.

Facial Animation Framework for the Web and Mobile Platforms

Igor S. Pandzic
Department of Electrical Engineering
Linköping University, SE-581 83 Linköping
+46-13-281 889
igor@isy.liu.se

ABSTRACT

Talking virtual characters are graphical simulations of real or imaginary persons capable of human-like behavior, most importantly talking and gesturing. They may find applications on the Internet and mobile platforms as newscasters, customer service representatives, sales representatives, guides etc. After briefly discussing the possible applications and the technical requirements for bringing such applications to life, we describe our approach to enable these applications: the Facial Animation Framework. This framework consists of (1) a lightweitht, portable, MPEG-4 compatible Facial Animation Player, (2) a system for fast production of ready-to-animate, MPEG-4 compatible face models and (3) a plethora of MPEG-4 compatible tools for Facial Animation content production. We believe that this kind of approach offers enough flexibility to rapidly adapt to a broad range of applications involving facial animation on various platforms.

Categories & Subject Descriptors

H.5.1 Multimedia Information Systems – *Animations*; *Artificial, augmented, and virtual realities*
H.5.2 User Interfaces (D.2.2, H.1.2, I.3.6) -- *Auditory (non-speech) feedback* , *Graphical user interfaces (GUI)* , *Natural language* , *User-centered design*

General Terms: Algorithms, Design, Human Factors

Keywords: Facial Animation, Virtual Characters, Virtual Humans, Talking Head, MPEG-4, FBA, VRML, Visual text-to-speech, facial motion cloning

1. Introduction

With the recent expansion of various services to the Internet, and expected expansion into mobile, new applications for Virtual Humans technologies can be identified. We believe in particular that talking virtual characters, such as [19, 23, 20, 30, 13, 6, 28, 12, 4, 21, 7, 5, 8, 16], can provide novel services such as virtual hosts, salespersons, newscasters and other.

We briefly discuss these potential applications and the technical requirements for bringing such applications to life in Section 2. As background information essential for understanding this article, we very shortly introduce the MPEG-4 Facial Animation standard in Section 3.

We describe our Facial Animation Framework for the Web and mobile platforms, as outlined in Figure 3, in three main sections. Section 4 deals with the Facial Animation Player. The player accepts facial animation input in the MPEG-4 FBA format from a wide variety of sources. Due to its simplicity, it is easy to implement as well as port and adapt to any platform. Current implementations include a Java applet and a prototype on the Symbian Quartz tablet communicator reference design [24]. Section 5 presents our approach to producing animatable face models, i.e. face models with the additional information necessary for them to be animated correctly in the Facial Animation Player. The face model production follows the widely used and well known principle of morph targets, but extends it to include low-level facial actions rather than just high level expressions and visemes. We also significantly simplify the task of the animator by introducing the Facial Motion Cloning method to automatically copy a whole set of morph targets from one face model to another. In Section 5 we present alternatives for Facial Animation content production ranging from text-to-speech to facial tracking, all feeding content to the Facial Animation Player. Finally, in the concluding section we summarize the advantages of our approach.

2. Applications

With the trend of moving services on-line and, in the near future, making them accessible through mobile terminals, we believe that human-like virtual characters can lend a personality and a more human touch. A talking person, even a virtual one, can be more pleasant and more persuasive than just text and graphics.

Additionally, as outlined in [15], the virtual character may in certain situations alleviate the (sometimes unavoidable) network waiting times by "entertaining" the user. The same study has also shown that the end users show a preference for a service enhanced with a virtual character to a plain text service.

If correctly implemented, virtual characters have low bandwidth requirements and high interaction capacity, making them a natural replacement for video streaming in "talking head" scenarios.

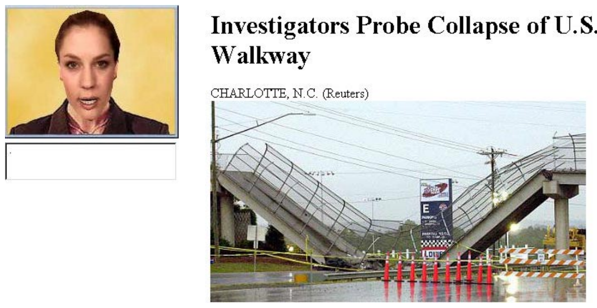
Combined with facial feature tracking, this technology can potentially be used for a very low bitrate visual communication, in a model based coding scenario [9, 10, 22].

We believe that we are just beginning to recognize possible practical applications, and what is presented here may well be just the tip of the iceberg. We can broadly classify the emerging application areas in the following categories:

- entertainment (interactive story telling, talking caricatures)

.

- personal communications (short messages, greetings, visual email, visual communication)
- navigation aid (virtual guide/host to web sites or services, tutorial presentations)
- newscasting (virtual newscaster (e.g. Figure 1), personalized interactive news)
- commerce (customer service, sales (e.g. Figure 2), marketing)
- education (interactive virtual teacher/trainer)
- advanced multimodal Human-Computer Interface


Investigators Probe Collapse of U.S. Walkway

CHARLOTTE, N.C. (Reuters)

Figure 1. A virtual newscaster; images on the right appear as she talks (© W Interactive SARL [31])

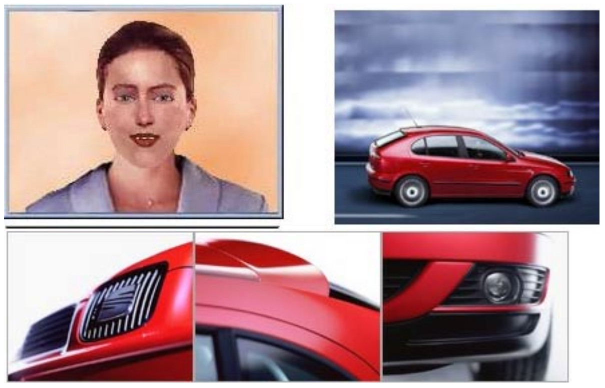


Figure 2. Virtual sales representative presenting a car model (© W Interactive SARL [31]) Requirements

Based on these application clases, we can identify the following requirements, expanded from the ones identified in [17], on the facial animation system to be used for such applications in the Web and mobile context:

2.1 Visual quality

It is obvious that virtual characters featured in any of the mentioned applications must have a visual appeal. They must look attractive, and animations should look natural. This does not necessarily mean photo-realistic models, or even very high-resolution models. A fairly simple cartoon character, if cleverly designed, can be very appealing. The real implication of this requirement is that any system hoping for success must provide means for visual artists to design models and animations, preferably using the tools that they are already used to, as this is the only reasonable way to promote creation of appealing content.

2.2 Easy installation

In many of the mentioned applications the virtual character is not in itself the main attraction of the proposed service, but rather an extra bonus, an improvement. The users who never used such an application might be hesitant to install software on their computers only to experience an unknown improvement of the service. This creates a barrier. To make the user jump over the barrier, one can either provide strong incentive in form of attractive marketing, or lower the barrier by making it easy to install the virtual character support. Ideally, no installation should be necessary. This means that the facial animation player must use, to the largest possible extent, the resources readily available on the target platform, or resources that are fairly widespread, like popular plug-ins.

2.3 Fast access

Although broadband-for-everyone is being promised by many, the reality is that most users do not have fast Internet access. Therefore the new applications involving virtual characters should not require high additional bandwidth. This implies several things. First, the virtual human models should not be too complex. Second, they need to be compressed for download. Third, both audio and animation data should not only be compressed, but also streamed, rather than downloaded and played. This allows for a faster response time.

2.4 Content generation

There is a need for an easy, preferably automatic way to generate content, i.e. speech and facial animation to be played by the facial animation system. This is because no interesting interaction can be achieved without enough variety of content. In simple words, a virtual character that can just say “yes” and “no” is not very interesting. If generating content is expensive, designing the whole application will be very expensive. It is preferable to have the possibility to generate content automatically. In the ideal case, the speech and behavior of the virtual character is completely generated on-the-fly, putting no limits to the variety of interaction. Unfortunately, this is in most cases opposing the requirement on the quality (the best content is still created with a lot of manual work), and a compromise must be found. It is therefore desirable to have a variety of options for content generation with varying levels of production time/cost and resulting quality. At the same time, it is essential that switching between these options be painless, which can be achieved through the use of common standards.

2.5 Platform integration

Whether it appears on the web page or on a mobile terminal, the virtual character must become an integral part of its environment, rather than an isolated application. In the context of a Web page, it should be able to react to users’ actions on the web page, and to control and change the content of the page. This implies that an open interface (in form of an API) should exist to make logical links between the virtual character and its surrounding platform, in both directions.

2.6 Decision-making

For many services it is desirable for a virtual character to have a fairly sophisticated decision-making mechanism, possibly based on AI algorithms like the descendants of Eliza [29] or the A.L.I.C.E. system [2], to deliver meaningful interactions. The facial animation system should have mechanisms to connect to such a system and generate audio-visual content based on the outputs of the decision-making module.

.

3. Introduction to MPEG-4 Facial Animation

MPEG-4 Facial Animation is central to our view of possible facial animation architectures, systems and applications (see Figure 3). For this reason, before going further with this article we need to introduce the basic notions of MPEG-4 Facial Animation. Beside the standard itself [11], there are other excellent references [27] [8] covering this subject. The purpose of this short section is therefore not to offer in-depth coverage, but to provide just enough background for understanding the rest of the article. Readers familiar with MPEG-4 FA may wish to skip this section, or use it only as a quick reference.

The MPEG-4 specification defines 66 low-level Facial Animation Parameters (FAPs) and two high-level FAPs. The low-level FAPs are based on the study of minimal facial actions and are closely related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of most natural facial expressions. Exaggerated values permit the definition of actions that are normally not possible for humans, but could be desirable for cartoon-like characters.

All low-level FAPs are expressed in terms of the *Facial Animation Parameter Units (FAPU)*. These units are defined in order to allow interpretation of the FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. They correspond to distances between key facial features and are defined in terms of distances between the MPEG-4 facial Feature Points (FPs). For each FAP it is defined on which FP it acts, in which direction it moves, and which FAPU is used as the unit for its movement. For example, FAP no. 3, `open_jaw`, moves the Feature Point 2.1 (bottom of the chin) downwards and is expressed in MNS (mouth-nose separation) units. The MNS unit is defined as the distance between the nose and the mouth divided by 1024. Therefore, in this example, a value of 512 for the FAP no. 3 means that the bottom of the chin moves down by half of the mouth-nose separation. The division by 1024 is introduced in order to have the units sufficiently small that FAPs can be represented in integer numbers.

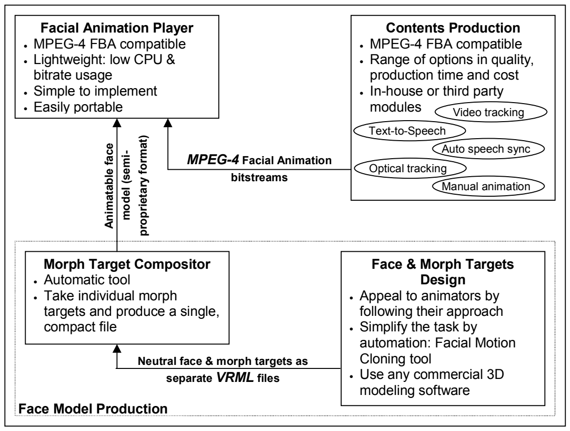
The specification includes two high-level FAPs: `expression` and `viseme`. Expression can contain two out of a predefined list of six basic expressions. Intensity values allow to blend the two expressions. Similarly, the Viseme parameter can contain two out of a predefined list of 14 visemes, and a blending factor to blend between them.

The specification also defines the Facial Animation Tables (FATs). The FATs allow to specify, for each FAP (high- and low-level), the exact motion of the vertices and/or transforms in the 3D model. This means that the expressions, visemes and low-level FAPs are described in a way that is essentially equivalent to the mentioned morph-targets. Animation systems then interpolate and blend between the values from the FAT.

4. The Facial Animation Player

As Figure 3 shows, the Facial Animation Player is the cornerstone of our facial animation framework for the Web and mobile platforms. The Player should be easy to port and adapt to any platform, accept Facial Animation input from a wide variety of sources, and be modest in usage of resources.

The first choice we made when designing the player was to make it MPEG-4 FBA compatible. This guarantees the variety of Facial Animation content sources, as many developers already support the standard and this trend is increasing. The choice of MPEG-4 also ensures very low bitrate needs. The MPEG-4 FBA decoding process itself is based on integer arithmetic, its implementation is very compact and it is very modest in CPU usage.



Facial Animation Player
• MPEG-4 FBA compatible
• Lightweight: low CPU & bitrate usage
• Simple to implement
• Easily portable
Contents Production
• MPEG-4 FBA compatible
• Range of options in quality, production time and cost
• In-house or third party modules
Video tracking
Text-to-Speech
Auto speech sync
Optical tracking
Manual animation
MPEG-4 Facial Animation bitstreams
Animatable face model (semi-proprietary format)
Morph Target Compositor
• Automatic tool
• Take individual morph targets and produce a single, compact file
Face & Morph Targets Design
• Appeal to animators by following their approach
• Simplify the task by automation: Facial Motion Cloning tool
• Use any commercial 3D modeling software
Neutral face & morph targets as separate VRML files
Face Model Production

Figure 3: The Facial Animation Framework for the Web and mobile platforms

When the MPEG-4 FAPs are decoded, the player needs to apply them to a face model. Our choice for the facial animation method is interpolation from key positions, essentially the same as the morph target approach widely used in computer animation and the MPEG-4 FAT approach (see previous section for explanation of FAT). Interpolation was probably the earliest approach to facial animation and it has been used extensively [19, 20, 3]. We prefer it to procedural approaches like [13, 6, 12, 8], and certainly to the more complex muscle based models like [23, 30, 28] for the following reasons:

- It is very simple to implement, and therefore easy to port to various platforms.
- It is modest in CPU time consumption
- The usage of key positions (morph targets) is close to the methodology used by computer animators and should be easily adopted by this community

The way it works is the following. Each FAP (both low- and high-level) is defined as a key position of the face, or *morph target*. To stay consistent with the computer animation terminology, we will use the term morph target throughout the article. Each morph target is described by the relative movement of each vertex with respect to its position in the neutral face, as well as the relative rotation and translation of each transform node in the scene graph of the face. The morph target is defined for a particular value of the FAP. The movement of vertices and transforms for other values of the FAP are then interpolated from the neutral face and the morph target. This can easily be extended to include several morph targets for each FAP and use a piecewise linear interpolation function, like the FAT approach defines. However, current implementations show simple linear interpolation to be sufficient in all situations encountered so far. The vertex and transform movements of the low-level FAPs are added together to produce final facial animation frames. In case of high-level faps, the movements are blended by averaging, rather than added together.

.

4.1 Implementation

Due to its simplicity and low requirements, the Facial Animation Player is expected to be easy to implement on a variety of platforms using various programming languages. The implementation we describe here is written as a Java applet and based on the Shout3D rendering engine [25]. An implementation on the Symbian Quartz [24] tablet communicator reference design (see Figure 4) is in an early prototype phase and we will not describe it in detail here.

Table 1: Bandwidth requirements

Total applet size (CAB file)	156K
Applet size (Shout3D)	~130K
Applet size (FA implementation)	~26K
Viseme-encoded FBA bitstreams	~0.3 kbit/s
Low-level FAPs FBA bitstreams	2-6 kbit/s
Audio (GSM 6.10)	13 kbit/s
Face models (medium complexity)	~50K

Table 2: Performance measurements for different face models on different computer configurations. C1 = P3/600 laptop; C2 = P3/1000; C3 = C2 with graphics acceleration hardware (ELSA GLoria II-64)

Model	Poly-gons	Size (KB)	Frames/second		
			C1	C2	C3
Demy	2800	32	15	24	31
Dummy	1362	50	20	31	41
Jörgen	168	40	32	40	60
Candide	168	4	40	60	60
MIRAface	3692	67	15	24	40
Cm. Lake	16917	284	1.4	1.5	2

4.2 Performance results

We have successfully tested the MPEG-4 Facial Animation player with several face models as illustrated in Figure 5. We can achieve interactive frame rates with models of up to 3000 polygons. The player has correctly interpreted the test FBA bitstreams (Marco, Wow, Emotions) as well as the bitstreams produced by the text-to-speech system. As the demonstration web page [18] shows, the applet is fully controllable from the web page by Javascript, making all interactions possible.

Table 1 summarizes the bandwidth requirements of this implementation. The size of the applet might be problematic, due mostly to the size of Shout3D renderer implementation (the Facial Animation implementation, including the MPEG-4 FBA decoder, is fairly small). The face model sizes, as well as facial animation and audio bitstreams, are very reasonable and content delivery is satisfactory even on modem connections.

Table 2 shows the performance measurements for different face models. Performance tests were done on two different computer configurations. A third measurement was performed using the graphics acceleration hardware. In order to use the graphics acceleration, a special plug-in for the browser needs to be installed; this plug-in allows the Shout3D renderer to use the graphics acceleration. The results show that the implementation can animate facial models of medium complexity at interactive frame rates on fairly modest hardware. Tests were made in the Internet Explorer browser; most of the measurements were confirmed in Netscape Navigator as well.

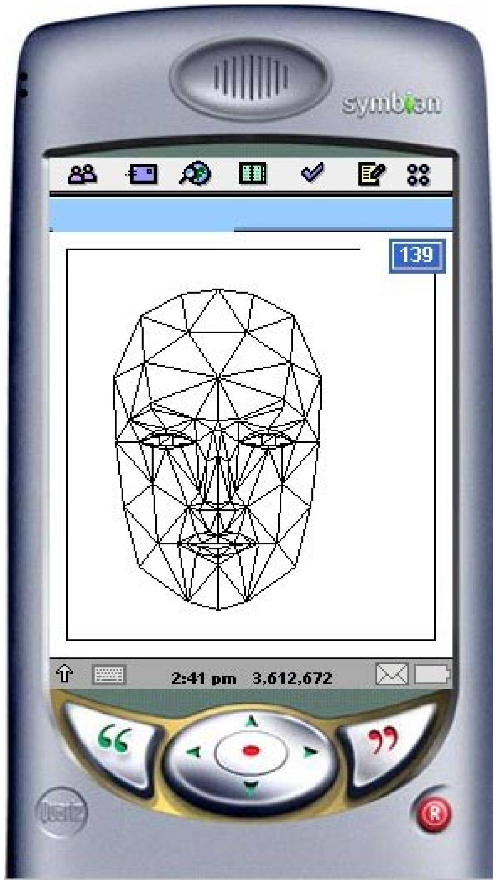


Figure 4: The Facial Animation Player early prototype implementation on Symbian Quartz using the Candide face model from Linköping University.

5. Producing Animatable Face Models

In this section we describe our approach to the production of face models that can be directly animated by the Facial Animation Player described in the previous section. Figure 3 illustrates the phases of face model production and how it fits within the Facial Animation Framework.

We believe that the most important requirement for achieving high visual quality is the openness of the system for visual artists. It should be convenient for them to design face models with the tools they are used to. While numerous algorithmic facial animation systems have been developed, the best-looking animations in current productions are done manually by artists or by facial tracking equipment and performing talent. This manual creation is painstakingly time-consuming, but some aspects can be automated.

The concept of morph targets as key building blocks of facial animation is already widely used in the animation community. However, morph targets are commonly used only for high level expressions (visemes, emotional expressions). In our approach we

.

follow the MPEG-4 FAT concept and use morph targets not only for the high level expressions, but also for low-level MPEG-4 FAPs. Once their morph targets are defined, the face is capable of full animation by limitless combinations of low-level FAPs. Furthermore, being MPEG-4 compatible offers access to a growing wealth of content and content sources.

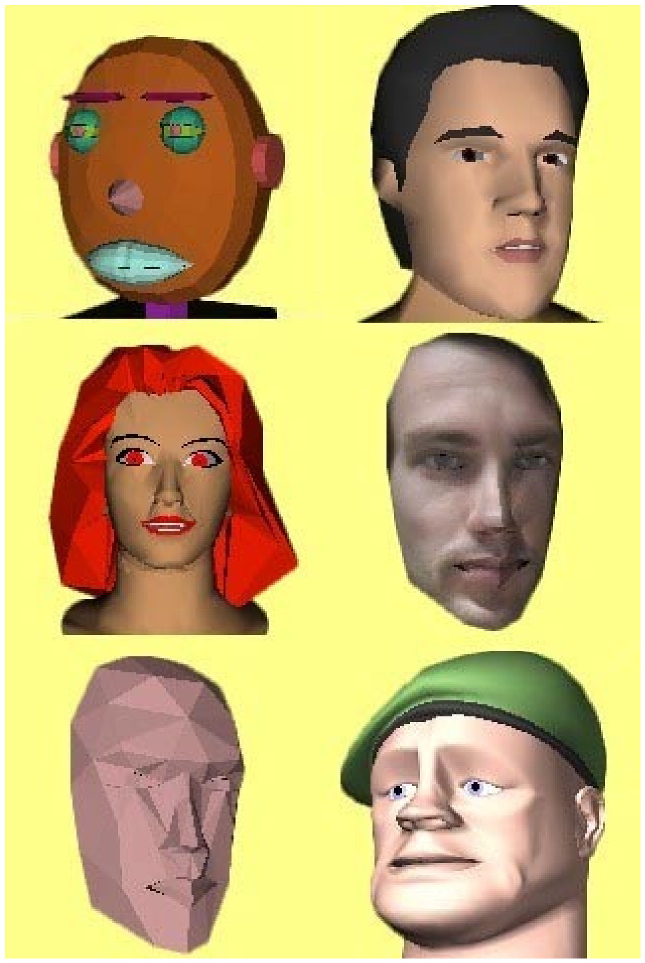


Figure 5. Examples of face models experimentally animated using the player: dummy, a model built using 3D modeling software; Miraface, a model donated by MIRALab, University of Geneva, to ISO as MPEG-4 reference software; Demy, designed by Sasa Galic; Jörgen, Candide, source Linköping University; Commander Lake, source 3DS Max.

Obviously, creating morph tragets not only for high level expressions, but also for low-level FAPs is a tedious task. We therefore propose a method to copy the complete range of morph targets, both low- and high-level, from one face to another. This means that an artist could produce one very detailed face with all morph targets, then use it to quickly produce the full set of morph targets for a new face. The automatically produced morph targets can still be edited to achieve final detail. It is concievable that libraries of facial models with morph targets suitable for copying to new face models will be available commercially. The method we propose for copying the morph targets is called Facial Motion Cloning. Our method is similar in goal to the Expression Cloning [14]. However, our method additionally preserves the MPEG-4 compatibility of cloned facial motion and it treats transforms for eyes, teeth and tongue. It is also substantially different in implementation.

Facial Motion Cloning can be schematically represented by Figure 7. The inputs to the method are the source and target face. The source face is available in neutral position (*source face*) as well as in a position containing some motion we want to copy (*animated source face*). The target face exists only as neutral (*target face*). The goal is to obtain the target face with the motion copied from the source face – the *animated target face*.

To reach this goal we first obtain *facial motion* as the difference of 3D vertex positions between the animated source face and the neutral source face. The facial motion is then added to the vertex positions of the target face, resulting in the animated target face.

In order for this to work, the facial motion must be normalized, which ensures that the scale of the motion is correct. In the *normalized facial space*, we compute facial motion by subtracting vertex positions of the animated and the neutral face. To map the facial motion correctly from one face to another, the faces need to be aligned with respect to the facial features. This is done in the *alignment space*. Once the faces have been aligned, we use interpolation to obtain facial motion vectors for vertices of the target face. The obtained facial motion vectors are applied by adding them to vertex positions, which is possible because we are working in the normalized facial space. Finally, the target face is de-normalized.

In the normalized facial space all faces have the same key proportions, e.g. same distance between the eyes etc. The benefit is that the magnitude of motion in the normalized space is the same for different faces and we can therefore apply motion from one face to another by simple addition. Figure 6B shows an example of a normalized face.

To define the proportions of the face we use the Facial Animation Parameter Units (FAPU, see section 3), as specified in the MPEG-4 standard, and the MPEG-4 Feature Points (FP, see section 3). The regions of the face are determined from the proximity to the FPs. The appropriate FAPUs are then used as normalization factors for each vertex. This amounts to scaling the face model with a regionally changing scaling factor, which is the local FAPU. By normalizing with FAPUs we make sure that the motion of the feature points in the normalized space corresponds to the MPEG-4 FAPs.

The facial motion is defined as the difference in vertex positions between the animated and neutral face. It is expressed by an array of *facial motion vectors*, each vector corresponding to one vertex of the face.

When transferring facial motion from source to target face, we need to transfer the motion to the corresponding facial region. In order to achieve this, we compute the motion mappings using the *alignment space* where the source and target face are aligned with respect to the feature points.

To achieve the alignment, the normalized faces are first mapped into 2D space using cylindrical projection around Y-axis. The center of projection is computed in such a way that the angular width of the mouth remains constant. The zero angle is always aligned with the tip of the nose. These two rules, together with previous normalization, ensure that the projected faces, shown in Figure 6C, are already roughly aligned.

.

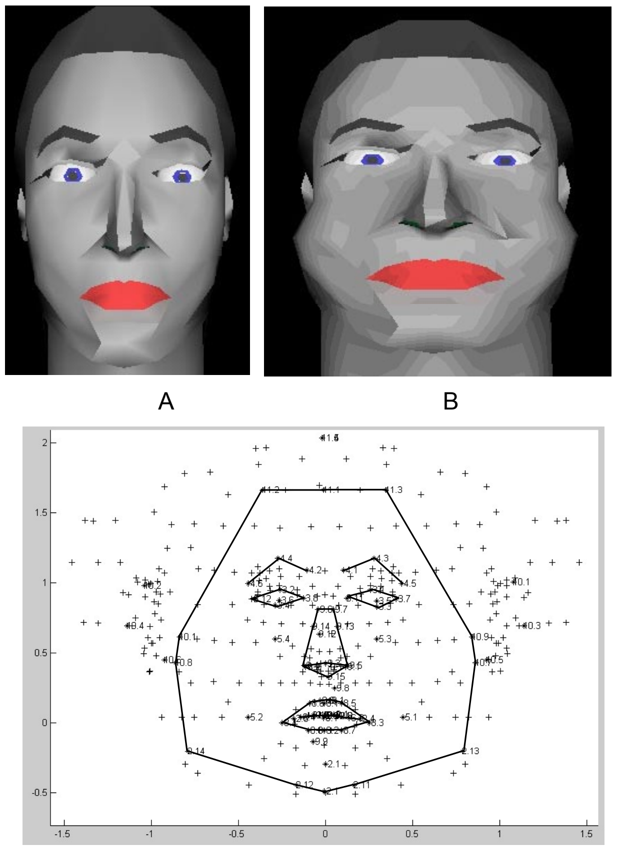


Figure 6: Normalizing and projecting the face. A: The face in initial state. B: Normalized face. C: The face projected into 2D. Facial features are outlined for better visibility.

In the final alignment, we move the feature points of the target face onto the corresponding feature points of the source face. The non-feature points are then *pulled* into position by the feature points, i.e. the position of each non-feature point is obtained by linear interpolation of the three surrounding feature points using barycentric coordinates.

We know the facial motion vector for each vertex of the source face, and we have mapped these vertices into the 2D alignment space. To obtain the motion vector for each vertex of the target face, we again interpolate from three surrounding vertices of the source face using the barycentric coordinates.

Once we have the facial motion vectors for each vertex of the target face, we simply add them to the vertex positions and de-normalize the target face. This is the animated target face.

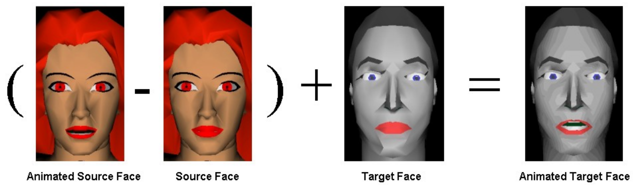


Figure 7: Overview of Facial Motion Cloning

The Morph Target Compositor (see Figure 3) is an automatic tool used to compose a facial model with animation based on a series of separate VRML [26] files containing morph targets. This means that each file contains the face with a different expression corresponding to high- and low-level MPEG-4 FAPs. For full MPEG-4 FBA compatibility, there is a total of 87 morph target files that correspond to high- and low-level MPEG-4 FAPs: the neutral face, 6 expressions (anger, fear, surprise, disgust, happiness and sadness), 14 visemes and 66 low-level FAPs as defined in MPEG-4. The final model that is used by the player is in a semi-proprietary format. It is a standard VRML file containing the animation information in Interpolator nodes. This information is registered in a compact way that can be interpreted by the player. This means that the model will be visible in any VRML player, albeit static; in order to view animations the Facial Animation Player has to be used.

We present the results of cloning on three face models. All motions were cloned from each model to each other model, producing a full grid of cloned animated models for each motion. One such grid is shown for the surprise expression in Figure 8. In this grid, the main diagonal contains the source faces, and the rest of each row shows the results of cloning from that source face to all other face models. Therefore, looking at each row shows how an expression is cloned from one face to all other faces; looking at each column shows how the expression is cloned onto the same face from different sources.

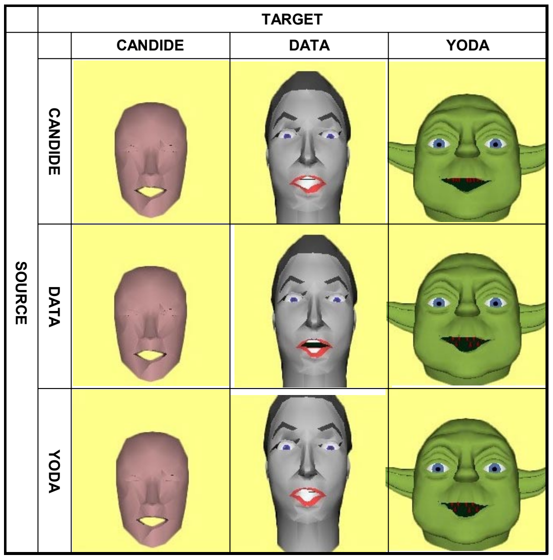


Figure 8: Cloning grid for surprise expression.

6. Producing Facial Animation Content

Choosing the MPEG-4 standard for the player makes it open for a wide variety of content sources that can be mixed and matched from various developers/vendors (see Figure 3). We foresee a broad spectrum of solutions varying in quality, cost and production time. Table 3 shows the foreseen classes of solutions with a comparison along these three parameters. This is a very general estimation and the particular implementations may vary, in particular as tools become more and more advanced.

Text-to-Speech (TTS) combined with facial animation, usually termed Visual TTS (VTTS), relies on phoneme timing information from the TTS to produce lip synchronization. It is particularly suitable for real time interactive applications as it can be fully

.

automated so it does not require human intervention. A drawback is the lack of expressiveness as only lips are animated, though this is usually improved by introducing random or rule-based facial motion, eye blinks and expressions. Another issue is the low speech quality of average TTS systems. Very high quality TTS systems are already available, but they are fairly expensive.

Table 3: Comparison of facial animation content production tools

Content production tool	Production cost	Production time	Animation quality
Visual text-to-speech	low	fast (real time)	fair
Automatic speech sync	low	fast (near real time)	fair
Video tracking	medium	medium	medium
Optical tracking	high	long	high
Manual animation	very high	very long	very high

Automatic speech synchronization relies on audio signal processing to synchronize the lips to the voice of a human speaker. This improves the sound quality from TTS, but obviously eliminates fully automatic applications as a human speaker is needed. The expressiveness problem is similar to the VTTS case.

Video tracking of facial features has been a challenging problem for a long time, and still remains without a full solution, though recent implementations [1] approach real time tracking while keeping a good tracking quality.

Optical tracking, using reflective markers glued on the face and a battery of calibrated cameras, can give high tracking quality. It involves expensive equipment and usually off-line processing increasing production time.

Manual animation still produces the highest quality, and will probably continue to do so in foreseeble future, albeit at a cost and production time that limits its application scope.

7. Conclusions

We have presented a framework for implementing applications involving talking virtual characters on the Web and mobile platforms. The framework consists of interchangable modules sharing standard data exchange formats (MPEG-4, VRML). It is biased towards the computer animation community by building upon the usual ways of face model/animation production. We believe that this framework:

- should appeal to the computer animators, which is important for quality content
- should easily adapt to a very broad range of applications with varying requirements
- is simple, robust and has low resource requirements on the target platform
- is easy to port to many different platforms

8. Acknowledgements

This research is partly supported by the VISIT programme of the Swedish Foundation for Strategic Research (SSF).

9. References

[1] J. Ahlberg, "Using the Active Appearance Algorithm for Face and Facial Feature Tracking," 2nd International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Realtime Systems (RATFFG-RTS), pp. 68 - 72, Vancouver, Canada, July 2001.

[2] A.L.I.C.E. natural language A.I. parser and chat robot, www.alicebot.org

[3] Kiyoshi Arai, Tsuneya Kurihara, Ken-ichi Anjyo, "Bilinear interpolation for facial expressions and methamrphosis in real-time animation", The Visual Computer, 12:105-116, 1996.

[4] 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.

[5] Cosatto E., Graf H.P., "Sample-Based Synthesis of Photo-Realistic Talking Heads", Proc. Computer Animation '98, Philadelphia, USA, pp. 103-110.

[6] Chadwick, "Layered construction for deformable animated characters", Computer Graphics, 23(3):234-243,1989

[7] P. Eisert, S. Chaudhuri and B. Girod, "Speech Driven Synthesis of Talking Head Sequences," 3D Image Analysis and Synthesis, pp. 51-56, Erlangen, November 1997.

[8] M. Escher, I.S. Pandzic, N. Magnenat-Thalmann, "Facial Deformations for MPEG-4", Computer Animation 98, Philadelphia, USA, pp. 138-145, IEEE Computer Society Press, 1998.

[9] Robert Forchheimer and Olov Fahlander, "Low Bit-rate Coding through Animation", Proceedings Picture Coding Symposium 83

[10] Robert Forchheimer, Olov Fahlander and Torbjörn Kronander, "A Semantic Approach to the Transmission of Face Images," Proceedings Picture Coding Symposium 84

[11] ISO/IEC 14496 - MPEG-4 International Standard, Moving Picture Experts Group, www.cselt.it/mpeg

[12] Kalra P., Mangili A., Magnenat-Thalmann N., Thalmann D., Simulation of Facial Muscle Actions based on Rational Free Form Deformation", Proceedings Eurographics 92, pp. 65-69

[13] N. Magnenat-Thalmann, N.E. Primeau, D. Thalmann, "Abstract muscle actions procedures for human face animation", Visual Computer, 3(5):290-297, 1988.

[14] Jun-yong Noh, Ulrich Neumann, "Expression Cloning", Proceedings of SIGGRAPH 2001, Los Angeles, USA

[15] Igor S. Pandzic, Joern Ostermann, David Millen, "Synthetic Faces: What are they good for?", The Visual Computer, 1999.

[16] Igor S. Pandzic, Gael Sannier, "From Photographs to Interactive Virtual Characters on the Web", Proc. Scanning 2000, Paris, France

[17] Igor S. Pandzic, "Life on the Web", Software Focus Journal, 2(2):52-59, John Wiley & Sons, 2001.

.

[18] I.S. Pandzic,”A Web-Based MPEG-4 Facial Animation System”, Proc. ICAV 3D 2001, demonstration at www.icg.isy.liu.se/~igor/MpegWeb

[19] F.I. Parke, “A Parametric Model for Human Faces”, PhD Thesis, University of Utah, Salt Lake City, USA, 1974. UTEC-CSc-75-047

[20] F.I. Parke, “Parametrized models for facial animation”, IEEE Computer Graphics and Applications, 2(9):61-68, November 1982.

[21] F.I. Parke, K. Waters, “Computer Facial Animation”, A K Peters Ltd. 1996., ISBN 1-56881-014-8

[22] Pearson, “Development in Model-Based Video Coding”, Proc. of the IEEE, 83(6):892-906, June 1995.

[23] S.M. Platt, N.I. Badler, “Animating Facial Expressions”, Computer Graphics, 15(3):245-252, 1981.

[24] Quartz Version 6.0, Symbian Technical Paper, Symbian Developer Network, www.symbiandevnet.com/techlib/techcomms/techpapers/papers/v6/over/quartz/index.html

[25] Shout 3D, Eyematic Interfaces Incorporated, http://www.shout3d.com/

[26] VRML, ISO/IEC 14772-1:1999, www.web3d.org/fs_specifications.htm

[27] Tekalp M.A., Ostermann J., “Face and 2-D Mesh Animation in MPEG-4”, Image Communication Journal, Tutorial Issue on MPEG-4 Standard, Elsevier, 2000.

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

[29] Weizenbaum, J., “ELIZA - A computer program for the study of natural language communication between man and machine”, Communications of the ACM 9(1):36-45, 1966.

[30] K. Waters, “A muscle model for animating three-dimensional facial expressions”, Computer Graphics (SIGGRAPH’87), 21(4):17-24, 1987.

[31] W Interactive SARL, www.winteractive.fr