Animatable Face Models from Uncalibrated Input Pictures

M. Fratarcangeli[*,**], M. Andolfi[*], K. Stanković[***] and I.S. Pandžić[***]

[*] University of Rome "LaSapienza", Department of Computer and Systems Science, Rome, Italy

[**] Linköping Institute of Technology, Department of Electrical Engineering, Image Coding Group, Linköping, Sweden

[***] University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Telecommunications, Zagreb, Croatia

frat@dis.uniroma1.it, andolfi.marco@gmail.com, kristina.stankovic@fer.hr, igor.pandzic@fer.hr

***Abstract*— In networked virtual environments, videoconferences or chatting over the Internet users are often graphically represented by virtual characters. Modeling realistic virtual heads of users suitable for animation implies a heavy artistic effort and resource cost. This paper introduces a system that generates a 3D model of a real human head with a little human intervention. The system receives five input orthogonal photographs of the human head and a generic template 3D model. It requires manual annotation of 94 feature points on each photograph. The same set of feature points must be selected on the template model in a preprocessing step that is done only once. Computing process consists of two phases: a morphing and a coloring phase. In the morphing phase the template model is morphed in two steps using a Radial Basis Function (RBF) to take a shape similar to the shape of the real human head. In the coloring phase the deformed model is colored using the input photographs based on a cubemap projection, which leads to a realistic appearance of the model while allowing for a real-time performance. We show the use of the output model by automatically copying facial motions from the template model to the deformed model, while preserving the compliance of the motion to the MPEG-4 FBA standard.**

I. INTRODUCTION

Personalized virtual characters are increasingly important for the modern telecommunications. They are often used as graphical representations of users in networked virtual environments, videoconferences, 3D email or chatting over the Internet. The faithfulness of the graphical representation depends a lot on how photorealistic is the virtual face of the user. Although dozens of research papers have been published on a 3D modeling of a human face it is still one of the most fundamental and one of the most difficult problems in computer graphics.

There are many approaches for a reconstruction of the 3D head model from photographs. A common approach is applying Principal Component Analysis (PCA) algorithm to a database of laser-scanned human faces described as shape and texture vectors. This results in two models – a shape and a texture model, which are linear combinations of the shape/texture vectors of an average face and a set of eigenheads. The Eigenheads reflect different modes of variation in a face geometry and texture, e.g. face size, gender, amount of beard etc. Every new face can be described with the shape and texture model by changing their parameters. Skoglund et al. apply the PCA algorithm three times – two times to get the shape and the texture model, and third time to combine these two models into a final appearance model [1]. Blantz and Vetter fit the shape and the texture model separately to one or more input photographs using a gradient descent optimization function [2]. Their system requires user's estimation of some rendering parameters, e.g. camera distance, light direction, surface shininess etc. Unlike [1] - [2] methods described in [3] - [4] only fit the shape model to face silhouettes to avoid errors caused by different illumination conditions in the input images or to avoid a specification of the rendering parameters. The face silhouettes are calculated from calibrated multiple views (8 or 11) using manually selected feature points [3] or automatically using a color-based segmentation algorithm [4]. Gong and Wang also use the PCA algorithm on a database of scanned human faces, but the face model is created from one frontal photograph [5]. A texture map of the final model is generated from the input image using a combined method. Regions in the image that are determined from manually selected feature points are sized using a bilinear interpolation and projected orthogonally to the model. Other regions are mapped using a sub-patch texture synthesis algorithm and a weighted interpolation method.

Some methods start with a database of generic models and choose a generic model that is most similar to the subject's face. In [6] the most similar generic model is selected by comparing horizontal and vertical distances between the generic model's feature points and points extracted from the input photograph using face detection method Active Shape Model (ASM). A method described in [7] selects the generic model that best matches ear features extracted from five input images (the frontal image, two profiles and two three-quarters portraits).

Unlike previously mentioned methods that require some sort of database there are a lot of approaches that receive as an input one generic model. The generic model is then deformed to fit the feature points that were detected automatically in the input photographs, e.g. using a corners matching technique [8], or they need to be selected manually like in [9] - [13]. Some of these methods use stereo images and require the calibration of one or more cameras [8] - [9]. Other methods like ours start with orthogonal views of the human head. Lee et al. use two orthogonal photographs (the front and side view) and a set of feature points that corresponds to the MPEG-4 Face Definition Parameters (FDPs) [10]. They deform the generic model using Dirichlet Free-Form Deformation (DFFD). The texture of the final model is generated automatically by connecting two input images along predefined piecewise lines between some FDPs and by

avoiding boundary effects along these lines using a multiresolution technique. The side photograph is used for both the right and the left view. A system described in [12] is very similar to [10] except for the detection of the hair and face outlines and features, which is being done in a semi-automatic way using a structured snake method. Also, the texture is created using a cylindrical projection.

Methods described in [11] and [13] are the most similar to our system. They generate the head model from several uncalibrated views of the human head. Starting from a few manually selected feature points these methods deform the generic model in two steps using RBF. Before the second step the initial set of points is expanded using a criterion like the Normalized Correlation Coefficient (NCC) [11] or with additional correspondences retrieved from manually drawn curves [13]. Our method starts with a bigger set of manually selected points and in the second step refines the model using a smaller subset of these points. A system described in [13] uses view-dependent textures to get high quality results, but this requires a recomputation of the textures every time the point of view or expression of the virtual face changes. We are using a method based on a cubemap system, a widely used extension that reduces deformation artifacts in a view-independent sense and thus provides a better performance during the animation.

In terms of the animation all the methods mentioned above use an average or generic model, which usually contains animation mechanisms, e.g. volume morphing anchor vertices or muscle-based information per vertex [9], MPEG-4 Face Animation Parameters (FAPs) [10], facial muscle actions using Rational Free Form Deformations (RFFDs) [12], pseudo-muscle system based on Minimum Perceptible Action (MPA) [14]. This way the final model can be animated immediately after its creation. But there are also methods that retrieve geometry of the subject's head using laser [15] or structured-light range scanners [16] so facial animation must be achieved on some other way. Guenter et al. generated a realistic animation of facial expressions by capturing the subject's facial expressions from videos recorded by 6 calibrated cameras [15]. The method requires gluing 182 fluorescent colored dots on the subject's face. After removing the colored landmarks from the video streams using image processing techniques the resulting streams are merged into the final texture map sequence of the model.

II. Our Approach

Our goal was to create a system that is able to produce the head model without using any particular instrument or an invasive technique on the subject. The system uses as input five orthogonal photographs taken from the front, left, right, back and top side of the subject's head as depicted in Fig. 1.

After the manual selection of features points on the input images the system automatically generates the 3D colored model of the subject in two phases – the morphing and the coloring phase. In the morphing phase the generic model is modified and adapted to fit the selected feature points from the images. The generic model is a model of the human head that is neutral in any way, i.e. without any expression, gender or race features (see Fig. 2). In the coloring phase color of each vertex is calculated from the images based on the modified cubemap system and the final texture of the model is created using the cylindrical projection.

Figure 1. Five input orthogonal pictures of the subject

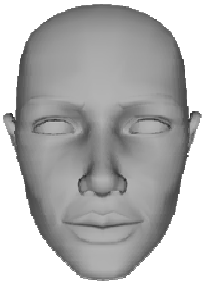

Figure 2. The generic model

III. Morphing Phase

As mentioned in the previous section the main idea of the morphing phase is to deform the generic model so that it gains the shape of the subject's head. This phase is composed of several subphases:

- an extraction of the generic model's features,
- an extraction of the subject's features,
- a modification of the generic model so that its features fit the subject's features.

Basically the feature extraction consists of the manual selection of the landmark points. The landmarks are characteristic points of the human head, such as the centers of the eyes, the corners of the lips, etc. Their position and the distances between them determine the head features.

A choice of feature points is crucial for the quality of the result, e.g. using a set of points that are mostly placed on the face but there are a few of them for the rest of the head would produce a model whose face would look like the subject's face, but the shape of the rest would be similar to the generic model. The set of points that we decided to use is very close to the one proposed in the MPEG-4 standard [17]. The MPEG-4 FDPs were chosen with facial animation in mind, so some of these points, e.g. some points on the teeth and over the tongue, are useless for our system. Also, they do not include so many points on the general shape of the head, so we used a modified set of points with additional points where the FDPs are not enough and without the points that are useless for us (see Fig. 3). We divided the chosen points in two subsets:

- High Quality Points, that are particularly distinctive, e.g. eyes' centers, lips' corners, nose tip, etc. (the green

points in Fig. 3), so it is easy to determine their position and the selection error will be small,

- Low Quality Points, that aren't so clear like the centers of cheeks (the yellow points in Fig. 3), so their exact detection is difficult and affected by noise.

This distinction is fundamental for the execution of the morphing.

A. Extraction of Generic Model's Features

In this subphase the features points described in the previous section are selected manually on the generic model, so they are also affected by noise. This operation can be done once in a preprocessing stage.

B. Extraction of Subject's Features

In this phase the user must select the features points on the input images. The images are taken with one uncalibrated camera, so the selected points must be corrected by removing camera errors (this kind of errors is defined in section 3.3). After the correction for each feature point there are maximally five 2D values (could be less if the point is not visible on some images). Each image is aligned with two axes so the values must be converted as described in Table 1.

The system calculates 3D coordinates [x, y, z] for each point as a weighted average of the 2D values. The most important image is the frontal photograph so the weights are defined in such a way that they increase the dependency of the final result on that image.

C. Correction of Camera Errors

The camera errors are mainly result of different camera's external parameters (the position and scaling) among the input images. There are three kinds of errors:

- a different distance between the camera and the subject,
- a different orientation of the camera,
- a wrong orientation of the camera.

While taking photographs of all sides of the subject's head it is possible that the distance between the camera and the head changes, which reduces or increases the global size of the head in the images as depicted in Fig. 4. Also, the camera orientation can change, which would produce a translation of the head as shown in Fig. 5. The third kind of errors happens when the subject is not looking straight into the camera, but with a little inclination. This results in a rotation of the head in the frontal photograph (see Fig. 6).

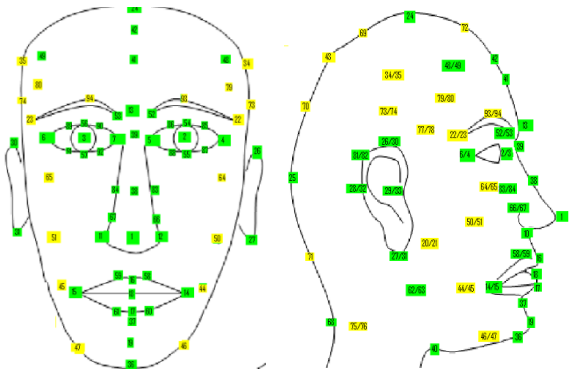

Figure 3. The set of adopted feature points.

TABLE I. CORRESPONDENCES OF IMAGE AXES AND AXES OF MODEL'S 3D COORDINATE SYSTEM

Image	[x, y] correspond to
Front	[x, y]
Left	[-z, y]
Right	[z, y]
Back	[-x, y]
Top	[x, -z]

The errors are corrected by aligning all input images to the frontal and the left image. The first kind of errors is corrected by scaling, the second by translation and the third by rotation. The scaling factor that aligns two images according to the size of the head is computed as the ratio of the sums of distances between all common points, i.e. the points that are visible on both images, and a reference point (usually the nose tip) on each image. That is the scaling factor of the image A that is aligned according to the image B is equal to:

$$sf = \frac{\sum_i dA_i \times oA_i \times oB_i}{\sum_i dB_i \times oA_i \times oB_i} \quad (1),$$

where:

$$oX_i = \begin{cases} 1 & point\ i\ is\ visible\ in\ the\ image\ X \\ 0 & otherwise \end{cases} \quad (2),$$

dA_i is a distance of some point from the reference point on the image A, and dB_i is a distance of some point from the reference point on the image B. The distances dA_i and dB_i are computed as differences between the coordinates along the axis that is common to the images A and B (it can be X, Y or Z axis).

The alignment of the images through translation consists of moving all the points so that the reference point is in the origin of the 3D coordinate system. The rotation of the head in the frontal photograph is corrected by reverse rotation for an angle between the line passing through the eyes' centers and the line y = 0.

D. Execution of Morphing

Just before the execution of the morphing operation the system has 3D coordinates of the generic model's feature points and the feature points that were selected on the input photographs. The morphing operation is executed as an interpolation using Radial Basis Function (RBF) so that the generic model's feature points fit the feature points extracted from the photographs.

A Radial Basis Function is a linear combination of the basis functions whose values depend only on the distance from the points [18]. Having two sets of points

$$P_j \in R^3 \qquad j = 1, 2, \dots, n$$
$$Q_j \in R^3 \qquad j = 1, 2, \dots, n \quad (3),$$

the RBF is a function $G : R^3 \to R^3$ defined as

$$G(P_j) = \sum_{i=1}^{n} h_i R_i(P_j) = Q_j \quad (4),$$

where $\{h_i\}$ are 3D coefficients, $\{R_i\}$ are the basis functions. As the basis function we use Hardy multiquadrics defined as:

$$R_i(P_j) = \sqrt{d_i^2(P_j) + r_i^2} \quad (5),$$

where d_i is the Euclidean distance, r_i is a stiffness parameter controlling the stiffness of the deformation around P_j. The final RBF is a linear combination of the Hardy multiquadrics and a linear term $L(P_j)$:

$$G(P_j) = \sum_{i=1}^{n} h_i \times \sqrt{d_i^2(P_j) + r_i^2} + L(P_j) \quad (6),$$

where $\{h_i\}$ $(h_i \in R^3,\ i = 1, \dots,\ n+4)$ are called Hardy coefficients, and $L(P_j)$ is defined as

$$L(P_j) = h_{n+1} \times x + h_{n+2} \times y + h_{n+3} \times z + h_{n+4} \quad (7),$$

which will produce a rotation, translation and scaling of all data points. In order to find the interpolation function $G(P_j)$, we define the source set of feature points (the generic model's feature points) $\{P_j\}$ and the target set of feature points (that were extracted from the input images) $\{Q_j\}$, and then we solve a linear equation system to compute the Hardy coefficients. The linear equation system is obtained by imposing n conditions $G(P_j) = Q_j$:

$$Q_j = \sum_{i=1}^{n} h_i \times \sqrt{d_i^2(P_j) + r_i^2} + L(P_j) \quad (8).$$
$$j = 1, \dots, n$$

Furthermore, for linear precision, we impose:

$$Q_j = \sum_{i=1}^{n} h_i \times \sqrt{d_i^2(P_j) + r_i^2} + L(P_j) \quad (9),$$
$$j = 1, \dots, n$$

$$\sum_{i=1}^{n} h_i \times x_i = 0 \quad (10),$$

$$\sum_{i=1}^{n} h_i \times y_i = 0 \quad (11),$$

$$\sum_{i=1}^{n} h_i \times z_i = 0 \quad (12),$$

$$\sum_{i=1}^{n} h_i = 0 \quad (13).$$

This results in a linear system of $n+4$ equations with $n+4$ unknowns $\{h_i\}$.

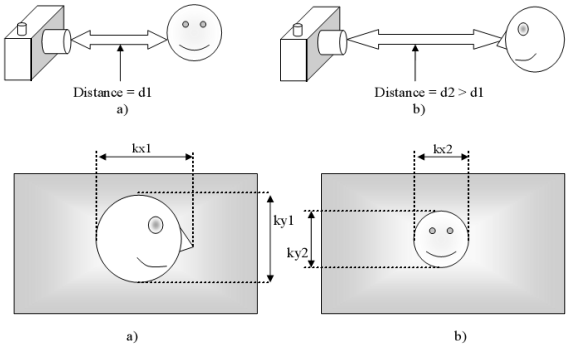

Figure 4. The different distance between the camera and the subject

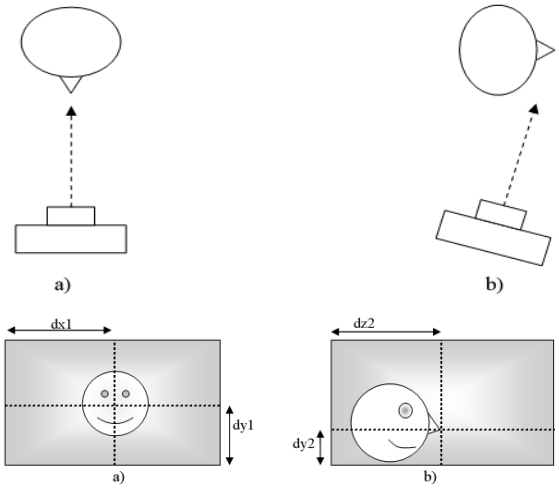

Figure 5. The different orientation of the camera

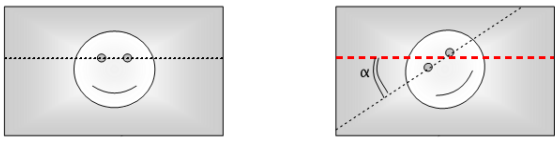

Figure 6. The wrong orientation of the camera

The sets of feature points are noisy because they were manually selected, so executing a simple interpolation would also reproduce the noise on the final shape of the head. So we execute the morphing operation in two steps (see Fig. 7) using the previously mentioned stiffness parameter and two sets of points that we defined earlier: the High and the Low Quality Points. The stiffness parameter determines how much close the starting points would be to the target points after the morphing. In the first step the morphing is executed using the High and the Low Quality Points with a high value of the stiffness parameter (0.7) to get an approximation of the target shape. The resulting shape is refined using only the High Quality Points with a very low value of the stiffness parameter (0.003125).

IV. Coloring Phase

As mentioned before the main idea used in the coloring phase is based on the cubemap system. Cube mapping [19], commonly used for environment mapping, consists of inserting a model into an ideal cube, mapping the pictures of the environment on the cube sides and

obtaining the color of each surface point using a radial projection (see Fig. 9b).

In our system each input photograph of the subject's head is mapped to the corresponding side of the cube as shown in Fig. 8. The cubemap system operates with six pictures so the bottom texture is generated automatically as explained in section 4.3. The input photographs are good approximations of orthogonal projections, so instead of using the radial projection the head model is colored using six orthogonal projections (see Fig. 9a). A final texture map of the model is produced based on the cylindrical projection as explained in section 4.4.

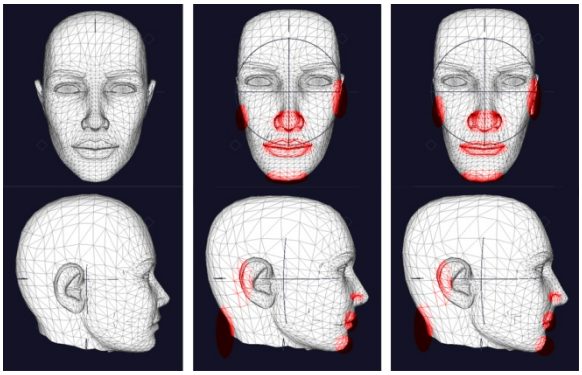

Figure 7. The pictures in the first column display the generic model. The pictures in the second/third column display the results of the first/second morphing step. A few visible improvements after the second morphing step compared to the first morphing step are highlighted with red color

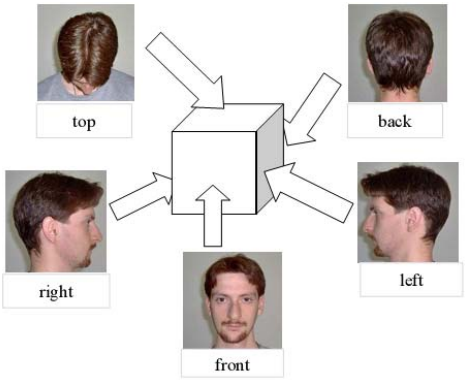

Figure 8. Five input photographs of the subject's head mapped on a cube

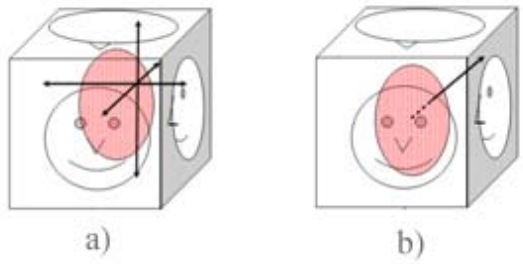

Figure 9. We use the orthogonal projection as shown in a); the cubemap system uses the radial projection as in b)

A. Modified Cubemap System

A certain region of the head is depicted in multiple input images, e.g. the frontal image covers the whole frontal side of the head, but the left, right, top and bottom side also depict some parts of the head's frontal side (see Fig. 10). So a color of each model's vertex is obtained as a weighted sum of color contributions of all images. The contribution of each image is retrieved by calculating the texture coordinates of the vertex using the orthogonal projection followed by the inverse operations used for the correction of the camera errors (see Fig. 11).

The weights of each vertex are computed using the scalar products of a vertex normal and the normals of the image planes, i.e. the weight w_i of the color taken from the image i is calculated as

$$w_i = \begin{cases} n_i \cdot n & if\ n_i \cdot n > 0 \\ 0 & otherwise \end{cases} \quad (14),$$

where n_i is a normal of the i image plane, and n the vertex normal. If the scalar product is negative the vertex normal points away from the image plane, i.e. looking from the image plane towards the head model it is possible to see the inner side of the surface around the vertex. This means that the vertex is not visible because it is covered by the other parts of the model's surface.

To emphasize the importance of the frontal image all input images are also weighted and the final weight wf_i of the vertex is calculated as

$$wf_i = \frac{w_i \cdot wp_i}{\sum_{j=1}^{6} w_j \cdot wp_j} \quad (15),$$

where

$$wp_i = \begin{cases} 10 & i = 1, \quad the\ front\ image \\ 5 & i = 2, \quad the\ left\ image \\ 5 & i = 3, \quad the\ right\ image \\ 5 & i = 4, \quad the\ back\ image \\ 1 & i = 5, \quad the\ top\ image \\ 1 & i = 6, \quad the\ bottom\ image \end{cases} \quad (16).$$

The weights wp_i are determined by experience.

B. Detection of Occluded Points

The coloring algorithm explained in the previous section does not consider all occluded points. An example of the occluded point is shown in Fig. 12. The blue point is visible looking from the frontal image plane, but is occluded by the eye looking from the left image plane. The algorithm will not take this in consideration because the scalar product of the point's normal and the image plane normal is positive.

The occluded points are usually detected by sending rays originating from the point in directions of the image normals (see the green and the red ray in Fig. 12) and checking if the ray intersects the model. But to preserve the fast coloring algorithm we used a simpler solution. The occluded areas like the ones between the nose and the eyes are defined using the feature points, e.g. the inner corners and centers of the eyes. Instead of sending rays and searching for intersections for each point the algorithm just checks if the point is in one of these areas and sets the corresponding weights of these points to zero. The results are displayed in Fig. 13.

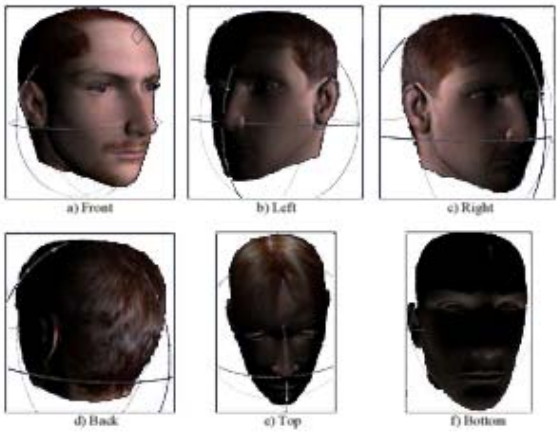

Figure 10. Each input image is mapped on a certain area of the target model

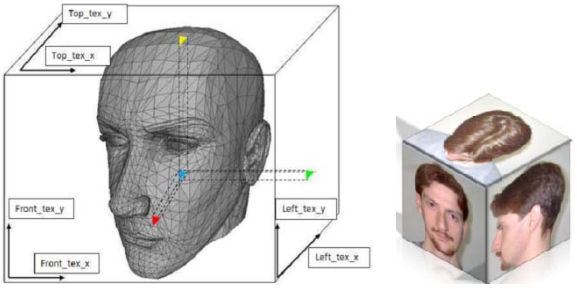

Figure 11. The computation of the texture coordinates: the image on the left shows the execution of the orthogonal projections toward the cube sides; the image on the right displays the aligned input photographs mapped to the cube sides

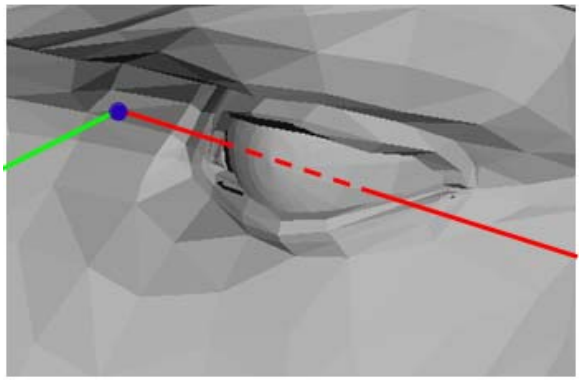

Figure 12. The detection of the occluded points

C. Bottom Texture Generation

As mentioned before the system automatically generates the bottom texture important for the head areas like the one under the chin. Most of these regions are hidden and the texture is blended with other five images, so it is not necessary that the final texture has a very good quality.

The bottom texture is generated in a simple and fast way by retrieving a little square area around the nose tip in the frontal photograph and by repeating this area to get a picture as large as the other images. The resulting image contains just the color of the skin and it is possible to see a repetition of the squares (see Fig. 14), but the quality is enough for our aim.

D. Final Texture Map Generation

The coloring of the model based on the cubemap system gives photorealistic results, but to get a faster rendering of the model in real-time animations the six textures are blended into one final texture map using the cylindrical projection.

The texture map is wrapped around an ideal cylinder surrounding the head model. For each texel on the cylinder surface a ray is casted from the medial axis of the head model towards the texel. A color of the texel is obtained at the intersection point between the ray and the head model. Usually the ray does not intersect a triangle vertex but an internal point of the triangle so the color of that point is assigned to the texel. The texture coordinates of each vertex are obtained in the same way, but the ray is casted from the axis through the vertex to find the intersection with the cylinder.

Although it guaranties a better performance the cylinder projection introduces some unavoidable stretching in the upper part of the head. But the main part of the animation happens in the frontal part of the head where the deformation artifacts are visually acceptable.

V. Facial Motion Cloning

After the coloring phase the system automatically copies facial motions from the generic model to the generated model using a method described in [20], which is schematically represented by Fig. 15. Every facial motion is computed as a difference of 3D vertex positions between the generic model containing the motion (an animated source face) and the generic model in neutral position (the source face). Calculated difference is then added to the vertex positions of the generated model in a neutral position (a target face), resulting in the generated model with the copied motion (the animated target face).

The described method insures the correct scale of motion by normalization of the motion with respect to the MPEG-4 normalization units (FAPUs). This preserves the compliance of the motion to the MPEG-4 FBA standard. The method requires an identification of a subset of the MPEG-4 feature points in the source and target faces. The feature points of the generic model are defined in the preprocessing step. The generated model has the same mesh topology as the generic model since it is created by its deformation. Thus, the feature points of the generated model are identified using the mesh and vertex indices of generic model's feature points.

VI. Results

Some of the results obtained with the system are reported in Fig. 16. Each case study shows the input set of five photographs of the subject's head and the generated 3D head model. The resulting models do not model the hair; they reproduce only the head shape.

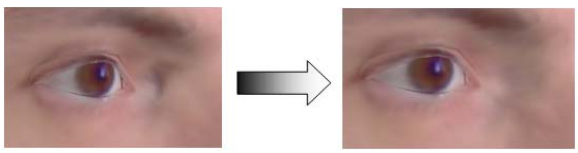

Figure 13. The occluded area of the nose before the correction (the left image) and after correction (the right image)

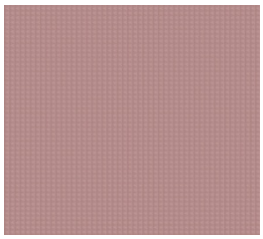

Figure 14. The generated bottom texture

VII. Conclusions and Future Work

Our system generates a photorealistic 3D model from five orthogonal photographs. It does not require any particular instrument or any invasive technique on the subject's head. Also, the system is very fast; after the initial interaction with the user it generates the 3D model in about ten seconds on a machine with Intel's processor Centrino Duo Core, 2 GHz, 2 GB RAM. The output model is colored using the cubemap projection, which introduces minimal deformation artifacts and thus leads to the realistic appearance of the model. Since the cube map extension is widely supported in hardware on commodity platforms, it makes the output models particularly suitable for real-time applications.

The main drawback of the system is the initial interaction with the user because he needs to manually select the set of 94 features points on the input photographs. Decreasing the number of input photographs from five to one or two can reduce the user interaction. Also, a smaller set of feature points would speed up the selection, but it would also reduce the quality of the result. The best solution would be an automatic detection of the feature points using computer vision techniques.

Acknowledgments

This work was partly carried out within the research project "Embodied Conversational Agents as interface for networked and mobile services" supported by the Ministry of Science, Education and Sports of the Republic of Croatia. Also, it was partly supported by the company Visage Technologies.

References

[1] K. Skoglund, R. Larsen, and B. Lading, "Building a 3-D appearance model of the human face," in *Proceedings of DSAGM*, 2003.

[2] V. Blantz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proceedings of SIGGRAPH '99*, 1999, pp. 187-194.

[3] J. Lee, B. Moghaddam, H. Pfister, and R. Machiraju, "Silhouette-based 3D face shape recovery," in *Proceedings of Graphics Interface*, 2003, pp. 21-30.

[4] Y.-C. Wong and W. H. Lau, "Fast model-based 3D human face reconstruction from silhouettes of multiple views," in *Proceedings of IEEE TENCON 2006*, 2006, pp. 1-4.

[5] X. Gong and G.-y. Wang, "Realistic face modeling based on multiple deformations," *Journal of China Universities of Posts and Telecommunications*, vol. 14, pp. 110-117, December 2007.

[6] M. Zhang, X. Zeng, P. Lu, and Y. Wang, "3D head retrieval based on geometrical measurement for modeling," in *Proceedings of Computer Graphics, Imaging and Visualization*, 2005, pp. 99-102.

[7] M. Dellepiane, N. Pietroni, N. Tsingos, M. Asselot, and R. Scopigno, "Reconstructing head models from photographs for individualized 3D-audio processing," *Computer Graphics Forum*, vol. 27, pp. 1719-1727, October 2008.

[8] R.-H. Liang, Z.-G. Pan, and C. Chen, "New algorithm for 3D facial model reconstruction and its application in virtual reality," *Journal of Computer Science and Technology*, vol. 19, pp. 501-509, July 2004.

[9] R. Enciso et al., "Synthesis of 3D faces," in *Proceedings of USF International Workshop on Digital and Computational Video*, 1999, pp. 146-151.

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

[11] T.-Y. Lee, P.-H. Lin, and T.-H. Yang, "Photo-realistic 3D head modeling using multi-view images," in *Proceedings of ICCSA '04*, 2004, vol. 3044, pp. 713-720.

[12] W.-S. Lee, P. Kalra, and N. Magnenat-Thalmann, "Model based face reconstruction for animation," in *Proceedings of MMM '97,* 1997, pp. 323-338.

[13] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin, "Synthesizing realistic facial expressions from photographs," in *Proceedings of SIGGRAPH '98*, 1998, pp. 75-84.

[14] W.-S. Lee and N. Magnenat-Thalmann, "Fast head modeling for animation," *Image and Vision Computing*, vol. 18, pp. 355-364, March 2000.

[15] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making faces," in *Proceedings of SIGGRAPH '98*, 1998, pp. 55-66.

[16] M. Tarini, H. Yamauchi, J. Haber, and H.-P. Seidel, "Texturing faces," in *Proceedings of Graphics Interface*, 2002, pp. 89-98.

[17] I. S. Pandžić and R. Forchheimer, *MPEG-4 Facial Animation – The Standard, Implementations and Applications*. Chichester: John Wiley & Sons, 2002.

[18] M. Fratarcangeli, M. Schaerf, and R. Forchheimer, "Facial motion cloning with radial basis functions in MPEG-4 FBA," *Graphical Models*, vol. 69, pp. 106-118, March 2007.

[19] N. Greene, "Environment mapping and other applications of world projections," *IEEE Computer Graphics and Applications*, vol. 6, pp. 21-29, November 1986.

[20] I. S. Pandžić, "Facial motion cloning," *Graphical Models*, vol. 65, pp. 385-404, November 2003.

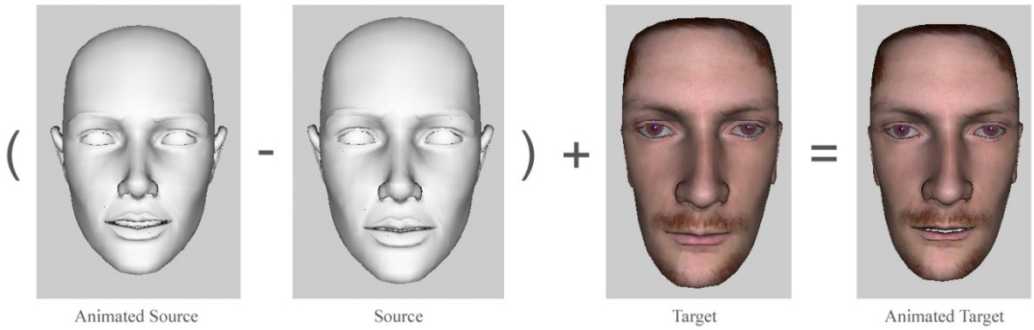

Figure 15. The overview of facial motion cloning

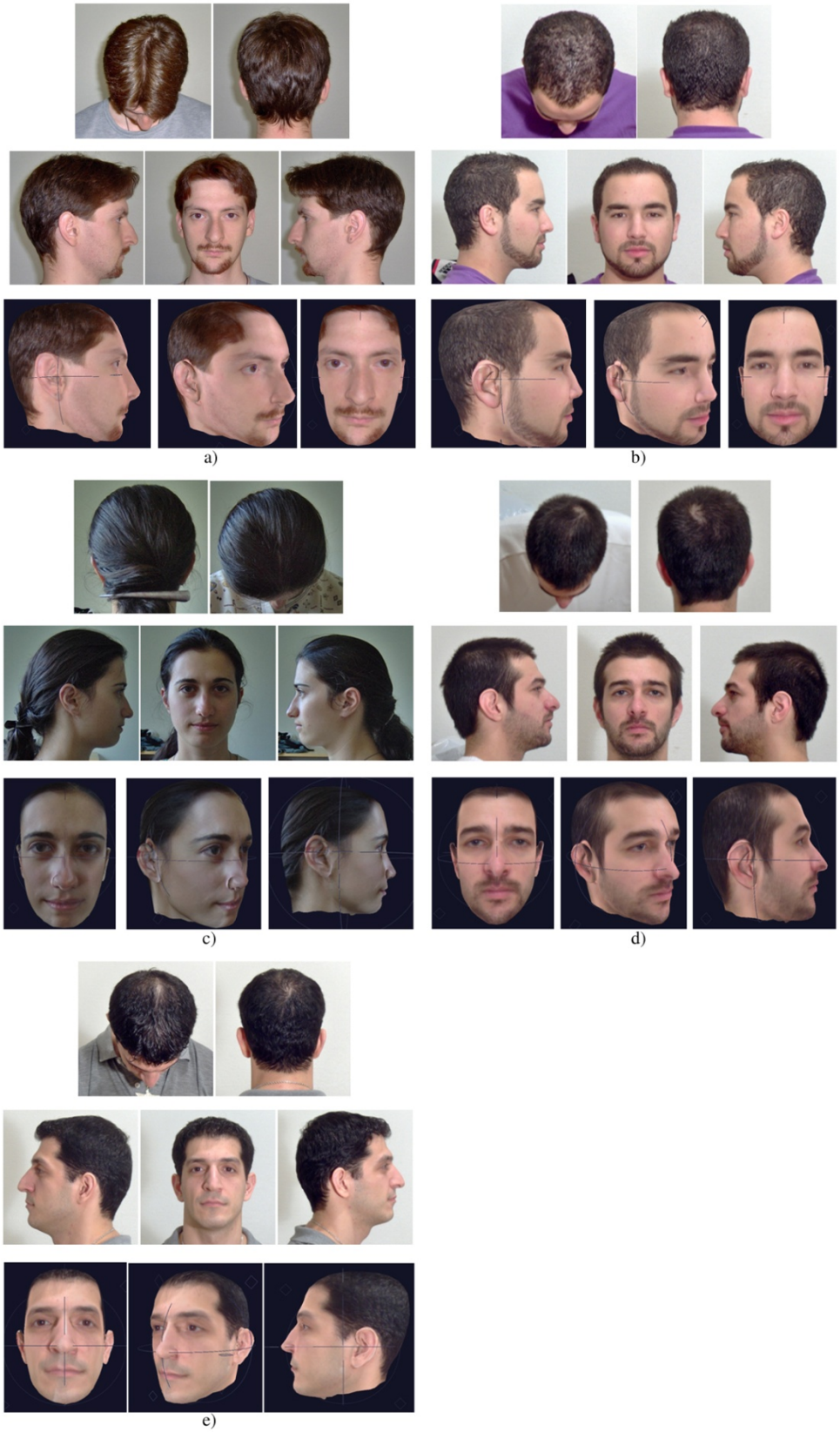

Figure 16. Five case studies of different human heads; the pictures in first two rows are the input photographs, and the pictures in the last row display the output models