Available online at www.sciencedirect.com

ScienceDirect

Graphical Models 69 (2007) 106–118

Graphical Models

www.elsevier.com/locate/gmod

Facial motion cloning with radial basis functions in MPEG-4 FBA

Marco Fratarcangeli [a,b,*], Marco Schaerf [a], Robert Forchheimer [b]

[a] University of Rome "La Sapienza", Department of Computer and Systems Science, Via Salaria 113, 00196, Rome, Italy
[b] Linköping Institute of Technology, Department of Electrical Engineering, Image Coding Group, 581 83 Linköping, Sweden

Received 25 November 2004; received in revised form 3 April 2006; accepted 25 September 2006
Available online 12 December 2006

Abstract

Facial Motion Cloning (FMC) is the technique employed to transfer the motion of a virtual face (namely the *source*) to a mesh representing another face (the *target*), generally having a different geometry and connectivity. In this paper, we describe a novel method based on the combination of the Radial Basis Functions (RBF) volume morphing with the encoding capabilities of the widely used MPEG-4 Facial and Body Animation (FBA) international standard. First, we find the morphing function $G(P)$ that precisely fits the shape of the source into the shape of the target face. Then, all the MPEG-4 encoded movements of the source face are transformed using the same function $G(P)$ and mapped to the corresponding vertices of the target mesh. By doing this, we obtain, in a straightforward and simple way, the whole set of the MPEG-4 encoded facial movements for the target face in a short time. This animatable version of the target face is able to perform generic face animation stored in a MPEG-4 FBA data stream.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Facial animation; Morph targets; MPEG-4 FBA; Virtual characters; Virtual humans

1. Introduction

Accurate animation of virtual actors is required in a wide range of fields like in entertainment, in distance learning and teleconferencing as well as in advanced human–machine interfaces. Currently, the best-looking animations of synthetic faces are done manually by visual artists or are tracked using motion capture equipment. Facial animation of characters is among the most tedious parts of creation of animations, it is time-consuming and it requires skilled animators while motion tracking hardware can be quite expensive. In order to save time and money, a trend is to develop methods to automatically simulate in the most realistic way the facial movements and then perform them in real-time. This is a challenging task because, on one side, the virtual faces present different characteristics (connectivity, shape), making difficult to define precisely an input space. On the other side, the human visual system is very sensitive to the

[*] Corresponding author. Fax: +39 06 85300849.
E-mail addresses: frat@dis.uniroma1.it (M. Fratarcangeli), schaerf@dis.uniroma1.it (M. Schaerf), robert@isy.liu.se (R. Forchheimer).
URLs: http://www.dis.uniroma1.it/~frat (M. Fratarcangeli), http://www.dis.uniroma1.it/~schaerf (M. Schaerf), http://www.icg.isy.liu.se/staff/robert (R. Forchheimer).

nuances of the perceived facial movements. Thus, the produced animation must be as accurate as possible, otherwise the risk is to loose the meaningful information hard-coded into the facial movements. As a last point, the computation to achieve the animation must not consume precious hardware resources like CPU-time or memory. This issue arise especially when the animation must be performed on modest equipped platforms, like mobile phones or handheld devices, or in the most demanding applications like games.

In this paper, we describe a Facial Motion Cloning (FMC) method that assists in solving these problems. FMC is used to copy the facial motion from one face to another. The facial movements are represented by a set of morph targets encoded by the MPEG-4 Facial and Body Animation (FBA) standard. A morph target is a variation of the face that has the same mesh topology, but different vertex positions. Essentially, it is the source face performing a particular key position. Each morph target corresponds to a basic facial action encoded by a MPEG-4 FBA parameter. The linear interpolation of the morph targets is able to represent a wide range of facial movements. Thus, an artist could produce one detailed face including all morph targets, then use this FMC implementation to quickly produce the corresponding full set of morph targets for a new and completely different face. The morph targets of the source face can be produced manually by artists, by motion capture techniques or in an automatic way, for example by applying physical simulation algorithms of the face anatomy [11,12].

There are two main advantages in using morph targets encoded by MPEG-4 FBA. The first one is in the low requirements of CPU-time usage, since the animation is achieved by linear interpolation of the morph targets. The second one lies in the capability of the MPEG-4 FBA to express and precisely control the whole range of facial expressions [24]. Thus, by computing the morph targets for the target face, we can use them to perform generic face animation encoded in a MPEG-4 FBA stream in a computationally inexpensive manner.

2. MPEG-4 facial and body animation standard

We provide a short introduction to MPEG-4 FBA specification, describing the main parameters sets cited in this paper. Readers familiar with MPEG-4 FBA may wish to skip the section, or use it only as a quick reference.

MPEG-4 is an international standard dealing with the efficient transmission of animation. A significant part of the MPEG-4 is the Face and Body Animation (FBA), a specification for efficient coding of shape and animation of human faces and bodies [24,1]. Instead of regarding animation as a sequence of images of faces with fixed shape and size, MPEG-4 FBA gives the possibility to express, for each animation frame, the facial movements through a set of weighted parameters. Thus, the animation is synthesized by a proper deformation of the face mesh according to such parameters.

MPEG-4 FBA specifies a face model in its neutral state together with two main data sets: 84 Facial Definition Parameters (FDPs), also called feature points, and 68 Facial Animation Parameters (FAPs). According to the standard, a generic face model is in neutral state when the gaze is in direction of the z-axis, all face muscles are relaxed, eyelids are tangent to the iris, lips are in contact and the mouth is closed in such a way that the upper teeth touch the lower ones [24].

FDPs (Fig. 1) are control points used to define the shape of a proprietary face model and to provide a spatial reference for defining FAPs. The location of FDPs has to be known for any MPEG-4 compliant face model. The FAP set is subdivided in high- and low-level parameters. Low-level FAPs are used to express basic action that the face can perform, like closing an eyelid or stretching a corner lip. All low-level FAPs are expressed in terms of the Facial Animation Parameter Units (FAPU). They correspond to distances between key facial features and are defined in terms of distances between the FDPs. High-level FAPs are useful to express in a compact way more complex movements like expressions and visemes.[1] FAPs allow for representation of the whole set of natural facial movements.

The FBA parameters, both FDPs either FAPs, can be extracted automatically from visual, audio and motion capture systems [4,3,13,16], therefore they can be properly compressed and used to achieve, for example, teleconferencing with a very low bandwidth (<3 kbps) [21,7].

3. Related work

Several different methods have been proposed through the years to achieve computer facial

[1] A viseme is the visual counterpart of a phoneme.

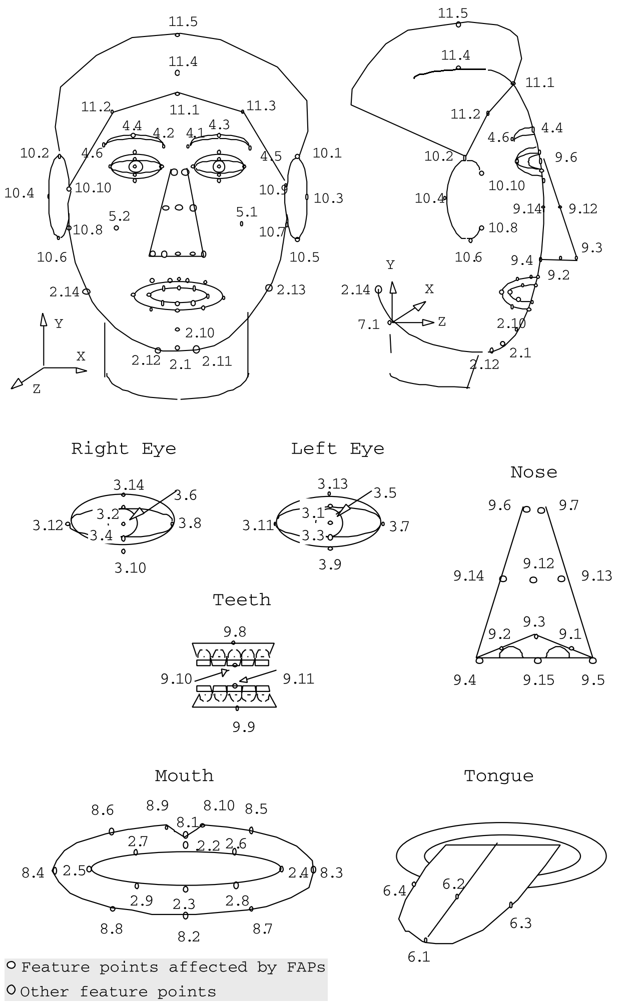

Fig. 1. MPEG-4 Facial Definition Parameters (or Feature Points) are used to define the shape of a face model [24,1].

animation. Key-frame interpolation technique proposed by Parke [25] is one of the most intuitive and still widely used. Basic expressions in 3D are defined at different moments in the animated sequence and intermediate frames are simply interpolated between two successive basic expressions. Physics-based approaches [17,18,26,14,15] animate faces by simulating the influence of muscle contraction onto the skin surface. The general approach is to build a model respecting the anatomical structure of the human head. By doing this, the produced motion is very realistic. The drawback is that these methods involve a massive use of CPU-resources to advance the simulation. Thus, they are not suitable for interactive applications on modestly equipped platforms.

Synthesis of facial motion by reusing existing data (Facial Motion Cloning), has recently become popular in computer graphics. Briefly, the main concept is to transfer the motion from a *source* face to a static *target* face, making this latter face animatable. Learning-based methods rely on a dataset of 3D scans of different facial expressions and mouth shapes to build a morphable model [6,5], that is, a vector space of 3D expressions. Difference vectors, such as a smile-vector, can be added to new individual faces. Unlike physical models, these methods address the appearance of expressions, rather than simulating the muscle forces and tissue properties that cause the surface deformations.

Escher et al. [8,9], developed a cloning method based on Dirichlet Free Form Deformation (FFD) and applied it to the synthesis of a virtual face in order to obtain a virtual avatar of a real human head. In FFD algorithms, the deformation is controlled by a few external points. To achieve volume morphing between the source and the target face meshes, the control points are usually difficult to define and not very intuitive for manipulation. Deformation based on scattered data interpolation, like Shepard-based methods and radial basis function methods [10], are easier to manipulate for our purpose. In particular, radial basis functions are more effective when landmarks are scarce and they are also computationally efficient.

To develop our method, we were inspired mainly by the two following methods. In Expression Cloning developed by Noh [20], the movements of a source face are expressed as motion vectors applied to the mesh vertices. The source mesh is morphed, through the use of RBF volume morphing and neural networks, to match the shape of the target mesh. The motion vectors are transferred from source model vertices to the corresponding target model vertices. The magnitude and direction of the transferred motion vectors are properly adjusted to account for the local shape of the model. The FMC approach developed by Pandzic [23] relies on the fact that the movements of the source face are coded as morph targets. In Pandzic's approach, each morph target is described as the relative movement of each vertex with respect to its position in the neutral face. Facial cloning is obtained by computing the difference of 3D vertex positions between the source morph targets and the neutral source face. The facial motion is then added to the vertex positions of the target face, resulting into the animated target face. The key positions represented by morph targets are expressed by the MPEG-4 Facial Animation Parameters (FAPs). Each morph target corresponds to a particular value of one FAP. By interpolating the different morph targets frame-by-frame, animation of the source face is achieved.

4. Our approach

Our FMC method is schematically represented by Figs. 2 and 3 and can be summarized as follows.

Given a manually picked set of 48 feature points on the input face meshes, we compute a scattered data interpolation function $G(P)$ employed to precisely fit the shape of the source into the shape of the target mesh through a volume morphing. Then, we map each vertex of the target face into the corresponding triangular facet of the deformed source mesh in its neutral state through a proper projection. The target vertex position is thus expressed in a function of the vertex positions of the source triangular facet through barycentric coordinates. At this point, we deform all the source MTs by applying to each one of them the same morphing function $G(P)$ used to fit the neutral source into the neutral target mesh. Hence, we can compute the new position of each target vertex considering the location of the corresponding deformed source facet, obtaining the target MT. The whole set of morph targets is called Animatable Face Model (AFM) and it can be directly animated by commercial MPEG-4 FBA players [2].

Inputs to the method are the source and target face meshes. The source face is available in neutral

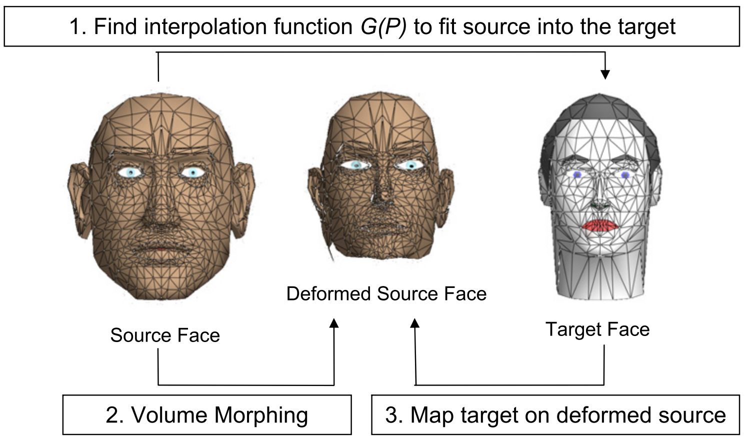
1. Find interpolation function G(P) to fit source into the target
Source Face
Deformed Source Face
Target Face
2. Volume Morphing
3. Map target on deformed source

Fig. 2. Facial Motion Cloning mechanism. A RBF volume morphing $G(P)$ is performed between the source and the target face.

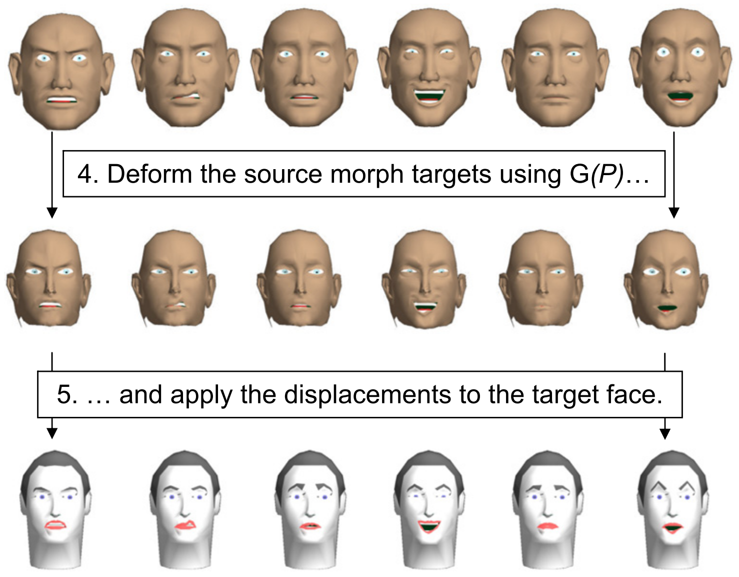
4. Deform the source morph targets using G(P)…
5. … and apply the displacements to the target face.

Fig. 3. Facial Motion Cloning mechanism. Using the same deformation function $G(P)$, all the source morph targets are cloned to the target face.

state as defined in the MPEG-4 FBA specification. The morph targets (MTs), corresponding to the MPEG-4 FAPs, of the source face are available as well. The target face exists only in the neutral state. For each neutral face mesh, the corresponding MPEG-4 Feature Definition Point (FDP) set must be defined, that is, each FDP is manually mapped onto a vertex of the input face meshes. We need 48 out of 84 FDPs since we do not consider the FDPs corresponding to group 10 (ears), 6 (tongue), FDPs from 9.8 to 9.11 (teeth) and 11.5 (top of the head). The process to manually map the 48 FDPs on the input faces requires 10–30 min according to the user skills. The goal is to obtain the target face with the motion copied from the source face. We do not make any assumption about the number of vertices or their connectivity in the input models. For simplicity, we will consider source and target as triangular meshes.

The remainder of this paper is organized as follows. In Section 5, we detail how to find the fitting function $G(P)$ and in Section 6 we describe the cloning process. Section 7 presents our experimental results and in Section 8 we conclude by providing directions for future work.

5. Shape fitting

The task of the shape fitting process is to adapt the generic source face mesh to fit the target face mesh. As input, we take the source and target face meshes. The source and the target faces are both available in neutral position as defined in the MPEG-4 FBA specification (Section 2 and [24]). For each neutral face mesh, the corresponding subset of 48 MPEG-4 Facial Definition Points (FDPs) must be defined as specified in Section 4. Taking these feature points as a reference, we are able to extract some face features, that is, the eye contour and the inner lip contour (Section 5.1). We use the FDPs together with the vertices belonging to the eye and lip contour features to find a first guess of the scattered data interpolation function $G(P)$ that roughly fits the source into the target mesh (Section 5.2). The precise fitting of the moving zone of the deformed source model is obtained by iteratively refining $G(P)$ (Section 5.3).

5.1. Eye and lip feature extraction

Starting from the manually picked MPEG-4 FDPs, we extract the eye and lip features from the input meshes through automatic algorithms.

Driven by some of the MPEG-4 FDPs belonging to the eye group, we find a proper path in the mesh graph going from a start point to an end point, where the start and the end point are defined according to the round robin UP ⇒ LEFT ⇒ DOWN ⇒ RIGHT ⇒ UP. To find the path between the start and the end point, we use a greedy strategy:

```
Right Eye FDPs: 3.2 ⇒ 3.8 ⇒ 3.4 ⇒ 3.12 ⇒ 3.2
Left Eye FDPs: 3.1 ⇒ 3.7 ⇒ 3.3 ⇒ 3.11 ⇒ 3.1

v = start_point
while (v ≠ end_point)
   $d_{min}$ = very big value
   for each neighbor n of v do
      $n_n$ = normalize (n − v)
      d = distance ((v + $n_n$), end_point)
      if ($d_{min}$ > d)
         $d_{min}$ = d
         $v_{min}$ = n
      end if
   end for
   insert $v_{min}$ in the eye contour set
   v = $v_{min}$
end while
```

We find the inner lip contours by applying the following algorithm, once for the upper lip and once for the lower one:

```
start_point = FP 2.5 (inner right corner lip)
end_point = FP 2.4 (inner left corner lip)
direction = end_point − start_point
$direction_{xy}$ = ($direction_x$, $direction_y$, 0)
v = start_point
insert v in the lip contour set
while (v ≠ end_point)
   for each neighbor n of v
      a = angle ($direction_{xy}$, $(n − v)_{xy}$)
   end for
   v = n having smaller [greatest] a, a∈ $(-\frac{\Pi}{2}, \frac{\Pi}{2})$
   insert v in the upper [lower] lip contour set
end while
```

Note that a MPEG-4 FBA compliant synthetic face has the gaze and the nose tip towards the positive z-axis and that the lips are in contact but they are not connected [24]. We do not provide a method to assess the correctness of such extraction algorithms, however they work in a satisfactory way with all our test face models.

5.2. Scattered data interpolation

Having computed the face feature points on both the source and the target face mesh in their neutral state, we construct a smooth interpolation function $G(P)$ that precisely fits the source model into the target face model. In mathematical terms, for given data pairs (P_i, Q_i), $(i = 1, \dots, n, P_i \in \mathbb{R}^3, Q_i \in \mathbb{R}^3)$, find a function $G : \mathbb{R}^3 \to \mathbb{R}^3$, $G \in \mathscr{C}^1$ so that

$$Q_i = G(P_i) \qquad i = 1, \dots, n \tag{1}$$

In our case, $\{P_i\}$ is a subset of the vertices of the *source* mesh and $\{Q_i\}$ is the subset of correspondent vertices on the *target* mesh. Applying $G(P)$ to the whole set of vertices of the source mesh, this latter will deform in such a way that it will fit the target shape.

A radial basis function is defined as a basis function whose value depends only on the distance from the data point. A radial basis function method is simply the linear combination of such basis functions:

$$G(P) = \sum_{i=1}^{n} h_i R_i(P) \tag{2}$$

where h_i are 3D coefficients, R_i are the radial basis functions and $P \in \mathbb{R}^3$ is a vertex of the source mesh. We use as basis function the Hardy multiquadrics, defined as:

$$R_i(P) = (d_i^2(P) + r_i^2)^{\frac{1}{2}} \tag{3}$$

where d_i is the euclidean distance from P_i, r_i is the stiffness radius controlling the stiffness of the deformation around P_i. The Hardy method is the combination of the sum of the Hardy multiquadrics and a linear term $\mathscr{L}$ used to absorb the linear part of the morphing transformation:

$$G(P) = \sum_{i=1}^{n} h_i \cdot (d_i^2(P) + r_i^2)^{\frac{1}{2}} + \mathscr{L}(P) \tag{4}$$

where $\mathscr{L}(P) = h_{n+1} \cdot x + h_{n+2} \cdot y + h_{n+3} \cdot z + h_{n+4}$ is the affine transformation, $\{h_i\}(h_i \in \mathbb{R}^3, i = 1, \dots, n+4)$ are called Hardy coefficients. Introducing the $\mathscr{L}(P)$ term, the source mesh will be translated, rotated and scaled according to the position of the target interpolation point set $\{Q_i\}$. For smooth results, the parameters $\{r_i\}$ are set to:

$$r_i = \begin{cases} \min d_i(P_j) & \text{if } i \neq j, \\ 0 & \text{if } i = j. \end{cases} \tag{5}$$

In order to find the volume morphing function $G(P)$, we define the interpolation point sets $\{P_i\}$ and $\{Q_i\}$ and then solve a linear equation system to compute the Hardy coefficients. The linear equation system to find the $\{h_i\}$ values is obtained by imposing the n conditions $G(P_j) = Q_j$:

$$Q_j = \sum_{i=1}^{n} h_i \cdot (d_i^2(P_j) + r_i^2)^{\frac{1}{2}} + \mathscr{L}(P) \qquad (j = 1, \dots, n) \tag{6}$$

Furthermore, for linear precision, we impose:

$$\sum_{i=1}^{n} h_i \cdot x_i = 0 \tag{7}$$

$$\sum_{i=1}^{n} h_i \cdot y_i = 0 \tag{8}$$

$$\sum_{i=1}^{n} h_i \cdot z_i = 0 \tag{9}$$

$$\sum_{i=1}^{n} h_i = 0 \tag{10}$$

This results in a linear system of $n+4$ equations with $n+4$ unknowns $\{h_i\}(i = 1,\ldots,n+4)$. The solution exists and it is unique [19].

Thus, $G(P)$ can be computed once the correspondence set $\{(P_i,Q_i)\}$ has been defined. The denser the correspondence set is the closer the resulting fit. We consider the 48 FDPs specified in Section 4 as a first guess of the interpolation point set $\{(P_i,Q_i)\}$. We compute $G(P)$ and apply it to the source mesh obtaining a rough fitting. Then, we enrich the correspondence set by inserting the vertices belonging to the eye and lip contours of the source and the point lying on the nearest edge of the correspondent target contours. We recompute $G(P)$ with the enriched set of correspondences and apply it again on the source mesh in its neutral state obtaining again a rough fitting but this time with a correct alignment of the eye and lip features.

5.3. Correspondence set refinement

After morphing the source face mesh to roughly fit the target, we further improve the fitting by specifying additional correspondences. A vertex P of the source face is called a *movable* vertex, if its position in one of the morph targets is not equal to its position in the neutral face. Thus, such a vertex will potentially move during the animation. We project the movable vertices from the deformed source mesh on the target face surface by casting rays with a fixed length along the vertex normals[2] and compute the intersection of the ray with the target mesh. We found a fixed length of $\text{ENS0} \times 0.3125$ for the casted rays, where ENS0 is the MPEG-4 FAPU defining the distance between the eye and nose.

[2] The normal of a vertex is considered as the average of the normals of the faces the vertex is part of.

By doing this, we are able to know for each movable vertex of the source face P_i, the corresponding intersection point Q_i on the target surface mesh. Having chosen a fixed length, only a part of the movable vertices will have a corresponding point on the target surface. This is because, after the initial rough fit, only the nearest vertices will be close enough to the target mesh surface to permit a ray-facet intersection. We also experimented using outward rays with unlimited length. We achieved better results with the fixed ray length, probably because in this way the interpolated surface smoothly stick to the target surface without inserting high-frequency elements.

We consider the first 125 moving vertices having greatest error $e_i = \|Q_i - P_i\|$ and we insert the pair (P_i, Q_i) in the correspondence set. If a point P_i is already in the set, then only the position of the correspondent Q_i is updated. We solve again the linear system Eqs. (6)–(10) to find $G(P)$ for the enriched correspondence set and we re-run the scattered data interpolation algorithm to update the whole source face model from its neutral state. This process is iterated until no more movable vertices are inserted in the correspondence set. This may happen for two different reasons:

- all the movable vertices have been inserted in the correspondence set;
- the actual set has enough correspondence pairs to fit the source to the target mesh, sufficiently well.

By applying the refined $G(P)$ to all the vertices of the neutral source face mesh, this latter fits precisely the target mesh in the area where there are vertices that potentially move when the animation is performed. We consider only the moving zone because we want to copy the source face motion and we are not interested in the static zones. Fig. 4 shows an example of this iterative fitting process for some test models. In the central column, the source face is solid while the target mesh is wireframe rendered. Step 0 is the initial rough fit. In Step 1, the eye and the lip features are aligned. After some iterative steps, the source fits to the target face in the moving zone.

6. Cloning process

As input of the motion cloning process, we need the scattered data interpolation function $G(P)$ just

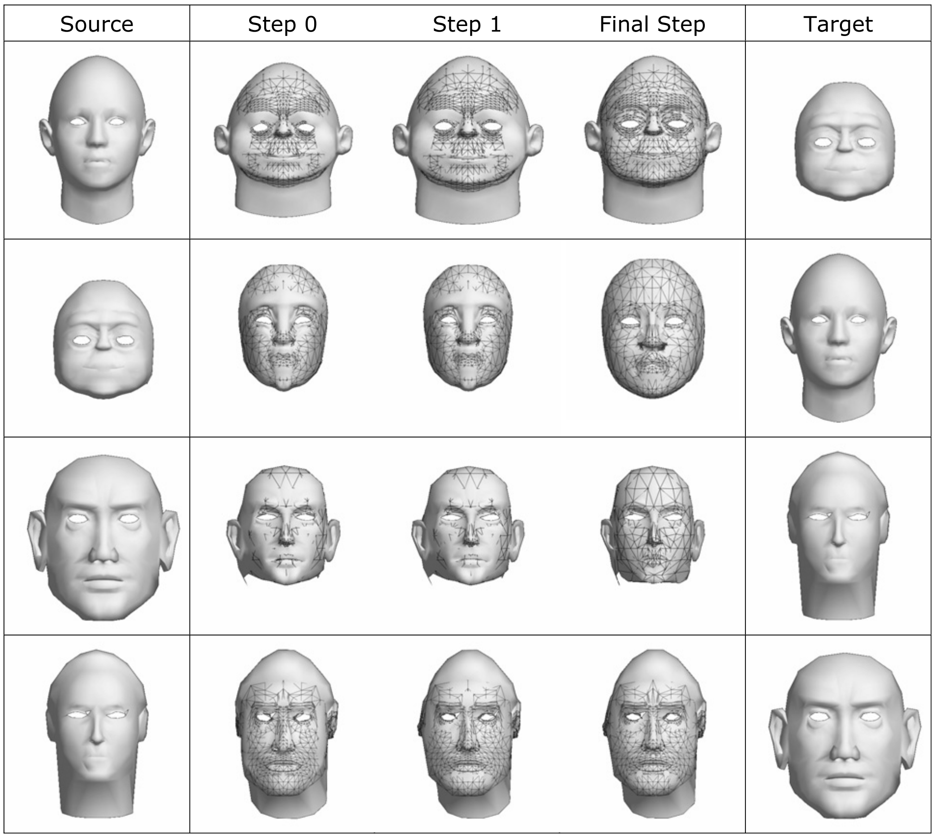

Fig. 4. Source shape fitting iterative process.

refined and the morph targets, corresponding to the MPEG-4 FAPs, of the source face. To be precise, we need the 64 MTs corresponding to low-level FAPs and the 20 MTs corresponding to high-level FAPs (14 visemes and 6 emotions), for a total of 84 MTs. We map the target vertices on the deformed source mesh through the same projection used in Section 5.2, this time casting the fixed-length rays from the target vertices towards the deformed source mesh. At this stage of the process, the deformed source and the target meshes are very similar to each other and each target vertex can be considered as located where the casted ray intersects the proper triangular face of the source mesh. Thus, we can compute the barycentric coordinates of each target vertex considering the vertices of the correspondent source triangular facet. The position of the target vertices can be expressed as a linear combination of the positions of the three corresponding source vertices.

Then, for each particular source MT S_i we compute the corresponding target MT T_i by applying the following algorithm:

```
$T_i$ = neutral target face (stored in $T_0$);
apply $G(P)$ to $S_i$ only in each vertex $P \in D_i$;
for each vertex $Q \in T_i$
    if ($P_a \in D_i$) $\vee$ ($P_b \in D_i$) $\vee$ ($P_c \in D_i$)
        $Q = P_a \cdot b + P_b \cdot c + P_c \cdot a$;
end for;
```

where D_i is the set of moving vertices of S_i, P_a, P_b and P_c are the vertices of the source triangular facet corresponding to the vertex Q and a, b and c are the proper barycentric coordinates. Note that we apply the scattered data interpolation function $G(P)$ *only to the movable vertices* D_i of each particular source MT S_i in order to speed up the cloning process. That is, we apply the same global deformation function $G(P)$ used to fit the source into the target, to the local part of the source MTs S_i that will move when S_i is employed during the animation. Then, we compute the position of the corresponding target vertices by linear combination of the barycentric coordinates with the new positions of the source triangular facet obtaining, finally, the resulting target MT T_i.

As a final step, we clone the global motion of the head as well as the movements of the tongue and teeth (if present) through affine transformations like in Pandzic [23].

7. Results

We performed our experiments on an Intel Pentium M 1.73 GHz processor with 512 MB RAM. Some of the face models that we used for the experiments are shown in Fig. 5. The test models have different polygonal resolutions, shapes and connectivity, both symmetric as well as asymmetric. A full set of high- and low-level FAP motions (i.e., morph targets), was available for each model. All motion was cloned from model `joakim` and `beta` to each other model, producing the grids of cloned animated models for each motion in Figs. 6–9. Looking at each row shows how an expression is cloned from one face to all the other faces; looking at each column shows how the expression is cloned onto the same face from different sources.

The comparison in terms of computation time between Pandzic's method [23] and ours is straightforward, since we use the same input and produce the same kind of output. In Table 1, we present the computation time for some cloning processes performed during our tests along with other interesting data. For comparison, the last column presents the computation time for Pandzic's cloning approach on the same PC.

To assess the quality of our approach, we cloned the same source animatable faces to themselves and we computed the error between the source and the output animatable model as the average of the error of each of the 84 morph targets:

$$e = \left(\frac{1}{84} \sum_{j=1}^{84} \frac{\sum_{i=1}^{N_v} \| v_i^S - v_i^T \|}{\sum_{i=1}^{N_v} \| v_i^S \|} \right) \times 100 \tag{11}$$

where N_v is the number of the vertices of the face mesh, v_i^S and v_i^T are, respectively, the i-th vertex of

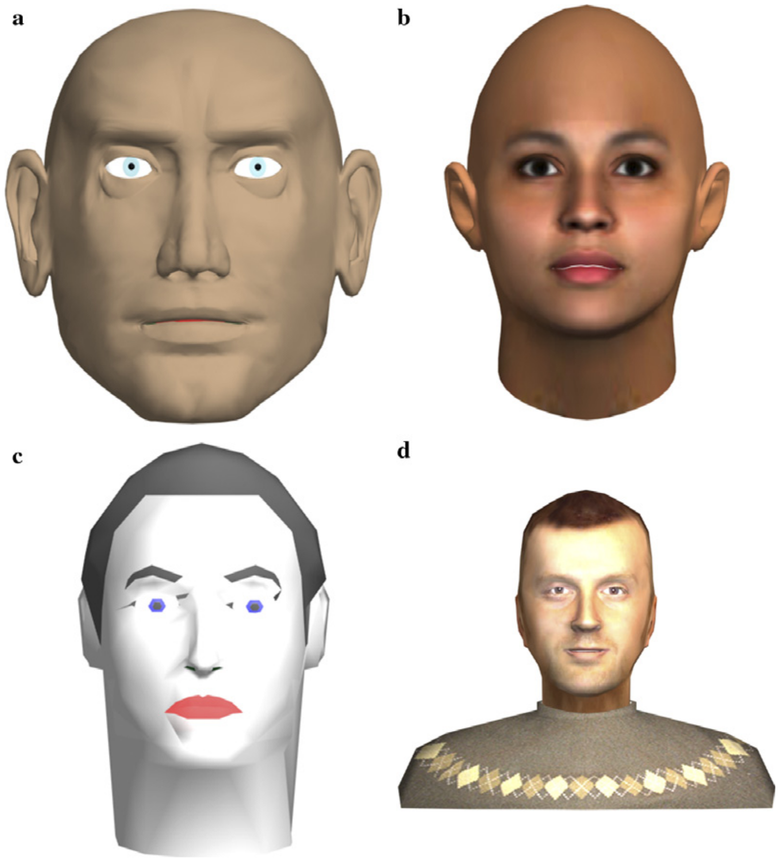
Fig. 5. Test models used in our experiments. (a) `joakim`, 909 vertices and 1752 faces; (b) `beta`, 2197 vertices and 4118 faces; (c) `data`, 367 vertices and 677 faces; (d) `kevin`, 498 vertices and 956 faces.

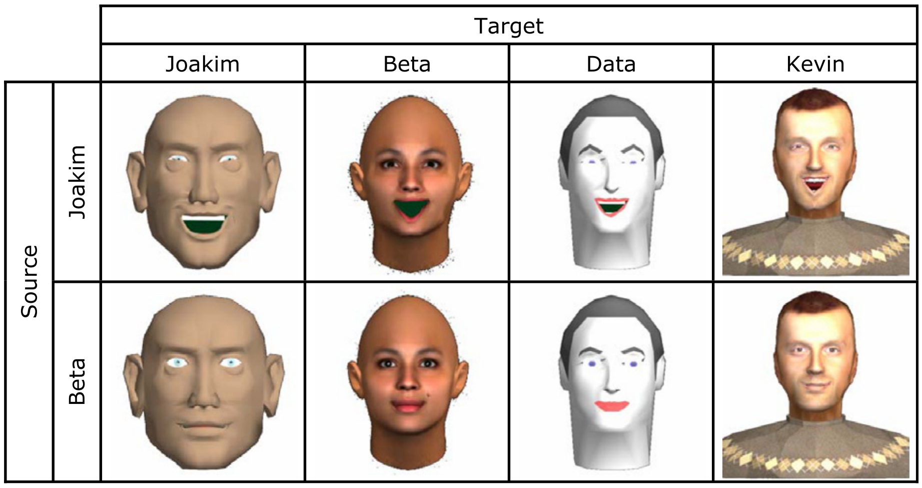

Fig. 6. Cloning grids for expression joy.

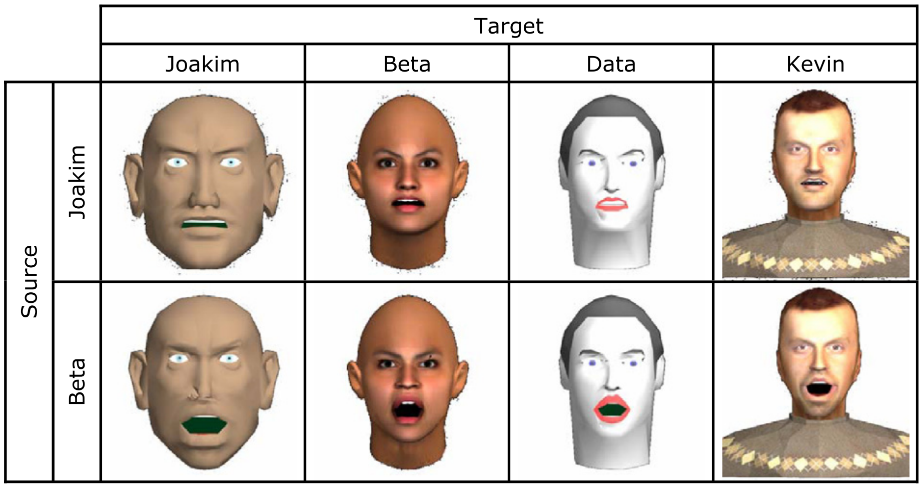

Fig. 7. Cloning grids for expression anger.

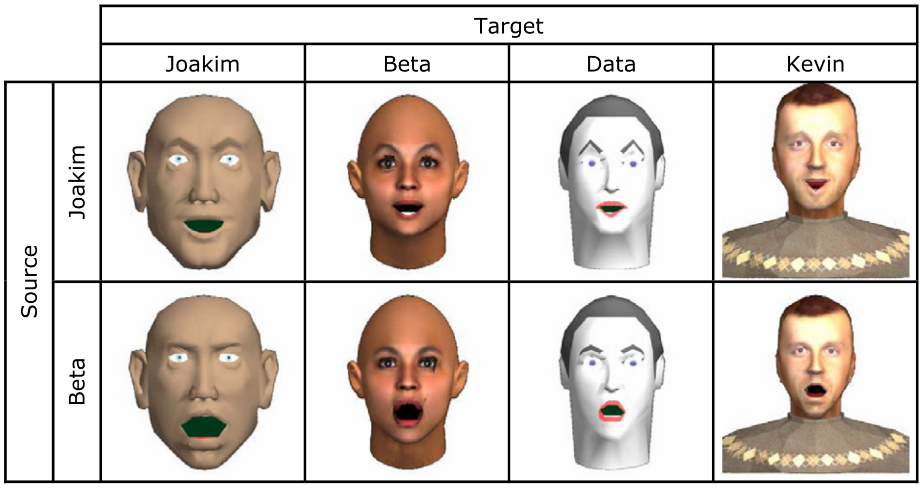

Fig. 8. Cloning grids for expression surprise.

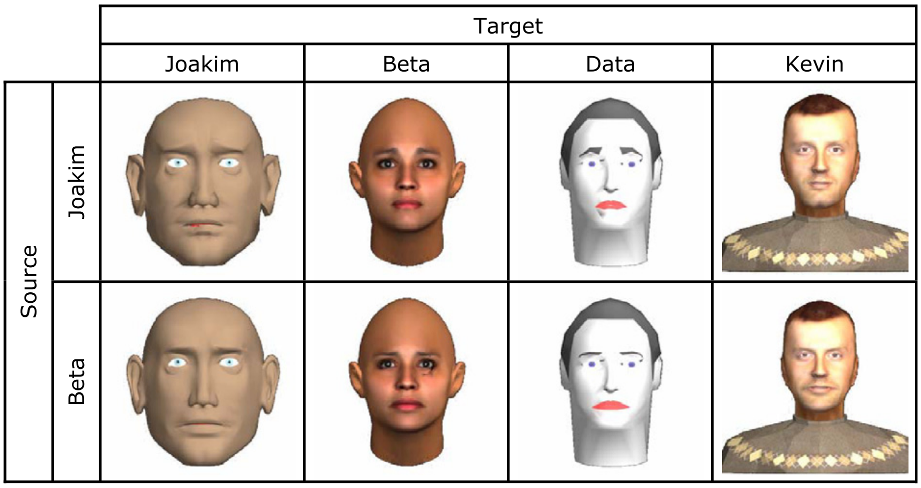

Fig. 9. Cloning grids for expression sadness.

Table 1
MVs, moving vertices; CPs, correspondence points; Is, iterations to refine; RT, $G(P)$ refinement time; CT, cloning time; TT, total time

	MVs	CPs	Is	RT (s)	CT (s)	TT (s)	TT_Pzc [23]
`beta` ⇒ `data`	396	377	8	4.0	1.0	5.0	5.3 s
`data` ⇒ `beta`	280	280	5	1.3	3.6	5.0	12.4 s
`joakim` ⇒ `data`	479	461	8	4.4	1.0	5.4	4.5 s
`data` ⇒ `joakim`	280	275	4	1.0	1.2	2.3	14.1 s
`beta` ⇒ `kevin`	396	382	9	4.6	0.8	5.4	9.1 s
`kevin` ⇒ `beta`	294	290	5	10.6	0.6	11.3	13.6

the source and the corresponding one on the cloned output model. Table 2 presents the obtained values. Fig. 10 shows the error distribution over the test models.

While differences between our visual results and [23,20] are subjective, we still can consider differences in implementation:

- Noh uses RBF, together with neural networks, to align the source to the target face and then the motion vectors for each target vertex are computed *locally* using proper affine transformations. In our approach, we need the RBF method $G(P)$ to align the source and the target with an iterative enrichment of the correspondence set. Then, $G(P)$ is reused to deform the source MTs and copy the motion on the target face. The fact that the cloning is carried out only where needed make this process computationally light and probably faster than Noh's approach. However, Noh's approach is almost fully automatic, while we still need to manually map 48 FDPs;
- our approach provide a MPEG-4 FBA compliant talking face able to perform generic animation, while Noh's approach is able to clone an animation given *a priori*.

Table 2
Average error on cloning source faces with themselves

	source ⇒ source
`beta`	0.028%
`data`	0.280%
`joakim`	0.084%
`kevin`	0.141%

8. Conclusions

We proposed a method dealing with the problem of reusing existing facial motion to produce in a short amount of time a ready-to-be-animated MPEG-4 FBA compliant talking head. Apart from an initial manual picking of 48 correspondence points all the techniques presented here are fully automatic. In terms of visual results shown, even though only a small subset of them could be presented here, we believe that most facial movements for expression and low-level FAPs are copied correctly to the target face models.

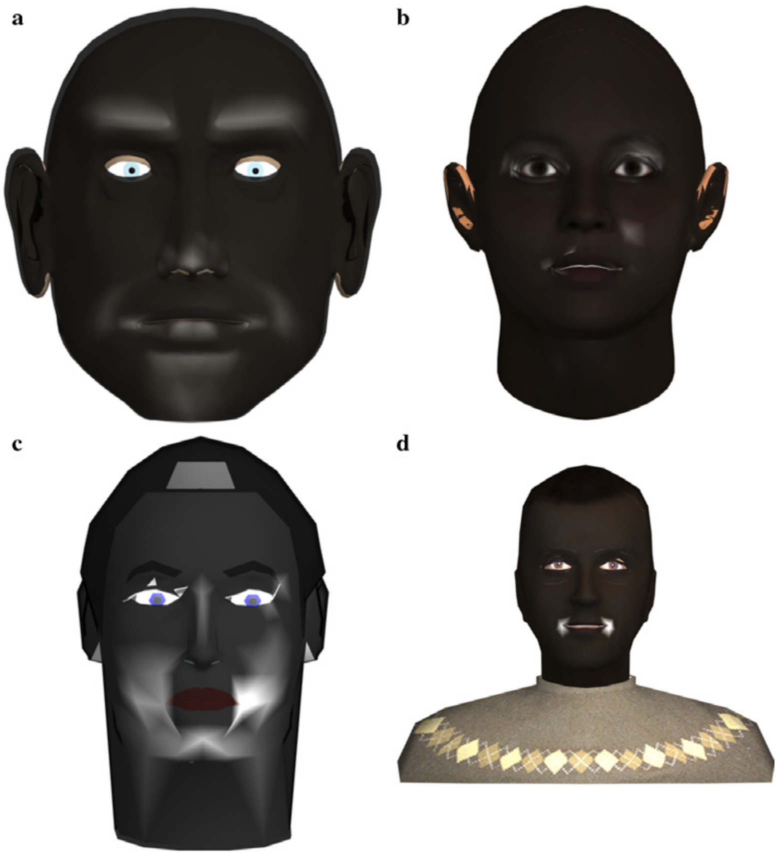

Fig. 10. Visual distribution of the error. Error magnitude is proportional to the color brightness.

One limitation of our method might be that the target mesh cannot be of higher resolution than the source otherwise the barycentric coordinates will just pull any vertices in the interior of that control triangle onto the plane of that triangle. However, using a high-resolution source should not cause major problems. In Figs. 6–9, higher-resolution models `joakim` and `beta` are cloned to lower resolution models `data` and `kevin`. The main computational effort during the cloning process lies in the refinement process of the $G(P)$ interpolation function. This is because each pair of correspondence points corresponds to a linear equation in the system to resolve in order to obtain $G(P)$. The asymptotic behavior of the linear equation-solving algorithm (LU decomposition) is $O(n^3)$, where n is the number of moving vertices of the source face. Since the correspondences can be very close to each other, we think that not all of them are necessary and, as a future work, we would identify and not consider the less significant ones. Furthermore, we would apply face feature tracking to the polygonal meshes in order to retrieve the FDPs making the whole process fully automatic.

After the cloning process is finished, the target face is ready to perform *generic* facial animation encoded into a MPEG-4 FBA stream. The computational cost of the animation depends on the player employed. In our case, we used the lightweight MPEG-4 FBA player provided by [2] in which the animation is achieved, for each frame, through linear interpolation between the proper morph targets and rendered in OpenGL. Thus, the computational cost is rather low. Combined with facial feature tracking, MPEG-4 FBA talking heads can potentially be used for a very low bitrate visual communication in a model-based coding scenario [7,22] (teleconferencing, games, web interfaces).

Acknowledgments

The authors thank Prof. Igor Pandzic and Dr. Jörghen Ahlberg for fruitful discussions. This work was supported by the European Union Research

Training Network MUHCI (HPRN-CT-2000-00111).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.gmod.2006.09.006.

References

[1] Moving Pictures Expert Group, MPEG-4 International standard, ISO/IEC 14496, <http://www.cselt.it/mpeg/>.

[2] Visage Technologies AB, URL: <http://www.visagetechnologies.com/>, March 2006.

[3] J. Ahlberg, Model-based coding—Extraction, coding and evaluation of face model parameters. Ph.d. thesis no. 761, Department of Electrical Engineering, Linköping University, Linköping, Sweden, September 2002.

[4] J. Ahlberg, F. Dornaika, Efficient active appearance model for real-time head and facial feature tracking, in: IEEE International Workshop on Analysis and Modelling of Faces and Gestures (AMFG), Nice, October 2003.

[5] V. Blanz, C. Basso, T. Poggio, T. Vetter. Reanimating faces in images and video, in: P. Brunet, D. Fellner (Eds.), Proceedings of EUROGRAPHICS, Granada, Spain, 2003.

[6] V. Blanz, T. Vetter, A morphable model for the synthesis of 3d faces, in: SIGGRAPH'99 Conference Proceedings, 1999.

[7] T.K. Capin, E. Petajan, J. Ostermann, Very low bitrate coding of virtual human animation in MPEG-4, in: IEEE International Conference on Multimedia and Expo (II), 2000, pp. 1107–1110.

[8] M. Escher, N. Magnenat-Thalmann, Automatic 3D cloning and real-time animation of a human face, in: Computer Animation, Geneva, Switzerland, June 1997.

[9] M. Escher, I.S. Pandzic, N. Magnenat-Thalmann, Facial deformations for MPEG-4, in: Computer Animation 98, IEEE Computer Society Press, Philadelphia, USA, 1998, pp. 138–145.

[10] S. Fang, R. Raghavan, J. Richtsmeier, Volume morphing methods for landmark based 3D image deformation, in: SPIE International Symposium on Medical Imaging, vol. 2710, Newport Beach, CA, 1996, pp. 404–415.

[11] M. Fratarcangeli, M. Schaerf, Realistic modeling of animatable faces in MPEG-4, in: Computer Animation and Social Agents, MIRALAB, Computer Graphics Society (CGS), Geneva, Switzerland, 2004, pp. 285–297.

[12] Marco Fratarcangeli, Physically based synthesis of animatable face models, in: Proceedings of the Second International Workshop on Virtual Reality and Physical Simulation (VRIPHYS05), Pisa, Italy, ISTI-CNR, The Eurographics Association, 2005, pp. 32–39.

[13] T. Goto, M. Escher, C. Zanardi, N. Magnenat-Thalmann, MPEG-4 based animation with face feature tracking, in: Computer Animation and Simulation '99, 1999, pp. 89–98.

[14] K. Kähler, J. Haber, H.-P. Seidel, Geometry-based muscle modeling for facial animation, in: Proceedings Graphics Interface 2001 (GI-2001), Morgan Kaufmann, Ottawa, Ontario, 2001, pp. 37–46.

[15] K. Kähler, J. Haber, H. Yamauchi, H.-P. Seidel, Head shop: Generating animated head models with anatomical structure, in: Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation, San Antonio, USA, ACM SIGGRAPH, 2002, pp. 55–64.

[16] W.-S. Lee, M. Escher, G. Sannier, N. Magnenat-Thalmann, MPEG-4 compatible faces from orthogonal photos, in: Proceedings of the Computer Animation, 1999, pp. 186–194.

[17] Y. Lee, D. Terzopoulos, K. Waters, Constructing physics-based facial models of individuals, in: Proceedings of the Graphics Interface '93 Conference, Toronto, ON, Canada, 1993, pp. 1–8.

[18] Y. Lee, D. Terzopoulos, K. Waters, Realistic modeling for facial animation, in: Computer Graphics Proceedings, SIGGRAPH 95, Annual Conference Series, Los Angeles, CA, ACM SIGGRAPH, 1995, pp. 55–62.

[19] C.A. Micchelli, Interpolation of scattered data: distance matrices and conditionally positive definite functions, Constructive Approximations 2 (1986) 11–22.

[20] J. Noh, U. Neumann, Expression cloning, in: SIGGRAPH, ACM SIGGRAPH, 2001, pp. 277–288.

[21] J. Ostermann, Animation of synthetic faces in MPEG-4, in: Computer Animation, Philadelphia, Pennsylvania, 1998, pp. 49–51.

[22] I.S. Pandzic, Facial animation framework for the web and mobile platforms, in: Web3D Symposium, Tempe, AZ, USA, 2002.

[23] I.S. Pandzic, Facial motion cloning, Graphical Models Journal, Elsevier 65 (6) (2003) 385–404.

[24] I.S. Pandzic, R. Forchheimer, MPEG-4 Facial Animation—The Standard, Implementation and Applications, first ed., John Wiley & Sons, LTD Linköping, Sweden, 2002.

[25] F. Parke, Computer generated animation of faces, in: ACM National Conference, ACM, 1972, pp. 451–457.

[26] K. Waters, A muscle model for animating three-dimensional facial expressions, in: Proceedings of Siggraph, vol. 21, 1987, pp. 17–24.