Real-Time Interactive Facial Animation

Marc Escher
Gael Sannier
Nadia Magnenat-Thalmann

MIRALab
University of Geneva
24 rue General-Dufour
CH-1211 Geneve 4
SWITZERLAND

Marc.Escher@cui.unige.ch,
Gael.Sannier@cui.unige.ch,
Thalmann@cui.unige.ch

Abstract

In this paper we describe methods for the generation of real-time facial animation for various virtual actor using high level-actions. These high-level actions allow the user to forget the technical side of the animation, and focus only on the more abstract, more natural, and intuitive part of the facial animation. The mechanisms developed to generate interactive high-level real-time animation are described and details are given on the method we use to blend several action to a smooth facial animation. These techniques have been integrated in a C++ library which functionalities are described. We integrate the result in an interactive real-time application. This application allows a user-friendly control of virtual humans' facial animation and speech in real-time. It provides tools for an interactive creation of virtual stories or storyboard.

Keywords: Virtual Human, Virtual Face, Real Time Facial Animation, MPEG-4

Introduction

The face is a relatively small part of a virtual human, but it plays an essential role in communication. We look at faces for clues about a person's emotions or even to read lips. It is a particular challenge to simulate these aspects. The ultimate objective is, therefore, to model human facial anatomy exactly, including its movements, with respect to both structural and functional aspects. Developments in facial animation include surface-based facial deformation [Park96], [Nahas88], [Wait89] or physically based approximation to facial tissue [Magnenat-Thalmann88], [Terzopoulos90], [Platt81], [Waters87] for better simulation of muscle contractions of human facial expressions. In the context of interactive animation systems, the relationship between the user (animator) and the virtual human (acting as a synthetic actor) should be emphasized. With the availability of graphics workstations powerfull enough to display complex scenes rendering about a million polygons per second, it is possible to create computer-generated characters based on a complete interaction in which deformations or motion specifications are given in real-time [Thorisson97], [Cassell94]. When producing, in real-time, animation of virtual heads, it is often necessary to use pre-defined high-level actions. These actions represent usual expressions and emotions. The complexity of these high-level actions may vary from a simple eye closed to a more sophisticated emotion such as fear or happiness. In order to produce real-time lip synchronization with speech, a set of actions that produce the mouth deformation corresponding to the articulation of phonemes is also necessary. The following sections detail how we interactively animate faces using high-level actions.

Hierarchy of actions

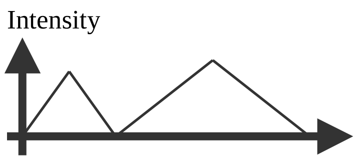
A multi-layered approach is used in our application [Kalra92], where at each level of the hierarchy, the degree of abstraction increases. It offers the advantage of providing the user with a relatively low level of complexity with concepts more global than the previous layers. For instance, complex expressions such as basic emotions or even speech can then be described more easily in this multi-layered system.
In our whole system we can distinguish 4 levels of abstraction (Fig. 1).

High-level actions	Description of the Intensity over time
Mid-level	Definition of movements
Low-level	Muscle or point based action units
Deformation	Mesh deformation

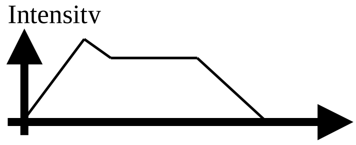
Figure 1 Levels hierarchy and effects

High-level actions

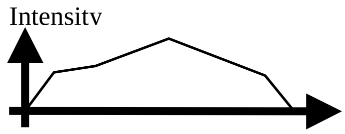
The purpose of this level is to allow an animator to work on a task level where she/he can define animation in terms of its inherent sources. Three types of high-level actions are defined, using three different envelopes describing the evolution of the deformation of a face through time. (Fig. 2) Starting from a neutral state, passing through a sequence of visible changes, and returning to a neutral state. It is important to start from a neutral position and end at a neutral one, this guaranties a smooth animation of the face without discontinuities. When the action starts there is no jump to the starting position, and when an action is finished, it does not leave the face in an unknown position from which there will be a jump to the starting position of the next action.



a) Triangle shape: For punctual short timed actions such as eye blinking or phonemes.



b) Attack – Decay – Sustain - Release shape: Suited for every natural expressions with a big emphasis at it's start such as surprise or smile.



c) Free shape: For any other type of movements one can think of.

Figure 2 Three available envelopes of high-level actions

In other terms a high-level action is a description of how to animate a static expression (defined in the mid-level) through time.

Mid-level

A mid-level action is a snapshot of the face, a static position/expression that has been defined by the user. (Fig. 3)

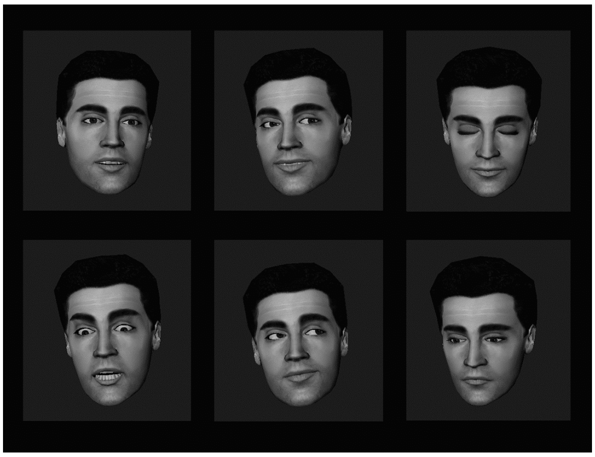


Figure 3. Mid-level expressions.

The user has interactively set a number of values to the desired low-level action units to get the desired expression. They are generally composed/defined manually using 3D modeling softwares although they can also be extracted more automatically with the help of external devices such as cameras [Terzopoulos93], [Blake94], [Pandzic94] or 3D scanners.[Kleiser89], [Lee95]

Low-level

Low-level actions are considered as action units. They are composed of a fixed set of parameters, each one describing a local deformation of the face (region on the face mesh). They all are placed on the face in order to allow every deformation humanly possible. Two type of action units have been implemented:

- **Minimum Perceptible Actions (MPA):** which is our proprietary model of action units. They are region based, each region corresponding to the location of a specific muscle or an articulated organ (eyes, teeth, jaw). They are driven by intensities. (Fig 4)

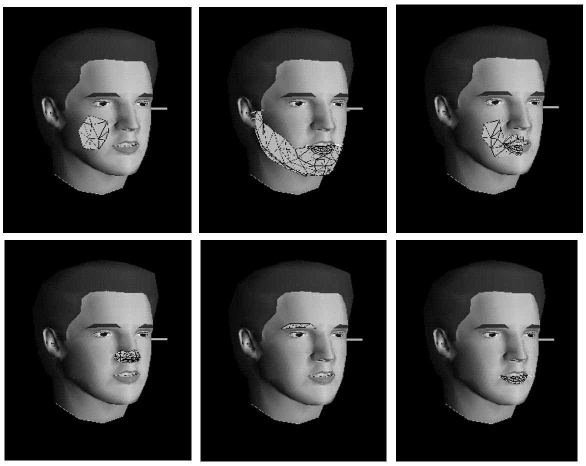


Figure 4. Regions location.

- **Facial Animation Parameters (FAP):** They have been defined in the MPEG-4 standard. They are represented as a set of feature points located at morphological places on the face. They uses 3D translation to specify the mesh deformation. (Fig. 5)

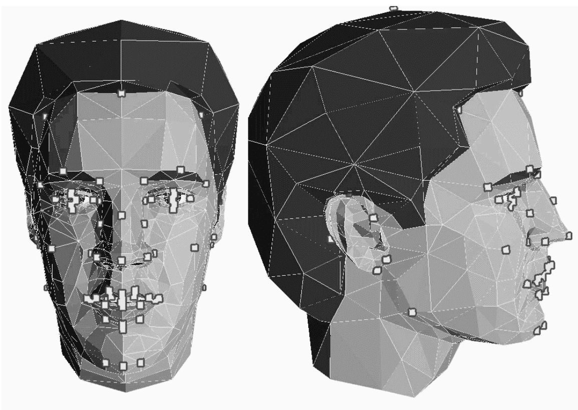


Figure 5. FAP location.

Deformation level

Getting into detail about the deformation level is not the purpose of this paper, as all the concepts exposed in it can be implemented using any kind of deformation method. The following high-level animation library has been used as a top layer to our two kinds deformation methods driven by MPAs and FAPs. In our case the deformation of the mesh is implemented using Free Form Deformations.

FaceAgentLib, a high-level facial animation library

In order to animate efficiently several faces, we have built up a C++ library that allows the user to associate a number of high-level actions to a specific head. These high-level actions are activated at the time the user decides, to produce a corresponding real-time facial animation.

At the activation time, the duration, intensity and weight of the action are specified. The intensity correspond here to the amplitude of the deformation, while the weight correspond to the “importance” of the action when combined with another one. This raises a key problem, which is the blending between several high-level actions to produce a smooth animation.

Action blending

The blending of actions problem surges when several high-level actions are active at the same time and are setting each a value to the same “action-unit” at the same moment. If it is not handled with care it can produce discontinuities in the animation. This is particularly important if we want to get a natural facial animation. A discontinuity can happen at the activation or at the termination of each high-level action when combined with another one. The solution we have chosen is to add to each high-level action a weighting curve, which minimizes the contribution of each high-level action at its activation and at its end. (Fig. 6) The weighting curve values are generally set between 0 and 1.

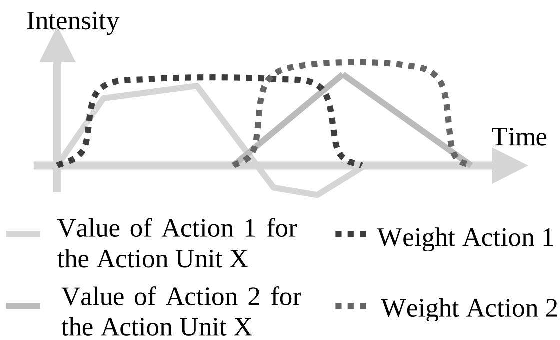


Figure 6. Weight curves

The composition formula used to compute the value of an action-unit at a specific time **t** modified by **n** high-level actions is given by Eq. 1.

$$V_{Final} = \frac{\sum_{i=1}^{n} Weight_{i,t} \cdot Intensity_{i} \cdot ValueActionUnit_{i,t}}{\sum_{i=1}^{n} Weight_{i,t}}$$

Equation 1. Action-unit composition formula

Where *Intensity* is the global intensity of the high-level action **i** and *ValueActionUnit* is the action unit intensity of the high-level action **i** at time **t**

This blending function is acting only at the action-unit level and only when several high-level actions are setting values to the same action-unit.

Functionalities of the library

The library is composed of all the functions described previously and layered as explained before. This library meets the necessary to an easy animation of a virtual heads:

- Completely graphic-library independent: The library is written in C++ and does not include any reference to any external graphical structure such as OpenGL, Performer or Inventor.
- Read high-level actions: All the animation of heads is done through high-level actions. It should simplifies the animation task by making it more intuitive and natural.
- Support multiple heads: Within one session, all the tools are available to manipulate and animate several heads.
- Enables global manipulation of high-level actions: The user can decide the starting time of a high-level action, its duration, intensity and weight.
- Allows interruption of high-level actions: At any time the user can decide to stop an activated high-level action. He can also choose to fixe a decay time during which the face will go down to neutral position.
- Allows reactivation of interrupted actions: After a high-level action has been stopped it is possible to resume its animation.
- Output frames of MPAs or FAPs: The user can choose either to use the proprietary MIRALab MPA parameters to animate the virtual face or use the newly defined MPEG-4 FAPs.

These functions are sufficient to enable the construction of an interactive facial animation software. It does not include the synchronization method. If you want to make lip synchronization you will have to add a process that select the time-right frame generated by the library. Neither does it integrate any facial deformation or generates any display. It is up to the user to integrate the results of the library (MPAs or FAPs) in his own facial rendering software.

Internal Structure

Some words about the representation of the datas in the library. The internal structure of the library within several dynamic links dedicated to each declared heads. Each head has two dynamically created lists of high-level actions. The first list contains, associated to each high-level action, the envelope and a list of action-units values corresponding to the face deformation. The second list is composed of all the activated high-level actions and their associated list of action-units frames. These frames are computed using the duration, envelope, weight and intensity values of the corresponding high-level action. Every frame has a list of values of action-units and a weighting value. Figure 7 presents a graphical representation of these two lists.

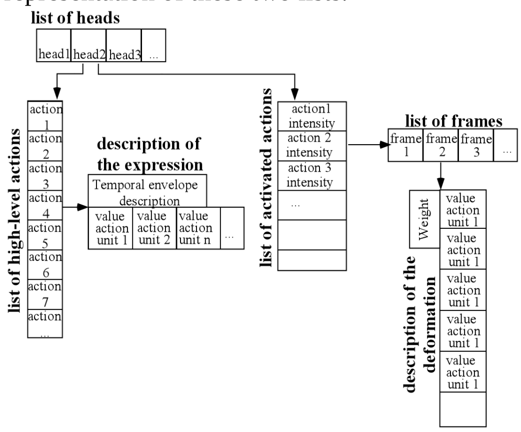


Figure 7. Internal structure of the library

Use and integration in an interactive application

This section presents an interactive interface for directing virtual humans (VHD) based on the high-level facial animation library presented in this paper. This software integrates several components and focuses on two important issues:

- *Fully integrated virtual humans with facial, speech and body animation.*
- *A straightforward user interface for designers and directors.*

VHD system provides a range of virtual human animation and interaction capabilities integrated into one single environment. Its software architecture allows the integration of various interfaces from distinct environments into the software. VHD actors can be controlled via a standard TCP/IP connection over any network by using the virtual human message protocol

Speech

The high-level facial animation system can be easily extended to the speech by associating audio output with the facial animation. In order to generate the speech automatically, the facial animation must be driven by the audio input. Each sentence is first decomposed into phonemes, to each phoneme corresponds a high-level action viseme, which is the phoneme counterpart in terms of facial deformation. 44 visemes are used for the Standard English language.

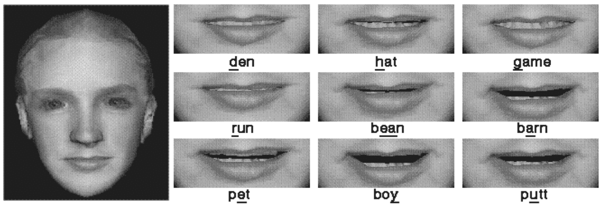


Figure 8. A virtual head model with its corresponding visemes for the indicated words.

Figure 8 shows examples of visemes. From the animation point of vue, a sentence is a sequence of temporized visemes. Nevertheless, during a conversation, a human mouth is not able to articulate precisely each phoneme: Using the weighting function available in the library, each visemes are blended together through time. To smooth the animation, the beginning of a viseme is blended with the end of the previous one. In Figure 9, we have displayed the generation of the word “hello” with the overlapping of each visemes over its neighbors and their corresponding weighting curve. One can observe that not every weighting curve has the same maximum, since some visemes must be more articulated thus allowing a more humanlike lip motion. Plosives phonemes for instance require the complete closure of the mouth.

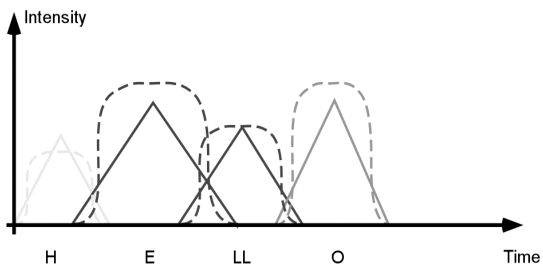


Figure 9. High-level action composition to articulate “HELLO”

We implemented this speech methodology in a graphical text-to-speech program. The input text is being converted into temporized phonemes using a text-to-speech synthesis system. In this case we are using the Festival Speech Synthesis System developed at the University of Edinburg [Black97]. It generates also the audio stream that will be subsequently playback in synchronization with the facial animation system. Using temporized phonemes, we are able to create the facial animation by concatenating the corresponding visemes through time. We are using the set of phonemes present in the Oxford English Dictionary. We can extend our model to any language by redefining the set of visemes. As a case study, we also extended our model to German language.

The two processes are launched simultaneously to synchronize the face movement with the sound. In order to keep the synchronization, sound playback is used as a time reference for the facial animation during the speech. By knowing beforehand the total length of the sound, the number of frames of the animation (given by the temporized visemes), and the current time, the synchronization is done easily by skipping frames in case of delay in the rendering process of the animation.

Interface design issues

In previous sections, we described a system for animating virtual humans with real-time constraint. In this section, we present how interactivity is integrated in the Virtual Human Director system (VHD).

The issue of easy control of a virtual face was raised in [Magnenat-Thalmann98][1]. VHD interface provides an extension of this concept through simple and easy control of multiple actors and cameras in real-time. The idea is to provide tools for interactive creation of virtual stories and storyboard. A key aspect is that the user can trigger actions in real-time, similar to a producer directing actors during a shooting but with much more since everything is decided and controlled by the same person. To make this complicated task possible and useable, the virtual actors have to be controlled by sets of high-level actions. Tools were developed to be able to predefine activation of actions in advance so that during the real-time playing, the producer can concentrate on the main guidelines of the scenario (sentences for instance).

Only high-level actions can be selected within the interface. The speech is activated by typing/selecting simple text-based sentences. The facial animation is controlled by pre-recorded sequences. These facial animations can be mixed in real-time and also mixed with the speech. Because we can mix together the facial animations, a simple set of facial animations is enough to generate the main usual facial expressions of a human's face. In our experiments, we are using a set of fifteen expressions. For a specific use (virtual twitch for example), new facial expressions can be designed easily. Control of the body is done through keyframing and motion motors for the walking.

As soon as we have many virtual actors to control, the number of available actions will make the task of the director more and more complex. In order to ease this complicated task for the real-time interaction, we provide several tools for pre-programming actions in advance. The user has the ability to set a delay between activation through the interface of an action and its activation on the virtual actors. This is very useful in case of repeated actions such as eye blinking. However all actions are not required to be programmed this way. Since the goal is to be able to play scenario in real-time, the control of the main actions must be in the hands of the director. Nevertheless, a lot of actions may result from a few main events. For example, sentences of a dialogue can be used as guidelines for other actions the specific behavior of the virtual actor. The user can predefine this behavior at the same time as the sentence is typed. Then, just by selecting main actions, complex behaviors of the virtual actors will be completely pre-determined. The user will however still be able to mix other actions to the pre-programmed ones.

In order to improve the creation of virtual stories, a scripting tool was developed for recording actions applied to an actor. Each action is recorded with its activation time. The user can then adjust the timing within each sequence and play it again in real-time. During playback, new actions can be also triggered. This way, an incremental methodology can be used to create complex behavior and interaction of virtual humans. This provides the appealing ability to completely program background actors while only important actors will be directed "live".

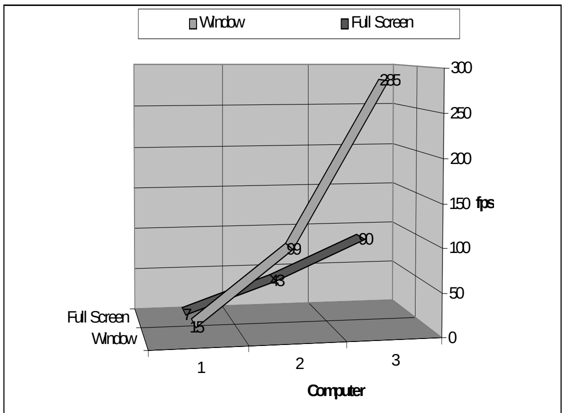
VHD results

We tested our software extensively within the European project VISTA together with broadcasting partners, to produce TV programs. After several tests with designers and directors, our concept of one single integrated platform for virtual humans has obtained good feedbacks. Our interface is a very straightforward concept and highest level of control allows designers to control virtual humans in a natural fashion. Figure 10 shows the interface controlling the virtual actors and the resulting 3D animation. We included several traditional camera operations into our interface to allow directors to use VHD without a steep learning curve.

Real-time performances

We tested VHD with a close-up to a virtual head made of 2313 triangles and fully textured. Table 1 presents the results of our trials on the following computers using full screen (1280x1024) and a 400x500 pixels window:

1- O2 R5000 at 200MHz
2- Octane R10000 at 195 MHz



	1	2	3
Window	15	99	285
Full Screen	7	43	90

3- Onyx2 2x R10000 at 195MHz

Table 1: Number of frames per second in VHD.

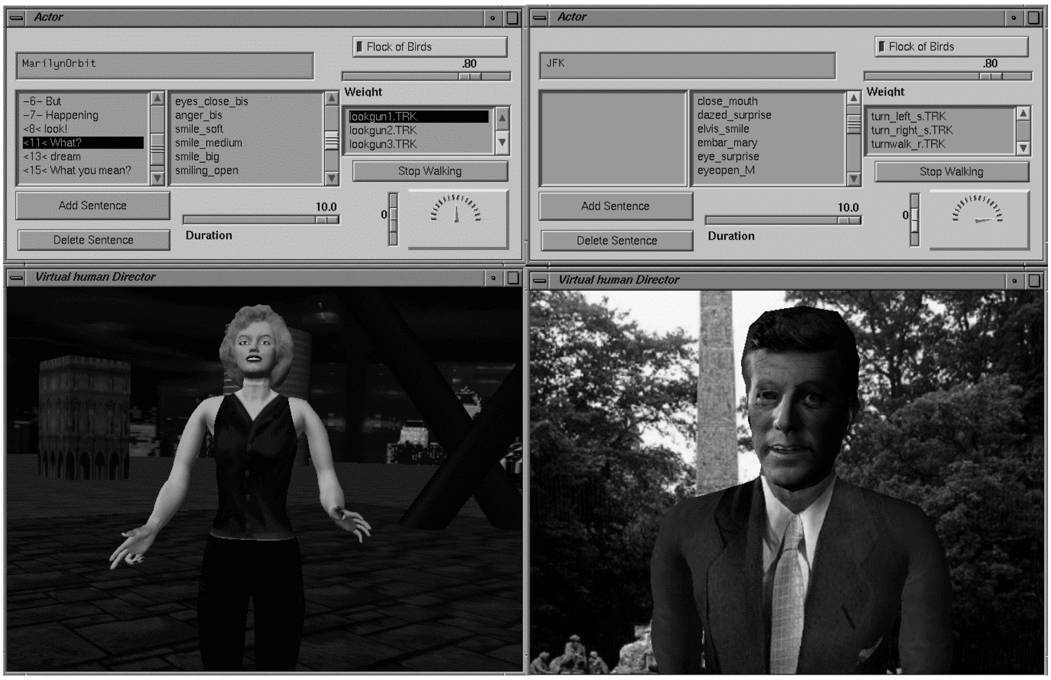
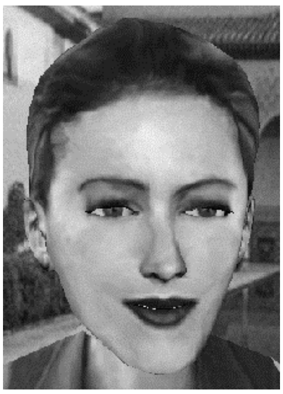


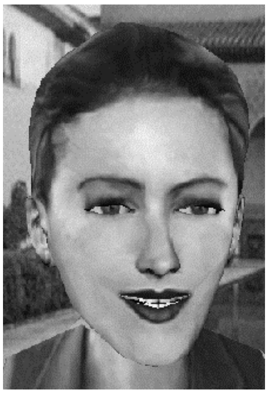
Figure 10. Examples of interface controlling virtual actors.

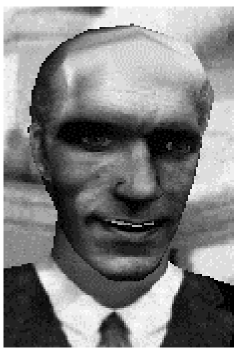
This case study presents how VHD can be used to make interactive events. The first stage is compilation of the scenario. According to the scenario, the production team records necessary predefined body gestures, facial animations and sentences. After this, the first trials may start with VHD. The director generates specific camera shots and checks all parts of the scenario. The director and the production team record and edit complete scenario with the scripting capabilities of VHD. Some complex parts of a real-time show can be recorded as a script to have a perfect synchronized results

Results

The following pictures shows various snapshots of synthetic faces animated using the methods described before.





Conclusions and future work

We have shown how our multi-layered animation system can be used to easily generate a facial animation driven by high-level actions. We have described the advantages and functionalities of this library. The description of the method developed to generate smooth blending between several high-level actions has also been provided. The library was used in an interactive application. This application allows the real-time animation of multiple synthetic actors driven by high-level actions.

The porting to other platform such as NT machines, the improvement of the speech coarticulation, and the development of some simpler synchronization tools is envisioned as further improvement of the library.

Acknowledgements

I will like to thank all the staff of MIRALab for their help and contribution to this work. This research was supported by the European projects VISTA and PAVR, and the Swiss National Foundation for Scientific Research.

References

A. Black, P. Taylor (1997) Festival Speech Synthesis System: System Documentation (1.1.1), Technical Report HCRC/TR-83, Human Communication Research Center, University of Edinburgh.

Blake A, Isard M (1994), “3D position, attitude and shape input using video tracking of hands and lips”, *Computer Graphics (Proc. SIGGRAPH ‘94)*, 28:185-192, July 1994.

Cassell J, Pelachaud C, Badler NI, Steedman M, Achorn B, Becket T, Deouville B, Prevost S, Stone M, “Animated Conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents”, *Proc. SIGGRAPH ‘94*, pp. 413-420.

Kalra P, Mangili A, Magnenat Thalmann N, Thalmann D (1992), “Simulation of facial muscle actions based on rational free form deformations”, *Proc. Eurographics ‘92*, pp. 59-69.

Kleiser J (1989), “A fast, efficient, accurate way to represent the human face”. In *State of the Art in Facial Animation, SIGGRAPH ‘89 Tutorial*, Vol. 22, pp. 37-40.

Lee Y, Terzopoulos D, Waters K (1995), “Realistic modeling for facial animation”, *Computer Graphics (Proc. SIGGRAPH ‘95)*, 29(4):55-62.

Magnenat Thalmann N, Primeau NE, Thalmann D (1988), “Abstract muscle actions procedures for human face animation”, *The Visual Computer*, 3(5):290-297, 1988.

N. Magnenat Thalmann, P. Kalra, M. Escher (1998) Face to Virtual Face, in Proc. of the IEEE. Vol.86, No. 5, May

Nahas M, Huitric H, Sanintourens M (1988), “Animation of a B-spline figure”, *The Visual Computer*, 3(5):272-276, March 1988.

Pandzic IS, Kalra P, Magnenat Thalmann N, Thalmann D (1994), “Real time facial interaction”, *Displays* 15 (3):157-163.

Parke FI, Waters K (1996), *Computer Facial Animation*, A K Peters, Wellesley, Massachusetts, USA.

Platt SM, Badler NI (1981), “Animating facial expressions”, *Computer Graphics (Proc. SIGGRAPH ‘81)*, 15(3):245-252, 1981.

Terzopoulos D, Waters K (1990), “Physically based facial modeling, analysis, and animation”, *The Journal of Visualization and Computer Animation*, 1(4):73-80.

Terzopoulos D, Waters K (1993), “Analysis and synthesis of facial image sequences using physical and anatomical models”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):569-579, 1993.

Thorisson Kristinn R (1997), “Layered modular action control for communicative humanoids”, *Proc. Computer Animation ‘97*, IEEE Computer Society, pp. 134-143.

Waite CT (1989) *The Facial Action Control Editor, FACE: A parametric facial expression editor for computer generated animation*, Master’s Thesis, MIT, Media Arts and Sciences, Cambridge, MA, Feb. 1989.