Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

Animation of Synthetic Faces in MPEG-4

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

Abstract

MPEG-4 is the first international standard that standardizes true multimedia communication – including natural and synthetic audio, natural and synthetic video, as well as 3D graphics. Integrated into this standard is the capability to define and animate virtual humans consisting of synthetic heads and bodies. For the head, more than 70 model-independent animation parameters defining low-level actions like move left mouth corner up to high-level parameters like facial expressions and visemes are standardized. In a communication application, the encoder can define the face model using MPEG-4 BInary Format for Scenes (BIFS) and transmit it to the decoder. Alternatively, the encoder can rely on a face model available at the decoder. The animation parameters are quantized, predictively encoded using an arithmetic encoder or a DCT. The decoder receives the model and the animation parameters in order to animate the model. Since MPEG-4 defines the minimum MPEG-4 terminal capabilities in profiles and levels, the encoder knows the quality of the animation at the decoder.

1. Introduction

The goal of MPEG-4 is to provide a new kind of standardization that responds to the evolution of technology, when it does not always make sense to specify a rigid standard addressing just one application. MPEG-4 will allow the user to configure and build systems for many applications by allowing flexibility in the system configurations, by providing various levels of interactivity with audio-visual content of a scene, and by integrating as many as possible audio visual data types like natural and synthetic audio, video and graphics [1][2]. MPEG-4 will become an International Standard in spring 1999, just in time for the new faster and more powerful media processors and in time for using the upcoming narrow- and broadband wired and wireless networks for audio-visual applications like database browsing, information retrieval and interactive communications.

As far as synthetic multimedia contents are concerned, MPEG-4 will provide synthetic audio like structured audio and a text-to-speech interface (TTSI). For synthetic visual contents, MPEG-4 allows to build 2D and 3D objects composed of primitives like rectangles, spheres, indexed face sets and arbitrarily shaped 2D objects. The 3D-object description is based on a subset of VRML nodes [3] and extended to enable seamless integration of 2D and 3D objects. Objects can be composed into 2D and 3D scenes using the Binary Format for Scenes (BIFS). BIFS also allows to animate objects and their properties.

Special 3D objects are human faces and bodies. MPEG-4 allows using decoder resident proprietary models as well as to transmit 3D models to the decoder such that the encoder can predict the quality of the presentation at the decoder. This paper focuses on the tools MPEG-4 provides to describe and animate 3D face models.

In Sections 2, we explain how MPEG-4 defines the specification of a face model and its animation using facial animation parameters (FAP). Section 3 provides details on how to efficiently encode FAPs. The integration of face animation into the overall architecture of an MPEG-4 terminal with text-to-speech capabilities is shown in Section 4. The capabilities of an MPEG-4 terminal are defined in profiles specifying a predefined set of tools and performance parameters for these tools as discussed in Section 5.

2. Specification and Animation of Faces

MPEG4 specifies a set of face animation parameters (FAPs), each corresponding to a particular facial action deforming a face model in its neutral state. The FAP value for a particular FAP indicates the magnitude of the corresponding action, e.g., a big versus a small smile. A particular facial action sequence is generated by deforming the face model in its neutral state according to the specified FAP values for the corresponding time instant. Then the model is rendered onto the screen.

The head in its neutral state is defined as follows (Figure 1): Gaze is in direction of Z axis; all face muscles are relaxed; eyelids are tangent to the iris; the pupil is one third of IRISD0;lips are in contact; the line of the lips is horizontal and at the same height of lip corners; the mouth is closed and the upper teeth touch the lower ones; the tongue is flat, horizontal with the tip of tongue touching the boundary between upper and lower teeth.

Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

For the renderer to interpret the FAP values using its face model, the renderer has to have predefined model specific animation rules to produce the facial action corresponding to each FAP. Since the FAPs are required to animate faces of different sizes and proportions, the FAP values are defined in face animation parameter units (FAPU). FAPU are defined as fractions of distances between key facial features (Figure 1). These features like eye separation, eye-nose separation, mouth nose separation, and mouth width, are defined for the face in its neutral state. They allow interpretation of the FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation.

MPEG-4 defines the animation rule for each FAP by specifying feature points and their direction of movement. The renderer can either use its own animation rules for his proprietary model or download the face model and the FaceDefTables that define the animation rules for the model.

In the following sections, we describe first how MPEG-4 defines the shape of a generic face model in its neutral state using feature points. Then we explain the facial animation parameters for this generic model. Finally we show how to define an MPEG-4 compliant face model that can be transmitted from the encoder to the decoder for animation.

2.1. Face Feature Points

In order to define face animation parameters for arbitrary face models, MPEG-4 specifies 84 feature points located in a face according to Figure 2 in order to provide a reference for defining facial animation parameters. Some feature points like the ones along the hairline are not affected by FAPs. They are required for defining the shape of a proprietary face model using feature points (Section 5). Feature points are arranged in groups like cheeks, eyes, and mouth (Table 1). The location of these feature points has to be known for any MPEG-4 compliant face model.

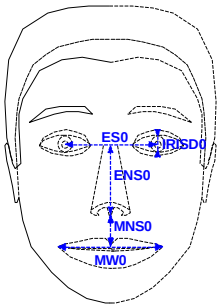

Figure 1: FAPUs

2.2. Face Animation Parameters

The FAPs are based on the study of minimal perceptible actions and are closely related to muscle action [4]. The 68 parameters are categorized into 10 groups related to parts of the face (Table 1). FAPs represent a complete set of basic facial actions including head motion, tongue, eye, and mouth control. They allow the representation of natural facial expressions. They can also be used to define facial action units [5]. Exaggerated values permit the definition of actions that are normally not possible for humans, but are desirable for cartoon-like characters.

The FAP set contains the two high-level parameters visemes and expressions (FAP group 1). A viseme is a visual correlate to a phoneme. Only 14 static visemes that are clearly distinguished are included in the standard set (Table 2). In order to allow for coarticulation of speech and mouth movement [6], transitions from one viseme to the next are defined by blending the two visemes with a weighting factor. Similarly, the expression parameter defines 6 high level facial expressions like joy and sadness (Figure 3). In contrast to visemes, facial expressions are animated with a value defining the excitation of the expression. Two facial expressions can be blended with a weighting factor. Since expressions are high-level animation parameters, they allow animating unknown models with high subjective quality.

Table 1: FAP groups

Group	Number of FAPs
1: visemes and expressions	2
2: jaw, chin, inner lowerlip, cornerlips, midlip	16
3: eyeballs, pupils, eyelids	12
4: eyebrow	8
5: cheeks	4
6: tongue	5
7: head rotation	3
8: outer lip positions	10
9: nose	4
10: ears	4

2.3. Face Model Specification

MPEG-4 allows the encoder to completely specify the face model the decoder has to animate. This involves defining the static geometry of the face model in its neutral state using a scene graph and defining the animation rules using FaceDefTables that specify how this model gets deformed by the facial animation parameters [7].

2.3.1. Static Geometry using a Scene Graph

The static geometry of the head model is defined with a scene graph specified using MPEG-4 BIFS [1]. For the purpose of defining a head model, BIFS provides the same nodes as VRML. VRML and BIFS describe geometrical scenes with objects as a collection of nodes, 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. For objects to

Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

move together as a group, they need to be in the same *Transform* group. The *Transform* node defines geometric

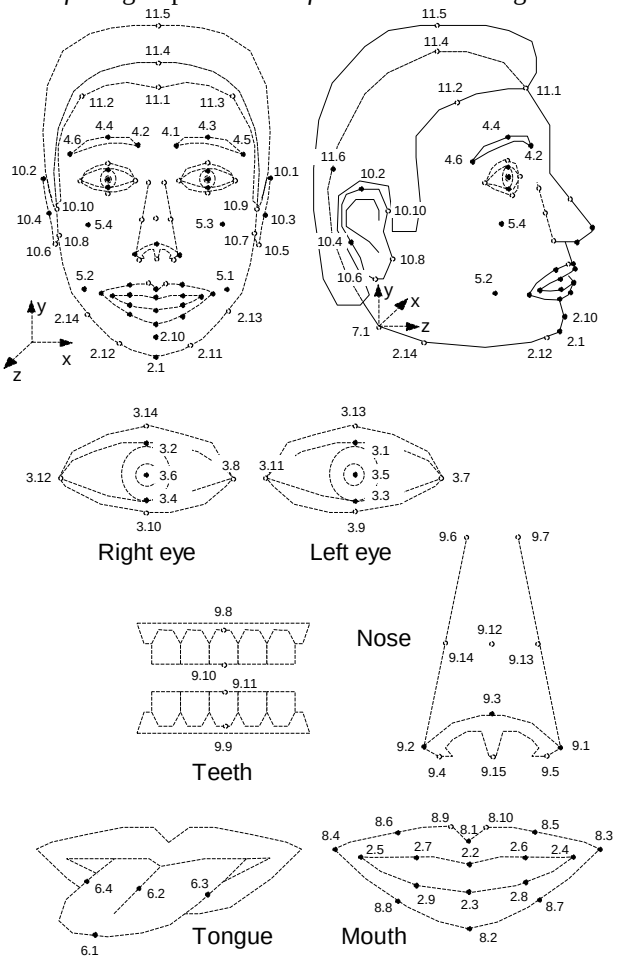

Figure 2: Feature points may be used to define the shape of a proprietary face model. FAPs are defined by motion of feature points.

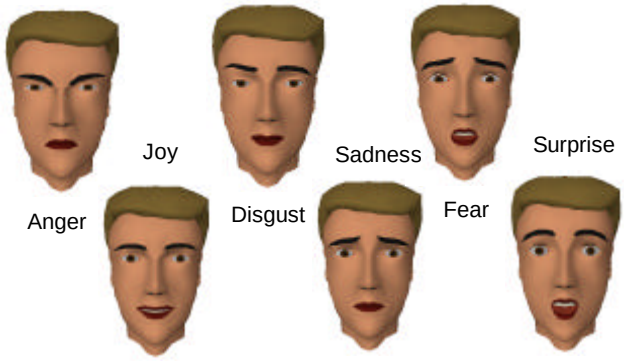

Figure 3: Primary facial expressions.

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 (3D mesh) and surface attributes (color, texture) of a polygonal object. Texture maps are coded the wavelet coder of the MPEG texture coder. Since the face model is specified with a scene graph, this face model can be easily extended to a head and shoulder model.

Table 2: Visemes and related phonemes

#	phonemes	example	#	phonemes	example
1	p, b, m	put, bed, mill	8	n, l	lot, not
2	f, v	far, voice	9	r	red
3	T,D	think, that	10	A:	car
4	t, d	tip, doll	11	e	bed
5	k, g	call, gas	12	I	tip
6	tS,dZ,S	chair,join, she	13	Q	top
7	s, z	sir, zeal	14	U	book

2.3.2. Animation Rules using FaceDefTables

A FaceDefTable defines how a model is deformed as a function of the amplitude of the FAP. It specifies, for a FAP, which Transform nodes and which vertices of an IndexedFaceSet node are animated by it and how. FaceDefTables are considered to be part of the face model.

Animation Definition for a Transform Node: If a FAP causes solely a transformation like rotation, translation or scale, a Transform node can describe this animation. The FaceDefTable specifies the type of transformation and a neutral value for the chosen transformation. During animation, the received value for the FAP and the neutral value determine the actual value.

Animation Definition for an IndexedFaceSet Node: If a FAP like smile causes flexible deformations of the face model, the animation results in updating vertex positions of the affected IndexedFaceSet nodes. The affected vertices move along piece-wise linear trajectories that approximate flexible deformations of a face. A vertex moves along this trajectory as the amplitude of the FAP varies. The FaceDefTable defines for each affected vertex its own piece-wise linear trajectory by specifying intervals of the FAP amplitude and 3D displacements for each interval (Table 3).

If $\boldsymbol{P}_m$ is the position of the m^{th} vertex in the IndexedFaceSet in neutral state (FAP = 0), $\boldsymbol{P'}_m$ the position of the same vertex after animation with the given FAP and $\boldsymbol{D}_{m,k}$ the 3D displacement in the k^{th} Interval, following algorithm must be applied to determine the new position $\boldsymbol{P'}_m$:

Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

1. Determine, in which of the intervals listed in the table the received FAP is lying.
2. If the received FAP is lying in the j^{th} interval $[I_j, I_{j+1}]$ and $0=I_k \le I_j$, the new vertex position $\boldsymbol{P'}_m$ of the m^{th} vertex of the IndexedFaceSet is given by:
   $\boldsymbol{P'}_m = FAPU * ((I_{k+1}-0) * \boldsymbol{D_{m,k}} + (I_{k+2}-I_{k+1}) * \boldsymbol{D_{m,k+1}} + \ldots (I_j - I_{j-1}) * \boldsymbol{D_{m,j-1}} + (FAP-I_j) * \boldsymbol{D_{m,j}}) + \boldsymbol{P}_m$.
3. If $FAP > I_{max}$, then $\boldsymbol{P'}_m$ is calculated by using the equation given in 2 and setting the index j = max-1.
4. If the received FAP is lying in the j^{th} interval $[I_j, I_{j+1}]$ and $I_{j+1} \le I_k=0$, the new vertex position $\boldsymbol{P'}_m$ is given by:
   $\boldsymbol{P'}_m = FAPU * ((I_{j+1} - FAP) * \boldsymbol{D_{m,j}} + (I_{j+2} - I_{j+1}) * \boldsymbol{D_{m,j+1}} + \ldots (I_{k-1} - I_{k-2}) * \boldsymbol{D_{m,k-2}} + (0 - I_{k-1}) * \boldsymbol{D_{m,k-1}}) + \boldsymbol{P}_m$.
5. If $FAP < I_1$, then $\boldsymbol{P'}_m$ is calculated by using the equation 4 in step 4 and setting the index j = 1.
6. If for a given FAP and 'IndexedFaceSet' the table contains only one interval, the motion is strictly linear:
   $\boldsymbol{P'}_m = FAPU * FAP * \boldsymbol{D_{m1}} + \boldsymbol{P}_m$.

Table 3: Simplified example of a FaceDefTable.

#FaceDefTable
FAP 6 (stretch left corner lip)
IndexedFaceSet: ***Face***
interval borders: -1000, 0, 500, 1000
displacements:
vertex 50 1 0 0, 0.9 0 0, 1.5 0 4
vertex 51 0.8 0 0, 0.7 0 0, 2 0 0
FAP 23 (yaw left eye ball)
Transform: ***LeftEyeX***
rotation neutral value: 0 -1 0 **(axis)** 1 **(angle)**

Example for an FaceDefTable: In Table 3, two FAPs are defined by FaceDefTables: FAP 6, which stretches the left corner lip, and FAP 23, which manipulates the horizontal orientation of the left eyeball.

FAP 6 deforms the IndexedFaceSet *Face*. For the piecewise-linear motion function three intervals are defined: [-1000, 0], [0, 500] and [500, 1000]. Displacements are given for the vertices with indices 50 and 51. The displacements for vertex 50 are: (1 0 0), (0.9 0 0) and (1.5 0 4), the displacements for vertex 51 are (0.8 0 0), (0.7 0 0) and (2 0 0). Given a FAP Value of 600, the resulting displacement for vertex 50 would be:

$\boldsymbol{P'}_{50} = \boldsymbol{P}_{50} + 500*(0.9\ 0\ 0)^T + 100 * (1.5\ 0\ 4)^T$
$= \boldsymbol{P}_{50} + (600\ 0\ 400)^T$.

FAP 23 updates the rotation field of the Transform node *LeftEyeX*. The rotation axis is (0, -1, 0), and the multiplication factor for the angle is 1. The FAP value determines the rotation angle.

Figure 4 shows 2 phases of an left eye blink (plus the neutral phase) which have been generated using a simple animation architecture [7].

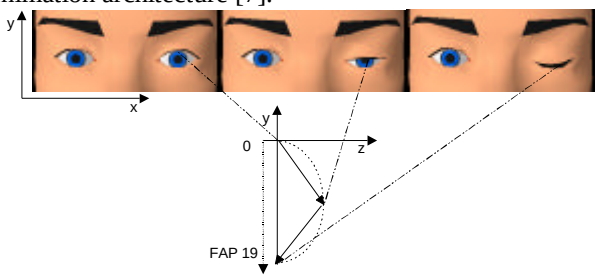

Figure 4: Neutral state of the left eye (left) and two deformed animation phases for the eye blink (FAP 19). The FAP definition defines the motion of the eyelid in negative y-direction, the FaceDefTable defines the motion in of the vertices of the eyelid in x and z direction. For FAP 19, positive FAP values move the vertices downwards.

3. Coding of Face Animation Parameters

For coding of facial animation parameters, MPEG-4 provides two tools. Coding of quantized and temporally predicted FAPs using an arithmetic coder allows for coding of FAPs introducing small delay only. Using a discrete cosine transform (DCT) for coding a sequence of FAPs introduces significant delay but achieves higher coding efficiency.

3.1. Arithmetic Coding of FAPs

Figure 5 shows the block diagram for encoding FAPs. The first set of FAP values $FAP(i)_0$ at time instant 0 is coded in intra mode. The value of an FAP at time instant *k* $FAP(i)_k$ is predicted using the previously decoded value $FAP(i)_{k-1}$. The prediction error *e* is quantized using a quantization stepsize that is specified for each FAP multiplied by a quantization parameter FAP_QUANT with 0<FAP_QUANT<9. FAP_QUANT is identical for all FAP values of one time instant *k*. Using the FAP dependent quantization stepsize and FAP_QUANT assures that quantization errors are subjectively evenly distributed between different FAPs. The quantized prediction error *e'* is arithmetically encoded using a separate adaptive probability model for each FAP. Since the encoding of the current FAP value depends only on one previously coded FAP value, this coding scheme allows for low-delay communications. At the decoder, the received data is arithmetically decoded, dequantized and added to the previously decoded value in order to recover the encoded FAP value.

In order to avoid transmitting all FAPs for every frame, the encoder can transmit a mask indicating for which groups (Table 1) FAP values are transmitted. The

Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

encoder can also specify for which FAPs within a group values will be transmitted. This allows the encoder to send incomplete sets of FAPs to the decoder.

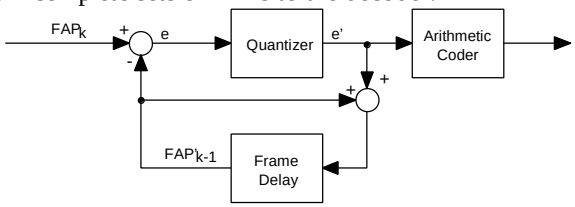

Figure 5: Block diagram of the encoder for FAPs.

The decoder can extrapolate values of unspecified FAPs, in order to create a more complete set of FAPs. The standard is vague in specifying how the decoder is supposed to extrapolate FAP values. Examples are that if only FAPs for the left half of a face are transmitted, the corresponding FAPs of the right side have to be set such that the face moves symmetrically. If the encoder only specifies motion of the inner lip (FAP group 2), the motion of the outer lip (FAP group 8) has to be extrapolated. Letting the decoder extrapolate FAP values may create unexpected results unless FAP interpolation functions are defined (Section 3.3).

3.2. DCT Coding of FAPs

The second coding tool that is provided for coding FAPs is the discrete cosine transform applied to 16 consecutive FAP values (Figure 6). This introduces a significant delay into the coding and decoding process. Hence, this coding method is mainly useful for application where animation parameter streams are retrieved from a database. This coder replaces the coder shown in Figure 5. After computing the DCT of 16 consecutive values of one FAP, DC and AC coefficients are coded differently. Whereas the DC value is coded predictively using the previous DC coefficient as prediction, the AC coefficient is directly coded. The AC coefficient and the prediction error of the DC coefficient are linearly quantized. Whereas the quantizer stepsize can be controlled, the ratio between the quantizer stepsize of the DC coefficients and the AC coefficients is set to ¼. The quantized AC coefficients are encoded with one variable length code word (VLC) defining the number of zero-coefficients prior to the next non-zero coefficient and one VLC for the amplitude of this non-zero coefficient. The handling of the decoded FAPs is not changed (see Section 3.1).

3.3. FAP Interpolation Tables

As mentioned in Section 3.1, the encoder may allow the decoder to extrapolate the values of some FAPs from the transmitted FAPs. Alternatively, the decoder can specify the interpolation rules using FAP interpolation tables (FIT). A FIT allows a smaller set of FAPs to be sent during a facial animation. This small set can then be used to determine the values of other FAPs, using a rational polynomial mapping between parameters. For example, the top inner lip FAPs can be sent and then used to determine the top outer lip FAPs. The inner lip FAPs would be mapped to the outer lip FAPs using a rational polynomial function that is specified in the FIT.

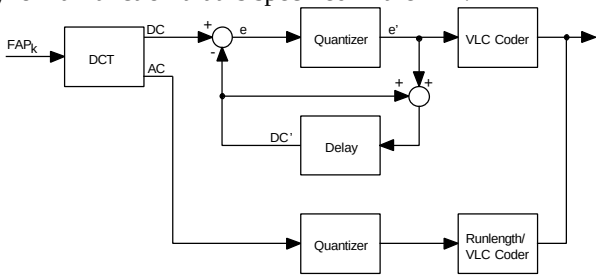

Figure 6: Block diagram of the FAP encoder using DCT. DC coefficients are predictively coded. AC coefficients are directly coded.

To make the scheme general, sets of FAPs are specified, along with a FAP interpolation graph (FIG) between the sets that specifies which sets are used to determine which other sets. The FIG is a graph with directed links. Each node contains a set of FAPs. Each link from a parent node to a child node indicates that the FAPs in child node can be interpolated from parent node.

In FIG, a FAP may appear in several nodes, and a node may have multiple parents. For a node that has multiple parent nodes, the parent nodes are ordered as 1st parent node, 2nd parent node, etc. During the interpolation process, if this child node needs to be interpolated, it is first interpolated from 1st parent node if all FAPs in that parent node are available. Otherwise, it is interpolated from 2nd parent node, and so on. An example of FIG is shown in Figure 7. Each node has an ID. The numerical label on each incoming link indicates the order of these links.

Each directed link in a FIG is a set of interpolation functions. Suppose F_1, F_2, …, F_n are the FAPs in a parent set and f_1, f_2, …, f_m are the FAPs in a child set. Then, there are m interpolation functions denoted as: $f_1 = I_1(F_1, F_2, \ldots, F_n)$, $f_2 = I_2(F_1, F_2, \ldots, F_n)$, $f_m = I_m(F_1, F_2, \ldots, F_n)$. Each interpolation function I_k () is in a rational polynomial form

$$I(F_1, F_2, ..., F_n) = \sum_{i=0}^{K-1} \left(c_i \prod_{j=1}^{n} F_j^{l_{ij}} \right) \Big/ \sum_{i=0}^{P-1} \left(b_i \prod_{j=1}^{n} F_j^{m_{ij}} \right) \quad (1)$$

where K and P are the numbers of polynomial products, c_i and b_i are the coefficient of the ith product. l_{ij} and m_{ij} are the power of F_j in the i^{th} product The encoder should send an interpolation function table which contains all $K, P, c_i, b_i, l_{ij}, m_{ij}$ to the decoder for each link in the FIG.

Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

4. Integration of Face Animation into an MPEG-4 Terminal

MPEG-4 defines a user terminal that allows decoding, composing and presenting multiple audio-visual objects. These A/V objects can be music, speech, synthesized speech from a text-to-speech (TTS) synthesizer, synthetic audio, video sequences, arbitrarily shaped moving video objects, images, 3D computer animated models or synthetic face models. MPEG-4 arranges and renders these objects into an audio-visual scene according to a scene description. This scene description allows defining variable dependencies of objects. As an example, it is not only possible to map an image as a texture onto a 3D model but also map video onto this object and align the position of the visual object and the position of the related sound source. Synchronization between the different media is achieved using the timing information of the individual media. The scene description allows also for interactivity inside the MPEG-4 player as well as for feedback to the encoder using a return channel.

Of particular interest to face animation is the MPEG-4 capability to map images as texture maps onto a face model, the synchronization of facial animation using FAPs and related audio, and the integration with a text-to-speech synthesizer. Synchronization with audio and speech streams with the FAP stream is achieved by evaluating the timing information that these streams have. Synchronization of an FAP stream with TTS synthesizers is currently only possible, if the encoder sends prosody and timing information. This is due to the fact that a conventional TTS system driven by text only behaves as an asynchronous source where the encoder does not know the exact timing behavior. The following section discusses the current integration of face animation and TTS and the ongoing work in this area.

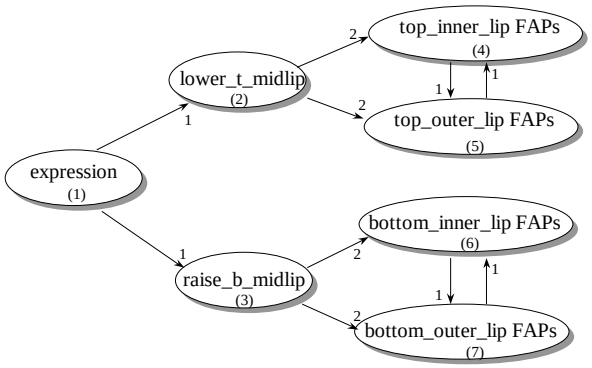

Figure 7: A FIG example for interpolating unspecified FAP values of the lip. If only the expression is defined, the FAPs get interpolated from the expression. If all inner lip FAPs are specified, they are used to interpolate the outer lip FAPs.

4.1. Text-to-Speech Interface(TTSI)

MPEG-4 foresees that talking heads will serve an important role in future customer service applications. Therefore, MPEG-4 provides interfaces to proprietary text-to-speech (TTS) synthesizer that allows driving a talking head from text (Figure 8) [8]. A TTS stream contains text or prosody in binary form. The decoder decodes the text and prosody information according to the interface defined for the TTS synthesizer. The synthesizer creates speech samples that are handed to the compositor. The compositor presents audio and if required video to the user. The second output interface of the synthesizer sends the phonemes of the synthesized speech as well as start time and duration information for each phoneme to the Phoneme/Bookmark-to-FAP-Converter. The converter translates the phonemes and timing information into face animation parameters that the face renderer uses in order to animate the face model. The precise method of how the converter derives visemes from phonemes is not specified by MPEG and left to the implementation of the decoder

In the current MPEG4 standard, the encoder is expected to send a FAP stream containing FAP number and value for every frame, to enable the receiver to produce desired facial actions. Since the TTS synthesizer can behave like an asynchronous source, synchronization of speech parameters with facial expressions of the FAP stream is usually not exact – unless the encoder transmits timing information for the synthesizer. An on-going working item in MPEG-4 is to provide a more efficient means for overlaying facial expression, by inserting bookmarks in the TTS stream. In addition to the phonemes, the synthesizer identifies bookmarks in the text that convey non-speech related facial expressions to the face renderer. The timing information of the bookmarks is derived from their position in the synthesized speech. The precise method of how the converter derives a continuous stream of FAPs from bookmarks is not specified by MPEG and left to the implementation of the decoder.

Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

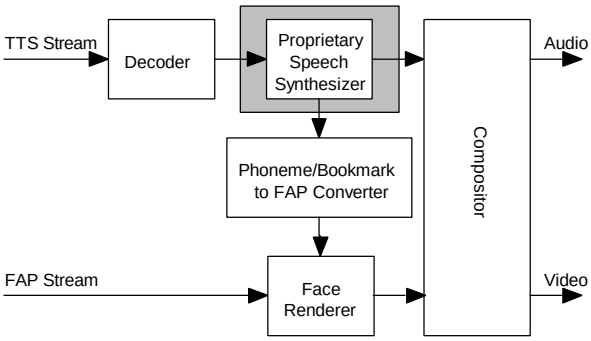

Figure 8: Block diagram showing the integration of a proprietary Text-to-Speech Synthesizer into an MPEG-4 face animation system.

5. Profiles

MPEG-4 foresees that applications for face animation fall in three scenarios and defines tools for these scenarios in three corresponding profiles.

Simple Profile: The decoder has its own proprietary model that is animated by a coded FAP stream (Sections 3.1 and 3.2.)

Calibration Profile: This profile includes the simple profile. The encoder transmits to the decoder calibration data for some or all of the predefined feature points (Figure 2). The decoder adapts its proprietary face model such that it aligns with the position of these feature points. This allows for customization of the model although the result is not predictable since the standard does not define a minimum quality for a decoder face model and the adaptation process of an arbitrary face model to these feature points is not specified in the standard. The calibration profile also requires the decoder to understand FITs (Section 3.3). Hence this profile allows for a higher coding efficiency than the simple profile.

Predictable Profile: This profile includes the calibration profile. MPEG-4 also provides a mechanism for downloading a model to the decoder according to Section 2.3 and animating this model. This gives the encoder control over the presentation at the receiver and allows for sensitive applications like web-based customer service.

In order to provide guranteed levels of performance at the decoder, MPEG-4 defines for each profile levels that specify minimum requirements in terms of FAP decoding speed and render speed. Only terminals that pass conformance tests for a defined profiles and level are MPEG-4 compliant.

6. Conclusions

MPEG-4 integrates animation of synthetic talking faces into audio-visual multimedia communications. A face model is a representation of the human face that is structured for portraying the visual manifestations of speech and facial expressions adequate to achieve visual speech intelligibility and the recognition of the mood of the speaker. A face model is defined as a static 3D model and related animation rules that define how the model deforms if it is animated with face animation parameters. The model is defined using a scene graph. Therefore, a customized model with head and shoulders can be defined for games or web-based customer service applications. MPEG-4 defines a complete set of animation parameters tailored towards animation of the human face. However, face animation parameters are defined independent of the proportions of the animated face model. Therefore, a face animation parameter stream can be used to animate different models. Successful animations of humans, animals and cartoon characters have been demonstrated.

In order to enable animation of a face model over low bitrate communication channels, for point to point as well as multi-point connections, MPEG-4 encodes the facial animation parameters using temporal prediction, quantization and coding of the prediction error. For low-delay applications, the prediction error is coded using an adaptive arithmetic coder, for other applications a discrete cosine transform is applied to a sequence of each facial animation parameter. In order to avoid coding of the entire set of more than 70 animation parameters for each frame, undefined animation parameters can be interpolated from the coded parameters. The encoder can specify these interpolation rules using rational polynomials. Face models can be animated with a data rate of 300 – 2000 bits/s

For talking head applications, MPEG-4 defines application program interfaces for TTS synthesizer. Using these interfaces, the synthesizer provides phonemes and related timing information to the face model enabling simple talking head applications.

In order to use facial animation for entertainment and business applications, the performance of the MPEG-4 player has to be known to the content creator. Therefore, MPEG defined 3 profiles and conformance points for facial animation that allow different levels of configuration of the decoder. This will make MPEG-4 face animation an attractive platform for many applications.

Acknowledgements: The author would like to thank Yao Wang and Ariel Fischer for the review of this paper.

7. References

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

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

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

[4] Kalra P., Mangili A, Magnenat-Thalmann N, Thalmann D. “Simulation of Facial Muscle Actions Based on Ra-

Reprint of J. Ostermann, "Animation of Synthetic Faces in MPEG-4",
Computer Animation, pp. 49-51, Philadelphia, Pennsylvania,
June 8-10, 1998. Please observe copyright.

tional Free Form Deformations", Proc. Eurographics 92, pp. 59-69, 1992.

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

[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] 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.

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