitable for facial animation where features are not straight lines and deform as the result of facial actions. Also the original algorithm uses a global approach, finding the correspondent points based on all feature lines. In case of face animation, not all feature lines are related to every face point and some degree of localization might be necessary. A possible solution is to consider facial regions and their related features that will be more expressive and high-level than general meshes.

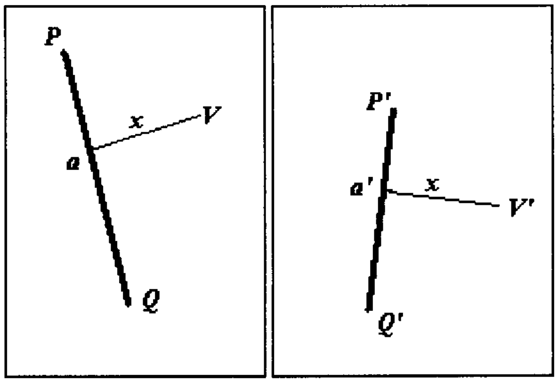


Figure 2-5. Feature-based Morphing

***PQ* and *P'Q'* are the feature line in first and second images. *V* and *V'* are the corresponding points. *a* and *a'* are projections of *V* and *V'* onto the feature lines. The distance *x* is considered the same in both images [15].**

Optical Flow (OF) was originally formulated by Horn and Schunck in the context of measuring the motion of objects in images [14]. This motion is captured as a 2D vector field $\{d_x, d_y\}$ that describes how each pixel has moved between a pair of source and destination images. From our perspective, optical flow is important because it allows for the automatic determination of correspondence between images. In addition, since each pixel is effectively a feature point, optical flow allows us to bypass the need for hand-coding any ad-hoc feature points.

Differential (gradient-based) OF methods, as used originally by Horn et al. are the most widely used optical flow techniques. They compute the velocity from spatio-temporal derivatives of image intensities. These methods, in general, are highly under-constrained and an additional set of assumptions about the underlying motion needs to be made to solve the related equations. The limited range of motion which can be detected by these methods is also another disadvantage of using them.

Correlation-based methods, used in stereo matching [42], can also be applied to find the flow field. This is done, for each pixel of the source image, by performing a correlation-based search in the corresponding neighbourhood in the target image. After finding the best match, the flow vector can be easily and precisely calculated. Though more powerful in handling large movements, these methods have a higher computational cost due to the search algorithm to find the best match. Hierarchical image representations, originally used as a compression mechanism [23], have been suggested to reduce the computational cost of differential and correlation-based optical flow computations [14,41]. Multiple-motion, boundaries of moving objects, and lighting conditions which change the intensities of points in different images are among the problems which has to be dealt with in all optical flow-based systems.

The optical flow methods (especially correlation-based ones) can be used as a basic tool to achieve an automated solution for the correspondence problem as the key issue in view morphing [15,91]. MikeTalk [41] is an image-based face animation system, which uses optical flow to solve the correspondence problem. It uses a pre-recorded library of visemes belonging to a specific person, and calculates the optical flow vectors for each pair of these images. Visual speech can then be created by the procedure shown in Figure 2-6. Figure 1-1-b illustrates some examples of the generated visemes.

The main unresolved issues in MikeTalk are:

- Limited ability in creating different facial images (e.g. moves and expressions)

- Weakness of optical flow in detecting corresponding facial features specially in movements (with large changes and new and hidden areas)
- Required image database for each person

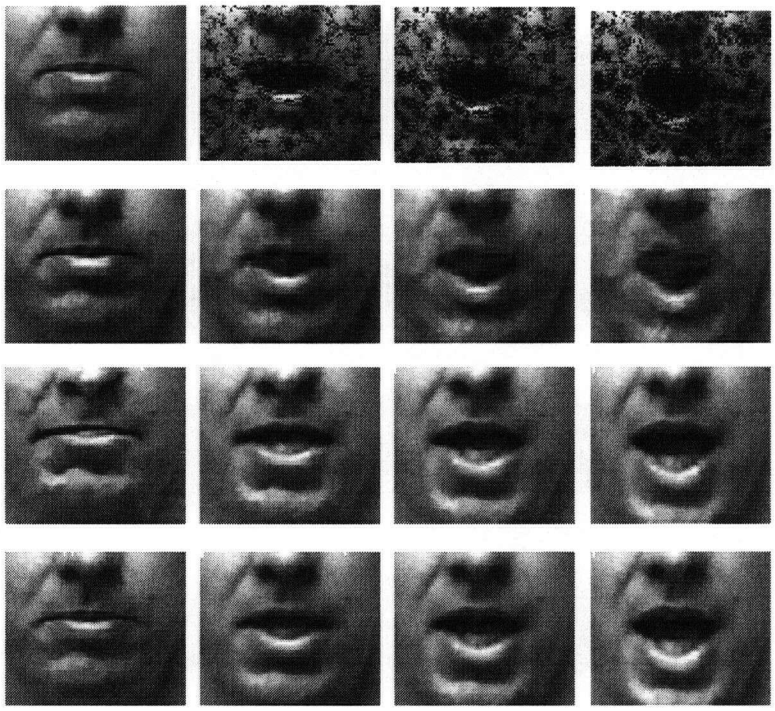
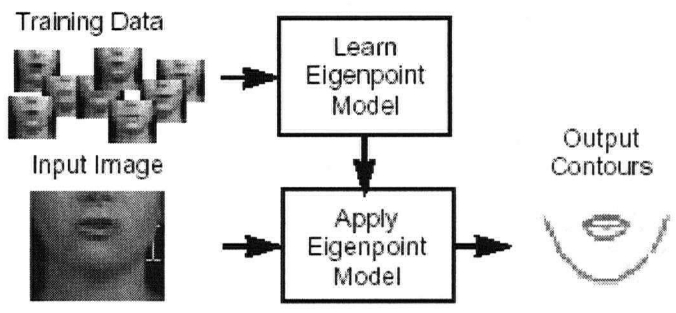


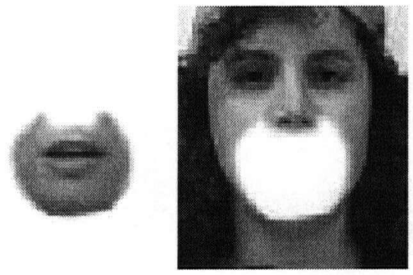
Figure 2-6. Creating Visual Speech in MikeTalk

**Row-1: Forward Warping, Viseme A (first image) towards B, without hole filling.
Row-2: Forward Warping, Viseme A (first image) towards B, with hole filling.
Row-3: Backward Warping, Viseme B (last image) towards A, with hole filling.
Row-4: A Morph between Viseme A and Viseme B.**

Bregler et al. [19] combine a new image with parts of existing footage (mouth and jaw) to create new talking views (see Figure 2-7). 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 or to create more general facial actions (e.g. head movements and expressions). In a more recent work, Graf et al. [46] 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 different views can be animated but the method requires a considerably large database. Other issues mentioned for MikeTalk still apply here, as well.



(a)



(b)

Figure 2-7. Video Rewrite

(a) Finding the Jaw-Mouth Area,
(b) Replacing the Masked Area with Speaking Image Data.

At the same time as the research described in this thesis, Tiddeman et al. [97] have used facial features to overcome some of these issues. Their objective is to apply effects such as aging and facial expressions as done by some prototype images. The system performs image transformation (warping) on a facial picture based on the changes learned from a previous pair of prototype images. Such changes govern the geometric shape and are the results of the movement of features. The system also applies a colour transformation (again learned from prototype images) to introduce the effects such as aging and wrinkles in facial expressions, as shown in Figure 1-1-a. Some normalization for shape and colour is considered, in this regard. Figure 2-8 illustrates this process where the new image is created by:

- Defining new shape, moving features by a scaled version of difference between source and destination prototypes
- Warping the image based on new shape
- Colour transformation for all pixels, again by adding the difference of prototypes

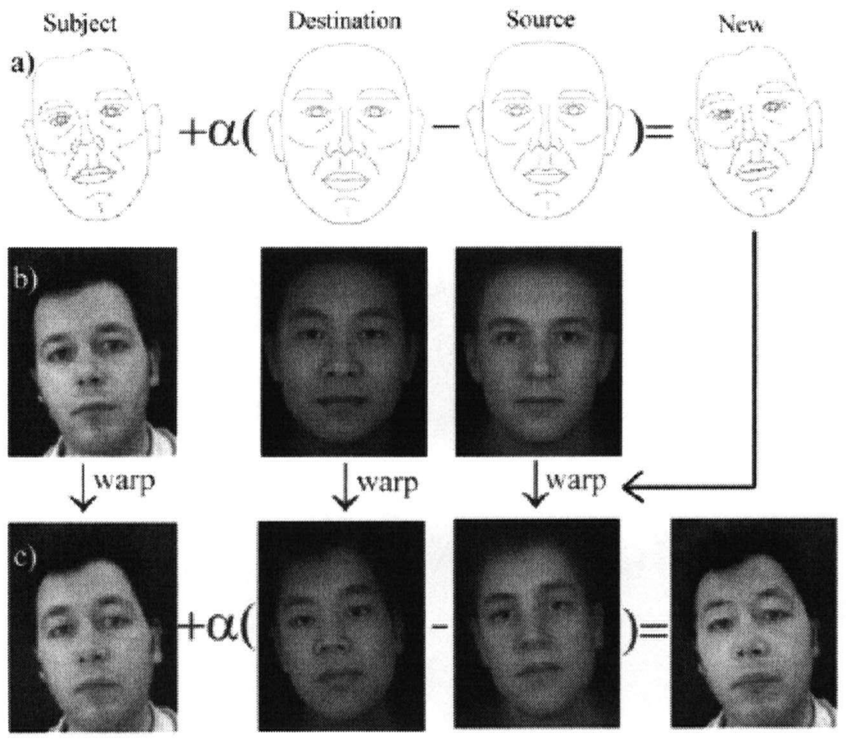


Figure 2-8. Facial Image Transformations Based On Static 2D Prototypes

The method does not address the effect of 3D head movements such as turning left or right, and can be applied only to 2D shape changes. The delta transformation between the source and destination prototypes cannot be applied to the subject by only a linear scaling when 3D head movements are present. Also, the texture transformation done by colour mapping is not very effective considering issues such as lighting and personalized skin texture. Details of the warping function are not available for this work. Knowing the mapping vectors for feature points and lines, this function determines such vector for other points.

2.2.2. Audio Content

Text-To-Speech (TTS) systems which produce an audio output based on a text input, have long been studied [37,56]. There are several algorithms for TTS conversion. The choice depends on the task for which they are used. The easiest way is to record the voice of a person speaking the desired phrases. This is useful if only a restricted set of phrases and sentences is necessary, e.g. messages in a train station or specified information via telephone. The quality depends on the way recording is done.

More sophisticated algorithms split the speech into smaller pieces. The smaller those units are, the fewer they will be in number, and the wider range of audio they can create, but the quality may also decrease due to artificial generation or concatenation of the units. An often-used unit of speech is the phoneme, the smallest linguistic unit. Depending on the language used, there can be anywhere from 30 to 60 phonemes in western European languages, i.e. 30 to 60 single recordings. The problem is combining them because fluent speech requires fluent transitions between the elements. The intelligibility is therefore lower for smaller units used, but the memory required is much smaller.

A solution to this dilemma is using diphones (two consecutive phonemes) as the unit of speech. Instead of splitting at the transitions, the cut is done at the centre of the phonemes, leaving the transitions themselves intact. Considering the possible combination of phonemes, this provides about 400 elements (20*20) and increases the quality. The longer the units become, the more elements there are, but the quality increases along with the memory required. Other units that are widely used are half-syllables, syllables, words, or combinations of them, e.g. word stems and inflectional endings. These small units are usually extracted manually from recorded sound by experts in a cumbersome process. Automated tools for speech segmentation are challenging research topics in this regard [66]. Smooth connection of the speech units [28], and their equalization and prosodic modification are also other areas of research [67].

Another group of TTS systems use models of vocal tract and synthesize the desired audio based on the phoneme list rather than using a pre-recorded sound [37]. In either case, the basic functions of phoneme/diphone systems are text normalization (separating words), word pronunciation (creating a sequence of phonemes), prosodic analysis (determining the pitch, speed, and volume of syllables), and finally audio production (synthetic or concatenative) [37, 56].

Commercial TTS products are available now as a result of extensive research in the 1980's and the 1990's. It should be noted that a personalized TTS system can be built by just changing the model parameters or audio database without any need to change the core TTS engine.

2.3. System Architecture

Different architectures are also proposed to perform facial animation, especially as an MPEG-4 decoder/player [18,79]. Although they try to use platform-independent and/or standard technologies (e.g. Java3D 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.

In a more general view, a variety of structures and standards have been proposed for multimedia streaming. Currently, the most considerable platforms available are Windows Media, Real Player, and QuickTime [59]. These technologies try to comply with the industry standards like MPEG, as much as possible. Relying on one of them, although results in some limitations, can provide the benefit of using existing system support and functionality, e.g. in multimedia streaming.

2.4. Evaluation Criteria

Any scientific or engineering system must have a clear set of objectives, precise definition for them, and measures of evaluation regarding the system objectives. Research projects, by their very nature, may not be expected to have such a precise structure, as they aim at solving unknown aspects of a final system. So the objectives of a research project can be only a subset of an "actual system" and the definitions and measures may still be vague and incomplete. But with maturity of works, it is expected and necessary that these diverse objectives and considerations converge to a well established and precisely defined set of requirements and evaluation indices corresponding to them. Digital Face Animation is now mature enough to do so [1].

Face animation systems proposed in literature have considered some objectives and requirements as their research target and have evaluated their performance, accordingly. Although acceptable for a research project, most of the time they are either not precise or incomplete for a comprehensive animation framework. Realistic appearance (Realism), real-time operation, and ability to animate a wide range of facial actions seem to be among the

[1] Although this thesis focuses on face animation but most of the discussions apply to other areas in computer animation and scene generation, as well

most widely mentioned objectives in face animation. Compatibility with existing multimedia and graphic standards, and also computational efficiency and simplicity are among other objectives noted in the literature. Few researchers have also paid attention to modeling and authoring capabilities. In most cases, precise and preferably quantitative measures are not defined for these objectives in order to evaluate the systems. In many cases, these are expressed merely as conceptual objectives rather than precise evaluation criteria, but we can consider them as a "starting-point" alternative. Needless to say introducing an objective or evaluation rule in a research work does not mean that the related requirement is satisfied by the proposed method.

Realism is the major objective in face animation systems, and has been used as main (or even the only) evaluation criterion. Examples are works by Blanz and Vetter [17], DeCarlo et al. [34], Guenter et al. [47], and Pighin et al. [82]. Ease of user interaction, ability to generate different face actions and applying them to specific pictures, and use of minimum input data are implied but not thoroughly discussed.

View Morphing and Interpolation are popular methods for creating new images based on existing ones. Seitz and Dyer [90,91] discuss the physical/geometrical validity of views in image interpolation as a basic requirement for realistic appearance. They explain and apply constraints such as visibility and user interaction to guaranty it in absence of 3D information. Graf et al. [46] and Ezzat et al. [41] also consider realistic output the main objective in their systems, and try to achieve that by using a relatively large database of images and performing view morphing between them. This implies a second objective: using simpler input types as opposed to 3D sensor data and/or 3D models. Ezzat et al. [41] have also considered lip-synchronization as an added realistic feature and try to introduce audio information to their system in order to provide a lip-synchronized talking head. In work by Graf et al. [46] head movement is considered as a new requirement and added capability.

Video Rewrite [19] is another face animation system that also uses existing images to create realistic lip-synchronized talking heads. It explicitly defines a set of criteria for evaluation, mainly:

- Lip synchronization and movement
- Proper attachment of modified lip area to the rest of the face, and other related details
- Overall quality and believability

Although these evaluation criteria are still qualitative and somehow vague but they clearly show a positive step at least for the quality evaluation.

Thalman and Thalman [96] have studied the animation of digital actors. Their main objective is again realistic output and flexible motions (ability to provide a wide range of actions), but other objectives are also mentioned, explicitly or implicitly. They are:

- Real-time interaction
- Defining behaviours
- Authoring tools (not well explained and defined)

Chung and Hahn [26] have also worked on the animation of human walking and proposed following objectives which can be considered as evaluation criteria from their point of view:

- Capability (range of actions)
- Responsiveness (real-time operation)
- Controllability (user-friendliness)
- Realism

They provide some experimental evaluation mainly for the first two objectives which are more "testable". In evaluating the system regarding the timeliness, the concept of computational complexity is also mentioned which may imply an independent evaluation criterion. In their Real-time Speech-driven system, Hong et al. [51] have also proposed the reduction of computational complexity as a major objective not only in "run-time" but also in learning phase. It should be noted that computational complexity can affect not only timeliness but also system development, maintenance, and upgrade, and has to be considered an independent factor by itself, as noted by some other researchers [102].

Lee et al. [62] have used a configurable 3D head model for face animation. Along with a powerful animation system, they have considered a set of objectives in their system, including:

- Realism
- Robustness (applicable to many faces)

- Personalization (applicable to specific people)
- Simplicity (using limited normal images as input)
- Scripting (being able to write macros for certain actions)
- MPEG-4 Compatibility

The simplicity requirement applies only to the input data not the system structure and computation. The scripting objective is not well discussed and does not seem to be a very important part of the system, but it could yield to the authoring tool considered by some others, and in general a content description method. This requirement will be discussed in detail later.

A behavioural approach to face animation has been considered by some researchers, as mentioned before. More recently, Funge et al. [43] have studied a cognitive model for character animation. This demonstrates a new objective in animation systems, i.e. being able to define and model behaviours to be used later in a programmable way. Cassell et al. [24] have proposed BEAT, a behavioural toolkit that uses an XML-based language for defining the relation between a text to be spoken and certain actions during the speech. They specifically aim at authoring and interaction purposes to ease up content description and development. Here, other than content description capability, the compatibility with existing standards, e.g. XML, is also considered.

Compatibility with existing standards and technologies (e.g. Web, XML, MPEG-4, VRML, Java3D) is also a major concern as seen in works of Lee et al. [62], Ostermann [78], and Pandzic [79]. MPEG-4 standard with its Face Definition and Animation Parameters plays a special role in this regard, and compatibility with it can be considered an important objective and so evaluation criterion.

3. Face Modeling Language

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, 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. 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 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," and finally, 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) [4,6] is a Structured Content Description mechanism based on Extensible Markup Language (XML). 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 XMT and FAPs
- Compatibility with XML and related web technologies and existing tools
- Support for different content generation (animation) methods

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 such as SMIL do this in a general-purpose form for any multimedia presentation and are not customized for specific applications such as face animation. A language bringing the best of these two together, i.e. customized for face animation, seems to be an important requirement. FML is designed to do so, filling the gap in XMT framework for a face animation language.

FML is an XML-based language. The choice of XML as the base for FML is due to its capabilities as a markup language, growing acceptance, and available system support in different platforms. Fundamental to FML is the idea of Structured Content Description. It means a hierarchical view of face animation capable of representing a range of activities, from 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 or uttering a diphone)
- Action, a "meaningful" combination of moves (e.g. a piece of speech)
- 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). These 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 3-1) 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 the 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

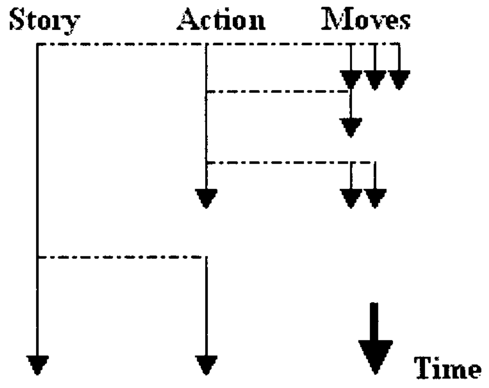


Figure 3-1. FML Timeline and Temporal Relation of Face Activities

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. Satisfying these requirements make FML a good candidate for being a part of MPEG-4 XMT framework.

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

- FML Document
- FML-compatible Player and its components (e.g. FML Processor and Animation Generator)

- Owner Application (e.g. web browser or a GUI application; may include FML playback functionality in itself without separate player object)

3.2. FML Document Structure

Figure 3-2 shows typical structure of FML documents. 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. This element groups other FML elements (model items) described in next section.

```
<fml>
    <model> <!-- Model Info -->
    <model-item />
    </model>
    <story> <!— Story TimeLine -->
         <act>
              <time-container>
                   <FML-move />
              </time-container>
         </act>
    </story>
</fml>
```

Figure 3-2. FML Document Map

Model-item, time-container, and FML-move represent parts to be replaced by actual FML elements.

A **`story`** element, on the other hand, represents the timeline of events in face animation in terms of individual Actions (FML **`act`** elements). Face animation timeline consists of facial activities and their temporal relations. These activities are themselves sets of simple Moves. These sets are grouped together within Time Containers (e.g. **`seq`** and **`par`** representing sequential and parallel move-sets).

FML supports three basic face moves: talking, expressions, and 3D head movements. Combined through time containers, they form an FML **act** which is a logically related set of activities. Details of these moves and other FML elements and constructs will be discussed in the following sections.

3.3. Modeling Elements

The model element encloses all the face modeling information. As described in FML specification [1], some important model elements are:

- **img**: An image to be used for animation; This element has two major attributes: **src** and **type**. It provides an image and tells the player where to use it. For instance the image can be a frontal or a profile picture used for creating a 3D geometric model.
- **range**: Acceptable range of head movement in a specific direction; It has two major attributes: **type** and **val**, specifying the direction and the related range value.
- **param**: Any player-specific parameter (e.g. MPEG-4 FDP); **param** has two attributes: **name** and **val** .
- **event**: external events used in decision-making; described later.
- **template**: defines a set of parameterized activities to be recalled inside story using **behaviour** element.
- **character**: The person to be displayed in the animation; This element has one major attribute name and is used to initialize the animation player database.
- **sound**: The sound data to be used in animation; This element also has a **src** attribute that points to a player-dependent audio data file/directory.

Figure 3-3 shows a sample model module. FML templates will be extended in later versions to include advanced behavioural modeling.

[1] http://www.ece.ubc.ca/~alia/Multimedia/fml_1.html

```
<model>
    <img src="me.jpg" type="front" />
    <range type="left" val="60" />
    <template name="hi" >
        <seq begin="0">
            <talk>Hello</talk>
            <hdmv type="0" begin="0" end="3s" val="30" />
        </seq>
    </template>
</model>
<story>
    <behaviour name="hi" />
</story>
```

Figure 3-3. FML Model and Templates

3.4. Story-related Language Constructs

3.4.1. FML Time Containers and Moves

FML timeline, presented in Stories, consists primarily of Acts which are purposeful set of Moves. The Acts are performed sequentially but may contain parallel Moves within themselves. Time Containers are FML elements that represent the temporal relation between moves. The basic Time Containers are **seq** and **par**, corresponding to sequential and parallel activities. The former contains moves that begin at the same time and the latter contains moves that start one after another. The Time Containers include primitive moves and also other Time Containers in a nested way. The **repeat** attribute of Time Container elements allows iteration in FML documents. They also have three other attributes **begin**, **duration**, and **end** (default value for **begin** is zero, and **duration** is an alternative to **end**) that specify the related times in milliseconds.

FML also has a third type of Time Containers, **excl**, used for implementing exclusive activities and decision-making as discussed later.

FML version 1.0 supports three types of primitive moves plus MPEG-4 FAPs:

- **talk** is a non-empty XML element and its content is the text to be spoken.
- **expr** specifies facial expressions with attributes **type** and **val**. The expression types can be smile, anger, surprise, sadness, fear, and normal.
- **hdmv** handles 3D head movements. Similar to **expr**, this move is an empty element and has the same attributes.
- **fap** inserts an MPEG-4 FAP into the document. It is also an empty element with attributes **type** and **val**.
- **nop** performs no operation. It is an empty element with only timing attributes.

All primitive moves have three timing attributes **begin**, **duration**, and **end**. In a sequential time container, **begin** is relative to start time of the previous move, and in a parallel container it is relative to the start time of the container. In case of a conflict, duration of moves is set according to their own settings rather than the container. Figure 3-4 illustrates the use of time containers and primitive moves.

```
<act>
    <seq begin="0">
         <talk>Hello</talk>
         <hdmv end="5s" type="0" val="30" />
    </seq>
    <par begin="0">
         <talk>Hello</talk>
         <expr end="3s" type="3" val="50" />
    </par>
</act>
```

Figure 3-4. FML Time Containers and Primitive Moves

3.4.2. Iterations in FML

Iteration in FML is provided by **`repeat`** attribute of Time Container elements that simply cycles through the content for the specified number of times (in Definite Loops) or until a certain condition is satisfied (Indefinite Loops). For a Definite Loop, repeat is either a number or the name of an external event with a numeric non-negative value.

Indefinite Loops are formed when the **`repeat`** attribute has a negative value. In such cases, the iteration continues until the value becomes non-negative. After that, the loop continues like a Definite Loop. Of course, setting the value to zero will stop the iteration. Figure 3-5 shows an example of FML iteration.

```
<event name="select" val="-1" />
< ... >
<act repeat="select">
    <seq>
        <talk begin="1">Come In</talk>
        < ... >
    </seq>
</act>
```

Figure 3-5. FML Iteration

3.5. *Event Handling and Decision Making*

Dynamic interactive applications require the FML document 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 **`excl`** time container associates with an event and allows waiting until the event has one of the given values, then it continues with exclusive execution of the action corresponding to that value, as illustrated in Figure 3-6.

The FML Processor exposes proper interface function to allow event values to be set in run time. **`event`** is the FML counterpart of familiar `if-else` constructs in normal programming languages.

```
<!-- in model part -->
<event name="user" val="-1" />
<!-- in story part -->
<excl ev_name="user">
    <talk ev_val="0">Hello</talk>
    <talk ev_val="1">Bye</talk>
</excl>
```

Figure 3-6. FML Decision Making and Event Handling

3.6. 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 systems, as illustrated in 3.7.2.

Another major aspect of FML is its compatibility with MPEG-4 XMT framework and face definition/animation parameters. This has been achieved by using XML as the base for FML and also sharing language concepts with SMIL. As the result, FML fits properly within the XMT framework. FML documents can work as an XMT-Ω code and be compiled to MPEG-4 native features, i.e. FDPs and FAPs.

FML is a high-level abstraction on top of MPEG-4 Face Animation Parameters. FAPs can be grouped into the following categories:

- Visemes
- Expressions
- Head Movements
- Low-level Facial Movements

In FML, visemes are handled implicitly through **`talk`** element. The FML processor translates the input text to a set of phonemes and visemes compatible with those defined in MPEG-4 standard. A typical Text-To-Speech engine can find the phonemes and using the tables defined in MPEG-4 standard, these phonemes will be mapped to visemes. FML facial expressions are defined in direct correspondence to those in MPEG FAPs. FML also provides **`hdmv`** element corresponding to FACS/FAPs basic 3D head movements. For other face animation parameters, the **`fap`** element is used. This element works like other FML moves, and its **`type`** and **`val`** attributes are compatible with FAP numbers and values. As a result, FML processor can translate an FML document to a set of MPEG-4 compatible movements to be animated by the player components.

Unlike MPEG-4 FAPs, in FML version 1.0, FDPs are not explicitly supported. This support is provided indirectly through **`param`** model element that can define any modeling parameter including FDPs. FML version 1.0 has been designed as part of the *ShowFace* system that uses 2D photos for animation and does not need FDPs. As a result, inclusion of a better support for FDPs was left to later versions of FML.

3.7. CASE STUDIES

FML can be used in a variety of ways and applications. It can be used as a high-level authoring tool within XMT framework to create MPEG-4 streams (after translation), or be used directly by compatible players for static or interactive face animation scenarios. Here we discuss three sample cases to illustrate the use of FML documents.

3.7.1. 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 iterations, using **`repeat`** attribute. An example of this case can be animating the image of a person who is not available for real recording. The **`img`** element specifies the frontal (base) view of the character and the story is a simple one: saying hello then smiling (Figure 3-7).

To add a limited dynamic behaviour, the image, text to be spoken, and the iteration count can be set by the user, and then a simple program (e.g. a script on a web page) can create the FML document. This document will then be sent to an FML-compatible player to generate and show the animation. *ShowFace* system is an example of such a player, discussed in Chapter 5.

```
<fml>
    <model>
        <img src="ali.jpg" type="frontal"/>
    </model>
    <srory>
        <act>
        <seq repeat="2">
          <talk begin="0">Hello</talk>
          <expr begin="0" end="2s" type="smile" val="80" />
          <expr begin="0" end="1s" type="normal" />
        </seq>
        </act>
    </story>
</fml>
```

Figure 3-7. Static Iterative FML Document

3.7.2. Event Handling

The second case shows how to define an external event, wait for a change in its value, and then perform certain activities based on that value (i.e. event handling and decision making). 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.

A good example of this case can be a virtual agent answering users' questions online. Depending on the selected question (assuming a fixed set of questions), a value is set for the external event and the agent speaks the related answer. The FML document in Figure 3-8 uses two events: one governs the indefinite loop to process the user inputs, and the second selects the proper action (reply to user question in mentioned example).

The FML-compatible player reads the input, initializes the animation (by showing the character in initial state), and when it reaches the decision point, waits for user input because the `select` event does not match any of the values inside **`excl`**. After the event is set through proper API (see Chapter 5), the related action is performed. This will continue until `quit` event used by repeat is set to a non-negative value. If the value is zero, it stops, otherwise continues for defined number of times.

```
<event name="quit" val="-1" />
<event name="select" val="-1" />
< ... >
<act repeat="quit">
    <excl ev_name="select">
         <seq ev_val="0">
              <talk>Text One</talk>
              <expr type="smile" val="100" end="2s" />
         </seq>
         <seq ev_val="1">
              <talk> Text Two</talk>
              <expr type="smile" val="100" end="1s" />
         </seq>
    </excl>
</act>
```

Figure 3-8. Events and Decision Making in FML

```
function onAdd()
{
    //fmldoc is FML (XML) document
    //loaded at startup

    //get the root (fml) element
    var fml =
         fmldoc.documentElement;

    //find the proper element by
    //navigating through fml node
    //details not shown
    var fmlnode;
    . . .

    //create/add a new element
    var new =
    fmldoc.createElement("hdmv");
    new.setAttribute("type","0");
    new.setAttribute("val","30");
    fmlnode.appendChild(new);
}
```

Figure 3-9. JavaScript Code for FML Document Modification

3.7.3. Dynamic Content Generation

The last FML case to be presented illustrates the use of XML Document Object Model (DOM) to dynamically modify the FML document and generate new animation activities. Example can be adding a new choice of action to the previous example, dynamically.

Figure 3-9 shows a sample JavaScript code that accesses an XML document, finds a particular node, and adds a new child to it. Since this case uses standard XML DOM features, we do not discuss the details. It only shows how the XML base can be helpful in FML documents. More information on XML DOM can be found at many online references such as http://www.xml.org/xml/resources_focus_dom.shtml.

The same DOM methods can be accessed from within the FML Player to modify the document while/before playing it. The FML Player can expose proper interface functions to provide these capabilities to users/applications in an easier way. In Chapter 5, the *ShowFace* system and the *ShowFacePlayer* component are introduced as examples of such user-friendly environments for working with FML documents.

4. Feature-based Image Transformations

4.1. Background: Optical Flow-based Approach

4.1.1. Structural Overview

View Morphing has been a useful technique for generating new images based on some existing ones. One of the main issues in morphing, as mentioned in Chapter 2, is finding the corresponding points that work as controls to find out the mapping values for other pixels in image [15,103]. Algorithms like optical flow have been used to automate the detection of mapping vectors with or without such control points [41]. Optical flow method (especially correlation-based) [14] can provide mapping vectors for all image pixels so animation will be possible without any need for control points. This minimizes the required interaction by the animator and makes the method suitable for any type of image (not just facial).

In the first stages of this research, a correlation-based optical flow algorithm was used to create a moving/talking head [8,9]. The basic structure of the proposed system is illustrated in Figure 4-1. The procedure is applied to a pair of given images (e.g. two visemes) in order to create a video stream of head changing from one state (represented by the first image) to another (represented by the second image).

Considering the case of moving/talking head, image pixels can have different 2D movements. So no unique epi-polar line [1] can be specified for the whole image pair. The key concept of this approach is to solve the correspondence problem by performing a 2D correlation-based search in a 2D neighbourhood around each pixel.

Due to huge amount of computation required for the 2D search, a hierarchical mechanism based on image pyramids is utilized [23]. The input image is resampled at different resolutions to make a pyramid. The search is started at lower resolutions and the result of each level is used as the initial guess for the next higher level.

[1] A line along that all movements happen.

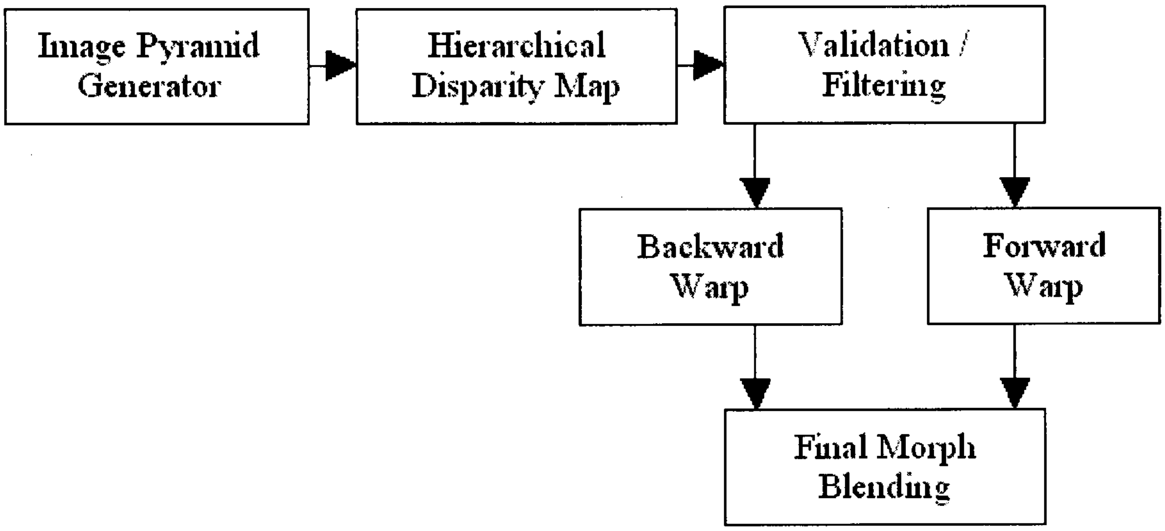


Figure 4-1. Moving/Talking Head with Correlation-based Optical Flow

The result of the correlation-based search is a pair of disparity maps, corresponding to forward and backward warps. After applying some validity checks and filtering, these maps are used to perform the forward and backward warps to create the required number of intermediate frames in both directions. The intermediate frames are generated by incrementally moving the source image points along the optical flow vector (2D disparity). The final morph is then made by blending these two sets of frames to do the hole-filling [1] and create a smooth transition.

4.1.2. Optical Flow Field

Optical flow field is a set of 2D vectors corresponding to each pixel in the source image which determines its motion. In our approach, this field is the disparity map resulted from solving the correspondence problem by a correlation-based search. In stereovision cases, disparity values are found by searching along an epi-polar line. This line specifies the relative movement of camera/object which is not necessarily vertical or horizontal but is unique for all image points.

Non-rigid and multi-component motions prevent us from using the concept of a unique epi-polar line. In case of a moving/talking head, for instance, the mouth area has a multi-

[1] Forward Mapping has been used.

component movement which is different from other parts of the head. To allow such movements, the proposed approach does not use any epi-polar line. A similarity score is computed for every point (i,j) in image with respect to all points of the other image belonging to a 2D neighbourhood centred at (i, j), and a disparity range parameter. The similarity score, c, is calculated based on the correlation scheme used by Fua [42]:

$$s = \max(0, 1 - c)$$
$$d_1 = (I_1(x+i, y+j) - I_{1a})$$
$$d_2 = (I_2(x+dx+i, y+dy+j) - I_{2a})$$

$$c = \frac{\sum_{i,j}(d_1 - d_2)^2}{\sqrt{\sum_{i,j} d_1^2 \sum_{i,j} d_2^2}} \tag{4.1}$$

where I_1 and I_2 are the image pair and I_{1a} and I_{2a} are their average value over the correlation window. *dx* and *dy* are displacement values in two dimensions.

As we can see, for a given 200x200 image and a maximum disparity of 20 in each direction (search window of size 41x41), the total number of correlation calculations needed to be done is approximately 200x200x41x41 = 67,240,000 (the searchable window shrinks when approaching the image boundaries). Having a reasonable initial guess removes the need for searching the entire 41x41 window for the best match. Image pyramids were originally introduced as image compression tools [23]. They have been used in optical flow computations both to reduce the computational cost and more importantly to allow detection of larger motions which do not satisfy small motion constraints of some optical flow methods [14]. An image pyramid consists of different levels of coarse to fine resolutions made by resampling the input image. As an example, if maximum disparity is 20, corresponding to the maximum amount of motion, a lower resolution image resampled at a 4:1 rate needs to be searched only for a disparity range of 5.

Although more advanced and complicated mechanisms for creating the image pyramid exist (e.g. Gaussian and Laplacian), in this work an image window is simply replaced by its average to make a lower resolution image. The pyramid has three levels: low, medium, and high resolutions, resampled at 4:1, 2:1, and 1:1 rates. Total amount of correlation operations required at each level is:

• Low Level:	302,500
• Medium Level:	250,000
• High Level:	1,000,000

This results in a total number of 1,552,500 operations that is almost 30 times less than non-hierarchical approach.

The calculated disparity vector at each level gives the centre point of matching search for the next level. For a disparity range of 20, the lowest level uses a range of 5 and others only 2. The initial value of disparity vectors for the low resolution level is zero.

4.1.3. Map Validation and Filtering

The correlation-based matching is not guaranteed to find the real corresponding point. It is always necessary to perform a validity check on the results. Two types of validation are used in our system: uni-directional and bi-directional. Uni-directional validation consists of two steps:

- Limit Check: The mapping should not change the relative positioning of points. If a point is inside a group of others in one image it cannot be outside the same group in other image. Working with the disparity maps, this means that the value of x or y component of flow vector of each point must be limited to corresponding values of its neighbours.
- Filtering: To remove false matches (or reduce their effects) a standard Median or Low-Pass filter is applied to maps.

Bi-directional validation is performed by comparing the disparity vector of each pixel with the disparity vector of its corresponding point in the other image. Ideally, these two vectors should sum up to zero, but a small amount of error (one or two pixels) can be ignored. Unaccepted disparity vectors will be replaced by interpolation of their valid neighbours. Due to performance results, the bi-directional validity check was implemented but not used in this project.

4.1.4. Image Warping and Final Morph

Given the input images and the calculated disparity maps (optical flow fields), the N intermediate frames can be generated using the following mapping:

$$I_1(i,j) \Rightarrow I_{12,k}(i+d_x, j+d_y)$$

$$d_x = \frac{k}{N-1} D_x(i,j) \tag{4.2}$$

$$d_y = \frac{k}{N-1} D_y(i,j) \tag{4.3}$$

where I_1 and $I_{12,k}$ are source and k^{th} intermediate frames, and D_x and D_y are components of disparity vector.

Based on the way optical flow fields are calculated, a forward mapping is used in this approach that needs final interpolation. Since the target image is available, better hole-filling can be achieved if we initialize all the intermediate frames with the target image before applying the above map. The same mapping will be applied to both input images to create two sets of N frames. The final morphed k^{th} frame can be computed as follows:

$$I_{k,m}(i,j) = \frac{(N-k)I_{12,k}(i,j) + kI_{21,k}(i,j)}{N} \tag{4.4}$$

where k is the frame index in forward direction and $I_{k,m}$ is the k^{th} frame of the final morph.

4.1.5. Experimental Results

The proposed approach is implemented as a Windows application with the capability of reading an input script which contains information about the input images and required settings. It then applies the described algorithm to create the final morph and play them as a video stream. Figure 4-2 shows a pair of input face images (a and f), four intermediate (b to e), another input pair (g and j) and two intermediate frames (h and i). We can see that for small movements like those in visemes (g to j), the optical flow works reasonably well but in larger movements, illustrated by the first group, the mismatch in optical flow-based algorithm causes noise-like pixels. Another major disadvantage of this method is the need for all input images and their mapping database which makes the method hard to personalize.

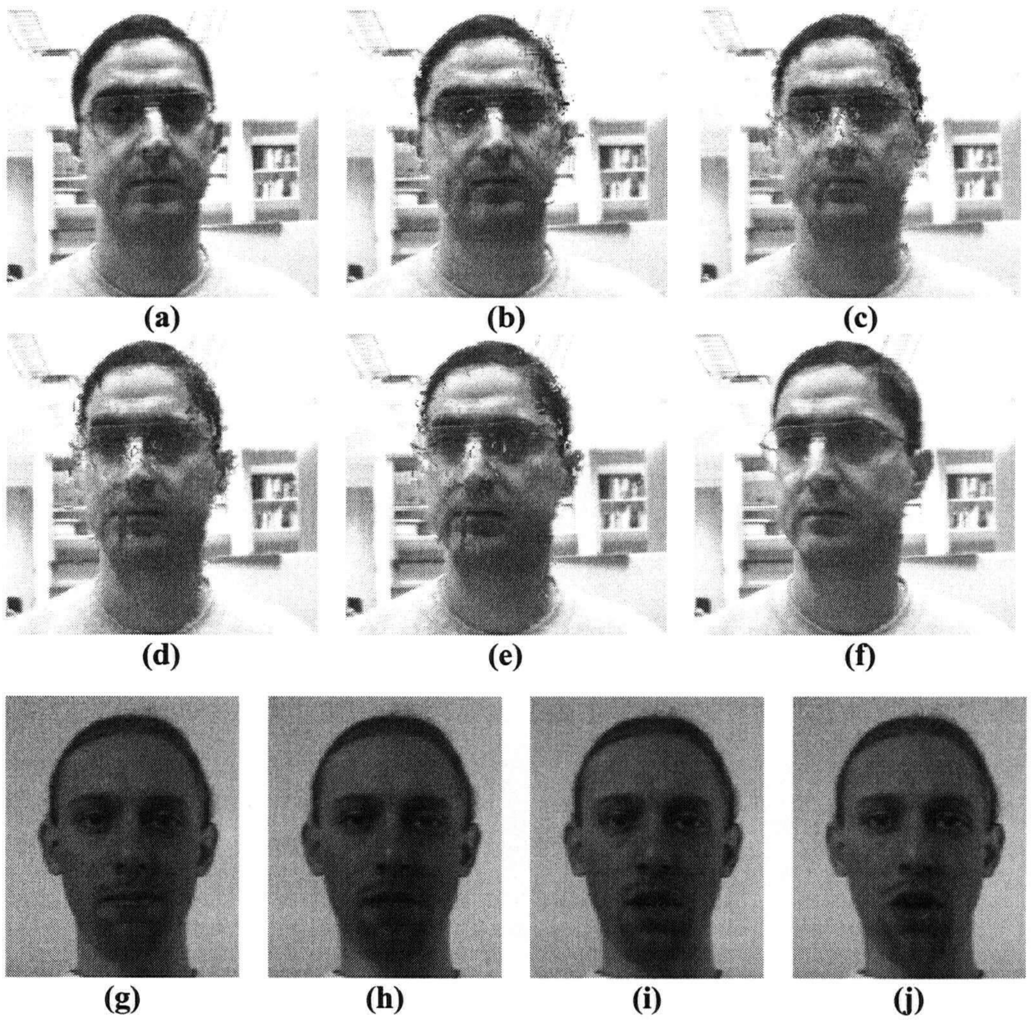


Figure 4-2. Sample Results of Optical Flow-based Approach

Morphing from (a) to (f), and (g) to (j)

The moving/talking head based on this approach uses two groups of input images:

- Visemes
- Head turning left and right (profile and two intermediate positions, e.g. Figure 4-2-f)

All the mapping vectors between each image pair of each group is created and stored. This results in a relatively large database that works only for one person. The visemes used in this system (and also in feature-based approach discussed later) are shown in Table 4-1.

For audio, pre-recorded diphones and a TTS engine are used. The engine is used only to convert the input text to a phoneme array. More details on the audio processing can be found in Chapter 5.

Table 4-1. Visemes List

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

4.2. *Feature-based Approach: Basic Concepts*

The rest of this chapter is devoted to the discussion of Feature-based Image Transformation (FIX) as the proposed alternative to optical flow-based method described in Section 4.1. As far as the content generation is concerned, the main objective of this research has been the development of methods that allow photo-realism and yet minimize the need for modeling data and also algorithmic complexity. To avoid the lack of realism and complexity of algorithms, modeling data, and possibly hardware (as explained in Chapter 2), 3D approaches have not been used. As discussed in 4.1, the research started with using view morphing as the main technique. The optical flow-based approach has several advantages including automated nature (no need to manually specify control points) and reasonable quality for limited movements. But it suffers from some major drawbacks:

- Large amount of data (Each facial action has to be stored as a mapping vector for each pixel)
- Lack of personalization (The mapping vectors are defined for a specific face and cannot be applied to another one)

- Optical flow mismatch (The matching error shows itself as noise-like pixels and will be more noticeable in larger movements)

It should be noted that OF-based approach is a "blind" method, i.e. there is no information regarding the content of image. This is beneficial for automation purposes but makes it impossible to perform scaling and similar operations required for applying the mapping to another face. Based on this idea, a primary modification to OF-based approach can be introduction of facial feaures which can help:

- Apply the mapping to a new face (rotation, scaling, etc)
- Enhance the correlation search by providing initial values or extra constraints

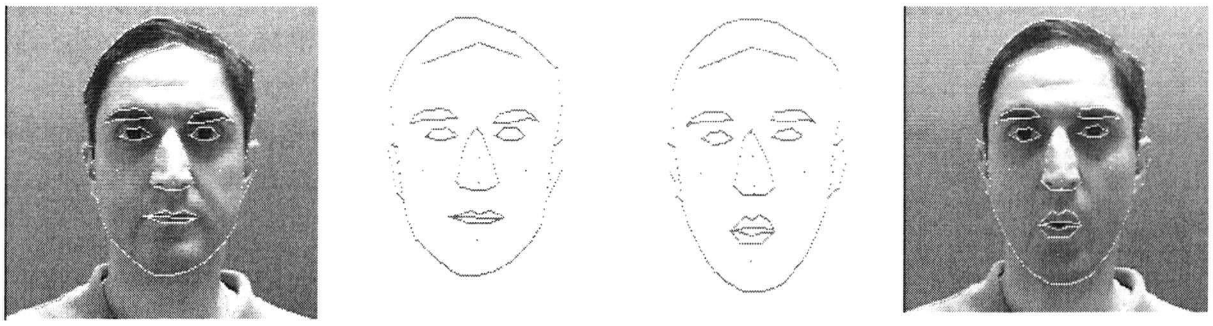
On the other hand, as experienced by FACS and MPEG-4 FAPs, knowing the movements of some important feature points and lines can be enough for generating new images (such as talking and facial expressions) at some satisfactory level of quality. This means that instead of storing mapping information for 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 the 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 reducing 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 T_{12} as mapping vectors that take each point in I_1 to its best match in I_2:

$$I_2 = T_{12}(I_1) \tag{4.5}$$

Here T_{12} has to be stored with all the mapping vectors for each pixel and due to its "blind" nature, cannot 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 F_i for each image I_i. The translation function will now be applied to these new data structures:

$$F_2 = T_{f,12}(F_1) \tag{4.6}$$

This idea is illustrated in Figure 4-3. Knowing the mapping vectors for feature points (a) and (b), first the feature points are mapped (c), and then the other points are mapped by interpolating their movements as function of the movement of feature points (d). This is basically a warping operation, as introduced in Chapter 2. The warping function creates the new image based on the pixel values of input image and has the limitation of not adding new information like change of texture. Wrinkles are good examples of such new information that cannot be handled by pure shape warping. In most of facial actions (including talking and head movements) texture/colour changes are negligible. But in some cases like facial expressions, they are more important. A secondary colour transformation can be added to further enhance the transformed image (as discussed later).



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

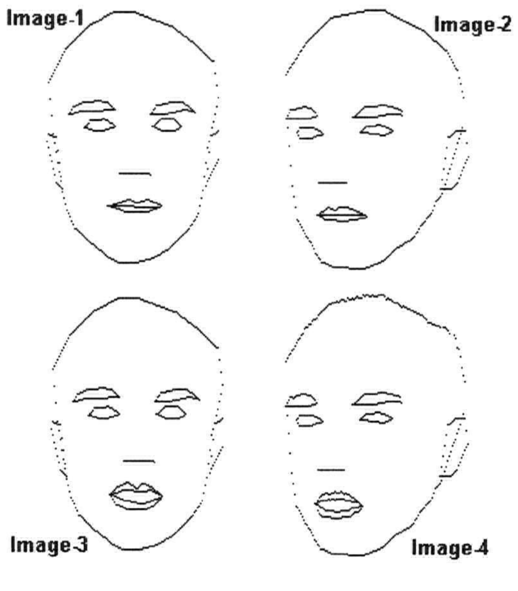
Figure 4-3. Feature-based View Generation

(a) Base Image with Features Detected, (b) Features of Base Image, (c) Features after Applying the Known Mapping, (d) Warped Image.

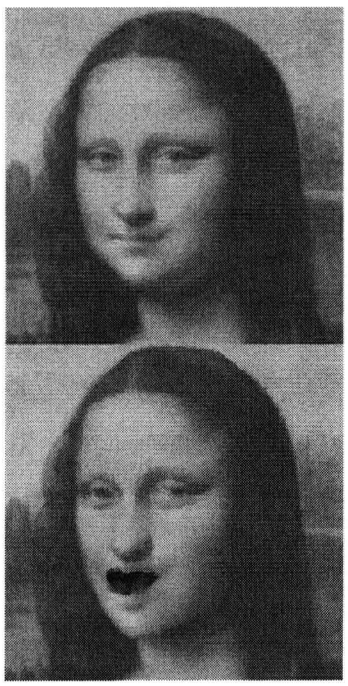
With availability of geometrical information of the face, the Feature Translation Function (FTF), T_f, can now be processed to handle normalization (scaling, rotation, and even change of orientation) and it needs less data to be stored. With proper normalization, the translation functions can be used for new characters or new images of the same character. This idea is illustrated in Figure 4-4. Within the context of FIX, the terms (1) *model* image, (2) *base* image, and (3) *normalization* are used respectively to refer to (1) images used to learn the FTFs, (2) images that FTFs are applied to, and (3) processes like scaling needed to make FTFs applicable to source images. *Model* and *base* images may also be called *prototype* and *input*, respectively.

To summarize, the Feature-based Image Transformation for Face Animation (FIX) consists of:

- Learning Feature Translation Function (FTF) between different facial states, using a set of model images [1]
- Detecting the facial features for a given base image
- Applying the translation function to the feature points of the base image
- Proper interpolation to find mapping vectors for non-feature points
- Optional colour/texture transformation
- View morphing to generate any number of intermediate images
- Filling newly appeared regions of face (after head movements) for each frame



(a)



(b)

Figure 4-4. Feature-based Image Transformation

(a) The FTF learned between Images 1 and 3 is applied to features of Image 2 to create a feature set for Image 4.
(b) Pre-learned "talking" FTF is applied to a new base image.

[1] For the sake of computational simplicity, co-articulation, i.e. the effect of neighbouring phonemes and visemes on each other, is not considered in this work.

4.3. *Facial States and Features*

4.3.1. Identifying Facial States and Transitions

Facial activities are transitions between certain face "states" such as 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 a model person is performing the related transitions. A library of these functions is created based on following groups of facial states:

- Visemes in frontal view (Table 4-1)
- Facial expressions in frontal view (Figure 4-5)
- Head movements including 2D rotation to left and right, tilting forward and backward, and turning left and right that poses more problems due to hidden and new areas and is the focus of our work (Figure 4-6)

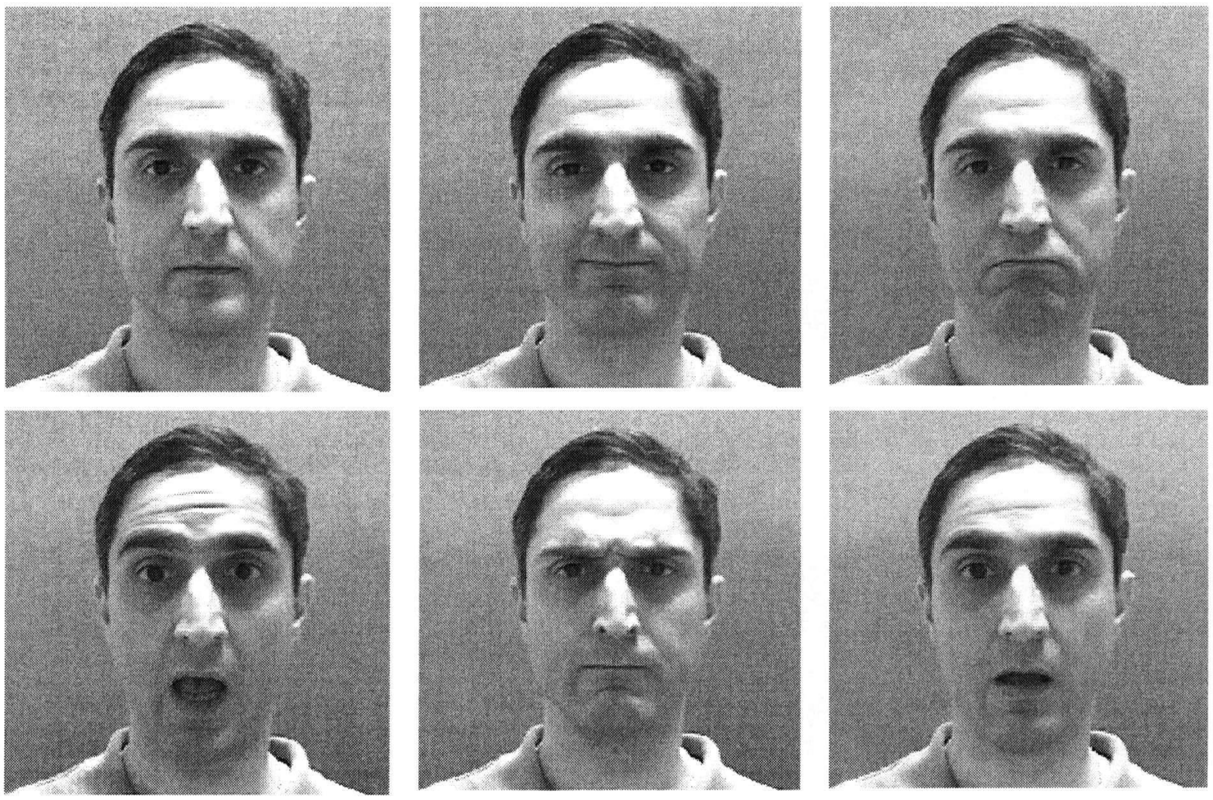


Figure 4-5. Facial Expressions

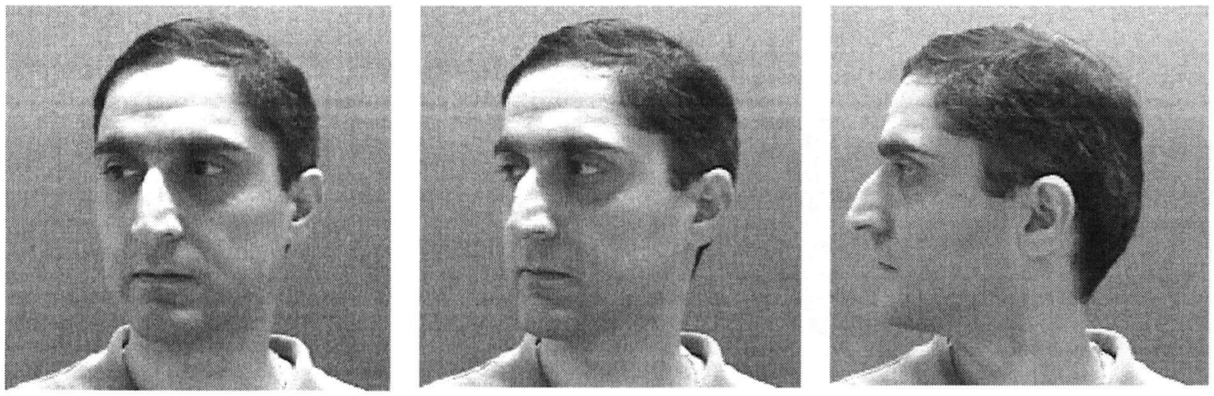


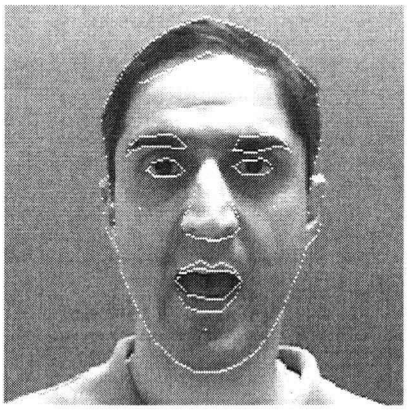
Figure 4-6. 3D Head Movements

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 including frontal view).

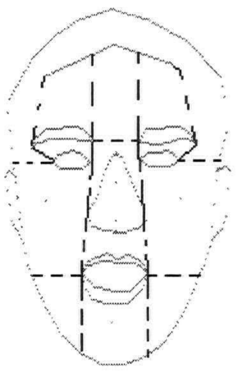
4.3.2. Detecting Facial Features

Each transformation is associated with one facial state transition with a source and destination image/state. It is defined in the 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 further normalization (scaling and calibration, explained later). The feature set for each image includes face boundary, hair line, eyes and eye-brows, nose, ears, lips, and some extra points as defined in MPEG-4 FDPs. These feature lines, and the facial regions created by them are shown in Figure 4-7.

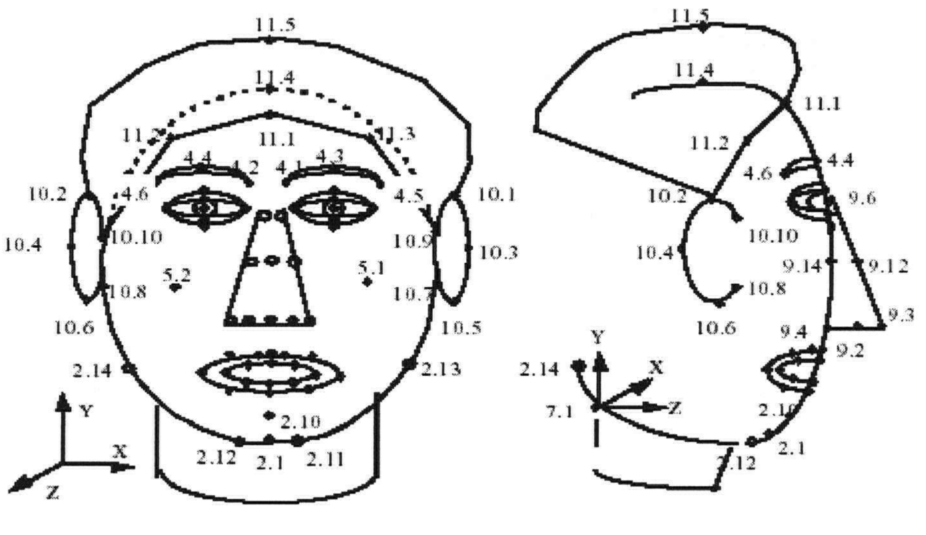
Off-line processing of facial features and their translation functions are done by a specifically designed tool, *ShowFaceStudio*. Facial features in model and base images are detected in a semi-automated way. Special points corresponding to major FAPs, are marked on the face for the model images. For base images, these points will be specified manually. Additional points will also be specified on the feature lines (Figure 4-8.b). The entire feature line is then determined by the software applying a low pass filter for smoothing purposes. The concept is similar to Snake [54] algorithm that tries to detect a contour by staying on the edge of an image area while maximizing the smoothness. In our case, a poly-line is formed based on the specified lines and then the low pass filter is used to smoothen the line. Due to offline nature of this operation and the fact that it is done only once for a limited set of images, manually specifying some control points is more effective than normal edge detection methods used in standard Snake algorithm. Figure 4-8 shows this process for upper lip line. For the sake of visual simplicity, it does not show the marks on the face except in (b).



(a)



(b)



(c)

Figure 4-7. Facial Features

(a) Image with Features, (b) Features with Facial Regions and Patches, (c) FDPs.

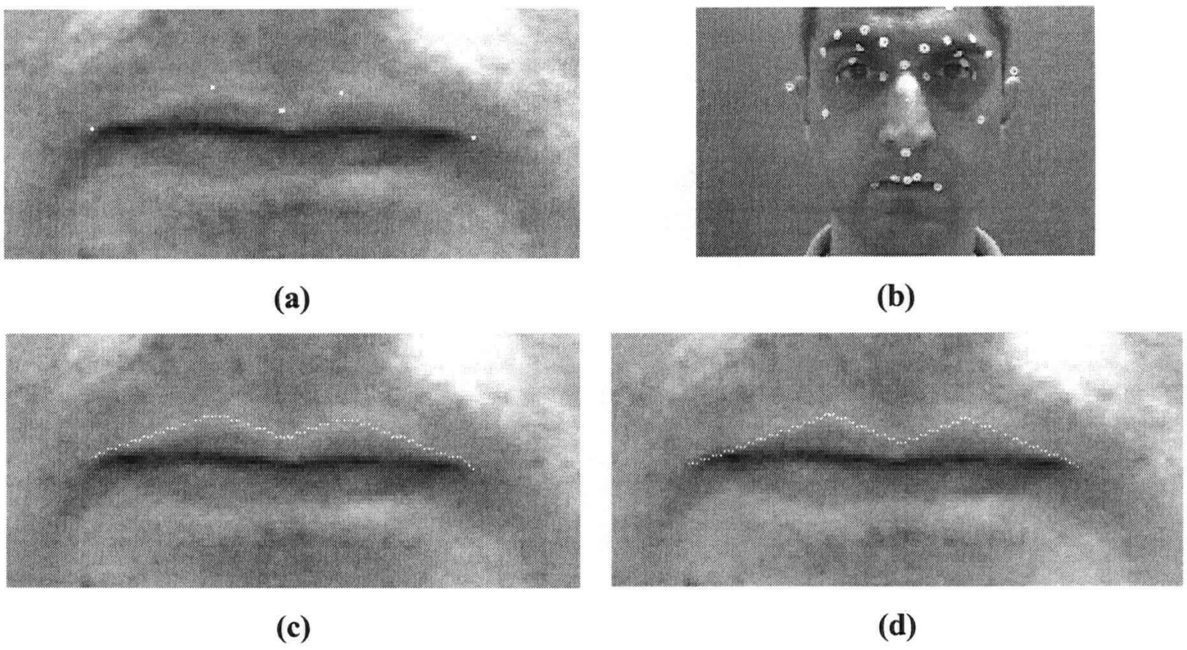


Figure 4-8. Feature Detection.

(a) Lip Points specified, (b) Marked Feature Points for Precision, (c) Feature Line Detected and Filtered, (d) Lip Line without Filtering.

Table 4-2 lists the facial features identified and used in FIX. The features are mostly lines, but some additional points are added to improve the warp function, as discussed later. Also, the lines are divided into segments based on important points on them (e.g. the vertical maximum for eyes and brows). These control points and segments help find the corresponding points on feature lines when calculating the FTFs. The control points are found automatically using their geometric properties. Associated with each image, there is a `FaceFeature` structure including these lines, points, and segments information [1].

[1] Actual number of points in each feature line can be different depending on the image.

Table 4-2. Feature Lines and Points (FaceFeature Structure)

Feature	**Type**	**Segments**	**Description**
Head	Line	4	Around head
Hair	Line	2	Hairline Above Forehead
LeftBrowTop	Line	2	
LeftBrowBottom	Line	2	
LeftEyeTop	Line	2	
LeftEyeBottom	Line	2	
RightBrowTop	Line	2	
RightBrowBottom	Line	2	
RightEyeTop	Line	2	
RightEyeBottom	Line	2	
Nose	Line	4	
HighLipTop	Line	3	
HighLipBottom	Line	3	
LowLipTop	Line	3	
LowLipBottom	Line	3	
LeftEar	Line	1	
RightEar	Line	1	
LeftForehead	Point		FDP 11.2
RightForehead	Point		FDP 11.3
LeftCheek	Point		FDP 5.1
RightCheek	Point		FDP 5.2
Chin	Point		FDP 2.10

4.3.3. Feature Translation Functions

Feature Translation Functions define how feature points are mapped from one location in the source model image to another in the destination model image, as a result of a facial action. Because these mapping values are later applied to the feature points of a new base image, the complete transformation includes the source model features, as well. Each FTF has the same structure as the source feature (i.e. FaceFeature) with the same number of lines and points, but instead of point coordinates, it includes mapping vectors.

The main issue in calculating FTFs is to find the corresponding points on feature lines, since corresponding feature lines on two images may have different number of points. Three methods have been investigated in this regard:

- Proportional

- Correlation-based
- Piecewise Proportional

Proportional method simply assumes that the relative location of a point on the feature line remains the same on all images. This means that if F_s is the feature line in source image with N_s point, and F_d is the same feature on destination image with N_d points, then the corresponding point to ith point of F_s is jth point of F_d where j is defined as follows:

$$j = \frac{N_d}{N_s} \times i \tag{4.7}$$

In correlation-based method, a search is performed on F_d points to find the jth point that gives the highest correlation. The search starts from ith point or with an initial guess based on equation 4.7. This correlation can be similar to one described by equation 4.1. Such method results in a higher computational cost and also does not guaranty a good match due to head movements (similar to discussion on optical flow-based approach).

Piecewise proportional method (used in this research) is based on the assumption that although the relative location of points does not remain the same on the whole feature line, but their relative location on each segment of that line will remain the same, with some reasonable approximation. These segments are defined based on co-visibility criterion [1] and we can assume that they undergo a linear shrinking or extension as the result of facial action. So the equation 4.7 will be replaced with the following formula where subscript k specifies the kth segment:

$$j_k = \frac{N_{k,d}}{N_{k,s}} \times i_k \tag{4.8}$$

$$j = j_k + \sum_{n=0}^{k-1} N_{n,d} \tag{4.9}$$

4.4. Mapping Facial Features

The transformations are done by first applying the FTF to the base feature points, as shown in Figure 4-9. These transformations fall into two categories: Simple and Combined.

[1] Co-visibility in this context refers to the points of a facial region being seen together under normal facial actions.

Simple transformations are those which have already been learned, e.g. T_{12} and T_{13} (assuming we only have an image in position 1 as base image, and separate Images 1, 2 and 3 for model). Combined transformations are necessary in cases when the destination 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 T_{14}.

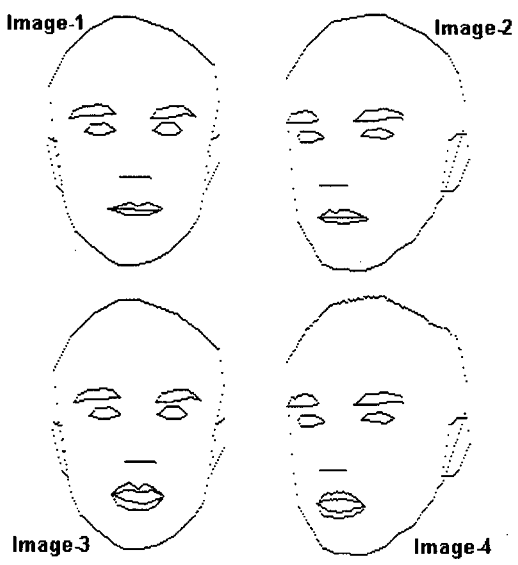


Figure 4-9. Using Feature Translation Functions

Before applying any FTF to a new set of features, the mapping vectors have to be normalized based on the difference between source model image in $T=(F,M)$ and the base image features. Normalization involves size scaling and 2D rotation (tilt left and right). If possible, the base image will be scaled to the same size as the model images and also rotated to full vertical position. This way all the pre-learned transformation can be used without any change. If it is necessary to use the base image in its original form, the model images will be normalized to base image space (size and 2D angle) and new set of transformations will be learned.

Assuming normalized transformation and images are being used, applying simple transformation T_{sd} (from model feature set F_s to model feature set F_d) to the base feature set F_b will create new feature set F_n:

$$F_n = F_b + a \times M_{sd} = F_b + a \times (F_d - F_s) \tag{4.10}$$

In this equation a (between 0 and 1) shows the portion of mapping we want to create. The corresponding points of each feature are found according to equation 4.9.

In case of a combined transformation without 3D head movements (e.g. talking and a facial expression), a linear combination can be used. Due to non-orthographic nature of 3D head movements, combined transformations involving 3D head rotation cannot 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. In Figure 4-9 to create Image-4 from Image-1, we have:

$$T_{14} = a \times T_{12} + b \times T_{13}' \tag{4.11}$$

$$T_{13}' = f_p(T_{12}, T_{13}) = (F_{m,3} + M_{12}) - (F_{m,1} + M_{12}) = T_{24} \tag{4.12}$$

f_p is Perspective Calibration Function (PCF) and involves moving $F_{m,1}$ and $F_{m,3}$ (model features) using T_{12}, and then re-calculating the FTF. Considering the way corresponding points are found in each step, this is not equal to a linear combination. If Image-2 is given as the base image, then we only need to apply T_{24}.

4.5. *Image Warping*

4.5.1. 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 (e.g. eye-brows, eyes, and lips)
- Face patches, covering the rest of the face as shown in Figure 4-7.
- New Regions, which appear as the result of head movement or talking (e.g. inside mouth)

The mapping vector for each point inside a feature island is found based on the mapping vector of the points on feature lines above and below it, using the following formula:

$$m_{r,c} = \frac{(abs(u-r) \times m_{u,c} + abs(r-l) \times m_{l,c})}{u-l} \quad (4.13)$$

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 ($m_{u,c}$ and $m_{l,c}$). 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 d is the mapping vector, as illustrated in Figure 4-10 for an eye-brow.

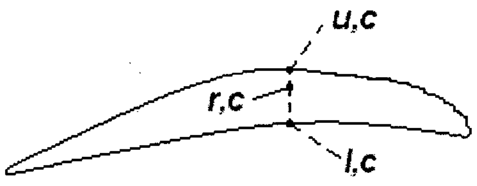


Figure 4-10. Mapping Vector for Feature Islands

4.5.2. Warp Function

The next step after mapping the feature points is to map other image points. This is done by the warp function. As discussed in Chapter 2, there are a few approaches to achieve warping. Mesh-based methods are the most common in animation but they do not allow any explicit control over critical features and the way they affect other points. Original feature-based approach by Beier and Neely [15] is one step toward such control to consider local information. But it's limited to straight lines. In this work, a modified version of that approach is used that:

- Works with any form of feature lines
- Finds the mapping vector of non-feature points as a function of feature point mapping (instead of directly calculating the coordinates of corresponding point)
- Uses only local features around a point to warp it (the mapping of a point is a function of limited features in its vicinity not all features as Beier-Neely consider)
- Does not use a fixed distance from the line

To perform the warp using the local information, face is divided into facial patches. In Figure 4-7, 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. Co-visibility is again the main concern when defining these face patches since it shows a common way of moving and being affected by the same feature lines/points. Points in each patch will be mapped to points in the corresponding patch of the target image, if visible. Face patches are defined based on co-visibility, 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 and allow local warping.

For each patch, a set controlling features (lines or points) are selected, and the mapping vector of each point in that patch is defined as a weighted average of the mapping vector of those features. For feature lines, the distance from the closest point on that line is considered:

$$m = \frac{\sum_{i=0}^{n} d_{p,i} \times m_{p,i}}{\sum_{i=0}^{n} d_{p,i}} \tag{4.14}$$

In this equation, $m_{p,i}$ is the mapping vector of the closest feature point on feature i, $d_{p,i}$ is its distance (to the patch point), and n is the number of features assigned to that patch. To further enhance this method, $m_{p,i}$ can be replaced by the average of mapping values on the feature line within a defined neighbourhood (e.g. 2 points on left and right). This adds a smoothing effect that compensates for possible correspondence errors and creates a smoother and more natural warping.

Table 4-3 shows the features that are considered for some examples of face patches. To allow continuous and smooth transition from one patch to another, these features are not limited to those around the patch but include some "farther" ones, as well. When the feature is a line, for each patch point, the closest point on a feature line is considered. Exceptions are closed feature lines such as Head and Nose, for them each patch point has four associated feature points on its top, bottom, left and right sides.

Table 4-3. Features Associated with Face Patches

Patch	Features	Comments
LeftForehead	HeadTop, HeadLeft Hair BrowLeft	The left side of forehead, above eyebrows
CentreForehead	HeadTop, HeadRight, HeadLeft Hair BrowRight, BrowLeft	The centre of forehead, above eyebrows
RightForehead	HeadTop, HeadRight Hair BrowRight	The right side of forehead, above eyebrows
LeftJaw	HeadBottom, HeadLeft LowerLip	The left side of lower face, below mouth
CentreJaw	HeadBottom, HeadLeft, HeadRight LowerLip	The centre of lower face, below mouth
RightJaw	HeadBottom, HeadRight LowerLip	The right side of lower face, below mouth

4.5.3. Newly Appeared Areas

The warp function described in 4.5.2 uses an inverse mapping, i.e. it goes through all the destination image pixels and finds the source pixels that correspond to them. This means that all the areas of new image are filled and there is no need for interpolation. But as the result of 3D head movements and also talking, there are certain areas in this newly created image that do not exist in the base image. For such areas the warp function finds the closest points on the base image. These points do not necessarily provide real data. Figure 4-11 illustrates this situation.

Filling the newly appeared regions is the only task that may not be done with a single input (base) image. In case of talking, the inside mouth image data (from base image or a second input image) is normalized properly and used to fill the open mouth area of new images. In case of head movements, a second image (e.g. a profile of the same person) is used that has the required region. We should map it to the target state, and use the data for that specific region. Mapping the second base image follows the exact procedure described above. The data from this new image can be used in two different ways:

- Create the final image as a blend of primary and secondary warp

- Use the secondary warp only to enhance regions of primary warp

To minimize the dependence of the secondary warp, the latter alternative is selected. The results can be seen in Figure 4-11.

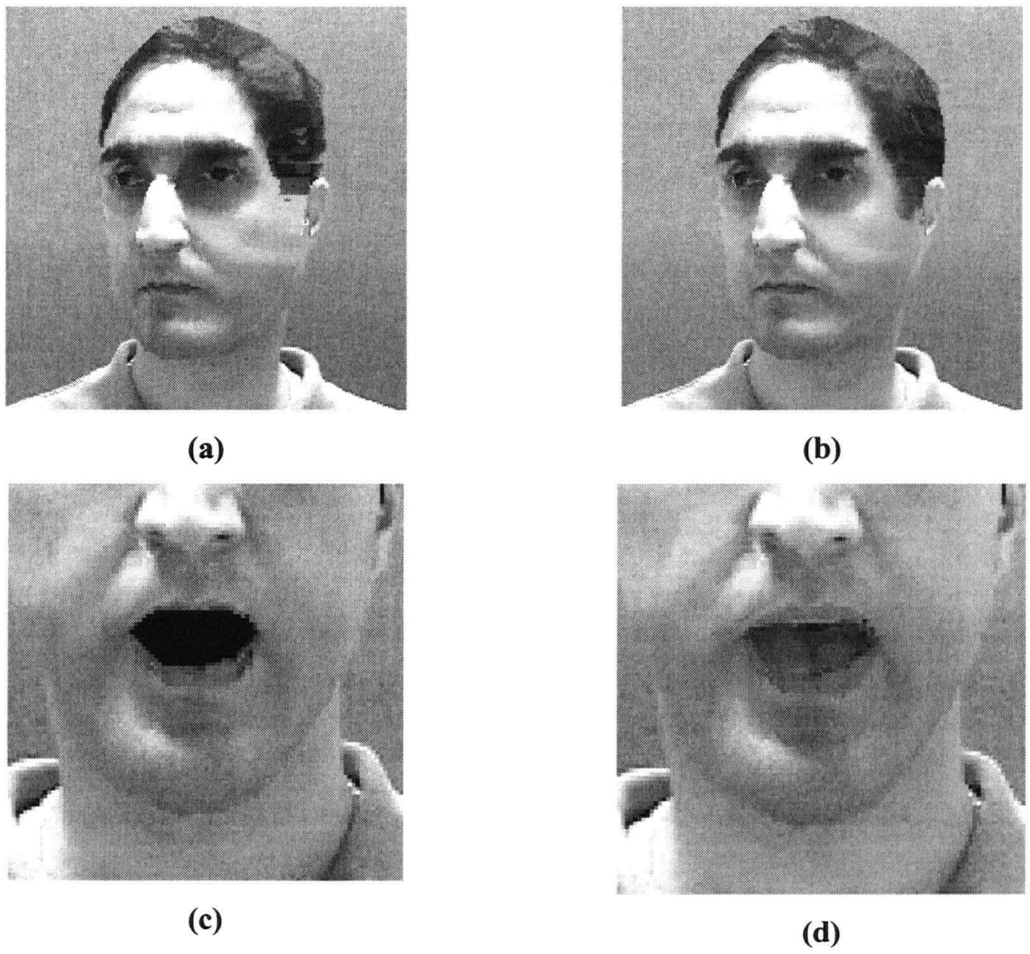


Figure 4-11. Newly Appeared Areas for Head Movement and Talking

(a) Using Existing Data, (b) Using Profile Image,
(c) Using Existing Data and Painting Inside Mouth Black, (d) Using In-Mouth Data.

4.6. *Texture Transformation*

In some cases like facial expressions, moving the existing face points may not be enough to cause the desired effects. Skin changes and wrinkles are the main issues, in this regard. This means that not only the movement but also the change in colour (or grey level) of pixels has to be considered. Tiddeman et al. [97] have proposed an approach to this colour/texture transformation. As discussed in Chapter 2, this approach has some weaknesses, for example:

- Larger database (not only feature but also pixel information has to be stored)
- Model-specific data (colour transformations are not easy to normalize)
- Global changes that are hard to normalize (transformations are defined globally for whole image rather than being associated with local features)

To address the need for texture changes, FIX has been extended to include local texture transformations around the features. This method has the advantages of:

- Storing only texture changes that are necessary for any specific facial action
- Being easy to normalize due to dependence on local features
- Reducing the computational cost by applying texture transformation only to the necessary areas

The colour change is stored in a normalized way (percentage of change) to be independent of skin brightness when applied to a new base image. It will undergo a size scaling as well (based on the feature size in the model and the base images) to match the size of the new features. The following algorithm explains this process:

```
// In Learning Phase
//
For each Feature $F_i$ in Destination Model Image
    Define Feature Area $S_i$ around $F_i$
    For each pixel in $S_i$
        Find Corresponding Pixel in Source Model Image
        Calculate Normalized Colour Change
            (destination - source)/source
    Store Feature Area
```

```
// In Run Time
//
For each Feature $F_i$
    Load Feature Area $S_i$
    Resize $S_i$ based on the size of Transformed Base Image
    For each pixel in $S_i$
        Find Corresponding Pixel in Transformed Base Image
        Apply Normalized Colour Change
            (colour += colour*change)
```

Figure 4-12 illustrates the result of such a texture transformation. For this example, only the colour changes in the area between the eyebrows needed to be stored, in order to amplify the effect of frown.

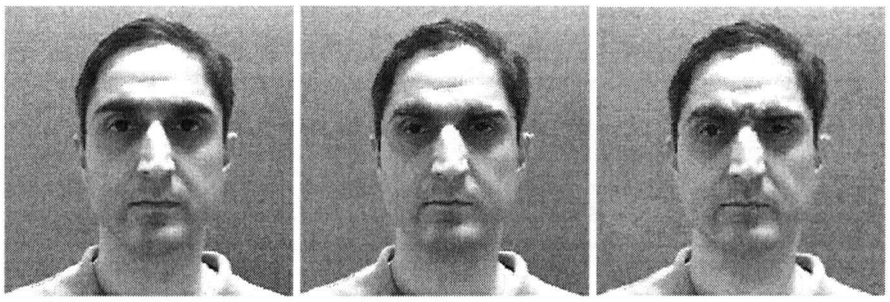


Figure 4-12. Texture Transformation

(a) Base Image, (b) and (c) Frown without and with Texture Transformation

4.7. Summary of FIX Features and Advantages

In chapter 2 the existing approaches to visual content generation for face animation were reviewed. In comparison to these methods, FIX offers a variety of advantages that can be summarized as follows:

- No need for complicated computationally-expensive 3D models
- No need for hard-to-acquire 3D data
- No need for a large database of 2D images for each person (unlike morphing-based 2D methods), due to use of only one set of learning images and storing only transformation information
- Realistic output due to use of photographs
- Personalization by applying transforms to any photograph

Table 4-4 compares the features supported by FIX with typical 3D (using 3D geometric models) and 2D (using image-based operations like morphing) methods.

Table 4-4. Features Comparison for FIX vs. Typical 3D and 2D Methods

Features	**FIX**	**3D**	**Image-based 2D**
Photo-Realism	Yes	No	Yes
Personalization	Yes	Somehow	Only with a large database
Data/Model Storage	Low	Low	High
Model Acquisition	Easy	Depends on method (some need 3D sensors)	Easy

5. ShowFace System

5.1. System Architecture

5.1.1. Requirements

As discussed in Chapter 2, one of the main weaknesses in existing face animation systems is the lack of a comprehensive framework. Most of the research in this area have been focused on graphic/animation techniques, limited compatibility (especially with MPEG-4), and occasionally content description and modeling. System-level concerns (e.g. architecture, openness, and streaming) and a "big picture" of face animation as a framework (not simply a set of tools and algorithms) have not been properly addressed.

Such a comprehensive framework should provide well-integrated solutions to following system-level issues:

- Modularity
- Compatibility
- Flexibility

A modular component-based architecture guarantees minimum interdependence between system parts, which in turn allows independent and parallel development and upgrade of one part with the least possible effect on the rest of the system. This is a major concern especially due to ever-changing features and functionality and also the need to introduce new technologies. A good example of such a modularity is encapsulation of visual content generation into one component so it is possible to switch from one approach (e.g. 2D image-based) to another (e.g. 3D geometry-based), without significant changes in the rest of the face animation system. A well-defined interface is the basic necessity in a successful modularization to make a component independent of internal implementation of the others.

Another important concern is compatibility with existing standards and technologies. This compatibility has double benefit of being able to (1) connect to frequently used systems, receive input in popular formats, provide output in an acceptable way, etc, and (2) use existing tools and facilities which are designed to provide the services within a standard framework. MPEG-4 is a good example of standards in the first category. Many applications now work with MPEG input/outputs and as discussed earlier, MPEG-4 standard has certain means of describing and animating faces. XML parsers and multimedia streaming environments such as Microsoft DirectShow [81] and Apple QuickTime [59] are examples of the second category that can facilitate face animation and the related processes, provided the system is compatible with their standards and interfaces.

Last but not least, flexibility in providing services to different types of applications is another major issue in face animation systems. Most of existing face animation solutions are either stand-alone application programs or single components (e.g. a Java applet) that has to be used within a certain container program. A well-defined framework should have different components and interfaces to allow different programs to connect and use its services.

Considering these general requirements, and all the capabilities mentioned in previous chapters, the following sections describe the *ShowFace* animation framework and its components and also underlying technologies and environments.

5.1.2. Basic Structure

The basic structure of the *ShowFace* system is shown in Figure 5-1. 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/Splitter, 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

ShowFace relies on Microsoft DirectShow [81] as the underlying multimedia technology for audio and video playback. DirectShow allows multimedia streaming through components called "filters" that process "frames" of multimedia data in a streaming scenario. DirectShow runtime environment provides a library of filters and utility functions for streaming purposes. DirectShow filters are objects based on Component Object Model

(COM) and work by exposing their functionality through COM interfaces. DirectShow will be installed as part of many application programs such as Internet Explorer web browser and many games. It comes with a set of filters such as audio and video decoders and renderers. Custom filters and interfaces can be designed and added to perform specific actions. DirectShow-based streaming is discussed in SubSection 5.1.3.

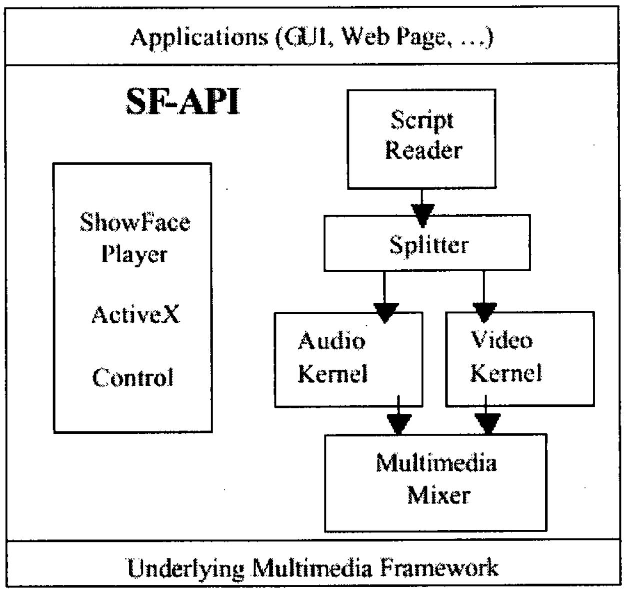

Applications (GUI, Web Page, ...)
SF-API
ShowFace Player ActiveX Control
Script Reader
Splitter
Audio Kernel
Video Kernel
Multimedia Mixer
Underlying Multimedia Framework

Figure 5-1. *ShowFace* System

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 the most of existing technologies, *ShowFace* components are implemented as DirectShow filters. This allows *ShowFace* objects to access the existing 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.

ShowFace components interact with each other, and client applications, using standard DirectShow interfaces and also *ShowFace* Application Programming Interface, SF-API, a set of custom interfaces exposed by the components and utility functions provided by *ShowFace* run-time environment (see SubSection 5.3.1). User applications can access system components and functions through SF-API, or use a wrapper object called *ShowFacePlayer*, which exposes the main functionality of the system, and hides programming details. Script Reader/Parser, FML Splitter, Video Kernel, and Audio Kernel are custom filters developed for *ShowFace* system. The mixing functionality is achieved by using DirectShow standard filters.

By using FML as the basic input mechanism, the script parser component of *ShowFace* efficiently uses standard XML parsers (installed by web browsers). After parsing the XML/FML input, the splitter filter interprets the FML commands, and creates detail information for audio and video actions.

Video Kernel uses FIX technique to create the desired images using the base image, transformation database, and facial actions description provided by splitter filter. The number of frames generated for each action is specified in FML input or is determined based on the duration of speech (to maintain synchronization). Audio Kernel uses a Text-To-Speech (TTS) engine and a set of pre-recorded diphones to generate the required audio data. Audio processing is described in Section 5.2. The *ShowFacePlayer* wrapper object is implemented as an ActiveX control, which can be easily used in web pages and other client applications. SubSection 5.3.2 describes this object and its usage.

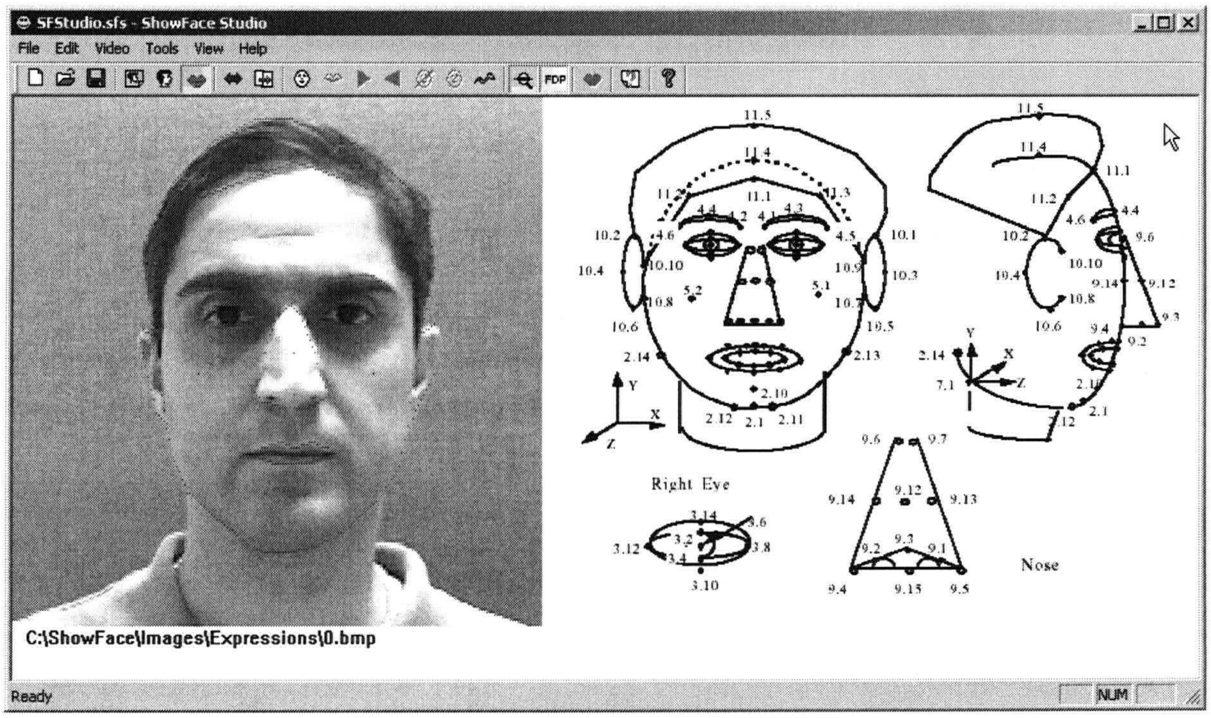

SFStudio.sfs - ShowFace Studio
File Edit Video Tools View Help
FDP
Right Eye
Nose
C:\ShowFace\Images\Expressions\0.bmp
Ready
NUM

Figure 5-2. ***ShowFaceStudio***

An off-line tool, *ShowFaceStudio*, is also developed to assist in detecting the features, creating the maps, and recording scripts. Using this tool, system developers can identify facial features in input images, create feature files and databases, perform normalization of images, run test mappings, and generate FML documents corresponding to their desired sequence of actions. A snapshot of this off-line tool is shown in Figure 5-2.

5.1.3. Streaming

The basic building block of DirectShow is a software component called "filter". A filter is a COM object that generally performs a single operation on a multimedia stream. For example, there are standard DirectShow filters that:

- Read multimedia files in popular formats
- Get video from a video capture device
- Decode a particular stream format, such as MPEG-1 video
- Pass decoded video or audio data to the graphics or sound card

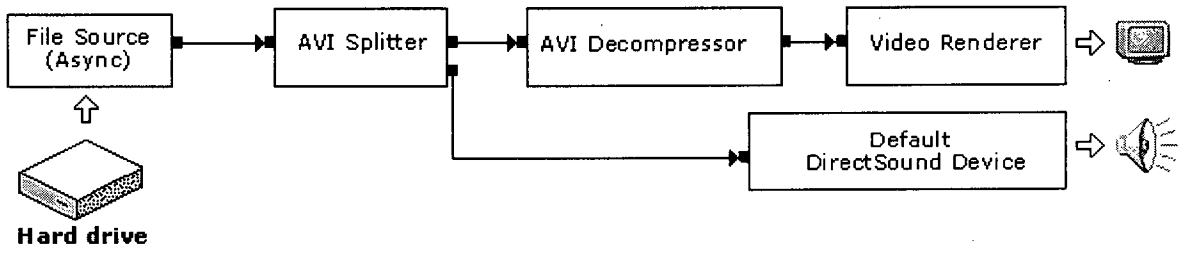


Figure 5-3. Sample DirectShow Filter Graph

Filters receive and deliver data through "pins," input and output objects associated with different media types. For example, if a filter decodes MPEG-1 video, the input is the MPEG-encoded stream and the output is an uncompressed RGB video stream. To perform a given task, an application connects several filters so that the output from one filter becomes the input for another. A set of connected filters is called a "filter graph". As an illustration of this concept, Figure 5-3 shows a filter graph for playing an AVI [1] file.

[1] Audio-Video Interleaved

The application program does not have to manage the individual filters in a filter graph. Instead, DirectShow provides a high-level component called the Filter Graph Manager. The Filter Graph Manager controls the flow of data through the graph (e.g. by asking video filters for a new frame of data). Applications make high-level API calls such as "Run" (to move data through the graph) or "Stop" (to stop the flow of data). If an application requires more direct control of stream operations, it can access the filters directly through COM interfaces. The Filter Graph Manager also passes event notifications to the application, so that they can respond to events, such as the end of a stream. DirectShow Software Development Kit (SDK) includes GraphEdit, a tool for creating, viewing, and testing filter graphs.

A typical DirectShow application performs following basic steps, as illustrated in Figure 5-4:

- Creates an instance of the Filter Graph Manager, using the CoCreateInstance COM function.
- Uses the Filter Graph Manager to build a filter graph (adding filters)
- Controls the filter graph and responds to events.

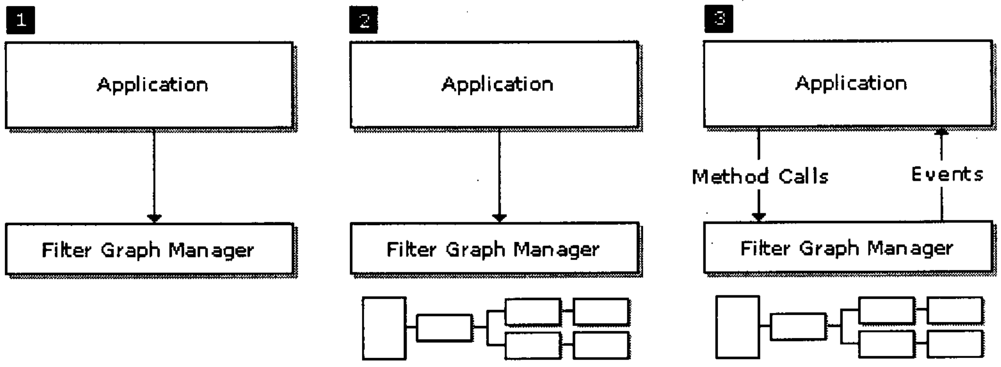


Figure 5-4. Using DirectShow Filters

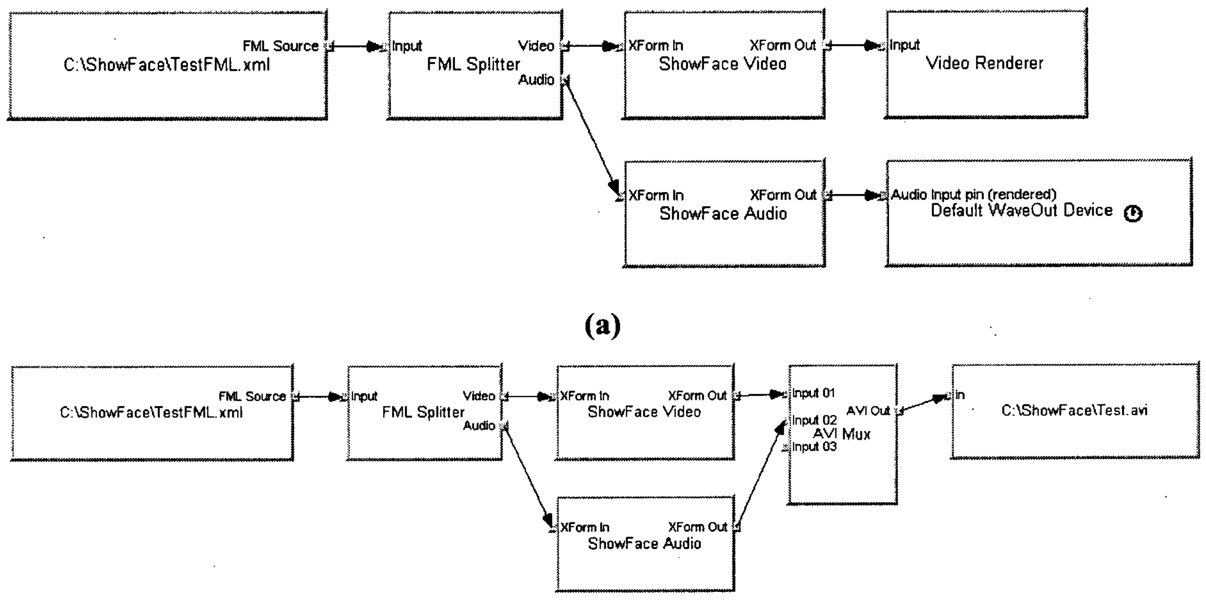
New filters can be developed and registered to be used by DirectShow runtime environment, provided they implement the proper interfaces that allow Filter Graph Manager perform filter creation and connection, and also data transfer. The most important of these interfaces are IMediaFilter and IBaseFilter for basic filter construction, and also IPin and IMemInputPin used for connections and data transfer. Some frequently used methods on these interfaces, to be implemented by the filter, are listed in Table 5-1.

Table 5-1. Basic Filter Interfaces

Interface Method	Description
IMediaFilter	
GetState	Retrieves the filter's state (Running, Stopped, or Paused)
SetSyncSource	Sets a reference clock for the filter
GetSyncSource	Retrieves the reference clock that the filter is using
Stop	Changes state to Stopped
Pause	Changes state to Paused
Run	Changes state to Running (actual data transfer in Receive method of IMemInputPin)
IBaseFilter	
EnumPins	Enumerates the pins on this filter
FindPin	Retrieves the pin with the specified identifier
QueryFilterInfo	Retrieves information about the filter
IPin	
Connect	Connects the pin to another pin
ReceiveConnection	Accepts a connection from another pin
Disconnect	Breaks the current pin connection.
ConnectedTo	Retrieves the pin connected to this pin
ConnectionMediaType	Retrieves the media type for the current pin connection
QueryPinInfo	Retrieves information about the pin, such as the name, the owning filter, and the direction
EnumMediaTypes	Enumerates the pin's preferred media types
EndOfStream	Notifies the pin that no additional data is expected
QueryDirection	Retrieves the direction of the pin (input or output)
IMemInputPin	
GetAllocator	Retrieves the memory allocator proposed by this pin
Receive	Receives the next media sample in the stream. Media sample is an object that implements IMediaSample interface

The filter graphs for normal playback and also writing to file are shown in Figure 5-5. The media types used as output of Video and Audio Kernels are image frame and block of audio data. The block size for audio data is determined according to its sample size, sample-per-second, and video frame-per-second, so that each audio and video "frame" corresponds to the same time period.

$$AudioFrameSize = (AudioChannels \times AudioSampleSize \times AudioSamplePerSecond) / VideoFramePerSecond \qquad (5.1)$$



(b)

Figure 5-5. *ShowFace* Filter Graph

(a) Playback, (b) Write to File

FML Splitter uses the text data to be spoken, read from FML TALK tags, as input to Audio Kernel. The video output of FML Splitter and also its input are in form of a block of binary data of the type FMLMove. This type is defined as a structure with following integer members:

- Move; Main category of supported facial actions, i.e. Talking, 3D Movement, and Facial Expressions.
- Type; Subcategory of Move, i.e. what viseme, expression, or movement.
- Val; Amount of movement, i.e. percentage of complete transformation.

- Begin and End; Start and End times in milliseconds. This determines the number of frames to be generated by the Video Kernel. In case of talking, these times are calculated based on the duration of corresponding diphone. For other moves, the timing is specified in FML input.

5.2. Speech Synthesis

The *ShowFace* Audio Kernel is responsible for speech synthesis. As reviewed in SubSection 2.2.2, there are two main approaches to speech synthesis: model-based and concatenative. Due to difficulty of modeling vocal tract and the need for a natural speech, the second method is used in our research. A database of diphones, spoken by the person to be animated, is created off-line by extracting speech modules from existing audio footage. This can be done manually or by an automated tool. The tool developed by Darbandi [31] is an example of the automated approaches to this task. It utilizes different speech models to detect and extract diphones from a given audio data.

The speech synthesis is performed with the following steps:

- Receive text from FML Splitter
- Convert the input text to a set of phonemes
- Create the sequential diphones list
- Find the diphone data from the database
- Connect the diphones in a smooth way
- Pass the speech to the audio playback filter, frame-by-frame

Any existing TTS engine can be used to translate the input text to phoneme list [1]. The most common on Windows platforms is the group of tools based on SpeechAPI (SAPI). SAPI guarantees a standard COM interface for the TTS engines but it needs the installation of SAPI-compatible software which is usually not free. The free version of TTS engine from Microsoft does not expose the phoneme translation to external applications, due to limited implementation. For this thesis, MBROLA open source speech synthesis tool

[1] It is not efficient to develop a new translator for this purpose, due to availability of commercial and free tools. It should be noted that only phoneme translation is done by the TTS engine. Actual speech synthesis has to be performed by *ShowFace* Audio object.

(http://tcts.fpms.ac.be/synthesis) has been used. It installs a small dictionary and can be added to the *ShowFace* software at the source-code level.

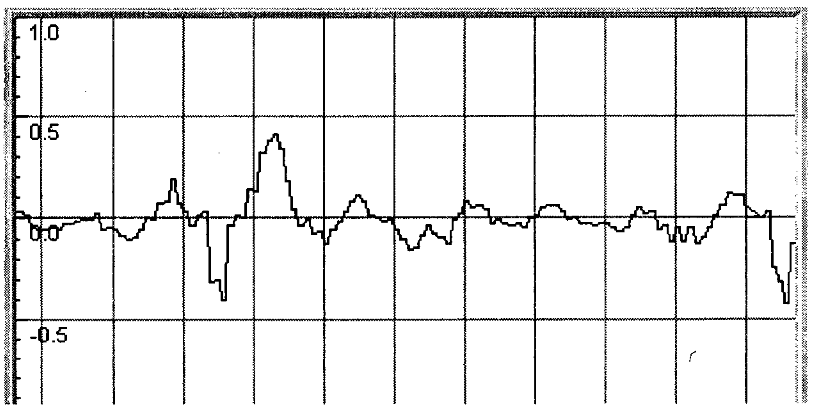
Due to the extraction of diphones from different parts of audio footage, their smooth connection needs three types of normalization to make the generated speech sound homogeneous and natural [37]:

- Power
- Pitch
- Phase

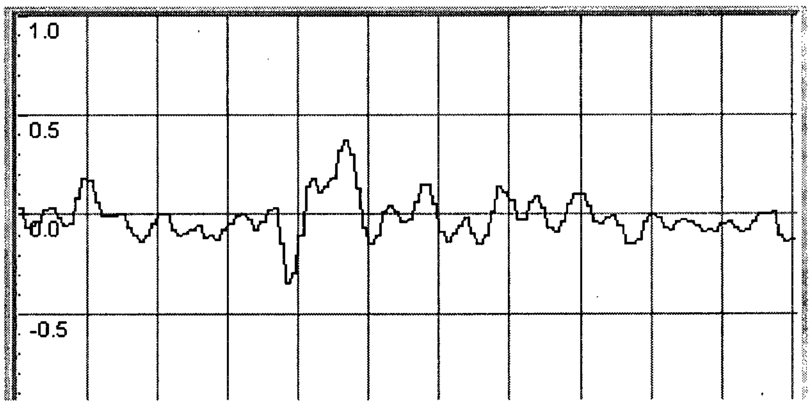
Different methods are proposed for time or frequency domain transformation on speech signal [56,86]. Current version of *ShowFace* does not provide on-demand transformation of audio power, pitch, and duration. Power and pitch of the extracted diphones are normalized off-line when creating database.

The phase normalization on the other hand has to be done in real time when connecting two diphones. The reason is that, due to the way diphones are extracted, the start and end points of diphones are the optimal connection point for the pair of phonems connected in the original audio footage but not necessarily optimal for connecting the diphone to another diphones/phonemes. Figure 5-6 illustrates this concept in a simplified manner. Diphones are usually made from the centre of a phoneme to the centre of another one. In *ShowFace*, this is done with an extra margin on each side. This allows a search to be done when connecting two diphones in order to find the optimal point of connection for that specific case.

The waveforms shown in Figure 5-6 are the last and the first 11 milliseconds of two diphones (each with a duration of about 200 milliseconds) which are supposed to be connected. Clearly, end of diphone (a) does not match the beginning of diphone (b). Optimal coupling seems to be possible by cutting 7 milliseconds off the end of (a) and 3 milliseconds from the beginning of (b).



(a)



(b)

Figure 5-6. Smooth Connection of Diphones

(a) Ending Part of a Diphone,
(b) Beginning of Another Diphone Starting with the Same Phoneme.

The search for optimal coupling point can be done in time domain (comparing windows of audio samples) or frequency domain (comparing windows of FFT [1] values). Due to the existence of noise, frequency domain comparison provides better results but requires more processing time. In either case, a simple distance function can be used to compare moving windows of two diphones (or the FFTs) and find the location that provides the minimum distance. 256-point windows seem to be suitable for comparison. Ideally, the windows should be moved 1 point each time, but for computational efficiency and without loss of too much accuracy, larger movements can be used.

[1] Fast Fourier Transform

It should be mentioned that the work presented in this thesis has focused mainly on visual aspects of face animation. Audio/speech processing has been the subject of considerable research. They are included in this work only to the minimum level necessary.

5.3. Application Development

5.3.1. SF-API

ShowFace components interact with each other, and client applications, through standard DirectShow interfaces and custom SF-API. Table 5-2 illustrates some important interfaces and methods of SF-API implemented by *ShowFace* components in addition to standard DirectShow interfaces.

Table 5-2. *ShowFace* API Methods for Filters

Interface Method	**Description**
IFMLSrc	
SetFMLEvent	Set the value of an FML Event for decision-making
ISFVideo	
SetImageFolder	Set the path to image/transformation database
ISFAudio	
SetWaveFolder	Set the path to audio database
SetWordInfo	Provide the input text (used by FML Splitter not applications)

Any program capable of using COM interfaces and DirectShow runtime environment can create and interact with the *ShowFace* filters. In a typical situation, this will be done through Filter Graph Manager, as described in SubSection 5.1.3. SF-API also includes CDSGraph class to make this interaction easier (Table 5-3). This class uses the standard and custom COM interfaces of Table 5-1 and Table 5-2 to perform the actions while isolating the applications from the COM programming details.

Table 5-3. CDSGraph SF-API Class

Method	Description
CreateFilterGraph	Create a Filter Graph using DirectShow COM interfaces
AddGraphFilter	Add a filter to graph
RenderFMLFile	Create a complete graph to render an FML file
PlayMedia	Start playing the graph
SetEvent	Set the value of an FML event
WaitForGraphEvent	Wait for an event from graph (e.g. stop) using a new thread

5.3.2. ShowFacePlayer

The COM interfaces and CDSGraph class have to be used at the source-code level in the client applications. To further simplify application development, especially for web-based systems where only simple scripting languages are available, SF-API includes a wrapper object. *ShowFacePlayer* is an ActiveX control that can be used in any application or web page, and interacts with the container program at binary level rather than source code.

ShowFacePlayer includes simple methods such as CreateMedia, Play, Stop, and SetEvent. A simplified HTML source for creating and using this control is shown here:

```
<INPUT onClick="Play()" type=button value="Play">
<OBJECT
     id=SFPlayer
     type=application/x-oleobject
     width=20 height=20 standby="Loading ShowFace Player..."
     classid=CLSID:71e54faa-d2ca-4a89-b62c-b60cd4191630>
<PARAM NAME="InputFile" VALUE="c:\ShowFace\TestFml.xml">
</OBJECT>
```

A simple script on that page is responsible for invoking methods on the object:

```
<script language="javascript">
function Play()
{
     SFPlayer.InputFile = "c:\\ShowFace\\TestFml.xml";
     SFPlayer.CreateMedia();
     SFPlayer.Play();
}
</script>
```

6. Evaluation and Experimental Results

6.1. Evaluation Criteria

6.1.1. Criteria Categories

Based on the survey in Chapter 2 and more studies, we categorize the basic requirements of a face animation system into the following groups:

- Content (e.g. realism)
- Architecture (e.g. compatibility)
- Development (e.g. minimum modeling data)

Table 6-1. Evaluation Criteria

Category	Criteria	Code
Content	Realism	C-1
	Graphic Capabilities	C-2
	Speech Generation	C-3
Architecture	Timeliness	C-4
	Descriptiveness	C-5
	Compatibility	C-6
	Modularity	C-7
Development	Computational Simplicity and Efficiency	C-8
	Input Requirements	C-9

Since the requirements should have a clear (and if possible, one-to-one) correspondence to test and evaluation criteria, we consider these requirements to be the base for a comprehensive set of evaluation criteria. Following subsections provide information on members of each category and try to give clear definition and some numerical metrics for them.

Table 6-1 summarizes our proposed evaluation criteria. Although it might not be possible in all cases, in compiling this set, we consider an evaluation measure to be:

- Clearly defined,
- Quantifiable and objective,
- Generally accepted as a requirement,
- Automatable

The last item in the above list refers to the ability to have a software tool performing the evaluation without any need for a human observer. This may be considered a result of the first and the second items but has been mentioned separately to emphasize its importance.

6.1.2. Content

6.1.2.1 Realism

Having a realistic output seems to be the most frequently mentioned feature of animation systems, although in some cases a non-photo-realistic output is acceptable or even desired. Under the umbrella of realism we consider some other aspects, which might seem not directly related, such as geometric validity and comprehensibility. Although a realistic multimedia data is supposed to be physically valid and comprehensible to the audience, not every valid or comprehensible data is necessarily realistic.

Realism can be defined as similarity to the real multimedia data with the same subject (people and actions). To what degree animation is realistic is primarily judged by the viewers. The evaluation can so be done by creating samples using the proposed method and analyzing the viewers' reactions. Some guidelines can be:

- Choosing unbiased random viewers
- Using standardized significant subjects
- Checking testable effects such as lip-synchronization and mapping of specific points

Selecting non-random viewers may also have its own advantages. For instance, animators can be good candidates since they pay attention more carefully and can provide feedback on more issues (e.g. user interface).

The evaluation can be improved by using "ground truth," i.e. existing footage of the animation subjects. Comparison can be done directly between the animation output and the existing footage showing the same actions by the same characters. This, of course, is limited to the cases when the corresponding "real" footage is available.

The subjective evaluation, regardless of such guidelines, is somewhat unreliable and more importantly inefficient. Objective methods for evaluation of synthesized and/or processed images have been studied by image processing and compression researchers [72] and also recently by computer graphics community [74]. Direct physical metrics based on pixel-wise comparison of image intensity and colour are usually combined with some means of considering viewer sensitivity to spatial frequency, brightness, and other image aspects [72]. Perception-based methods are more advanced methods in this regard. Using models of human visual system, and considering spatial frequency, contrast, and colour distribution, Visible Difference Predictor (VDP) and it's variations [74] are shown to be powerful means of defining image quality metrics. VDP maps images to be compared to a multidimensional representation and applies a model of human visual system to find the image difference in perceptual space rather than normal physical space.

Such methods can be used for global or regional comparison of real photos with computer-generated images. For instance pure 3D approaches may not be able to create realistic images due to complicated 3D models required for details such as hair and wrinkles. Another example can be optical flow-based view morphing techniques which suffer from "visual noises" caused by mismatching the image points.

Special characteristics of face animation also need more feature-based approaches. They should consider validity and correctness of movements for significant feature points and lines in face. Such feature-based considerations can be done in three different ways:

First, performing regional image comparison for certain regions specified by important facial features. The comparison itself can be done by above-mentioned image quality metrics. A difference measure, D, can be defined as follows:

$$D = \sum_k w_k d_{fk}(i,j) \qquad (6.1)$$

Where d_{fk} is regional distance function, calculated over all region points, for kth feature area of images i and j, and w_k is the weight for that feature area in the whole quality assessment.

Second, performing the image comparison only for feature points and lines.

$$D = \sum_k w_k d(f_{ki}, f_{kj}) \tag{6.2}$$

Where d is Euclidean distance function, and f_{ki} and f_{kj} are k^{th} feature set in images i and j (e.g. lip lines). These measures are calculated in units of misplaced pixels and can be further normalized using the image size.

Third, checking the geometric validity of feature movements according to a set of pre-defined physical rules. For instance horizontal alignment of feature points must remain unchanged after moving head to right or left.

6.1.2.2 Graphic Capabilities

Common capabilities required in face animation systems include:

- Creating a wide range of actions (such as talking, head movement/rotation in different directions, and facial expressions)
- Personalization, i.e. creating images for a certain person based on minimum input data (2D photos or 3D sensor data)
- Robustness, i.e. making faces for different people

Evaluation of a proposed system based on these objectives is more straightforward than evaluating realism due to clearer definition, but also more qualitative. It is still possible to assign numerical weights to each one of these desired capabilities in order to incorporate them into a quantitative set of evaluation criteria. Such weights can be defined based on the importance of the related capabilities in each application.

6.1.2.3 Speech Synthesis

The ability to create speech is another major evaluation item. It can include maximizing quality and minimizing training and database requirements. A concatenative Text-To-Speech (TTS) system used for personalized speech synthesis needs a proper database of speech segments (e.g. diphones). For such a system we can define:

$$SEM = SC \times SQM / SDS \tag{6.3}$$

Where *SEM*, *SQM*, and *SDS* are Speech Evaluation Index, Speech Quality Metric, and Speech Database Size, respectively, and *SC* is a proper scaling coefficient. Common measures of audio quality (such as signal-to-noise-ratio and comprehensibility) can be used for *SQM*. The database size can be measured in terms of the number of pre-recorded speech segments and their individual size.

6.1.3. Architecture

6.1.3.1 Timeliness and Streaming

Most of the new applications for face animation (such as video conferencing and interactive agents) need real-time operation. This means that the system has to receive continuous input (in some form) and produce output in real-time. A streaming real-time animation not only needs to be reasonably fast but also has to guaranty synchronization and work with input and output in standard formats. So this criterion includes some architectural concerns as well as pure performance measures.

Relying on existing real-time streaming technologies can be considered a positive point, in this regard. On the other hand, complex mathematical calculations, as common in 3D approaches, might be time consuming operations that make the system unsuitable for real-time operation, especially in applications with limited hardware like video phones.

Real-time operation and streaming capabilities can be simply translated to quantitative metrics (for example, the number of frames with certain number of colours and resolution, that can be processed in real time). The ability to receive streams of input and produce streams of output as opposed to input/output files with fixed sizes, and synchronization issues (e.g. lip-synch) are also important, in this regard.

6.1.3.2 Descriptiveness

A structured method for describing desired actions and content is absolutely necessary for authoring and interaction with the system. Such a Structured Content Description needs to be compatible with commonly used languages and methods, in order to make interfacing easier. As seen in Chapter 2, this objective is usually ignored in face animation systems.

Some desired features of such a description mechanism are:

- Hierarchical view of animation, including high-level "stories" to low-level "actions" (like mouth-open)
- Support for MPEG-4 FAPs
- Support for MPEG-4 XMT framework
- Behavioural modeling and templates
- Dynamic decision-making (allowing external events to change the sequence of actions)

Face Modeling Language (FML) is an example of such a content description mechanism. FML time containers **`par`**, **`seq`**, and **`excl`** make it possible to put actions together in parallel, sequential, and exclusive ways. The latter is considered to provide decision making based on an external event which selects one of exclusive actions to execute.

Another aspect of FML (also seen in CML [43] and BEAT [24]) is cognitive and behavioural modeling. Current version of FML defines simple configuration and behavioural templates consisting of standard face actions. The FML documents can then use these templates in their "scenarios". CML and BEAT extend this modeling approach by defining more complex behaviours.

Such languages can be used both for authoring animation applications and also as input to animation players. Use of XML Document Object Model (DOM) and introduction of events and decision-making structures provide dynamic and interactive content.

6.1.3.3 Compatibility

As mentioned before, existing multimedia technologies and standards can help improve the performance of face animation. In some cases compliance with these standards can be required, due to their popularity. MPEG-4 standard and Web-based systems are good examples, widely used in normal applications.

Another example can be the use of XML-based languages for animation and modeling. This allows utilizing existing XML parsers and also compatibility with XML-enabled systems.

It may be hard to define numeric measures for this and some other requirements as evaluation criteria. Assigning weights to each one of them and considering those numbers as metrics can be one approach.

6.1.3.4 Modularity

Rapid development of technologies means that a system should be designed in a way that can be upgraded with minimum effort, in order to work with new components. A modular architecture and a set of well-defined and possibly accepted interfaces make such upgrade easier to achieve, and allow parallel development and maintenance of modules.

ShowFace system, for instance, provides SF-API which acts as a communication interface between different components. Each part of the system can be upgraded/modified without affecting the other parts, as long as they all use SF-API. An example can be using 3D face models instead of 2D image transformation currently implemented in the *ShowFace*.

6.1.4. Development

6.1.4.1 Simplicity and Efficiency

Computational simplicity and efficiency of the methods and components used in the system are primarily related to issues such as timeliness. But they also make the system development, maintenance, and evolution easier. So it is worthwhile to consider them as independent evaluation criteria. Needless to say, such criteria are highly subjective and hard to measure. Measuring software efficiency and simplicity, in terms of parameters such as the number of lines of code and programming structures, is a major area of research which is out of scope of this thesis.

6.1.4.2 Minimum Input Requirements

As mentioned in speech synthesis subsection, the amount and type of data required to "train" the system is a major evaluation index. For example, some animation systems based on geometric head models need 3D data produced by 3D scanners which might be hard to provide. Some 2D image-based methods, on the other hand, need a relatively huge database of images to create new ones. MikeTalk [41] uses a complete set of "visemes" (visual presentations of phonemes) to create visual speech. Although the number of such images is limited for one person in one orientation of the head, the method cannot be used to generate animation for other faces and in other head orientations. Other extensions to this approach [46] use a more extended set of images to deal with head movement which increases the database size.

A weighted combination of difficulty of creating the database and its size is a necessary evaluation index for face animation systems. Methods that create the 3D head model based on 2D photos (e.g. [62]), and those which use one set of images to learn the necessary image transformations and apply them to multiple faces, will result in better evaluation indices, in this regard.

6.2. Experimental Results

6.2.1. Test Procedure

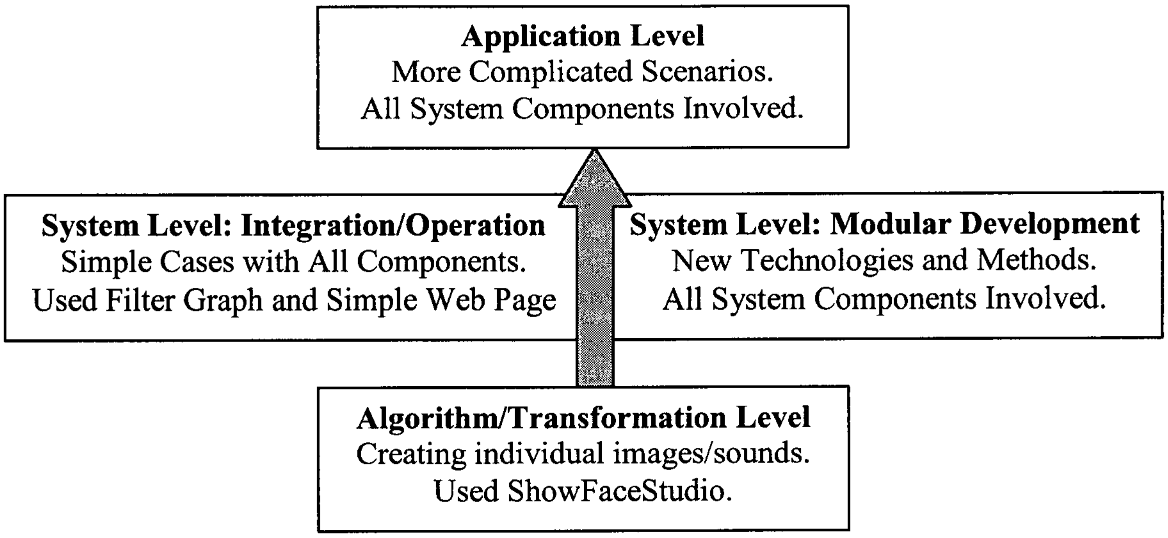

Application Level
More Complicated Scenarios.
All System Components Involved.
System Level: Integration/Operation
Simple Cases with All Components.
Used Filter Graph and Simple Web Page
System Level: Modular Development
New Technologies and Methods.
All System Components Involved.
Algorithm/Transformation Level
Creating individual images/sounds.
Used ShowFaceStudio.

Figure 6-1. *ShowFace* Test Procedure

Considering the criteria introduced in 6.1, as shown in Figure 6-1, the *ShowFace* system has undergone a series of test experiments, which fall into the following groups:

- Individual transformed images mainly to test the FIX approach (used *ShowFaceStudio*).
- Simple facial actions to evaluate the integration of system components, lip-synch, streaming, and other system-level concerns (used *ShowFace* components and GraphEdit SDK tool).

- System upgrades (with new technologies and methods) to verify openness, modularity, and component-based structure (used *ShowFace* components and sample applications).
- Animation applications to test FML, application-level concerns, system abilities in real-world scenarios involving more complicated tasks such as decision-making, event handling, and web-based operation (used *ShowFace* components and sample applications).

During these experiments, the system has been evaluated according to the criteria introduced in Section 6.1 (revisited in Table 6-2). Also, a comparison has been made between *ShowFace* and some of the most influential and accepted face animation systems. Three of these systems [19,41,97] use 2D image-based approaches which make them more suitable for comparison. A 3D system [62] is also selected due to its typical capabilities, to provide a more comprehensive comparison. The compared animation systems are:

- MikeTalk by Ezzat et al. [41].
- Video Rewrite by Bregler et al. [19].
- Perception Lab Face Animation System, by Tiddeman et al. [97].
- Virtual Human Director (VHD), by Lee et al. [62].

Table 6-2. Evaluation Criteria, Revisited

Criteria	**Code**
Realism (geometry, texture, lip-synch, etc)	C-1
Graphic Capabilities (personalization, variety of facial actions, etc)	C-2
Speech Generation (audio quality, lip-synch, etc)	C-3
Timeliness (streaming, real-time operation, etc)	C-4
Descriptiveness (structured content description)	C-5
Compatibility (with existing standards and technologies)	C-6
Modularity (independent components with well-defined interfaces)	C-7
Computational Simplicity and Efficiency (specially considering applications with limited hardware)	C-8
Input Requirements (for modeling and also in run-time)	C-9

Table 6-3 summarizes the results of this comparative evaluation. Although this thesis has proposed a comprehensive set of evaluation criteria for face animation systems, but due to their complex nature, quantitative measures for most of the proposed criteria are still in the offing or out of the scope of this work. Using qualitative evaluations has been an unavoidable consequence of this fact. Following sections describe the experiments, evaluation, and comparison, in more detail. The performance according to different criteria is evaluated at five levels of Not-Supported, Low, Medium, High, and Very High (0 to 4) quality.

Table 6-3. Summarized Comparative Evaluation

Criteria	ShowFace	MikeTalk	Video Rewrite	Perception Lab	Virtual Human Dir.
C-1	2.3	1.6	2	2	2.3
C-2	3	1.5	1.75	2.25	3
C-3	2	2	2	0	2
C-4	2	1	1	1	1
C-5	4	0	0	0	3
C-6	4	1	1	1	3
C-7	4	1	1	1	2
C-8	3	3	3	3	2
C-9	3	1	2	3	3

6.2.2. Realism

Figure 6-2 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, 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).

Subjective evaluation is performed by different viewers, including:

- Development team (author and his supervisor)
- Two other graduate students in the same ECE group/lab

- Two computer engineers from un-related fields
- Two other individuals who are not working in animation or computer fields

Evaluating the realistic appearance of the transformed images has been done by individually observing them, and also comparing the synthetic images with corresponding real ones (when they exist), in four different aspects:

- Mapping of facial feature lines and points
- General warping of existing image regions
- Filling newly appeared areas
- Texture/colour changes

Figure 6-3 shows two sets of sample base images, transformed versions and real images corresponding to those transformations. Using equations (6.1) and (6.2), and using feature points and selected non-feature points, we can see that the average misplacement for existing image pixels is less than 5%. Proper smoothing of transformed feature lines may enhance the visual appearance but not necessarily this measure. This shows that such a measure cannot be used by itself, and has to be accompanied by other measures. Also, the number is larger for newly appeared areas due to the fact that they are filled with pixels of the profile image and usually mapped corresponding to a large movement. Difficulty of applying a single texture transformation to different people is another cause of error.

Non-integer ratio of diphone time to frame time causes minor errors in lip-synchronization. *ShowFace* tries to overcome this by adding up the extra times and inserting a new frame when necessary. This makes the correct total number of frames (e.g. per word or sentence) but still suffers from lip-synch errors at diphone level. Further synchronization (e.g. reducing diphone time as suggested below) is necessary to resolve this quality issue.

$$NumberOfFrames = \mathrm{int}(\frac{DiphoneTime}{FrameTime}) \tag{6.4}$$

$$IdealDiphoneTime = NumberOfFrames \times FrameTime \tag{6.5}$$

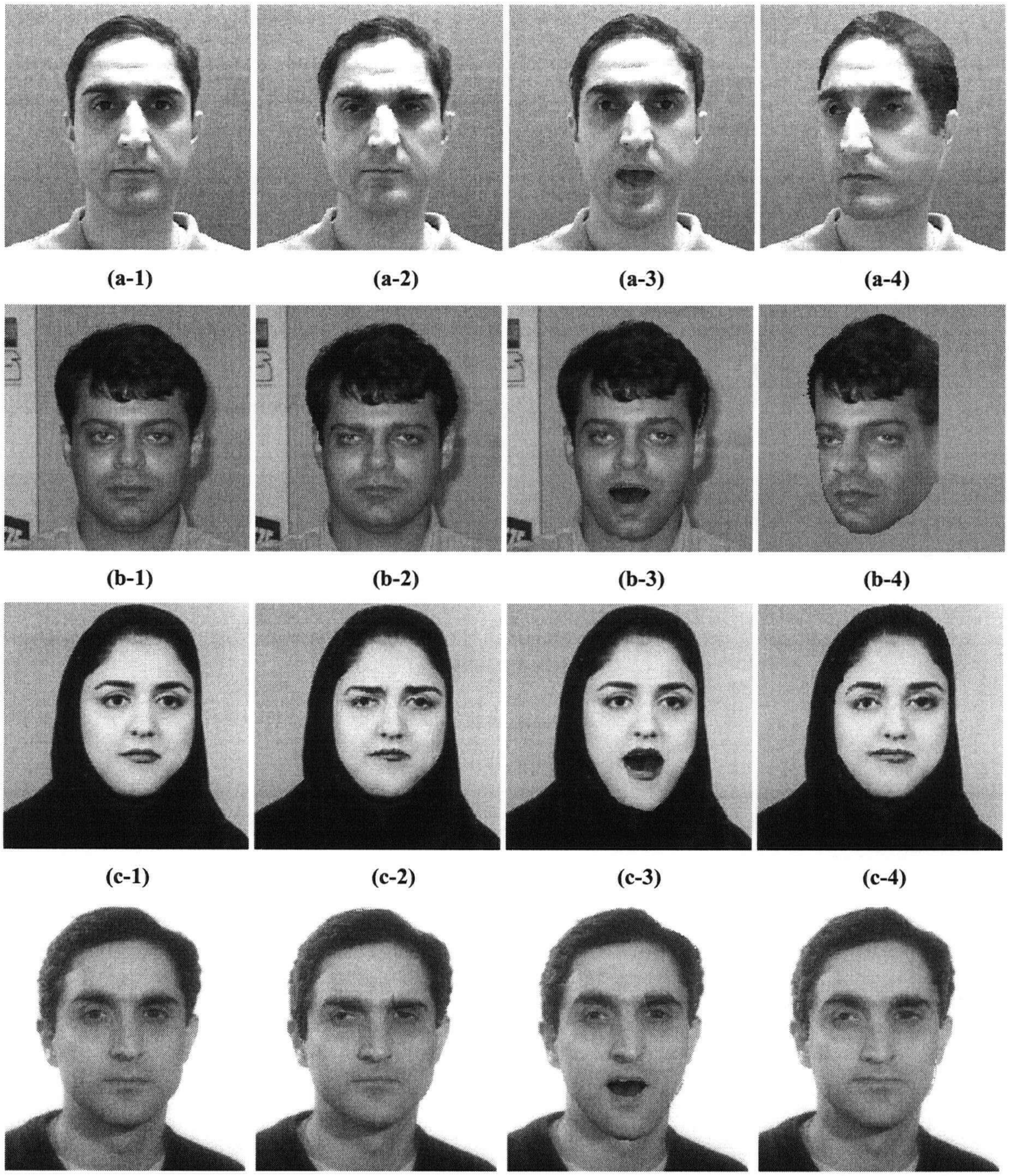


Figure 6-2. Image Transformation Results

Images at the left side are the base images. Others are "angry", "talking", and "moved" ("happy" in c-4 and d-4).

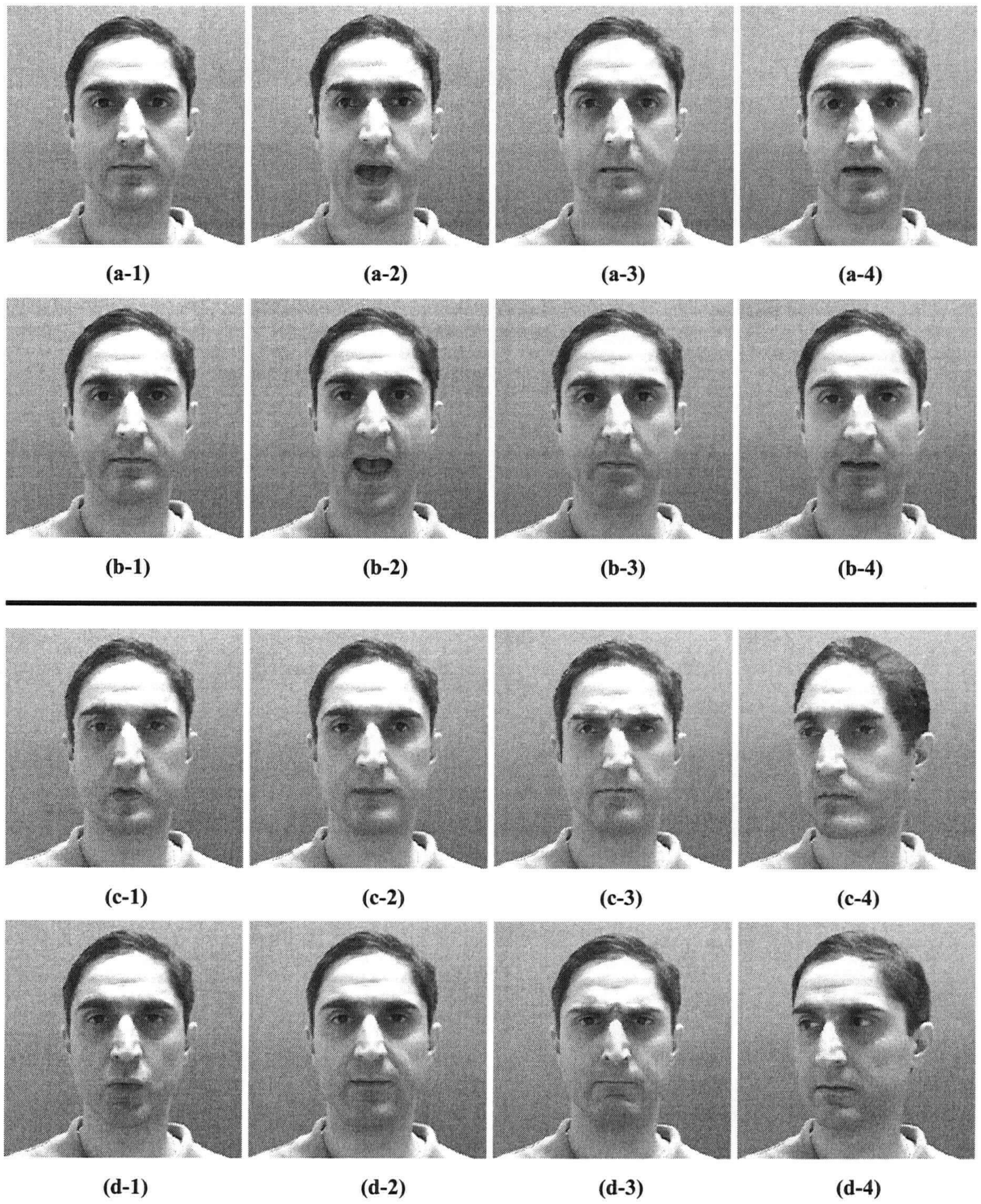


Figure 6-3. Comparing Transformed and Real Images

(a-1) to (a-4) and (c-1) to (c-4) Transformed Images, (b-1) to (b-4) and (d-1) to (d-4) corresponding Real Photo. Transformations, all applied to (b-1), are from left to right: Top Group (a): neutral, sound "a", sound "m", sound "n" Bottom Group (c): sound "u", sound "v", frown, and left-turn.

Another measure to consider was the amount of error when reading lips in absence of audio. This test was not possible due to lack of proper individuals. The video resembles the related speech for non-expert people, in a subjective way.

Considering the overall visual quality of output, and comparing with other animation systems in Table 6-3 (for which no detail measurement was available or possible), all of them are at a medium to high level of realism for visual speech, but facial expressions and head movements are not supported by all the systems. This results in the following table of comparison where a score of 1 means that the system has either low or no direct support for that sub-criterion of realism but it is possible to extend the approach for partial support:

Table 6-4. Realism in Different Animation Approaches

Sub-criterion	**ShowFace**	**MikeTalk**	**Video Rewrite**	**Perception Lab**	**Virtual Human Dir.**
Speech	3	3	3	2	3
Expressions	2	1	1	3	1
Movements	2	1	2	1	3
Average	*2.3*	*1.6*	*2*	*2*	*2.3*

6.2.3. Graphic Capabilities

For this criterion personalization, visual speech, facial expressions, and head movements are considered as sub-criteria. The scores for each is not necessarily related to quality and realism as presented in Table 6-4, but shows how each functionality is supported in the related system. In this regard, scores 0 to 4 mean:

- Not supported at all.
- Not supported directly but somewhat extensible to include.
- Partially supported.
- Typically supported but with limitations.
- Fully supported.

The optical flow-based approach in MikeTalk makes it impossible to apply transformations to other persons unless two characters are similar in geometry and texture. Feature-based approach of Video Rewrite allows a limited support for personalization although method is not specifically designed for it. On the other hand, Virtual Human Director uses a 3D head model that can be applied to any individual provided we have enough input information.

The result of this part of evaluation is shown in Table 6-5. It has been hard to come up with scores for the systems that do not support a specific functionality. For example, the Perception Lab Face Animation system has no direct involvement in visual speech but a logical extension to its visual content generation approach will be capable of supporting visual speech.

Table 6-5. Graphic Capabilities of Different Animation Approaches

Sub-criterion	**ShowFace**	**MikeTalk**	**Video Rewrite**	**Perception Lab**	**Virtual Human Dir.**
Personalization	3	1	2	3	4
Speech	3	3	3	1	3
Expressions	3	1	1	3	1
Movements	3	1	1	1	4
Average	*3*	*1.5*	*1.75*	*2.0*	*3*

6.2.4. Speech Generation

All the systems under review (except Perception Lab system) have direct support for speech generation. Unfortunately detail information and enough samples are not available for comparison purposes. Based on existing data, they all seem to be at a medium level of quality.

6.2.5. Timeliness

ShowFace has a streaming framework and lip-synch algorithms in place which are major issues in timeliness. Real-time performance needs optimization of algorithms and implementation. At this point generation of one single frame takes about 150 millisecond on a 1 GHz PC which makes it necessary to write data to a file and play it back later, for frame rates higher than 6 fps (frame per second).

No such data is available for other systems, but comparing the algorithms used, they are unlikely to have a real-time performance. On the other hand, *ShowFace* is the only system providing a streaming structure that makes a real-time operation possible after some optimizations.

6.2.6. Descriptiveness

The inclusion of FML in *ShowFace* provides a structured content description. VHD has the similar concept but no details are available on definition of spatial/temporal relations or support for decision-making. No specific way for content description is defined in other systems except VHD which has a hierarchical content description method, although it is not defined clearly and may not support higher level functionality such as event handling and parallel actions.

Several sample applications are developed for *ShowFace* to test and illustrate its descriptive power. These examples are reviewed in Section 3.7 and show the flexibility of FML to provide content description in simple and more complicated cases involving parallel actions, external events, and even dynamic content modification.

6.2.7. Compatibility

Due to expanding use and popularity of MPEG-4 FAPs, compatibility with these parameters to describe facial actions is widely considered by face animation systems. As discussed before, the input format and transformations used in *ShowFace* are fully compatible with MPEG-4 FAPs. Also, FML is capable of working in MPEG-4 XMT framework. MikeTalk, Video Rewrite, and Perception Lab system do not support FAPs explicitly, but the techniques they use provide basic means of working with them, at least partially (e.g. visemes in MikeTalk). VHD has MPEG-4 compatibility as an explicit requirement, but it is only at FAP level.

Another aspect of compatibility in *ShowFace* is the use of XML that allows utilization of available tools and also translation into XMT. None of the other methods have considered an XML basis, although it has been used by few other systems as mentioned in Section 2.1.

ShowFace is also compatible with DirectShow multimedia technology which provides a set of functionality to animation system. This aspect, of course, has its own limitations, i.e. certain extra requirements and limitations for the run-time platform. On the other hand, the modular structure of *ShowFace* allows it to be ported to another platform, with minimum effort.

6.2.8. Modularity

Component-based structure and well-defined interfaces have been major requirements of *ShowFace*. During the work on the system, several "upgrades" and "replacements" have occurred that illustrated the flexibility of system structure and its ability to replace a component with a new one without affecting the other parts. Examples are:

- Using FIX instead of OF-based approach.
- Using MBROLA instead of SpeechAPI.
- Using FML instead of simple FAPs.

ShowFace also provides interfacing mechanism, library functions, and wrapper objects, and relies on a fully component-based multimedia technology. Architectural information is not available for other systems but no such concern is explicitly mentioned in the related publications.

6.2.9. Computational Simplicity and Efficiency

Avoiding complex algorithms has been another requirement in designing *ShowFace*. Considering the fact that a major target application is videophone systems with limited hardware and relatively low-end processors, it is important to use simple computations and optimize the software to be as efficient as possible. Some of the features used in *ShowFace* are already supported by the operating platform (e.g. XML parsing and audio/video playback). Most of the remaining (core) components use simple algorithm for interpreting FML scripts, converting text to phonemes and diphones, and mapping the pixels (mainly adding a vector and averaging). Among more complicated tasks are correlation-based search and FFT for smooth coupling of diphones.

On the other hand, the implementation of the *ShowFace* algorithms has not been properly optimized, yet. As a result of this, the execution takes more than necessary time, memory, and disk space. Proper optimization is required (and possible) to reduce the algorithm execution time, number of disk access tasks, and database size. No detail is available on computational aspects of other systems but 2D approaches are likely to have the same level of complexity while VHD needs to have more complicated 3D graphics.

6.2.10. Input Requirements

Generally two sets of data are required for an animation system. In *ShowFace* terminology they are called *model* and *base*. Model data shows how to do different tasks (e.g. pre-learned transformations). Base data is about the character for which the animation is being created (e.g. input images). Combination of FML and FIX considerably reduces the model and base data sizes, compared to OF-based approaches like MikeTalk, due to the fact that only feature mapping needs to be stored as model, and it can be applied to any new character represented by only one base image.

3D approaches that utilize 3D sensors have major difficulty regarding the model data. This has been reduced in systems like VHD by using two 2D photographs to calibrate a generic model. On the other hand, having a 3D head model (with one or two images for texture mapping) results in almost the same amount of data requirements as in *ShowFace*, but the 3D model has to be modified/calibrated for each character to be animated.

7. Concluding Remarks

7.1. Objectives and Contributions

Within the diverse field of multimedia composition and presentation, from more traditional examples such as painting to state-of-the-art computer animation, *Making Faces* appears to be a very challenging task. Recent developments in the areas of Virtual Reality, Virtual Environment, Video Conferencing, Games, and Agent-based Online Applications (e.g. Shopping and Customer Support) have drawn considerable attention to character, and especially face animation. Replacing the audio-visual data of "real people" with multimedia presentations based on "virtual agents" seems 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 case, while creating new scenes with "unavailable" characters is an example of the latter. Figure 7-1 illustrates some application areas where face animation can be used.

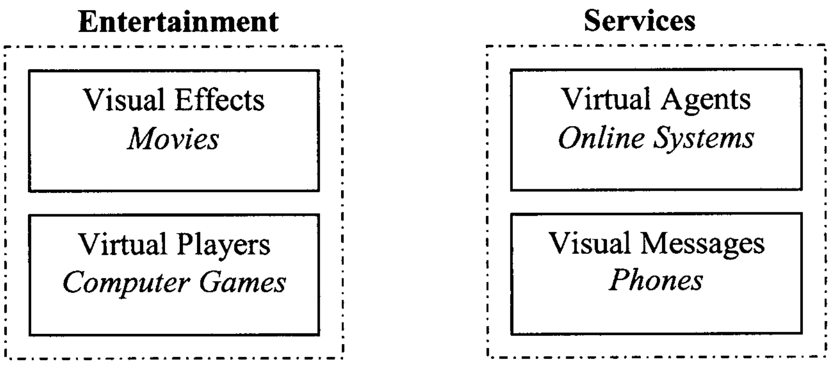


Figure 7-1. Some Applications of Face Animation

Personalized Face Animation includes all the information and activities required to create a multimedia presentation resembling a specific person. The input to such a system can be a combination of audio-visual data and textual commands and descriptions. Technological advances in multimedia systems, speech/image processing, and computer graphics have resulted in a rapidly growing number of publications. These research achievements, although very successful in their objectives, mostly address a limited subset of face animation requirements (Figure 7-2, right side). The *ShowFace* system that we proposed in this thesis, takes an important step toward a comprehensive framework for face animation by introducing the following major contributions:

- Component-based structure with well-defined interfaces, independent modules to be designed and upgraded with minimum effect on the other parts, and different means of application development.
- Hierarchical structured content description language (FML) compatible with MPEG-4 FAPs, XMT framework and XML parsers, with support for parallel actions, detailed timing, and external events.
- Feature-based Image Transformation for face animation in order to maximize realism and minimize the input data.
- Comprehensive evaluation criteria covering different aspects of face animation systems.

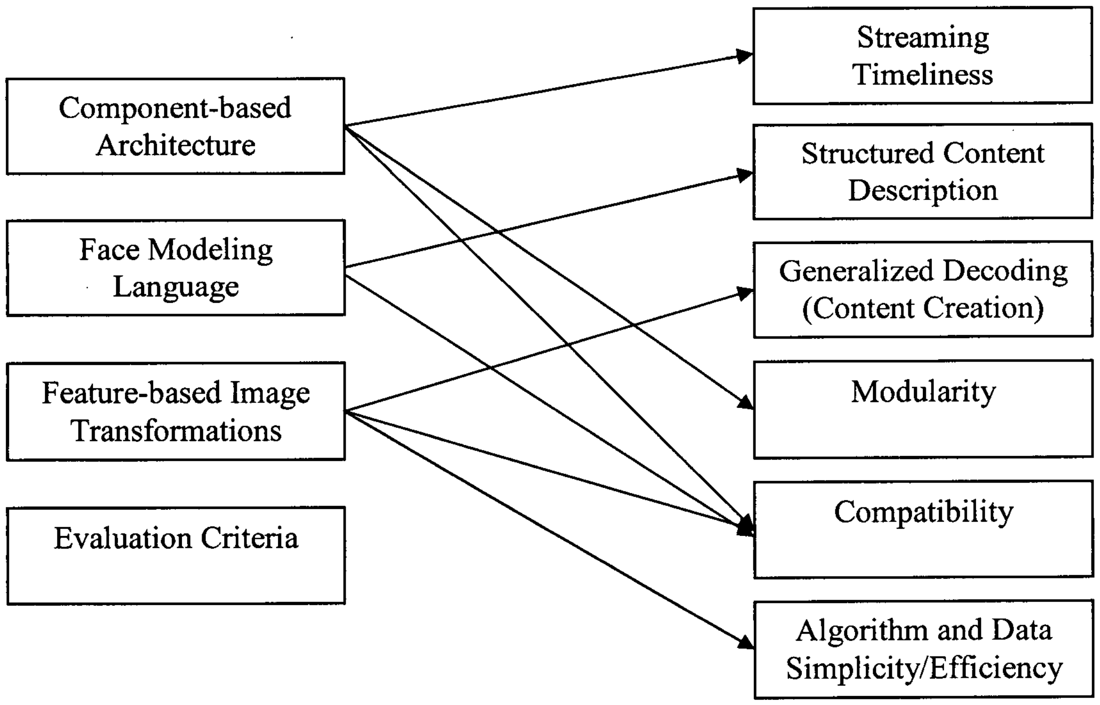

Component-based Architecture
Face Modeling Language
Feature-based Image Transformations
Evaluation Criteria
Streaming Timeliness
Structured Content Description
Generalized Decoding (Content Creation)
Modularity
Compatibility
Algorithm and Data Simplicity/Efficiency

Figure 7-2. Face Animation System Requirements and *ShowFace* Contributions

Figure 7-2 shows how these system features contribute to addressing animation requirements. The preceding chapters discuss major aspects of *ShowFace* in addition to other system details such as speech synthesis. The defined set of evaluation criteria has successfully shown the effectiveness of *ShowFace* in dealing with face animation requirements. Although defining quantitative and objective measures for all of these evaluation criteria is not yet completely possible [1], our criteria not only help evaluate the proposed system, but also qualify as a major contribution of this thesis, since no proper effort has been made to define such a comprehensive evaluation criteria for face animation systems. We cover the following criteria:

- Realism (geometry, texture, lip-synch, etc)
- Graphic Capabilities (personalization, variety of facial actions, etc)
- Speech Generation (audio quality, lip-synch, etc)
- Timeliness (streaming, real-time operation, etc)
- Descriptiveness (structured content description)
- Compatibility (with existing standards and technologies)
- Modularity (independent components with well-defined interfaces)
- Computational Simplicity and Efficiency (specially considering applications with limited hardware)
- Input Requirements (for modeling and also in run-time)

ShowFace architecture consists of independent components for reading and parsing the input, creating audio and video content, and playback or file storage. It provides comprehensive API and wrapper classes and components to simplify application development.

Feature-based Image Transformation (FIX) acts as the main visual content creation method in *ShowFace*. FIX is based on detecting the required mapping of facial features needed for specific facial actions and applying them to a base image combined with warping the non-feature points of the image. Speech, facial expressions, and 3D head movements are included in the FIX library of transformations in a way that is compatible with MPEG-4 FAPs. FIX maximizes the realism of generated images while minimizing input and modeling data. This approach may not be as powerful as a 3D model-based one in generating 3D movements, lighting, or facial skin effects, but it does provide an optimal solution when the application domain is constrained with hardware and software limitations and still requires a realistic appearance.

[1] Such measures are worthy of being the topic of separate research projects by themselves.

Last but not least, Face Modeling Language (FML) is presented as a high-level content description language for face animation, bringing together the strengths of MPEG-4 and XML, to handle spatial and temporal relations between face actions, event handling and decision-making, behavioural templates, and dynamic content generation.

The evaluation of the proposed system is performed by considering a comprehensive set of criteria and comparing the system with a group of face animation systems/approaches. This evaluation shows that *ShowFace* provides acceptable levels of quality in content together with a flexible modular streaming structure, minimum amount of modeling and runtime input data, and also computational simplicity. Although it may not have the same constructive power of a 3D head model for various head movements, the provided capabilities make it suitable for a variety of applications.

7.2. Future Research

This thesis does not claim to have answered all the questions related to face animation but is intended as a step toward a comprehensive framework for personalized face animation. Although quite successful in satisfying the requirements to some reasonable levels, *ShowFace* has certain areas for improvement. Further research will be mainly focused on:

- Optimizing FIX algorithm for real-time applications
- Replacing FIX with a 3D approach for less-constrained applications
- Making expression overlays (e.g. talking with frown) more realistic by studying the inter-effect of image transformations
- Better integration of FML with MPEG-4 XMT by providing automated translators from FML to native MPEG-4 format
- Adding behavioural modeling to FML
- Defining quantitative measures for evaluation criteria

The above list is certainly open to other enhancements, and the face animation community as a member of a larger group of virtual/virtualized/augmented reality has more questions to answer in the years to come. Here are some intriguing ones:

- What are reality and realism? How are they related to comprehensibility?

- What are the moral issues involved in creating "realistic" virtual worlds/people?
- How far can we go into virtual worlds without losing contact with the real world?
- Is virtual world a real world itself? Where is the boundary?
- Is a holographic character (animated in the "outside" world) real?
- Can virtual reality recreate the dead, or recreate the world in a better way?

It will take quite some time to debate and perhaps answer all of these questions, but for now it seems proper to end this discussion with a short fantasy that links the concepts of information and creation (and so animation).

Ev straightened and nodded to Reyn, then moved to a position beside the switch that would complete the contact when he threw it. The switch that would connect, all at once, all of the monster computing machines of all the populated planets in the universe, ninety-six billion planets, into the supercircuit that would connect them all into one supercalculator, one cybernetics machine that would combine all the knowledge of all the galaxies.
Ev threw the switch, ..., stepped back and drew a deep breath. "The honour of asking the first question is yours, Reyn."
"Thank you," said Reyn. "It should be a question which no single cybernetics machine has been able to answer."
He turned to face the machine. "Is there a God?"
The mighty voice answered without hesitation.
"Yes, NOW there is."

Sudden fear flashed on the face of Ev. He leaped to grab the switch.
A bolt of lightning from the cloudless sky struck him down and fused the switch shut.

From *Answer*
Frederic Brown (1954) [20]

Bibliography

1. Allbeck, J., and N. Badler, "Toward Representing Agent Behaviours Modified by Personality and Emotion," *Proceedings of First Intl Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, July 2002.
2. Arya, A., and B. Hamidzadeh, "ShowFace: A Framework for Personalized Face Animation," *Proceedings of IEEE/EURASIP RichMedia-2003*, Lausanne, Switzerland, October 16-17, 2003.
3. Arya, A., and B. Hamidzadeh, "FIX: Feature-based Image Transformations for Face Animation," *Proceedings of IEEE Intl Conf on Information Technology: Research and Education, ITRE-2003*, Newark, NJ, USA, August 10-13, 2003.
4. Arya, A., and B. Hamidzadeh, "Face Animation: A Case Study for Multimedia Modeling and Specification Languages," *Multimedia Systems and Content-based Image Retrieval*, S. Deb editor, pp 356-375, IGP, 2003.
5. Arya, A., and B. Hamidzadeh, "Personalized Face Animation in ShowFace System," *Int. Journal of Image and Graphics, Special Issue on Virtual Reality and Virtual Environments*, vol. 3, no. 2, pp 345-363, World Scientific Publishing, April, 2003.
6. Arya, A., and B. Hamidzadeh, "An XML-Based Language for Face Modeling and Animation," *Proceedings of IASTED Intl Conf on Visualization, Imaging and Image Processing, VIIP-2002*, pp 1-6, Malaga, Spain, 2002.
7. Arya, A., and B. Hamidzadeh, "ShowFace MPEG-4 Compatible Face Animation Framework," *Proceedings of IASTED Intl Conf on Computer Graphics and Imaging, CGIM-2002*, pp 32-37, Kauai, Hawaii, 2002.
8. Arya, A., and B. Hamidzadeh, "TalkingFace: Using Facial Feature Detection and Image Transformations for Visual Speech," *Proceedings of IEEE Intl Conf on Image Processing, ICIP-2001*, pp 943-946, Thessaloniki, Greece, 2001.
9. Arya, A., and B. Hamidzadeh, "A Personalized Talking/Moving Head Presentation Using Image-based Transformations," *Proceedings of Intl Conf on Augmented Virtual Environments and 3D Imaging*, pp 232-235, Mykonos, Greece, 2001.

10. Ankeney, J., "Non-linear Editing Comes of Age," *TV Technology*, May 1995.
11. Arafa, Y., et al., "Two approaches to Scripting Character Animation," *Proceedings of First Intl Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, July 2002.
12. Badler, N. I., "Animation 2000+," *IEEE Computer Graphics and Applications*, vol. 20, no. 1, pp 28-29, January 2000.
13. Battista, S., et al., "MPEG-4: A Multimedia Standard for the Third Millennium," Part 1 and 2, *IEEE Multimedia*, vol. 6, no. 4, pp 74-83, and vol. 7, no. 1, pp 76-84, October 1999 and January, 2000.
14. Beauchemin, S.S., and J. L. Barron, "The Computation of Optical Flow," *ACM Computing Surveys*, vol. 27, no. 3, pp 433-466, September 1995.
15. Beier, T., and S. Neely, "Feature-Based Image Metamorphosis," *Proceedings of ACM SIGGRAPH*, pp 35-42, 1992.
16. Besl, P. J., and R. C. Jain, "Three Dimensional Object Recognition," *ACM Computing Surveys*, vol. 17, no. 1, pp 75-145, March 1985.
17. Blanz, V., and T. Vetter, "A Morphable Model For The Synthesis Of 3D Faces," *Proceedings of ACM SIGGRAPH*, pp 187-194, 1999.
18. Bonamico, C., et al., "A Java-based MPEG-4 Facial Animation Player," *Proceedings of Intl Conf Augmented Virtual Reality& 3D Imaging*, pp 335-338, European Project INTERFACE IST Press, 2001.
19. Bregler, C., et al., "Video Rewrite: Driving Visual Speech with Audio," *ACM Computer Graphics*, pp 353-360, 1997.
20. Brown, F., "The Answer," *Inside Information (Computers in Fiction)*, short stories selected by Abbe Mowshowitz, Addison-Wesley, 1977.
21. Bulterman, D., "SMIL-2," *IEEE Multimedia*, vol. 8, no. 4, pp 82-88, October 2001.
22. Bulwer, J., *Pathomyotamia, or, A dissection of the significtive muscles of the affections of the minde*, Humphrey and Moseley, London, 1649.
23. Burt, P., and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans on Communications*, vol. 31, no. 4, pp 532-540, April 1983.
24. Cassell, J., et al., "BEAT: the Behavior Expression Animation Toolkit," *Proceedings of ACM SIGGRAPH*, pp 477-486, 2001.

25. Chen, S., and L. Williams, "View Interpolation for Image Synthesis," *Proceedings of ACM SIGGRAPH*, pp 279-288, 1993.

26. Chung, S., and J. K. Hahn, "Animation of Human Walking in Virtual Environments," *Proceedings of IEEE Conf Computer Animation*, pp 4-15, 1999.

27. Cohen, M., et al., "Image-Based Rendering: Really New or Deja Vu," Panel, *ACM SIGGRAPH*, 1997.

28. Corkie, A. D., and S. Isard, "Optimal Coupling of Diphones," *Progress in Speech Synthesis*, Springer, 1997.

29. Cosatto, E., and H. P. Graf, "Sample-Based Synthesis of Photo-Realistic Talking Heads," *Proceedings of IEEE Conf Computer Animation*, pp 103-110, 1998.

30. Damer, B., et al., "Putting a Human Face on Cyberspace: Designing Avatars and the Virtual Worlds They Live In," Panel, *ACM SIGGRAPH*, 1997.

31. Darbandi, H., *Speech Recognition & Diphone Extraction for Natural Speech Synthesis*, M.Sc. Thesis, University of British Columbia, February, 2002.

32. Darwin, C., *Expression of the Emotions in Men and Animals*, John Murray, London, 1872.

33. De Carolis, B., et al., "APML, a Markup Language for Believable Behaviour Generation," *Proceedings of First Intl Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, July 2002.

34. DeCarlo, D., et al., "An Anthropometric Face Model using Variational Techniques," *Proceedings of ACM SIGGRAPH*, pp 67-74, 1998.

35. DiPaola, S., "FaceSpace: A Facial Spatial-Domain Toolkit," *Proceedings of Info Viz-02*, 2002.

36. DiPaola, S., and D. Collins, "A 3D Virtual Environment for Social Telepresence," *Proceedings of Western Computer Graphics Symposium, Skigraph-02*, Silverstar, BC, Canada, 2002.

37. Dutoit, T., *An Introduction to TTS Synthesis*, Kluwer Academic Publishers, 1994.

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

39. ElNasr, M. S., et al., "Emotionally Expressive Agents," *Proceedings of IEEE Conf Computer Animation*, pp 48-57, 1999.

40. Escher, M., et al., "Facial Deformations for MPEG-4," *Proceedings of IEEE Conf Computer Animation*, pp 56-62, 1998.

41. Ezzat, T., and T. Poggio, "MikeTalk: A Talking Facial Display Based on Morphing Visemes," *Proceedings of IEEE Conf Computer Animation*, pp 96-102, 1998.

42. Fua, P., "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, no. 6, pp 35-49, 1993.

43. Funge, J., et al., "Cognitive Modeling: Knowledge, Reasoning, and Planning for Intelligent Characters," *Proceedings of ACM SIGGRAPH*, pp 29-38, 1999.

44. Gammuri, M., and A. Coglio, "An Architecture for Emotional Agents," *IEEE Multimedia*, vol. 5, no. 4, pp 24-33, October 1998.

45. Ghosal, S., "A Fast Scalable Algorithm for Discontinuous Optical Flow Estimation," *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp 181-194, February 1996.

46. Graf, H.P., et al., "Face Analysis for the Synthesis of Photo-Realistic Talking Heads," *Proceedings of IEEE Conf Automatic Face and Gesture Recognition*, pp 189-194, 2000.

47. Guenter, B., et al., "Making Faces," *Proceedings of ACM SIGGRAPH*, pp 55-66, 1998.

48. Haralick, R. M., and L. G. Shapiro, *Computer and Robot Vision*, vol. 2, Addison-Wesley, 1993.

49. Hirose, M., "Image-based Virtual World Generation," *IEEE Multimedia*, vol. 4, no. 1, pp 27-33, January 1997.

50. Hirzalla, N., et al., "A Temporal Model for Interactive Multimedia Scenarios," *IEEE Multimedia*, vol. 2, no. 3, pp 24-31, Fall 1995.

51. Hong, P., et al., "Real-Time Speech-Driven 3D Face Animation," *Proceedings of Intl Symp 3D Data Processing Visualization and Transmission,* 2002.

52. Kallmann, M., and D. Thalmann, "A Behavioral Interface to Simulate Agent-Object Interactions in Real Time," *Proceedings of IEEE Conf Computer Animation*, pp 138-146, 1999.

53. Kalra, P., et al., "SMILE: A Multi-layered Facial Animation System," *IFIP WG 5.10*, 1991.

54. Kass, M., et al., "Snakes: Active contour models." *Intl Journal of Computer Vision*, vol. 1, no. 4, 1987.

55. Kim, M, et al., "Extensible MPEG-4 Textual Format (XMT)," *Proceedings of ACM Conf Multimedia*, pp 71-74, 2000.
56. Kleijn, W. B., et al., *Speech Coding and Synthesis*, Elsevier Publishers, 1995.
57. Kouadio, C., et al., "Real-Time Facial Animation based upon a Bank of 3D Facial Expressions," *Proceedings of IEEE Conf Computer Animation*, pp 128-136, 1998.
58. Lande, C., and G. Francini, "An MPEG-4 Facial Animation System Driven by Synthetic Speech," *Proceedings of IEEE Conf Multimedia Modeling*, pp 203-212, 1998.
59. Lawton, G., "Video Streaming," *IEEE Computer*, vol. 33, no. 7, pp 120-122, July 2000.
60. Lee, G. S., "A General Specification for Scene Animation," *Proceedings of IEEE Conf SIBGRAPHI*, pp 216-223, 1998.
61. Lee, S., et al., "Polymorph: Morphing Among Multiple Images," *IEEE Computer Graphics and Applications*, vol. 18, no. 1, pp 58-71, January 1998.
62. Lee, W. S., et al., "MPEG-4 Compatible Faces from Orthogonal Photos," *Proceedings of IEEE Conf Computer Animation*, pp 186-194, 1999.
63. Lengyel, J., "The Convergence of Graphics and Vision," *IEEE Computer*, vol. 31, no. 7, pp 46-53, July 1998.
64. Lin, D., and H. Huang, "Facial Expression Morphing and Animation with Local Warping Methods," *Proceedings of IEEE Conf on Image Analysis and Processing*, pp 594-599, Sept. 1999.
65. Little, T.D.C., "Time-based Media Representation and Delivery," *Multimedia Systems*, J.F. Koegel Buford (ed), ACM Press, 1994.
66. Ljolje, A., et al., "Automatic Speech Segmentation for Concatenative Inventory Selection," *Progress in Speech Synthesis*, Springer 1997.
67. Macchi, M., "Issues in Text-to-Speech Synthesis," *Proceedings of IEEE Intl Conf Intelligent Systems*, pp 318-325, 1998.
68. Mann, N., and L. Syson, *The Image of the Individual, Portraits in the Renaissance*, The Trustees of British Museum, 1998.
69. Manning, R. A., and C. R. Dyer, "Interpolating View and Scene Motion by Dynamic View Morphing," *Proceedings of IEEE Conf Computer Vision and Pattern Recognition*, vol. 1, pp 394-400, 1999.

70. Manske, K., and R. Rudisch, "Comic Actors Representing Software Agents," *Proceedings of IEEE Conf Multimedia Modeling*, pp 213-222, 1998.

71. Marriott, A., and J. Stallo, "VHML: Uncertainties and Problems. A discussion," *Proceedings of First Intl Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, July 2002.

72. Mayache, A., et al., "A Comparison of Image Quality Models and Metrics," *Proceedings of IEEE Intl Conf Image Processing, ICIP-98*, vol. 3, pp 409-413, 1998.

73. Metaxas, D., et al., "Vision-based Animation of Digital Humans," *Proceedings of IEEE Computer Animation, CA-98*, pp 144-152, Philadelphia, 1998.

74. Myszkowski, K., "Perception-Based Global Illumination, Rendering, and Animation Techniques," *Proceedings of ACM Conf Computer Graphics*, pp 13-24, Budmerice, Slovakia, 2002.

75. Nack, F., and A.T. Lindsay, "Everything You Wanted To Know About MPEG-7," *IEEE Multimedia*, vol. 6, no. 3, pp 65-77, July 1999.

76. Noma, T., et al., "Design of a Virtual Human Presenter," *IEEE Computer Graphics and Applications*, vol. 20, no. 4, pp 79-85, July 2000.

77. Ohya, J., et al., "Virtual Metamorphosis," *IEEE Multimedia*, vol. 6, no. 2, pp 29-39, April 1999.

78. Ostermann, J., "Animation of Synthetic Faces in MPEG-4," *Proceedings of IEEE Conf Computer Animation*, pp 49-55, 1998.

79. Pandzic, I. S., "A Web-based MPEG-4 Facial Animation System," *Proceedings of Intl Conf Augmented Virtual Reality & 3D Imaging*, pp 323-326, 2001.

80. Parke, F. I., and K. Waters, *Computer Facial Animation*, A. K. Peters, 1996.

81. Pesce, M. D., *Programming Microsoft DirectShow for Digital Video and Television*, Microsoft Press, 2003.

82. Pighin, F., et al., "Synthesizing Realistic Facial Expressions from Photographs," *Proceedings of ACM SIGGRAPH*, pp 75-84, 1998.

83. Piwek, P., et al., "RRL: A Rich Representation Language for the Description of Agent Behaviour in NECA," *Proceedings of First Intl Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, July 2002.

84. 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.

85. Quek, F., et al., "A Multimedia Database System for Temporally Situated Perceptual Psycholinguistic Analysis," *Multimedia Tools and Applications*, vol. 18, no. 2, pp 91-113, Kluwer Academic Publishers, 2002. Also as VISLab Report: VISLab-00-03.

86. Rabiner, L. R., and R. W. Schaffer, *Digital Processing of Speech Signal*, Prentice-Hall, 1978.

87. Ram, A., et al., "PML: Adding Flexibility to Multimedia Presentation," *IEEE Multimedia*, vol. 6, no. 2, pp 40-52, April 1999.

88. Reeves, W. T., "Simple and Complex Facial Animation: Case Studies," *State of the Art in Facial Animation, ACM SIGGRAPH-90 Course Notes #26*, 1990.

89. Rowland, D. A., and D. I. Perrett, "Manipulating Facial Appearance through Shape and Color," *IEEE Computer Graphics and Applications*, vol. 15, no. 5, pp 70-76, September 1995.

90. Seitz, S. M., and C. R. Dyer, "Physically-Valid View Synthesis by Image Interpolation," *Proceedings of IEEE Workshop Presentation of Virtual Scenes*, pp 18-25, 1995.

91. Seitz, S. M., and C. R. Dyer, "View Morphing," *Proceedings of ACM SIGGRAPH*, pp 21-30, 1996.

92. Sturman, D. J., "Computer Puppetry," *IEEE Computer Graphics and Applications*, vol. 18, no. 1, pp 38-45, January 1998.

93. Takeda, K., et al., "User Interface and Agent Prototyping for Flexible Working," *IEEE Multimedia*, vol. 3, no. 2, pp 40-50, Summer 1996.

94. Terzopoulos, D., and K. Waters, "Analysis and Synthesis of Facial Image Sequences Using Physical and Anatomical Models," *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp 569-579, June 1993.

95. Terzopoulos, D., et al., "Facial Animation: Past, Present and Future," Panel, *ACM SIGGRAPH*, 1997.

96. Thalmann, N. M., and D. Thalmann, "Digital Actors for Interactive Television," *Proceedings of of IEEE*, 1995.

97. Tiddeman, B., et al., “Prototyping and Transforming Facial Textures for Perception Research,” *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp 42-50, September 2001.

98. Valente, S., and J. L. Dugelay, "Face Tracking and Realistic Animation for Telecommunicant Clones," *IEEE Multimedia*, vol. 7, no. 1, pp 34-43, January 2000.

99. Vetter, T., et al., "A Bootstrapping Algorithm for Learning Linear Models of Object Classes," *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp 40-46, 1997.

100. Wey, J. D., and J. A. Zuffo, "InterFace: a Real Time Facial Animation System," *Proceedings of IEEE Conf SIBGRAPHI*, pp 200-207, 1998.

101. Whitford, F., *Expressionist Portraits*, Thames and Hudson, London, 1987.

102. Williams, L., "Performance-Driven Facial Animation," *ACM Computer Graphics*, pp 235-242, August 1990.

103. Wolberg, G., *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1990.

104. Yacoob, Y., and L. S. Davis, "Recognizing Human Facial Expressions From Long Image Sequences Using Optical Flow," *IEEE Trans Pattern Analysis and Machine Intelligence*, pp 636-642, June 1996.