PCS 156 1997

Parameter Based Animation of Arbitrary 3D Head Models

Erich Haratsch[1], Technical University of Munich, erich@lis.e-technik.tu-muenchen.de
Jörn Ostermann, AT&T Labs Research, osterman@research.att.com

1. INTRODUCTION

Parameter based facial animation allows the convincing generation of facial expressions at a very low bitrate. This makes it very suitable for talking head applications where the channel capacity is highly limited and an exact reproduction is not imperative, like virtual meeting rooms in the Internet, or where the animated head does not represent a human being (e. g. networked human-machine-interface).

Today's facial animation systems use a generic head model which is fit to the geometric 3D data of an individual head in the setup phase [1]. The animation rules, which describe how each animation parameter deforms the generic model, is hard coded into the animation system and strongly related to the used generic model. This makes the animation system very inflexible: it cannot animate other head models (e. g. of other topology, higher resolution, etc.) without major changes. Hence, only proprietary solutions are available today.

In this contribution, we describe a new model-independent animation architecture, which allows to download an arbitrary head model and the information how to animate it. During the initialization, the encoder sends a VRML 2.0 head model and an animation table, which specifies the animation rules for the animation parameters, to the decoder. During animation, the receiver decodes the animation parameter stream and animates the downloaded head model according to the rules.

Since the model is defined by the encoder, no head fitting at the decoder has to be performed. Furthermore, this approach allows the encoder a precise control over the decoder enabling applications like virtual representatives where it is important not to distort visual information like attitudes or moods. Downloading a head model to the decoder does not slow down animation speed at the decoder.

2. MODELING THE HEAD FOR ANIMATION

2.1 Modeling in VRML 2.0

VRML 2.0 [2] already defines a file format for the modeling of 3D objects in virtual worlds. This file format can be used to define the shape, texture and transformation hierarchy of the head model in a scene graph.

The topology of the scene graph is given by the transformation hierarchy and the existence of different texture maps for single parts of the head.

Two node classes in VRML 2.0 are of particular interest for facial animation: *Transform* and *IndexedFaceSet* nodes: *Transform* nodes allow the definition of rigid transformations like translation, rotation and scaling. Whenever possible this node should be used for the definition of an animation as it takes full advantage of the hardware accelerated graphics rendering engine.

If the animation of an object includes flexible deformations, an *IndexedFaceSet* node has to be used to describe this object. This object is then animated by replacing the changed coordinate positions with the new values.

Figure 1 gives an example of a simplified scene graph for a head model. The head consists of 3 objects. The face and the eyes. As the face has to undergo flexible deformations during animation, it is modeled in an *IndexedFaceSet* node. The eyes are modeled with *Squere* nodes. *Transformation* nodes specify their rigid transformations. A *Transformation* node at the top of the scene graph is used to control global head movements (rotation, nodding, etc.).

2.2 Including Animation Related Information in the Scene Graph

The animation system decodes the incoming animation parameter stream and determines the update information for *IndexedFaceSet* and *Transform* nodes. Therefore it must be able to access these nodes unambiguously. This can be done by assigning them labels with the DEF statement in VRML 2.0.

3. DEFINING RULES FOR THE ANIMATION PARAMETERS IN AN ANIMATION TABLE

The animation table defines for each animation parameter, which nodes are animated by it and how.

[1] Erich Haratsch performed this work while he was a consultant to AT&T Labs Research

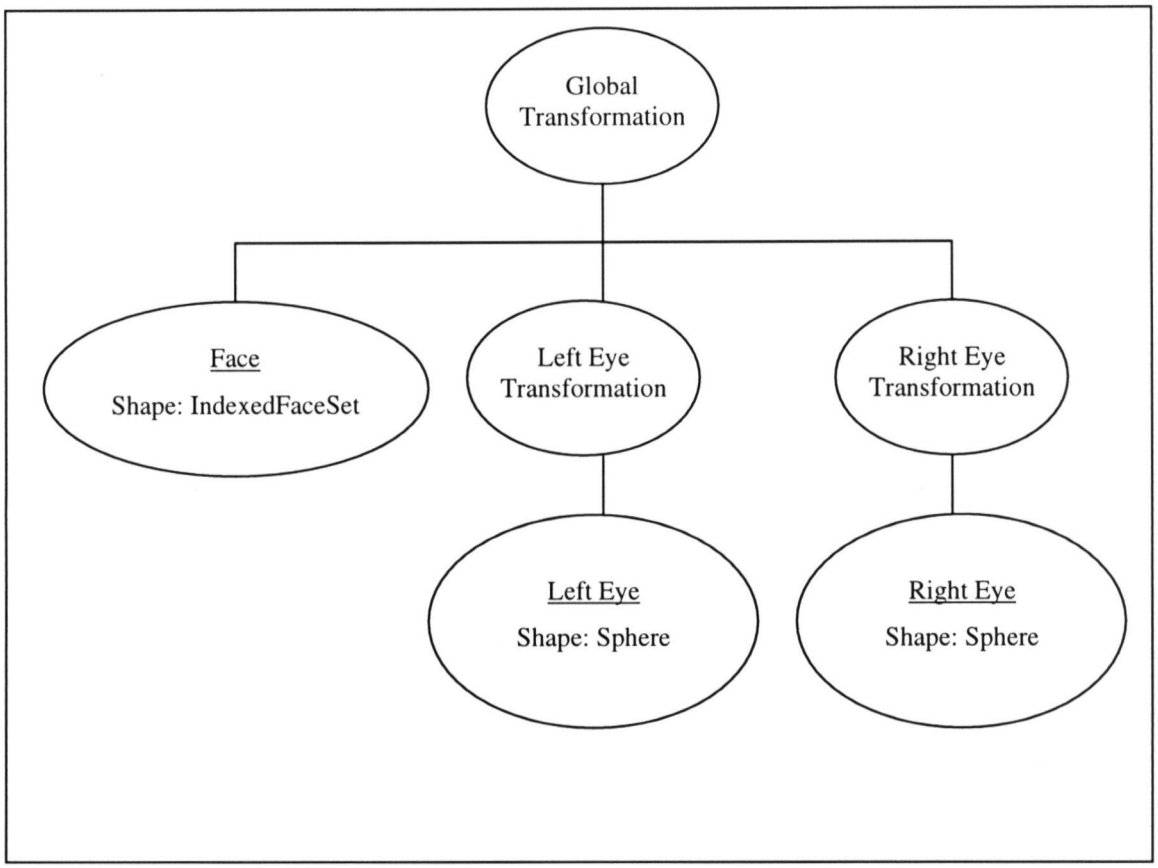

Figure 1: Scene Graph for a VRML Description of a Simple Head Model.

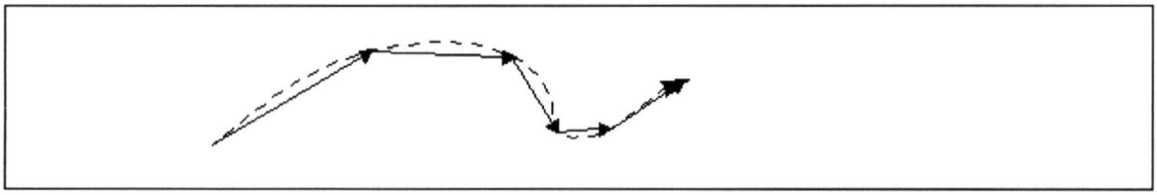

Figure 2: Piecewise-linear Approximation of a Complex Motion.

3.1 Transform nodes

If an animation parameter causes an update of an *Transform* node, the animation table has to specify, which field in the *Transform* node is updated by this animation parameter. Then, during animation, the value of the animation parameter is directly inserted into this field.

3.2 IndexedFaceSet nodes

Flexible Deformations of *IndexedFaceSet* nodes can be approximated by piecewise-linear approximation (Figure 2).

The animation table lists for an *IndexedFaceSet* node, which is deformed by a given animation parameter, the interval borders, affected vertices and displacement vectors (Table 1). During animation, the decoder determines for a received value of this animation parameter, in what interval it is lying, and piecewise-linearly interpolates the motion of the affected vertices.

3.3 Example of a Simple Animation Table

To make our concept of animation tables clearer, we give a simple example, in which the rules for two animation parameters are defined (Table 2). The animation parameter 1 is specified such that the values for it during animation replace the rotation field of the *Transform* node for the left eye. Animation Parameter 2 causes a flexible deformation of the *IndexedFaceSet* describing the face: The positions of vertex 23 and vertex 58 have to be

determined by linear interpolation. Two interpolation intervals have been specified.

Table 1: Animation Table Format for *IndexedFaceSet* Nodes.

Vertex no.	*1st Interval [I_1, I_2]*	*2nd Interval [I_2, I_3]*	.....
Index 1	Displacement D_{11}	Displacement D_{12}	...
Index 2	Displacement D_{21}	Displacement D_{22}	...
...	...	...	...

Table 2: Example of an Animation Table.

#FacialAnimationTable
Animation Parameter 1:
Transform node: left eye, field: rotation.
Animation Parameter 2:
IndexedFaceSet node: face.

	]-∞;0]	[0;∞[
Vertex 23	(-0.9; 0.1; 0.1)	(0.8; -0.1, -0.1)
Vertex 58	(-0.6; 0.2; 0.6)	(0.6; -0.1; -0,2)

4. CONTENT CREATION

How can arbitrary head models in VRML and animation tables be created? There are two ways: a natural for personalized head models and a synthetic one.

In the natural approach a VRML model of a person's head is created by cyberscan data. The animation table is generated by image analysis. Images of the person are taken in neutral state and for each facial expression corresponding to the different animation parameters. The method described in [3] could be used to calculate the displacement vectors for *IndexedFaceSet* nodes. Applying this algorithm for different intensities of the person's expression improves the realism of the facial movements during animation. An animation system which downloads a person's data obtained in this way could be seen as a new architecture for a primitive model-based decoder.

We have implemented the synthetic approach in our animation system [4]. We use professional 3D modeling software to create a head model in VRML. Then we deform the head for each animation parameter in different intensities and save the resulting head in a VRML file. An API reads the VRML head model in neutral state and all the deformed states and generates automatically the animation table. This procedure has been used to define the entire set of MPEG-4 Facial Animation Parameters [5] for AT&T's talking head system. The head model downloading and animation functionality integrate nicely with the talking capability of the system. Animated sequences using different personalities will be shown at the symposium. (Figure 3, 4)

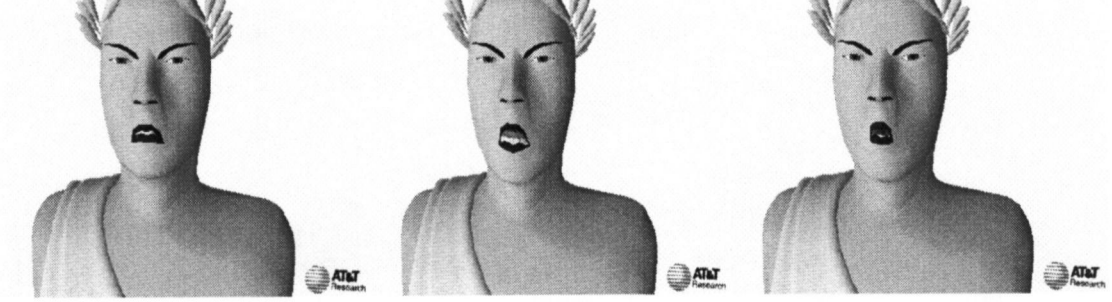

Figure 3: Visual Speech with a newly designed head and shoulder model.

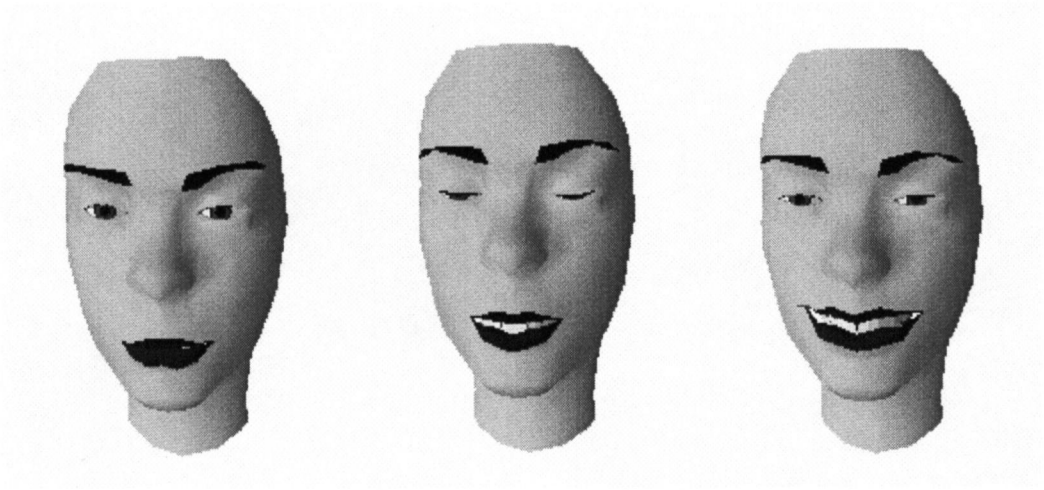

Figure 4: Animation of a downloaded face model with the MPEG-4 FAP test sequence "Marco".

5. CONCLUSIONS

We described a new animation architecture, which allows to download a head model and the definition of its animation parameters. After this initialization phase, the encoder sends animation parameters to the decoder. The decoder executes the animation parameters according to their downloaded definition. It is obvious that this approach allows for a significant flexibility for animation: The head can be human, animal or an artificial object, like a logo. Furthermore, this approach can be extended towards the animation of arbitrary objects.

6. REFERENCES

[1] Lawrence S. Chen, Jörn Ostermann, "Animated Talking Head with Personalized 3D Head Model", to appear in Proceedings of 1997 Workshop on Multimedia Signal Processing, June 23-25, 1997, Princeton, USA.

[2] "The Virtual Reality Modeling Language", ISO/IEC DIS 14772-1, April 4, 1997, http://www.vrml.org/VRML97/DIS/.

[3] L. Tang, T. S. Huang, "Quantifying Facial Expressions: Smiles", International Workshop on Very Low-Bitrate Video, Proceedings, Paper No. 2.2, April 7 - 8, 1994, Essex, UK.

[4] Jörn Ostermann, Erich Haratsch, "An Animation Definition Interface - Rapid Design of MPEG-4 Compliant Animated Faces and Bodies", submitted to The International Workshop on Synthetic-Natural Hybrid Coding and 3D Imaging, September 5-9, 1997, Rhodes, Greece.

[5] "Facial Animation Parameter Set", in: "SNHC Verification Model 4.0", ISO/IEC JTC1/SC29/WG11 N1666, April 1997, Bristol, UK.

Parameter-Based Model-Independent Animation of Personalized Talking Heads

Jörn Ostermann[1], Erich Haratsch[2]

[1]AT&T Labs - Research, Room 3-231, 100 Schultz Dr., Red Bank, NJ, 07701, email: ostermann@research.att.com

[2]Institut für Integrierte Schaltungen, Technische Universität München, Germany, email: erich@lis.e-technik.tu-muenchen.de

submitted to IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY

Abstract

A system is presented that defines personalized talking head models to consist of three parts: The model in its neutral position, animation definition tables that define the spatial deformation in response to the amplitude of different animation parameters and animation definition tables that define the temporal behavior of the model for actions like eye blinks and nodding but also for facial expressions. A process for quickly creating these models using modeling software is described. Since the models contain the knowledge of their behavior, the renderer for animating these models is easy to implement. In the prototype system, the models are animated using model-independent animation parameters as well as a speech synthesizer. Part of this proposal is implemented within the MPEG-4 visual, audio and systems standard.

Keywords: Talking head, face animation, avatar, VRML, MPEG-4, text-to-speech synthesizer, virtual reality

1 Introduction

Human Computer Interfaces is an application area where audio, text, graphics, and video are integrated to convey various types of information. Often conversion is necessary between different media [1]. The objective is to provide more natural interactions between the human user and the computer. One approach is to display an animated character or a life-like talking head on the computer screen. The characters should have the ability to receive input from the

user and respond in a natural and intelligent way. Using the character as a visual interface to an agent, such an agent may perform information retrieval, notification, reading email or replying to email messages.

Already available are software programs that display a generic animated talking head or a cartoon animal character on the screen to perform various tasks. One program is able to fetch songs at the user's request [2]. Another is able to carry on simple conversations [3]. Yet others are able to convert written text into visual speech using a Text-To-Speech (TTS) synthesizer [4] and a synchronized talking head with realistic lip and jaw movements as well as visible teeth and tongue[5][6][7][8][9]. Whereas some programs use simple polygon models with and without texture maps for animation, others use warped samples of 2D images [11]. Some animation programs consider models of the human muscles and skin in order to generate realistic facial motion[12][13]. A common feature of these programs is that the animation rules for the models are integrated into the animation program and not into the model.

Since facial animation becomes feasible in modern computers and settop boxes, MPEG-4 is also standardizing the facial animation parameters (FAP) that allow animating talking heads. With this standard becoming available, it seems desirable to provide MPEG-4 decoders with the capability to animate different talking head models without knowing details of the facial model itself. In this paper, we propose to extend the definition of a facial model to not only include the static 3D shape of the model but also the knowledge of the model's dynamic behavior like smiles, head motion, and eye blink. With this new definition of a talking head model, the animation program does not need any knowledge about the dynamic of the model or the topology of the model. It will also be possible to use the MPEG-4 high-level FAPs to animate facial expressions like joy, fear, disgust, surprise as well as visemes (mouth shapes corresponding to phonemes). Furthermore, we could animate with one FAP stream a group of talking heads and they all would preserve their own personalities.

In Section 2, we describe application scenarios where talking heads will be required to create or to improve a service. The personalized talking head models including their static and dynamic properties as well as a process for creating them is presented in Section 3. Section 4 describes the architecture for an implementation of a talking head system. Since part of this proposal was adopted by MPEG-4, Section 4 further outlines the differences between the proposed

system and what MPEG-4 will cover. In Section 5, experimental results of animated personalized face models are presented.

2 Application Scenarios

Talking head systems can be used as visual enhancements to human computer interfaces, customer service as well as games where the user wants to control artificial characters. Depending on the application, the talking head system will not only provide the visualization of a talking head but also of an upper torso or an entire body.

2.1 Human Computer Interface

Visual interfaces helping to enhance the look and feel of a program are becoming more and more standard. 10 years ago, Apple envisioned the future computer interface to have speech recognition, speech synthesis and a butler-like animated computer character interacting with the user. Today, we see the first steps into this direction. Speech recognition and speech synthesis software is coming to the desktop together with enhanced video and graphics capabilities. Microsoft office shows a small avatar that reacts visually to human interaction with the program, other applications provide video clips with a news speaker providing an introduction into the software or a recorded human at a help desk.

Users may want to select their own talking head as their interface to the computer. We expect the talking head to talk, smile, and have its own personality. It will be used as the interface with which the operating system and other applications like mail, calendar manager, answering machine, software agents, etc. interact with the user.

2.2 Virtual Representative

Some applications want to provide their own talking head as the visual interface to the computer user. This might become very important in web-based customer service, where a company not only uses it's own logo but also it's own animated character to create a corporate identity. It is critical that the talking head system behaves on the user terminal as designed by corporate marketing.

In virtual environments, talking heads will represent people. For interactive applications, the owner can control the avatar. For non-interactive applications or for applications where the

control of the avatar has to be at a high level with commands like *walk*, show *joy* or *sadness*, the talking head should behave according to the rules laid out by its creator.

As far as the talking head, perhaps the most important part of the virtual character, is concerned, we foresee two different environments: The user buys his/her own talking head or uses the one that is part of the operating system. For some applications a specialized talking head like a virtual company representative is downloaded to the computer in order to provide a customized interface. For both environments, animation parameters used to animate the character should be independent of the model itself; i.e. the talking head knows how to speak, laugh, blink with the eyes and how to move its head. This allows animating different models with just one parameter stream containing high-level commands. Assuming that the models are also allowed to have autonomous behavior like eye-blinks or head movements, one parameter stream can animate a group of heads and the group would act jointly, but still every character would act different to the others. We call these models *personalized models*.

2.3 Visual Text-To-Speech Synthesizer

The prerequisite for animating personalized models with a single parameter stream is a clean separation between properties of a model and functions of the renderer. Figure 1 shows the concept of a versatile Visual Text to Speech System (VTTS) that allows animating different personalized models, human like or avatars.

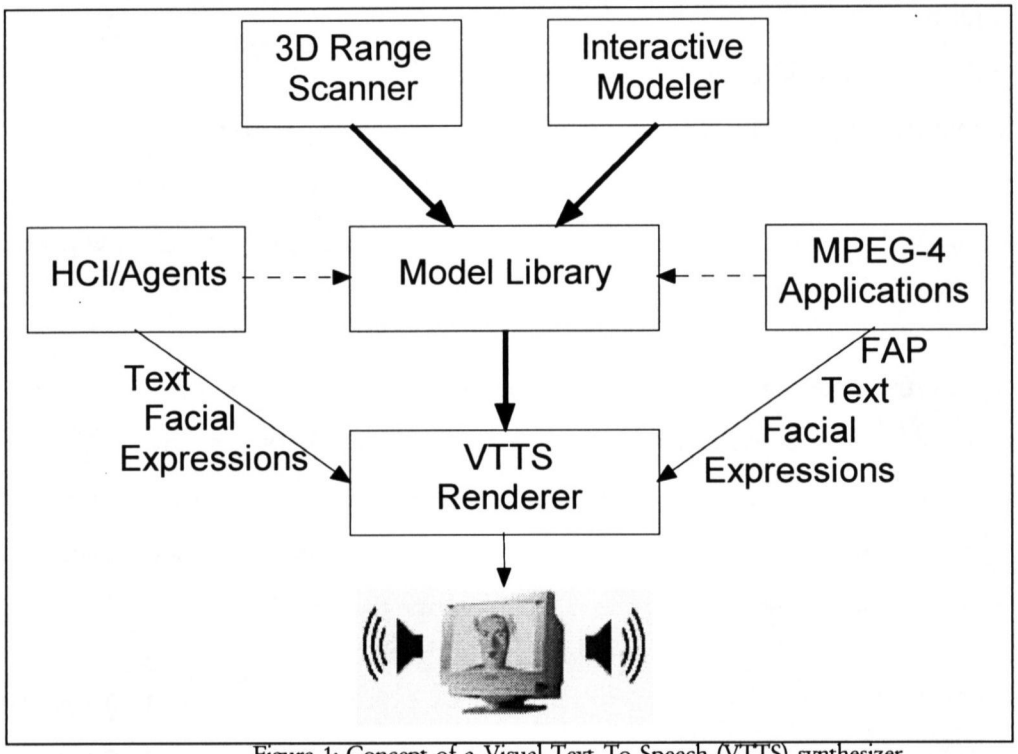

Figure 1: Concept of a Visual Text To Speech (VTTS) synthesizer with a library of personalized models controlled from an agent, a human computer interface (HCI) or an MPEG-4 terminal.

As can be seen from the diagram, a model library provides several models that can be read by the VTTS system. The application can be the human computer interface, an agent or an MPEG-4 communications terminal. The application selects a model from the library and sends it to the VTTS renderer. Depending on the application, this model can be described in VRML format [14] or in BIFS format [15] as defined by MPEG-4. After loading the model, the renderer can receive animation data from the application. Input is text and facial animation parameters like mimic or the FAPs as defined by MPEG-4 [16]. The renderer computes the animation of the face model and the TTS synthesizer creates the text that it receives from the application.

In sections 3 and 4, we describe personalized talking head models and the architecture of a renderer, respectively. They allow animation of different personalized models using just the same parameter stream.

3 Personalized Talking Head Models

3.1 Rapid Modeling and Animation Definition Architecture

We have designed and implemented an architecture that allows for the rapid modeling and animation definition of personalized talking heads (Figure 2) [8]. In the following it is assumed that head objects are modeled with polygonal surfaces. We implemented 2 methods for creating the talking head in its neutral position.

In the semi-automatic approach a 3D range finder captures the shape and texture information of a person's head. The model-fitting algorithm uses this data to adapt the shape of a generic head model to the range data (Section 3.2.1). In the manual approach, the 3D range data delivered by the 3D scanner or 2D images taken by a camera are imported into a professional modeler to conform a generic model to the range 3D shape (Section 3.2.2).

Based on the head in neutral position, different facial expressions according to the set of animation parameters are created and stored. An animation definition interface (ADI) evaluates the models and creates an animation definition table (ADT) that defines the behavior of the model. The neutral head model and the animation definition parameters define the personalized model (PM). The VTTS Renderer imports the PM for animation. It learns by reading the ADTs, how model independent animation parameters (MIAPs) animate this model.

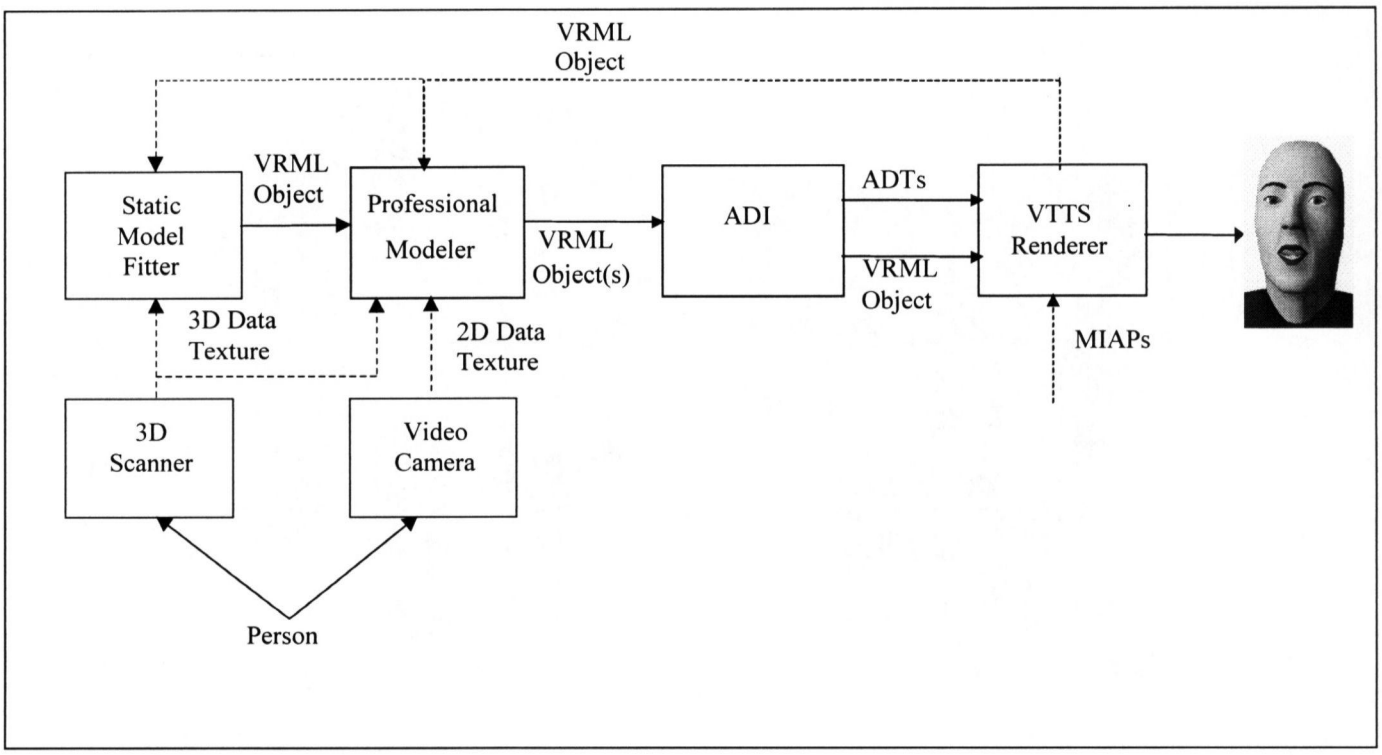

Figure 2: Rapid Modeling and Animation Definition Architecture

3.2 Defining the Static Model

The shape of the static 3D model can be defined in different ways. One method is to use real persons scanned by a range finder; alternatively, the model can be defined using standard modeling software.

3.2.1 Range Finder

As an example for defining the 3D model, we use the data from a 3D-laser range scanner [17]. Figure 3 shows the shape of a face and Figure 4 the texture map as captured at the time of recording the range data. After manually identifying the tip of the nose as a point of the vertical profile line on the range data, we adapt a generic face model (Figure 5) to the individual person represented by the range data. In a first step, several facial features along the profile line like chin, lower lip, upper lip, forehead, etc are identified. These features are used to vertically scale the generic head model to fit the distances between the feature points on the range data. In a second step, the face model is horizontally scaled such that it wrappes around the facial part of the range data. Figure 6 and Figure 7 show the adapted model. Similar

methods are described in [18][20]. This model is then used to create the animation definition tables as described in Section 3.3.

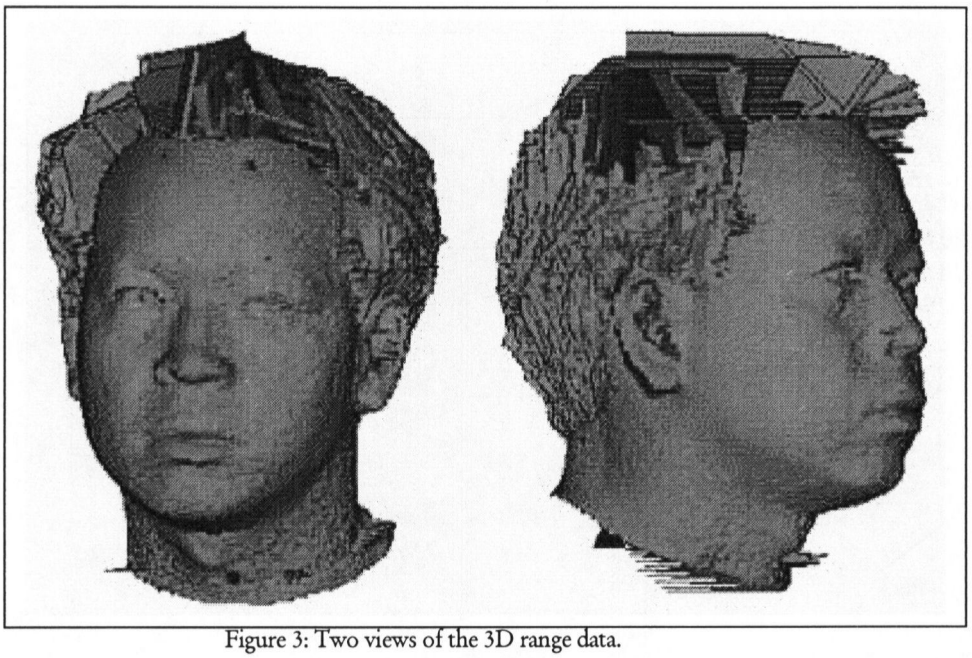

Figure 3: Two views of the 3D range data.

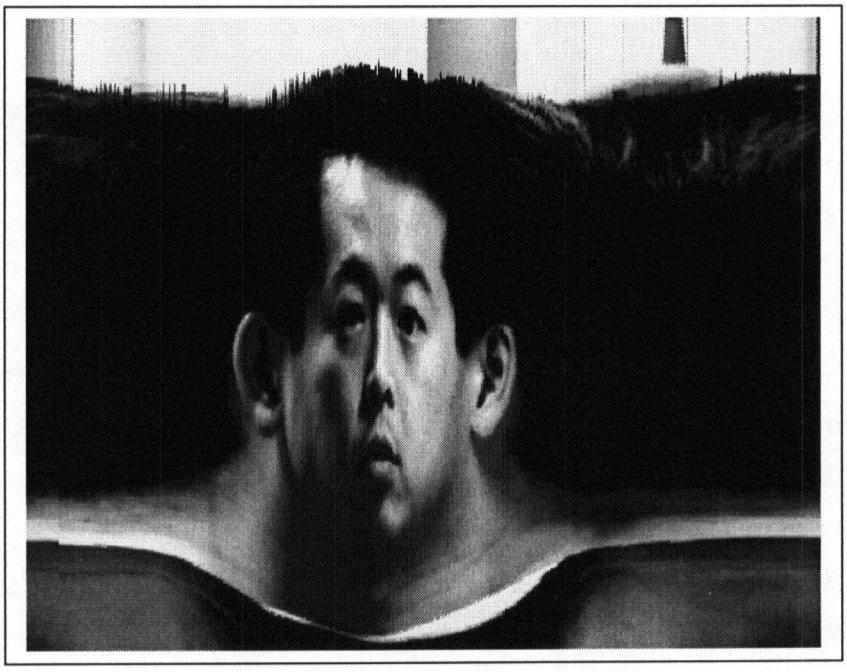

Figure 4: Texture map of the 3D range data shown in Figure 3.

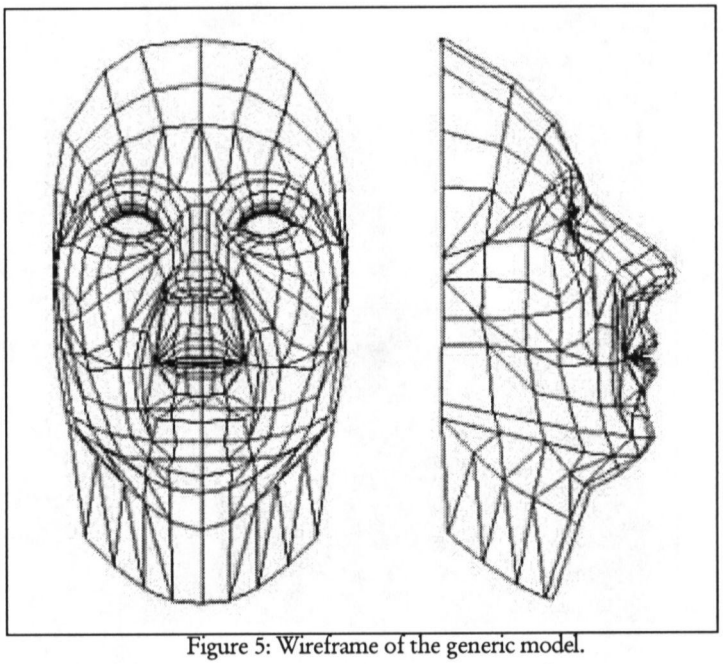

Figure 5: Wireframe of the generic model.

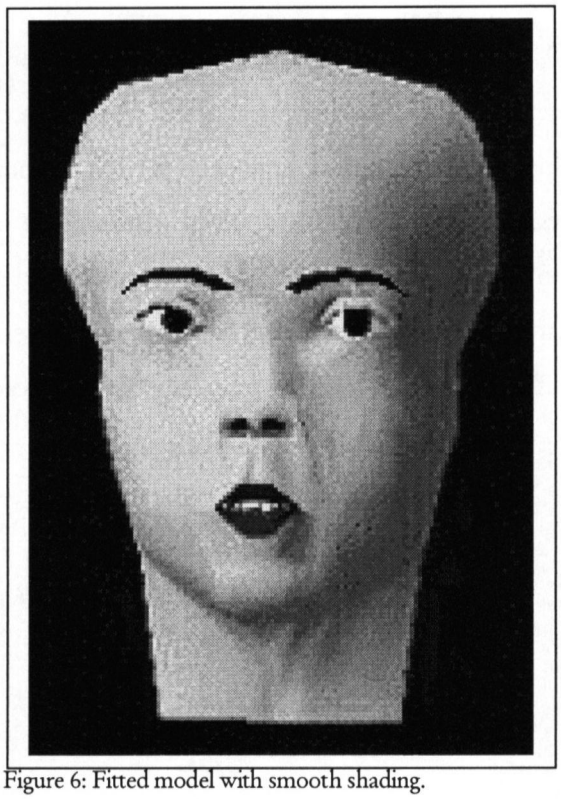

Figure 6: Fitted model with smooth shading.

Figure 7: Fitted model with texture mapping.

3.2.2 *Modeling Software*

The generic model of the VTTS renderer can be adapted to the shape of a real person's head in the professional modeler. In the 3D based approach, the range data of a scanner is therefore imported into the modeler. The 2D based approach works without expensive scanning equipment: a grid pattern corresponding to the topology of the generic model is painted onto the real person's head (Figure 8) [5]. Two pictures, of the front and side views, respectively, are taken with a camera and imported into the modeler. There, the generic model is adapted by using the grid pattern as a reference (Figure 9). A manually created head model with texture is shown in Figure 10.

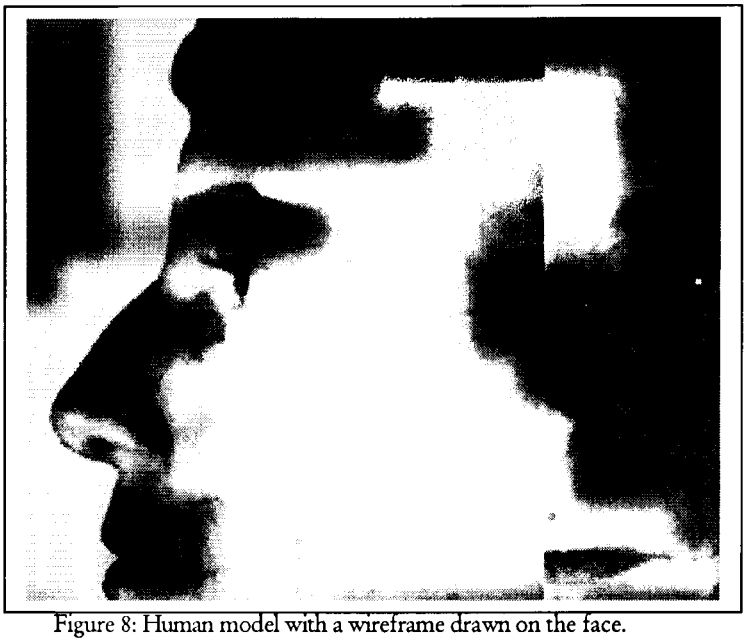

Figure 8: Human model with a wireframe drawn on the face.

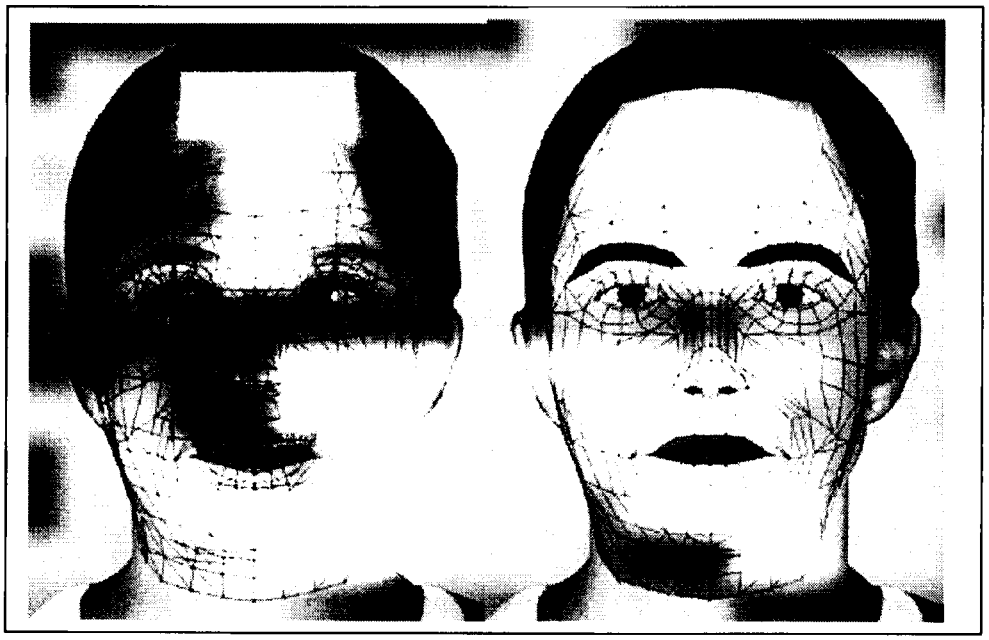

Figure 9: Photo of the human model with a wireframe and head model superimposed.

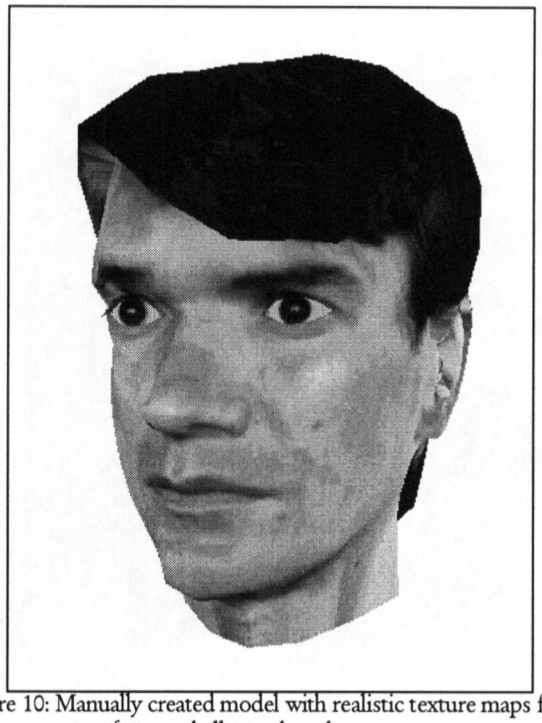

Figure 10: Manually created model with realistic texture maps for face, eyeballs, teeth and tongue.

3.2.3 Scenegraph Representation of the Static Model

The static personalized head model is defined in a VRML file. VRML describes geometrical scenes as a collection of objects, arranged in a scene graph. Three types of nodes are of particular interest for the definition of a static head model. A *Group* node is a container for collecting child objects: it allows for building hierarchical models. If objects are located in the same position in the scene, or if the objects move together as a group, they need to be in the same *Transform* group. The *Transform* node defines geometric affine 3D transformations like scaling, rotation and translation that are performed on its children. When *Transform* nodes contain other *Transforms*, their transformation settings have a cumulative effect. Nested *Transform* nodes can be used to build a transformation hierarchy. An *IndexedFaceSet* node defines the geometry and surface attributes (color, texture) of a polygonal object.

Figure 11 shows the simplified scene graph for a head object. Nested *Transforms* are used to apply rotations about the x, y, and z-axis one after another. Embedded into these global head movements are the rotations for the left and right eye. Separate IndexedFaceSets define the

shape and the surface of the face, hair, tongue, teeth, left eye and right eye, thus allowing for separate texture maps.

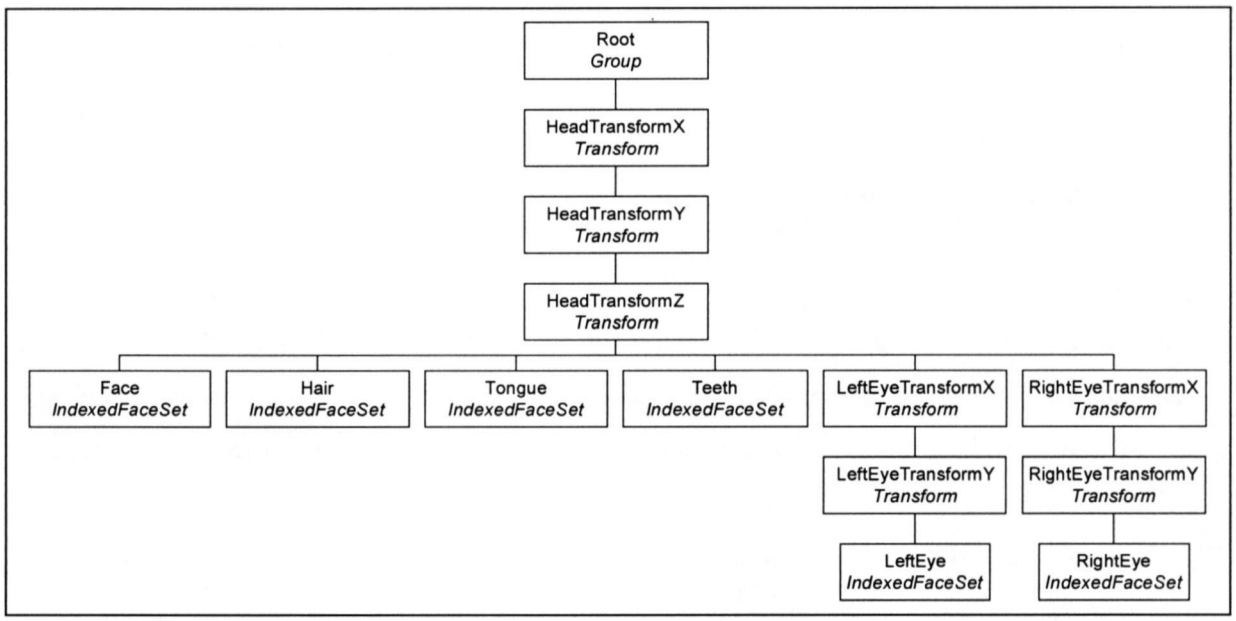

Figure 11: Simplified scene graph for a head model.

3.3 Defining the Dynamic Behavior

Personalized dynamic behavior must be defined in both the space and the time domain. The behavior in space defines how a model is deformed as a function of the amplitude of the facial animation parameters. The behavior in time defines how the object creates a time curve [25] for an animation given that the input to the model is just the length of the expression and the maximum amplitude. An example for such a request would be "Smile for 5 seconds".

3.3.1 Space

ADTs are read by the VTTS to learn how the MIAPs animate the PM. The ADT specifies, for a MIAP, which nodes are animated by it and how [10][23].

Animation Definition of a Transform node

If a MIAP causes solely a transformation like rotation, translation or scale, a Transform node can describe this animation. Whenever possible this node should be used for the definition of an animation as it takes full advantage of hardware accelerated graphics rendering engines. The ADT specifies the type of transformation and a neutral value for the chosen transformation. During animation, the received value for the MIAP and the neutral value determine the actual value.

Animation Definition of an IndexedFaceSet Node

If a MIAP causes flexible deformation (this is the case for most MIAPs), the animation results in updating vertex positions of the affected IndexedFaceSet nodes. Flexible deformations of an IndexedFaceSet are approximated by piece-wise linear motion trajectories. Therefore the ADT defines motion intervals for the motion trajectories, indices for the affected vertices, and 3D displacements (Table 1). When a value for the MIAP is received during animation, the VTTS calculates the new 3D position for the vertices specified in the ADT by simple linear superposition.

Example for an ADT

In Table 1, two MIAPs are defined by ADTs: MIAP 6, which stretches the left corner lip, and MIAP 23, which manipulates the horizontal orientation of the left eyeball.

ADT 6 deforms the IndexedFaceSet Face. Displacement vectors for two intervals are given to describe piece-wise linearly the motion of MIAP 6.

MIAP 23 updates the rotation field of the Transform node LeftEyeX. The neutral value is (0, -1, 0), and the neutral angle is 0 radians.

Table 1: Example of an Animation Definition Table

```
#AnimationDefinitionTables
MIAP 6 (stretch left corner lip)
        IndexedFaceSet: Face
                interval borders: -1000 0 1000
                displacements:
                        vertex 50  -1 0 0   1 0 0
                        vertex 51  -1 0 0  1 0 0
MIAP 23 (yaw left eye ball)
        Transform: LeftEyeX
                rotation   neutral value: 0 -1 0 (axis)  0 (angle)
```

Rapid Synthesis of ADTs

The architecture presented in Section 3.1 allows for the rapid synthesis of ADTs defining personalized MIAPs. To define a particular MIAP, the head model gets deformed in the professional modeler using its sophisticated tools. Then the head model is exported in one or more phases of the animation as separate VRML files and read by the ADI. The ADI compares the vertex positions of the deformed models with those of the non-deformed model and generates automatically the ADT. For linear movements, it is sufficient to export one deformed model. Non-linear deformations are approximated piece-wise linearly, which means that significant phases of the deformation must be exported. From our experiences, a maximum of three exported defined VRML head models are sufficient to generate an ADT with nice animation results for most non-linear animation parameters encountered in facial animation. Figure 12 shows 2 phases of an left eye blink (plus the neutral phase) which have been generated in the professional modeler.

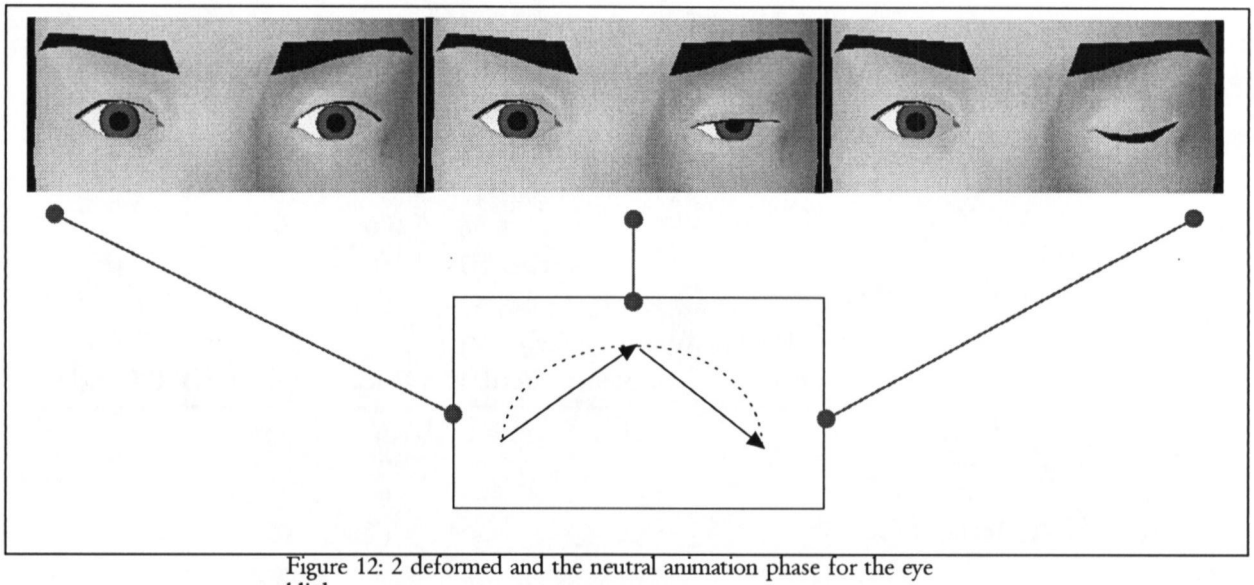

Figure 12: 2 deformed and the neutral animation phase for the eye blink.

3.3.2 Time

Combining ADTs with a personalized head model allows for personalized animation in the space domain, but to get full independence between personalized model and parameter stream also the time domain must be considered. Specifying temporal animation definition parameters that define how the personalized parameters control the model over time can do this. By including temporal animation definition parameters into the ADTs, a complete model independent parameter stream can be used to animate different models while maintaining their individual personality. A synthesized facial expression (like smiles) and non-emotional overlays (like eye blink, nodding) will not only have a different shape, but also have a personalized appearance from a temporal point of view. This capability does of course require a command language other than MPEG that defines the FAPs for each frame to be rendered. Since the personalized model has a time behavior it is sufficient to animate facial expressions by defining start, end and excitation during climax of this expression

4 Animation System for Personalized Talking Heads

First, we describe the architecture of our animation system. Then we compare it to the MPEG-4 architecture.

4.1 Architecture

Figure 13 shows the architecture for a real-time VTTS: The Text Analysis unit of the Text-To-Speech Synthesis module parses text into a timed sequence of phonemes and control parameters. This data is passed to the Speech Synthesis unit, which generates the synthetic audio. Before the animation, the Real-time Renderer imports the personalized head model and the ADTs that define how MIAPs animate it. During animation, two different mechanisms animate the head object. The speech animation is driven by the phonemes and control parameters passed by the Text Analysis unit. MIAPs for personalized emotional expression overlay (like joy, anger, etc.) are included as bookmarks in the text. The personalized model creates non-emotional overlays like eye blink and nodding itself. However, if bookmarks are found that animate this model specific behavior, the models acts according to the bookmarks.

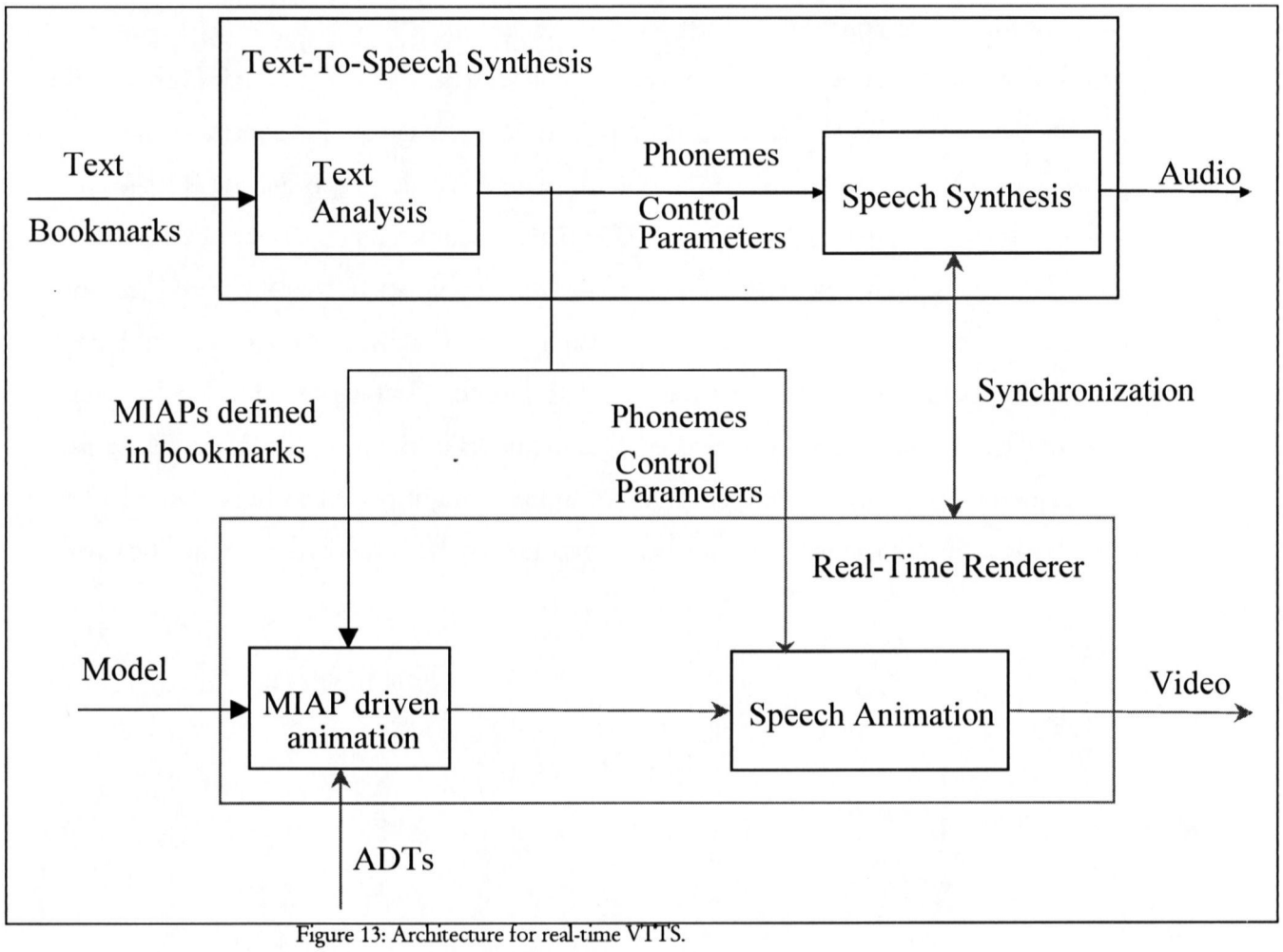

Figure 13: Architecture for real-time VTTS.

4.2 Comparison with MPEG-4

MPEG-4 mainly concentrated on defining a parameter set to animate faces. The facial animation parameters consist of 3 groups:

1. Facial Expression like joy, sadness (Figure 14).
2. Visemes for allowing the easy animation of talking heads from synthetic speech using a phoneme to viseme converter.
3. 66 low-level FAPs like move left mouth corner up.

The main emphasis was put on defining the low-level FAPs and using them to animate unknown facial models. In order to get somehow predictable results, the amplitude of the low-level FAPs are scaled according to distances of facial feature points in the animated face model. FAPs are coded at fixed or variable frame rate. In order to define the temporal behavior of a face model, FAPs are transmitted at a frame rate of up to 30Hz. The decoder is

not supposed to modify the transmitted parameters. However, FAPs can also be set to *interpolate* which means that the decoder can define their value at its leisure. Experiments were mainly carried out at a frame rate of 30 Hz.

As far as the actual face model is concerned, MPEG-4 foresees 3 decoder scenarios:

1. *Uncalibrated proprietary face model*: The encoder just sends FAPs to the decoder and hopes that the animation parameters give the desired effects. Due to the scaling of the parameters, attractive animations can be achieved. However, in critical applications like virtual company representatives, the appearance at the decoder depends to a large extend on the proprietary model that may or may not be appropriate. Using facial expressions and visemes to animate the model allows keeping the personality of the proprietary face model whereas the use of low-level FAPs gives defined movement of facial features but it destroys the personality of the model. A face model of a baby will have the same expressions as the model of its grandfather.
2. *Calibrated proprietary face model*: The encoder sends 3D-shape information in terms of feature point coordinates or a 3D mesh to the decoder. The decoder adapts its proprietary face model to the calibration data. Please note that this calibration destroys the delicate relationship of facial expressions and object shape. Hence, the animation using facial expressions and visemes is not desirable. It is unclear how the decoder can make use of FAPs set to *interpolate* since the decoder lost knowledge of the face model due to the calibration.
3. *Downloaded face model*: The encoder sends a 3D-face model of its choice to the decoder. Additionally, the encoder defines the animation definition interfaces as proposed in Section 3.3.1 and [10]. In this case, the encoder knows the decoder face model and can animate it using facial expressions, visemes, and low-level FAPs. When using facial expressions and visemes for animation, it is also possible to animate a plurality of face models with the same parameter stream and still have models that perform as expected.

It seems that the scenarios 1 and 3 are the most relevant because both allow for simple model independent animation of personalized talking head models (downloaded or proprietary) using visemes and facial expressions. This is also favorable for interactive applications where the user wants to manually animate a talking head or use an image analysis system that usually cannot extract more than 10 animation parameters from a face.

What MPEG is currently lacking is a method for defining the time behavior of models. Currently, the FAP stream defines the temporal functions of a smile or eye blink. If we use one parameter stream to animate several models, their facial actions will be completely synchronized. I.e., they will start and stop smiling at exactly the same time and they will blink with their eyes at the same time. Furthermore, MPEG currently lacks a bookmark mechanism. Hence, an MPEG-4 terminal will not be able to precisely synchronize facial expressions with the output of a TTS system. We hope that MPEG will enlarge the animation parameter syntax in Phase 2 such that MPEG-4 will allow animating personalized head models with model-independent parameter streams.

5 Experimental Results

Our real-time animation system runs at 30 Hz on a SGI O2 workstation. We implemented several face models with their own facial expressions (Figure 14, Figure 15). Following the approach chosen by MPEG-4, each expression is directly implemented. Other implementations choose to create facial expressions by linearly superimposing several action units [1][20][21]. Each facial expression is controlled by one parameter as defined by MPEG-4. If several parameters are set, the facial expressions are superimposed.

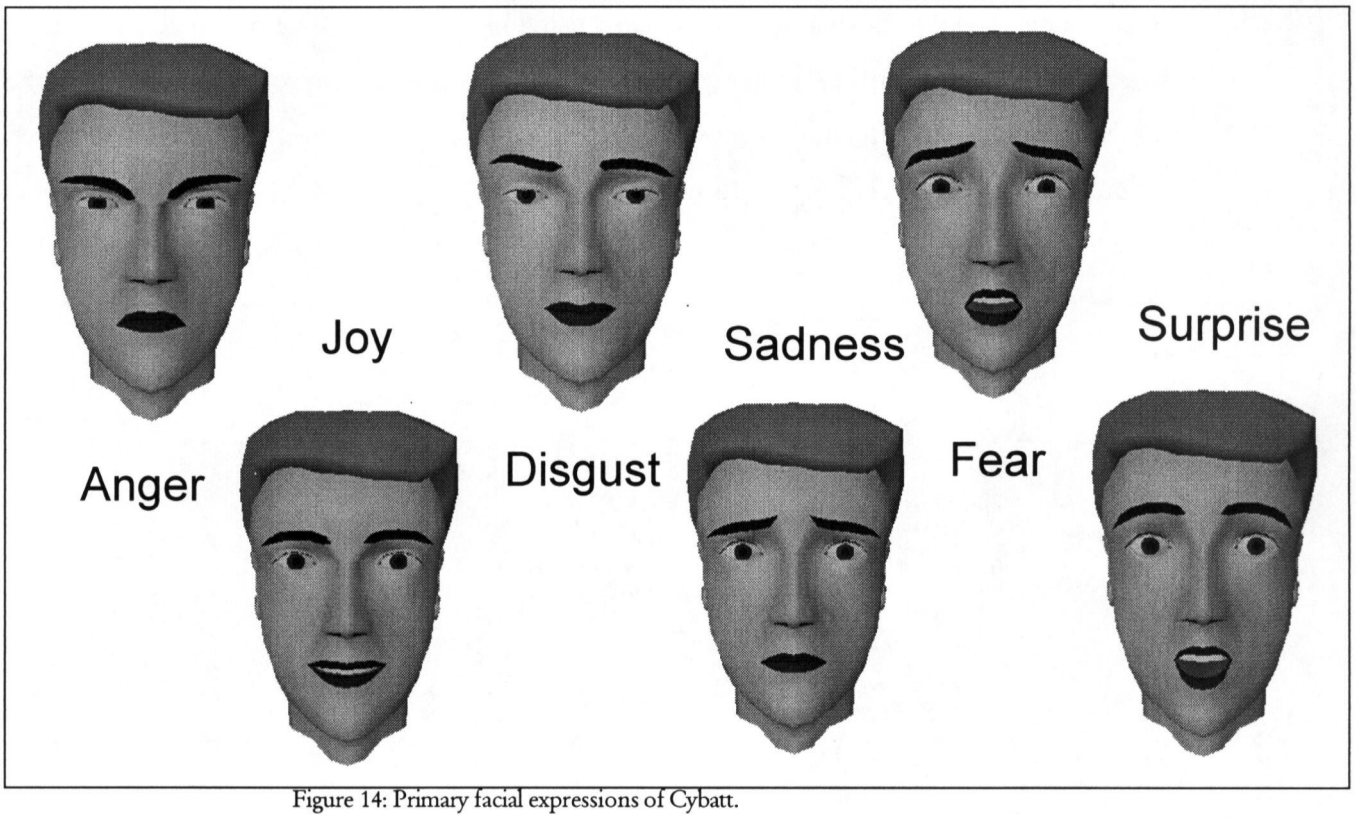

Figure 14: Primary facial expressions of Cybatt.

Figure 15 shows two face models that have different smiles. Both models were synthesized using the same animation parameters. Since each model has its own personalized facial expressions, they smile differently.

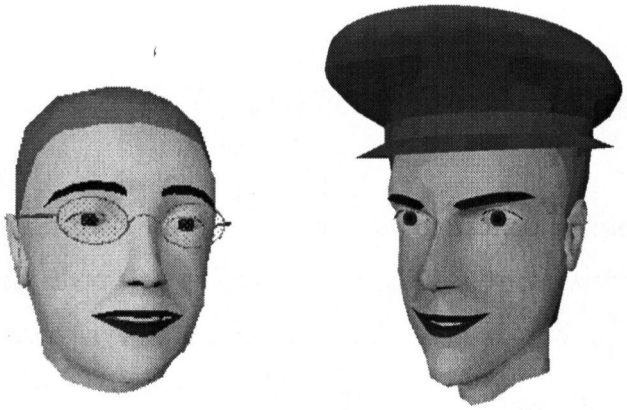

Figure 15: Two personalized talking head models with different smiles.

In case a personalized talking head is animated using high level temporal parameters indicating when a facial expression should start, when it should end and the climax of the expression, the model computes its own amplitude over time. For smiles, we implemented 3 stages, an onset, a climax and an offset [22]. Figure 16 shows an example.

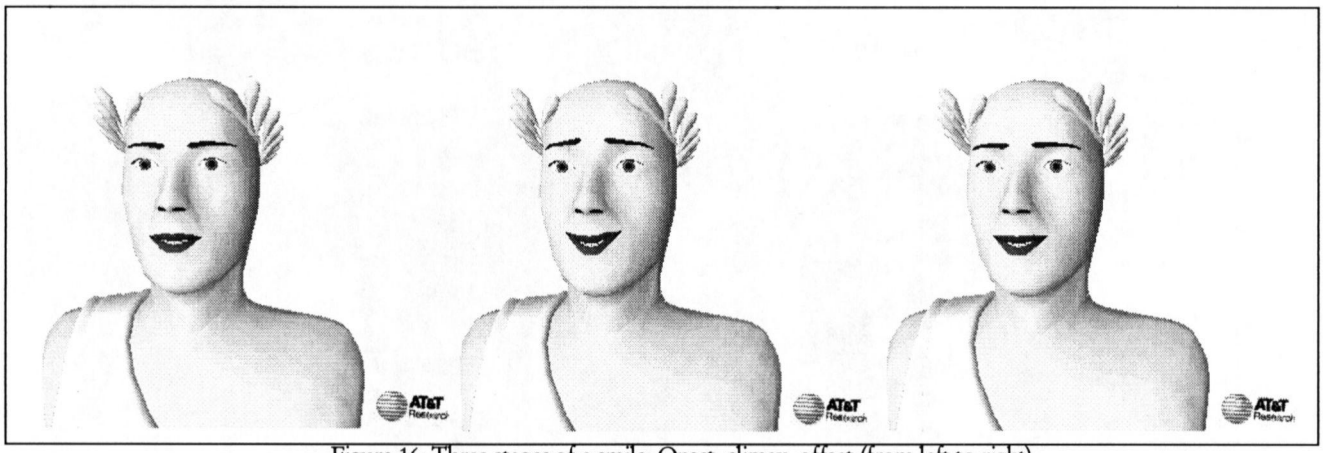

Figure 16: Three stages of a smile: Onset, climax, offset (from left to right).

As stated before, the expressions and the speech parameters are linearly superimposed and then animate the model. There is one exception. While talking, mouth closures are so important that facial expressions are prevented from opening the mouth during the pronunciation of a viseme that requires a mouth closure.

For a convincing talking head model, it is also essential to create eye blinks, head turns and head nodding. We create this animation with each model having its own time constants for these somehow arbitrary motions.

6 Conclusions

In this paper, we propose an animation system that allows the animation of personalized head models using model independent parameter streams. The parameter stream mainly contains high-level animation parameters like facial expressions and visemes. Although not implemented here, these high-level parameters could also be action units.

In order to use these model-independent animation parameter streams, the animated models have to be personalized, i.e. each model knows how to deform itself in the space time domain

in response to an animation parameter. Therefore, a personalized head model consists of three parts: A description of the static neutral face; a sequence of tables defining how the model deforms as a function of the amplitude of each animation parameter; a list of temporal animation definition parameters defining the temporal behavior for actions like facial expressions, head motion and eye blinks.

An architecture for creating these models was implemented. A range scanner or pictures of the person and modeling software are used to define the shape and texture of the model in neutral position. Professional animation software is used to define the behavior of the model. These models were tested in an animation system that allows animating the model by text using a text-to-speech synthesizer, model independent animation parameter as well as MPEG-4 facial animation parameters. Synchronization between facial expressions and the TTS synthesizer was maintained using bookmarks in the text to define the facial expressions. Since the personalized head models carry all their information with them, the renderer is model independent and easy to implement.

MPEG-4 has not yet embraced this approach fully. Currently, MPEG-4 only allows defining models as a static neutral face and a sequence of tables defining the deformation of the face. This enables to download and animate an arbitrary face model in an MPEG-4 terminal. MPEG-4 still needs to define temporally autonomous head models and the associated syntax.

Acknowledgements

The authors would like to thank Prof. Y. Wang and Dr. J. Schroeter for their help in reviewing this paper.

7 References

[1] S. Morishima and H. Harashima, "A Media Conversion from Speech to Facial Image for Intelligent Man-Machine Interface", IEEE Journal on Selected Areas in Communications, vol. 9, no. 4, pp. 594-600, May 1991.

[2] D. Kurlander and D. T. Ling, "Planning-Based Control of Interface Animation", Microsoft Research Technical Report MSR-TR-95-21, January 1995, Microsoft Research, Redmond, WA, USA.

[3] K. Nagao and A. Takeuchi, "Speech Dialogue With Facial Displays: Multimodal Human-Computer Conversation", Proceedings of the 32nd Annual meeting of the Association for Computational Linguistics,} pp. 102-109, 1994.

[4] R. Sproat and J. Olive, "An Approach To Text-to-speech Synthesis", In W.B. Kleijn & K.K. Paliwal (Eds.) Speech Coding and Synthesis, Elsevier Science, 1995.

[5] F. I. Parke, "Parameterized Models for Facial Animation", IEEE Computer Graphics and Applications, vol. 2, pp. 61-68, Nov. 1982.

[6] M. M. Cohen and D. W. Massaro, Modeling Coarticulation in Synthetic Visual Speech, In M. Thalmann & D. Thalmann (Eds.) *Computer Animation '93*, Tokyo: Springer-Verlag.

[7] K. Waters, T. Levergood, "An automatic lip-synchronization algorithm for synthetic faces", Proceedings of the Multimedia Conference, pages 149-156, San Francisco, California, September 1994. ACM

[8] J. Ostermann, E. Haratsch, "An animation definition interface: Rapid design of MPEG-4 compliant animated faces and bodies", International Workshop on synthetic - natural hybrid coding and three dimensional imaging, pp. 216-219, Rhodes, Greece, September 5-9, 1997.

[9] Potamianos, G., Cosatto, E., Graf, H. P. and Roe, D. B., ``Speaker independent audio-visual database for bimodal ASR," Proceedings of the ESCA/ESCOP Workshop on Audio-Visual Speech Processing, Rhodes, Greece, September 1997.

[10] J. Ostermann, E. Haratsch, "Let animals and furniture speak: A proposal for extending the scope of face and body animation", ISO/IEC JTC1/SC29/WG11 MPEG 97/1918, Bristol meeting, April 1997.

[11] C. Bregler, M. Covell, M. Slaney, "Video Rewrite: Driving Visual Speech with Audio", Proc. ACM SIGGRAPH 97, in Computer Graphics Proceedings, Annual Conference Series, 1997.

[12] D. Terzopolous, K. Waters, "Physically-based facial modeling, analysis and animation", Journal of Visualization and Computer Animation, vol. 1, pp. 73-80, 1990.

[13] K. Waters, "A muscle model of animating three dimensional facial expression", Computer Graphics, Vol. 22, No. 4, pp. 17-24, 1987.

[14] J. Hartman, J. Wernecke, *The VRML handbook*, Addison Wessley, 1996.

[15] ISO/IEC JTC1/WG11 N1901, Text for CD 14496-1 Systems, Fribourg meeting, November 1997.

[16] ISO/IEC JTC1/WG11 N1902, Text for CD 14496-2 Visual, Fribourg meeting, November 1997

[17] Chen, L., Ostermann, J. and Huang, T., ``Adaptation of a generic 3D human face model to 3D range data," First IEEE Workshop on Multimedia Signal Processing, Wang and et al. (Ed.), IEEE, Princeton, NJ, June 1997, pp. 274-279.

[18] M. Proesmans, L. van Gool, A. Oosterlink, "One-shot active 3D shape acquisition", Proceedings 13th IAPR international conference on pattern recognition: applications and robotic systems, vol III C, pp. 336-340, August 25-26, 1996, Vienna, Austria.

[19] M. Escher, N. M. Thalmann, "Automatic 3D cloning and real-time animation of a human face, Proc. Computer Animation '97, Geneva, 1997.

[20] S. Morishima, "Modeling of facial expression and emotion for human communication system", Display, Vol. 17, pp 15-25, 1996.

[21] P. Ekman, W.V. Friesen: Manual for the facial action coding system,. Consulting Psychologist Press, Inc. Palo Alto, CA, 1978.

[22] Ursula Hess, personal communication, December 1997.

[23] E. Haratsch, J. Ostermann, "Parameter Based Animation of Arbitrary 3D Head Models", Proceedings Picture Coding Symposium, Berlin, September 1997.

[24] Cyberware Laboratory Inc., "4020/RGB 3D Scanner with Color Digitize". Monterey, CA, 1990.

[25] S. Murakami, H. Tadokoro, "Generation of facial expressions based on time functions", Proceedings of IWSNHC3DI, Rhodes, September 97, pp. 212-215.