IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.1, January 2007

Anthropometric Modeling of Faces from Range Scans

Yu Zhang[†] and Chew Lim Tan[††],

[†]Department of Computer and Information Science, University of Pennsylvania, PA 19104-6389, USA
[††]School of Computing, National University of Singapore, 117543 Singapore

Summary
This paper presents new techniques for generating varied realistic geometric models of human faces by synthesizing local facial features according to anthropometric control parameters, and for generating a full-head texture from a face image of the scanned data for realistic rendering. We automatically register 3D face scans in a large database by deforming a generic head mesh to fit each scanned face shape. Once we have a common surface representation for each face scan, we form local feature shape spaces by applying principal component analysis to the data sets of facial feature shapes. We parameterize the example models using face anthropometric measurements and predefine the interpolation functions for the parameterized examples based on radial basis functions. At runtime, the new face geometry is generated at an interactive rate by evaluating the interpolation functions with the input anthropometric parameter values. We automatically generate a full-head texture from parameterized texture of the face region. In particular, we address the creation of individual textures for ears. Apart from the initial adjustment of feature point positions, our method is fully automated.

Key words:
Face Modeling, Anthropometric Control, Interpolation, Texture Mapping, 3D Scanned Data.

1. Introduction

Generation of realistic human face models is one of the most interesting problems in computer graphics. Many applications such as character animation for films and advertisement, computer games, video teleconference, user-interface agents and avatars require a large number of different face geometries. However, creating diverse 3D face models for these applications is a difficult and time-consuming task, particularly if realism is desired.

With a significant increase in the quality and availability of 3D capture methods, a common approach towards creating face models of real humans uses laser range scanners to acquire both the face geometry and texture simultaneously. Although the acquired face data is highly accurate, unfortunately, substantial effort is needed to process the noisy and incomplete data into a model suitable for modeling or animation. In addition, each new face must be found on a living subject. There is no simple means of automatically modifying the face model to different shapes as the user intends once it has been generated. Another limitation of the 3D scanning technology is that the complete head geometry cannot be easily captured as the hair in dark color absorbs all of the laser radiation. The top and back of the head are generally not digitized unless the hair is artificially colored white, but that destroys the texture. In most cases, only the face region can be properly textured. There is no automatic mechanism provided to generate a full-head texture from the acquired face image for rendering the model towards a "cloned" head.

In this paper, we present new techniques for creating varied realistic human face models subject to desired shape parameters, and for generating a parameterized full-head texture for rendering the model. The reported psychophysical evidence [1] suggests that internal facial features (e.g. eyes, nose, mouth and chin) are good for discriminating faces. Thus, we synthesize face geometry by perceiving the face as a set of feature regions. Such a feature-based face synthesis allows us to generate more diverse face geometries through different combinations of the synthesized feature shapes. Our method takes as examples 3D scanned face data to exploit the variations presented in the real human faces. In order to establish correspondences between face scans, we develop a two-step model fitting method for the 3D registration problem, where a generic head model is fitted to each example in a global-to-local fashion. The obtained correspondence enables the application of principal component analysis (PCA) to exemplar shapes of each facial feature to build a low dimensional shape space. We parameterize the example models using the face anthropometric measurements, and predefine the interpolation functions for the example models based on radial basis functions. At runtime, the interpolation functions are evaluated to efficiently generate the appropriate feature shapes by taking the anthropometric parameters as input. After having performed a vertex-to-image binding for vertices of the head mesh, we generate a cylindrical full-head texture from the parameterized texture of the face region. The individual ear textures are also created from a single input image.

The main contributions presented in this paper are:

- a feature-based face modeling method to automatically synthesize 3D face shapes according to anthropometric parameters. It regulates the naturalness of synthesized faces by

Manuscript received January 5, 2007
Manuscript revised January 25, 2007

exploiting the parameter-to-shape correlations that are presented in the real human faces. It is efficient in time complexity and performs at an interactive rate.

- a technique that uses a frontal face image of the scanned data to generate a full-head texture and individual ear textures which add greatly to a realistic personalized appearance of the face model.

This paper is organized as follows. Section 2 reviews the previous work on face modelling and texturing. Section 3 presents the face data we use. Section 4 describes the model fitting process. In Section 5, we present our feature-based synthesis technique. Section 6 elaborates on the generation of a full-head texture and ear textures. Experimental results are shown in Section 7. Section 8 concludes by discussing future work.

2. Previous Work

Face modeling and animation has been an active area of research in computer graphics since the early 1970's [2] (see [3] for an excellent survey). Regarding modeling of static face geometry in particular, several approaches have been proposed. Parametric conformation models have been invented very early [4-6]. The desire was to create an encapsulated model that could generate a wide range of faces based on a small set of input parameters. However, the choice of the parameter set depends on the face mesh topology and therefore the manual association of a group of vertices to a specific parameter is required. Furthermore, manual parameter tuning without constraints from real human faces for generating a realistic face is difficult and time-consuming.

The image-based technique [7-12] utilizes an existing 3D face model and information from few pictures (or video streams) for the reconstruction of both geometry and texture. Although this kind of technique can provide reconstructed face models easily, its drawbacks are the inaccurate geometry reconstruction and inability to generate new faces that have no image counterparts.

Another avenue for creating accurate human face models is 3D scanning technology [13-16]. However, the result of the range scan is a model corresponding to a single individual that tells us little about the space of face shapes. Moreover, in the absence of a characterization of this space, editing a face model in a way that yields a realistic, novel face shape is not trivial.

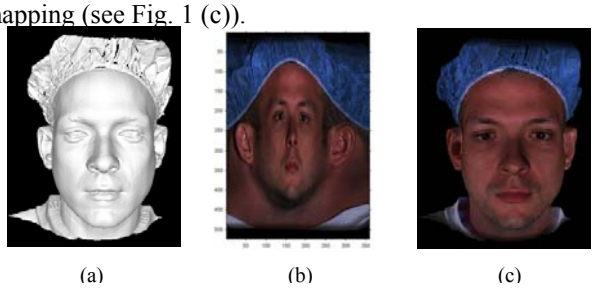
Decarlo *et al.* [17] constructed a range of face models with realistic proportions using a variational constrained optimization technique. However, because of the sparseness of the constraints compared to the high dimensionality of possible faces, realistic shape cannot be obtained in the facial regions where no desirable measurement has been specified as a constraint. Also, this approach requires minutes of computation for the optimization process to generate a new face.

Vlasic *et al.* [18] used multi-linear face models to study and synthesize variations in faces along several axes, such as identity and expression. Blanz and Vetter [19] used example database models from scanners and a linear function that maps facial attributes (gender, weight and expression) onto the 3D model. There are several key differences from our work. First, they manually assign the attribute values to face shape and texture and devise attribute controls using linear regression. We automatically compute the anthropometric measurements for describing face shape and synthesize facial features by learning a mapping between the measurement space and shape space through scattered data interpolation. Second, their morphable model is restricted to the face region. We synthesize an individual texture map for texturing the full head geometry. Third, they use a 3D variant of a gradient-based optical flow algorithm to derive the point-to-point correspondence. This approach will not work well for faces of different races or in different illumination given the inherent problem of using static textures. In contrast, our robust method of determining correspondences does not depend on the texture information.

Texturing for face modeling is still an underemphasized issue. Williams [20] generated and registered a texture map from a peripheral photograph. A more convenient way to create models of real persons uses the textures acquired by laser scanners [13-16]. However, the full head geometry can not be easily captured and textured due to complex reflectance properties of hair. Marschner *et al.* [21] described a technique that uses several input photographs taken under controlled illumination with known camera and light source locations to generate an albedo texture map of the human face along with the parameters of a BRDF. Creating face textures from multiple, unregistered images has more flexibility [9,12,22,23]. Depending on which images each vertex gets its color from, the vertices of a triangle mesh are partitioned into two types first. Then, colors in images are resampled for the frontal vertices. It results in a smooth and seamless joint between different images that map on adjacent triangles of surfaces when combining images together. Not requiring separate camera setups, we focus on synthesizing a full-head texture from a reflectance image acquired by a laser scanner which only captures color of the face region.

3. Face Data

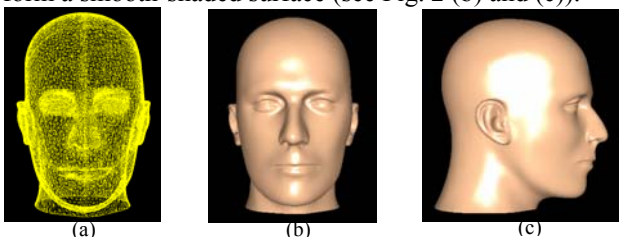
We use the USF face database [24] that consists of Cyberware scans of 186 human faces with a mixture of race and age. Each subject is captured wearing a bathing cap and with a neutral expression. The laser scans provide face structure data which contains approximately 140k surface points (see Fig. 1 (a)) and RGB-color values that are stored in a 360×524 image with 8 bit per channel (see Fig. 1 (b)). The image is registered against the range data in the scanning process and can be used for texture-mapping (see Fig. 1 (c)).



(a) (b) (c)

Fig. 1 Face data: (a) scanned face geometry; (b) acquired color image; (c) texture-mapped face scan.

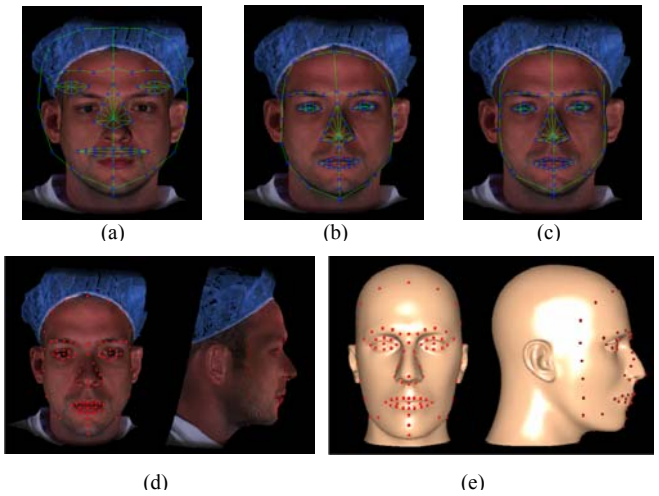
We use a generic head model created with Autodesk Maya^{TM}. It is a wire-frame of numbered vertices in 3D coordinate space (see Fig. 2 (a)). The generic model consists of 6,092 vertices and 12,274 triangles, with finer triangles over the highly curved and/or highly articulated regions of the face, such as the eyes and mouth, while larger triangles are used elsewhere, such as the cheeks and forehead. Prescribed colors are added to each triangle to form a smooth-shaded surface (see Fig. 2 (b) and (c)).



(a) (b) (c)

Fig. 2 Generic model: (a) wire-frame mesh; (b) and (c) smoothly-shaded surface.

Let each 3D face in the database be $\mathcal{F}_i$ (i =1, … ,M), M=186. Since the number of vertices in $\mathcal{F}_i$ varies, we resample all faces in the database so that they have the same number of vertices all in mutual correspondence. Feature points are identified to guide the resampling. In our method, the feature points are identified semi-automatically (see Fig. 3). A 2D feature mask consisting of polylines groups 83 feature points that correspond to the facial features such as the eyes, eyebrows, nose, mouth and face outline. It is superimposed onto the front-view face image obtained by orthographic projection of a texture-mapped face scan. The facial features in this image are detected by using the Active Shape Models [25] and the 2D feature mask is fitted to the detected features automatically. A little user interaction is utilized to tune the feature point positions in the image plane due to slight inaccuracy of the automatic facial feature detection. The 3D positions of feature points on the scanned surface are then recovered by re-projection to the 3D space. In this manner, we efficiently define a set of *target feature points* in a face scan $\mathcal{F}_i$ as $T_i = \{\mathbf{t}_{i,1}, \ldots, \mathbf{t}_{i,n}\}$, where n=83. Our generic model $\mathcal{G}$ is already tagged with the corresponding set of *source feature points* $S=\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ by default (see Fig. 3 (e)).



(a) (b) (c) (d) (e)

Fig. 3 Semi-automatic feature point identification: (a) initial outline of the feature template; (b) after automatic facial feature detection; (c) after interactive user tuning; (d) and (e) feature points identified on the scanned data and generic model, respectively.

4. Model Fitting

The problem of deriving full correspondence for all face scans $\mathcal{F}_i$ can be stated as: resample the surface for all $\mathcal{F}_i$ using $\mathcal{G}$ under the constraint that $\mathbf{s}_j$ is mapped to $\mathbf{t}_{i,j}$. We need to find a function $\mathbf{f}: \Re^3 \rightarrow \Re^3$ such that

$$\mathbf{t}_{i,j} = \mathbf{f}(\mathbf{s}_j) + \mathbf{s}_j \qquad j = 1, \ldots, n \tag{1}$$

The goal is to construct a smooth interpolating function that expresses the deformation of non-feature vertices of the generic model in terms of the changes in the feature points during deformation. Radial Basis Functions (RBFs) are a popular means for interpolating scattered data for its power to deal with irregular sets of data in multi-dimensional space in approximating high dimensional smooth surfaces [26]. In our case, the interpolant using RBFs is a function that returns the displacement value for each vertex of $\mathcal{G}$ that takes it from the original position to its position in the target form. The displacements $\mathbf{d}_{i,j}=\mathbf{t}_{i,j}-\mathbf{s}_j$ are known for the source feature points $\mathbf{s}_j$ and target

feature points $\mathbf{t}_{i,j}$. These displacements are utilized to construct the interpolating function $\mathbf{f}(\mathbf{v})$ that returns the displacement for each generic mesh vertex $\mathbf{v}$:

$$\mathbf{f}(\mathbf{v}) = \sum_{j=1}^{n} \mathbf{c}_j \phi_j(\|\mathbf{v} - \mathbf{s}_j\|) + \mathbf{M}\mathbf{v} + \mathbf{t} \qquad (2)$$

where $\mathbf{v} \in \Re^3$ is a vertex of $\mathcal{G}$, $\mathbf{c}_j \in \Re^3$ are (unknown) weights, Φ is the radial basis function which is a real valued function on [0,1), $\|\cdot\|$ denotes the Euclidean norm, $\mathbf{M} \in \Re^{3\times3}$ adds rotation, skew, and scaling, and $\mathbf{t} \in \Re^3$ is a translation component. The Φ_j are defined by the source feature points. Popular choices for Φ include the thin-plate spline $\phi(r) = r^2 \log(r)$, the Gaussian $\phi(r) = \exp(-\rho r^2)$, the multi-quadric $\phi(r) = \sqrt{r^2 + \rho^2}$, and the biharmonic $\phi(r) = r$. In our work, we use the multi-quadric function, which places no restrictions on the locations of the feature points. In this function, ρ is the locality parameter used to control how the basis function is influenced by neighboring feature points. It is determined as the Euclidean distance to the nearest other feature point.

Setting up a system of linear equations relating the source and target feature points, the unknowns $\mathbf{c}_j$, $\mathbf{M}$, and $\mathbf{t}$ can be solved for simultaneously. The interpolation conditions of Eq. 1 lead to a linear system of n equations:

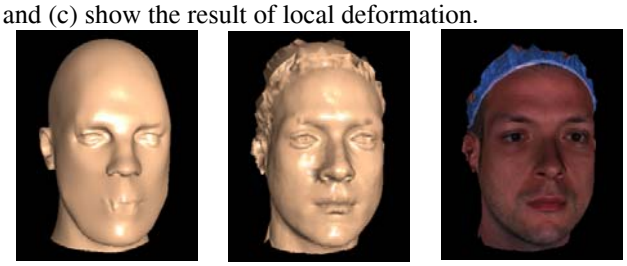
$$\mathbf{f}(\mathbf{s}_j) = \mathbf{t}_{i,j} - \mathbf{s}_j = \mathbf{d}_{i,j} \qquad j = 1,\ldots,n \qquad (3)$$

To remove affine contributions from the weighted sum of the basic functions, we include the additional constraints:

$$\sum_{j=1}^{n} \mathbf{c}_j = 0, \qquad \sum_{j=1}^{n} \mathbf{c}_j^T \mathbf{s}_j = 0 \qquad (4)$$

The system of linear equations (Eq. 3 and 4) is solved using an LU decomposition to obtain the unknown parameters. Using the predefined interpolating function as given in Eq. 2, we then calculate the displacement vectors for all vertices of the generic model to generate the output head model. Fig. 4 (a) shows the shape of the deformed model after having interpolated the set of computed 3D displacements at 83 feature points and applied them to the generic model.

As shown in Fig. 4 (a), the RBF-based deformation roughly aligns facial features of the generic model to the scanned data. We further improve the shape using a local deformation which ensures that all the generic mesh vertices are truly embedded in the scanned surface. The local deformation is based on the closest points on the surfaces of the generic model and the scanned data. The vertices of the generic model are displaced towards their closest positions on the surface of the scanned data. The polygons of the scanned data are organized into a Binary Space Partition tree in order to speed up the process of the closest point identification. As each generic mesh vertex samples a scanned data point, it takes the texture coordinates of that point for texture mapping. Fig. 4 (b) and (c) show the result of local deformation.



(a) (b) (c)

Fig. 4 Model fitting: (a) generic model after RBF-based deformation; (b) after local deformation; (c) texture-mapped appearance.

5. Feature-Based Face Shape Synthesis

5.1 Forming Local Shape Spaces

To synthesize the shape of facial features, we form the *local shape spaces* using PCA. The model fitting process generates the necessary vertex-to-vertex correspondence across 3D scans in the database, which is the prerequisite of PCA. Since all face scans are in correspondence through mapping onto the generic model, it is sufficient to define the regions of facial features on the generic model. We partition the generic mesh into four regions: eyes, nose, mouth and chin. The segmentation is transferred to the fitted generic meshes to generate individualized feature shapes with correspondences (see Fig. 5). Note that in order to isolate the shape variation from the position variation, we normalize all face scans with respect to the rigid rotation and translation of the face before fitting the generic model. Thus, PCA can be performed directly on the obtained data sets of feature shapes.

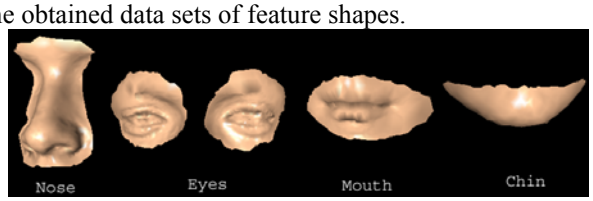


Fig. 5 Meshes of four facial features decomposed from the fitted generic mesh shown in Fig. 4 (b).

Given the set $\{F\}$ of facial features, we obtain a compact representation for the meshes of each facial feature using PCA. Let $\{F_i\}_{i=1,\ldots,M}$ be a set of meshes of feature F, each mesh being associated to one of the M 3D scans of the database. These meshes have the same dimensions. They are represented as vectors that contain the x, y, z coordinates of the n_F vertices $F_i = (x_1, y_1, z_1, \ldots, x_{n_F}, y_{n_F}, z_{n_F}) \in \Re^{3n_F}$. Each mesh of the feature can then be expressed as a linear combination of $M+1$ meshes $\{\psi_j^F\}_{j=0,\ldots,M}$:

$$F_i = \psi_0^F + \sum_{j=1}^{M} a_j^{F_i} \psi_j^F \quad (5)$$

where $\psi_0^F = \frac{1}{M}\sum_{i=1}^{M} F_i, \quad a_j^{F_i} = (F_i - \psi_0^F)\cdot\psi_j^F \quad (6)$

The meshes ψ_j^F ($1 \le j \le M$) are the eigenvectors of the covariance matrix of the set $\{F_i - \psi_0^F\}$. They are sorted by decreasing eigenvalue ($\lambda_p \ge \lambda_q$, $p<q$) and represent the principal components of the data set. By truncating the expansion of Eq. 5 at $j=k_F$ we introduce an error whose magnitude decreases when k_F is increased. We choose the k_F such that $\sum_{j=1}^{k_F}\lambda_j \ge \tau\sum_{j=1}^{M}\lambda_j$, where τ defines the proportion of the total shape variation (98% in our experiments). Each mesh of a facial feature in the data set can then be approximately described by the vector of the coefficients $\boldsymbol{a}^{F_i} = \{a_1^{F_i}, \ldots, a_{k_F}^{F_i}\}$. Each eigenvector ψ_j^F is a new coordinate axis for our existing data; thus each feature mesh can be restated as a point in the space spanned by the PCA-yielded orthogonal mesh basis. We call these axes *eigenmeshes*.

5.2 Anthropometric Control Parameters

Although eigenmeshes represent the most salient directions of the shape variation in the data set of a facial feature, they bear little resemblance to the underlying structure of biological forms. Face anthropometry provides a set of meaningful measurements or shape control parameters that allow the most complete control over the shape of the face. Anthropometric study [27] describes a widely used set of 132 measurements to characterize the human face. The measurements are taken between the landmarks defined in terms of visually-identifiable or palpable features on the subject's face. Such measurements use a total of 47 landmarks on the face. Following the conventions laid out in [27], we have chosen a subset of 38 landmarks for anthropometric measurements (see Fig. 6 (a)).

Facial measurements are categorized into five types. As shown in Fig. 6 (b), *ch-ch* refers to the **shortest distance** between the landmarks at the corners of the mouth, *n-sn* refers to the **axial distance** between the midpoint of the nasofrontal suture and junction between the lower border of the nasal septum, *al-prn* refers to the **tangential distance** measured on the face surface from the most lateral point on the nasal ala to the nose tip, the **angle of inclination** is exemplified by the inclination of the upper lip *sn-ls* with respect to the vertical axis, and the **angle between locations** is exemplified by the labiomental angle (the angle at the lower lip).

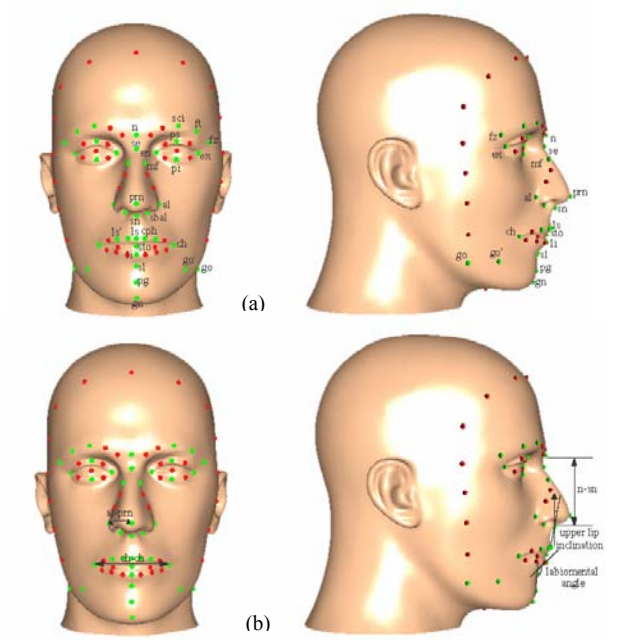


Fig. 6 (a) Head geometry with anthropometric landmarks (green dots). The landmark names are taken from [27]. (b) Anthropometric measurements.

Table 1: Anthropometric measurements of the nose

Landmarks	Measurement Name	Landmarks	Measurement Name
mf-mf	Nasal root width	n-prn	Nasal bridge length
al-al	Nose width	al-prn	Ala surface length
sbal-sbal	Alar base width	al-sn	Alar point-subnasale length
sbal-sn	Nostril floor width	n-prn	Inclination of the nasal bridge
sn-prn	Nasal tip protrusion	sn-prn	Inclination of the columella
en-se	Nasal root depth	al-prn	Inclination of the alar-slope line
en-se	Nasal root slope	n-se-prn	Nasofrontal angle
al-prn	Ala length	al-prn-al	Ala-slope angle
al-mf	Nasal bridge angle	se-prn-sn	Nasal tip angle
n-sn	Nose height	prn-sn-ls	Nasolabial angle

Supporting all 132 measurements is beyond the scope of this work. Instead, we are only concerned with those related to four facial features. As an example, Table 1 lists the nasal measurements used in our work. The anthropometric measurements are taken through all face examples in the database to determine their locations in a multi-dimensional measurement space. These locations are then used to guide the feature shape synthesis process.

5.3 Feature Shape Synthesis

From the previous stage we obtain a set of examples of each facial feature with measured shape characteristics, each of them consisting of the same set of dimensions, where every dimension is an anthropometric measurement. As each measurement has a different average and variation, it is prohibitive to define the measurement axis by directly using the measurements unit. Instead, the example measurements are normalized. Generally, we assume that an example model F_i of feature F has m_F dimensions,

where each dimension is represented by a value in the interval (0,1]. A value of 1 corresponds to the maximum measurement value of the dimension. The measurements of F_i can then be represented by the vector

$$\mathbf{q}_i^F = [q_{i1}^F, \ldots, q_{im_F}^F], \forall j \in [1, m_F]: q_{ij}^F \in (0,1] \quad (7)$$

This is equivalent to projecting each example model F_i into a *measurement space* spanned by the m_F selected anthropometric measurements. The location of each example in the measurement space is $\mathbf{q}_i^F$.

With the anthropometric shape control thus parameterized, our goal is to generate a new deformation of the face mesh by computing the corresponding eigenmesh coefficients with control through the anthropometric measurement parameters. Given an arbitrary input measurement $\mathbf{q}^F$ in the measurement space, such controlled deformation should interpolate the example models. To do this we interpolate the eigenmesh coefficients of the example models. For the truncated set of eigenmeshes of each facial feature, we need to solve for the coefficients in the truncated basis ($a_j^{F_i}$ in Eq. 5) which give shapes closest to the example models. We solve such an over constrained linear system using least squares to get the best fit to the example models.

Our feature shape synthesis problem is thus transformed to a scattered data interpolation problem. Again, the RBFs are employed in our shape interpolation scheme. The interpolation is multi-dimensional. Consider a $\Re^{m_F} \to \Re$ mapping, the interpolated eigenmesh coefficients $a_j^F(\cdot) \in \Re$, $1 \le j \le k_F$ at an input measurement vector $\mathbf{q}^F \in \Re^{m_F}$ are computed as:

$$a_j^F(\mathbf{q}^F) = \sum_{i=1}^{M} \gamma_{ij} R_i(\mathbf{q}^F) \quad \textbf{for} \quad 1 \le j \le k_F \quad (8)$$

where $\gamma_{ij} \in \Re$ are the radial coefficients and M is the number of example models. Let $\mathbf{q}_i^F$ be the measurement vector of an example model F_i. The radial basis function $R_i(\mathbf{q}^F)$ is a multi-quadric function of the Euclidean distance between $\mathbf{q}^F$ and $\mathbf{q}_i^F$ in the measurement space:

$$R_i(\mathbf{q}^F) = \sqrt{\|\mathbf{q}^F - \mathbf{q}_i^F\|^2 + \rho_i^2} \quad \textbf{for} \quad 1 \le i \le M \quad (9)$$

where ρ_i are the locality parameters used to control the behaviour of the basis function:

$$\rho_i = \min_{i \ne j} \|\mathbf{q}_i^F - \mathbf{q}_j^F\| \qquad i, j = 1, \ldots, M \quad (10)$$

The j-th eigenmesh coefficient of the i-th example model, $a_j^{F_i}$, corresponds to the measurement vector of the i-th example model, $\mathbf{q}_i^F$. Eq. 8 should be satisfied for $\mathbf{q}_i^F$ and $a_j^{F_i}$ ($1 \le i_F \le M$). Hence, by substituting $\mathbf{q}_i^F$ and $a_j^{F_i}$ for $\mathbf{q}^F$ and a_j respectively in Eq. 8, we have

$$a_j^{F_i}(\mathbf{q}_i^F) = \sum_{i=1}^{M} \gamma_{ij} R_i(\mathbf{q}_i^F) \quad \text{for} \quad 1 \le j \le k_F \quad (11)$$

Eq. 11 can be expressed in the matrix form

$$\mathbf{RY} = \mathbf{A} \quad (12)$$

where $\mathbf{Y} \in \Re^{M \times k_F}$ is the matrix of the unknown radial coefficients γ_{ij}, and $\mathbf{R} \in \Re^{M \times M}$ and $\mathbf{A} \in \Re^{M \times k_F}$ are the matrices defined by the radial bases and eigenmesh coefficients of the example models respectively, such that $\mathbf{R}_{ij} = R_j(\mathbf{q}_i^F)$ and $\mathbf{A}_{ij} = a_j^{F_i}$. The radial coefficients γ_{ij} are obtained by solving this linear system using an LU decomposition. We can then generate the eigenmesh coefficients, hence the shape of the facial feature, corresponding to the input measurement vector $\mathbf{q}^F$ according to Eq. 8.

5.4 Subregion Shape Blending

After the shape interpolation procedure, the surrounding facial areas should be blended with the deformed facial features to generate a seamlessly smooth face mesh. The position of a vertex $\mathbf{v}_i$ in the feature region F after deformation is $\mathbf{v}_i'$. Let V denote the set of vertices of the head mesh. For smooth blending, positions of the subset $\overline{V_F} = V \setminus V_F$ of vertices of V that are not inside the feature regions should be updated with deformation of the facial features. For each vertex $\mathbf{v}_j \in \overline{V_F}$, the vertex in each feature region that exerts influence on it, $\mathbf{v}_{k_i}^F$, is the one of minimal distance to it. Note that the distance is measured *offline* in the original undeformed generic mesh. For each vertex $\mathbf{v}_j \in \overline{V_F}$, the displacement vector for its corresponding closest feature vertex $\mathbf{v}_{k_i}^F$ is used to update its position in shape blending. The displacement is weighted by an exponential fall-off function according to the distance between $\mathbf{v}_j$ and $\mathbf{v}_{k_i}^F$:

$$\mathbf{v}_j' = \mathbf{v}_j + \sum_{F \in \Gamma} \exp\left(-\frac{1}{\beta}\|\mathbf{v}_j - \mathbf{v}_{k_i}^F\|\right)\|\mathbf{v}_{k_i}'^F - \mathbf{v}_{k_i}^F\| \quad (13)$$

where Γ is the set of facial features and β controls the size of the region influenced by the blending. We set β to 1/10 of the diagonal length of the bounding box of the head model. Fig. 7 shows the effect of our shape blending scheme utilized in synthesizing the nose shape.

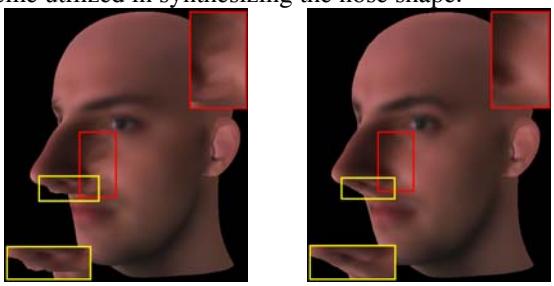


Fig. 7 Left: Without shape blending, the obvious geometric discontinuities around the border of the nose impair visual realism of the synthesis. Right: Using our approach, geometries of the feature region and surrounding area are smoothly blended at their boundary.

6. Texturing A Head

6.1 Mesh Parameterization

Our head skin texture is generated from the acquired color image shown in Fig. 1 (b). One of our goals is to readily generate 2D texture metamorphosis for head morphing. In general, morphing between two images requires pair-wise correspondences between image features. In our case, however, correspondences between the two textures are implicit in the texture coordinates of the two associated face meshes. Since every face generated from one generic model has a similar characteristic for texture coordinates, we can produce the shape-free face texture images by constructing a parameterization of the 3D generic mesh over a 2D image plane.

Given the vertex-wise correspondence between a fitted generic head mesh and the original undeformed generic mesh, it is trivial to transfer a texture image between them. Each vertex on the original generic mesh simply takes the texture coordinates of its corresponding vertex on the fitted mesh for texture mapping (see Fig. 8 (a) and (b)).

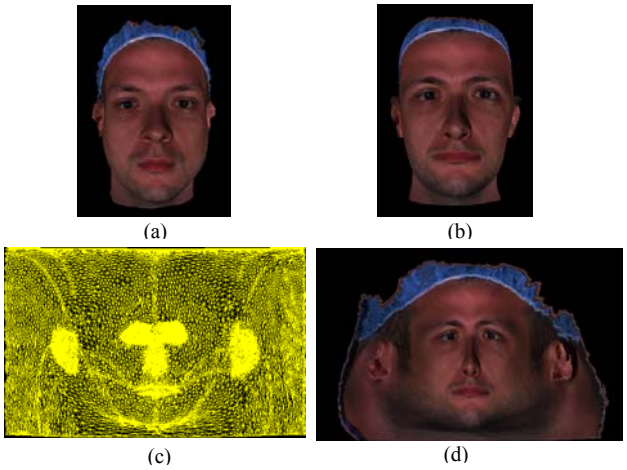


Fig. 8 (a) Fitted generic model with texture mapping. (b) Texture transferred to the original undeformed generic model. (c) Texture mesh parameterization. (d) Cylindrical texture image.

We parameterize the 3D generic head mesh over a 2D domain $[0,1]^2$ by implementing a cylindrical projection. The projection results in a cylindrical texture mesh (see Fig. 8 (c)). Each vertex of the 2D cylindrical texture mesh has cylindrical coordinates with corresponding longitude (0-360 degrees) along the u-axis and vertical height along the v-axis. The resulting (u, v) coordinates map to a suitable aspect and resolution image (800×500 in our experiment). We also map the original generic mesh rendered with the transferred texture to the image plane using the same cylindrical projection. The result is a 800×500 *cylindrical texture image* in which each pixel value represents the surface color of the texture-mapped face surface in cylindrical coordinates (see Fig. 8 (d)). The generic head mesh can be textured by this cylindrical texture image using normalized 2D cylindrical coordinates as the texture coordinates.

6.2 Synthesizing A Full-Head Skin Texture

After having created the 2D texture mesh in the mesh parameterization step, we perform a vertex-to-image binding for all vertices of the 3D head mesh. This step is carried out by taking into account removal of undesired textures of cap and dark hair. A vertex on the generic mesh is bound to the input image, if it samples the scanned surface and takes valid texture coordinates of the sampling point in the model fitting procedure. Removal of cap and hair textures is done by unbinding the vertices with a color too dissimilar to the color of the forehead. We compute the average color and standard deviation of the vertices in the forehead and unbind those vertices that are at least η times the standard deviation away from the average. The parameter η should be chosen within [1.5,3], as it empirically proved to remove the problematic (cap and hair) textures. Fig. 9 shows the remaining texture with 2D mesh parameterization and its vertex binding visualized with color coding.

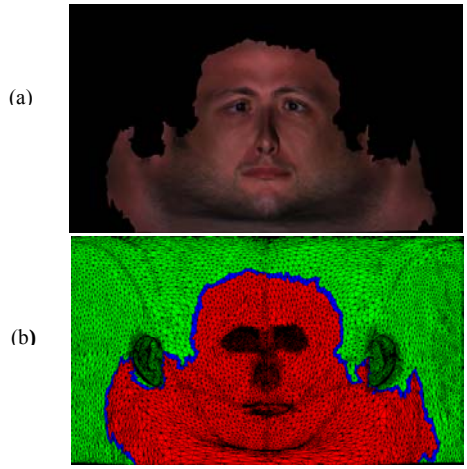


Fig. 9 (a) The resulting cylindrical texture image after cap and hair textures have been removed automatically. (b) Color-coded triangles of the texture mesh: each red triangle has all of its vertices bound to the input color image; blue triangles have at lease one bound vertex and one unbound vertex; the vertices of green triangles are all unbound.

Let $\Delta=(\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3)$ denote a triangle of the face mesh and $\tilde{\Delta}(\tilde{\mathbf{v}}_1,\tilde{\mathbf{v}}_2,\tilde{\mathbf{v}}_3)$ be the corresponding triangle in the texture mesh. For each triangle Δ, one of the following situations might occur (see Fig. 9 (b)):

1. There is a texture patch of the input image that can be mapped to Δ (red triangles).

2. Only one or two vertices of Δ are bound to the input image (blue triangles).

3. No vertex of Δ is bound to the input image (green triangles).

In the first case, we rasterize $\tilde{\Delta}$ in texture space. For each texel T_i, we color it with the color of the image pixel P_i that corresponds to T_i. In the second case, we color vertices of $\tilde{\Delta}$ that are bound to the input image with the colors of the corresponding pixels as in the first case. For each unbound vertex $\tilde{\mathbf{v}}_j$, we check the vertices in its one-ring neighbours that are colored by being bound to the input image. $\tilde{\mathbf{v}}_j$ is then colored by summing up the weighted colors of all the colored vertices $\tilde{\mathbf{v}}_i$ around it.

$$C(\tilde{\mathbf{v}}_j) = \frac{\sum_{i=1}^{n} \cos(\frac{d_i}{d_{\max}} \cdot \frac{\pi}{2}) C(\tilde{\mathbf{v}}_i)}{\sum_{i=1}^{n} \cos(\frac{d_i}{d_{\max}} \cdot \frac{\pi}{2})} \qquad (14)$$

where $C(\tilde{\mathbf{v}})$ is the color of the vertex $\tilde{\mathbf{v}}$, n is the number of colored neighbouring vertices, d_i is the length of the edge linking between $\mathbf{v}_i$ and $\mathbf{v}_j$ in the original 3D generic mesh, and d_{max} is the maximal edge length in the generic mesh. The weight term measures the normalized distance between two vertices, and favors the vertices that are much closer to the considered vertex. The texels of the rasterization of $\tilde{\Delta}$ can be grouped into two sets: T_t and T_c. Textured texel set T_t represents the set of texels that have a corresponding pixel in the input image. We thus color this set of texels with their corresponding pixel colors. If a texel T_i has no corresponding pixel, it is categorized into the colored texel set T_c. We determine its barycentric coordinates (ζ_i, κ_i, τ_i) with respect to $\tilde{\Delta}$ and compute the corresponding color $C(T_i)$ by interpolating the vertex colors of $\tilde{\Delta}$:

$$C(T_i) = \zeta_i C(\tilde{\mathbf{v}}_1) + \kappa_i C(\tilde{\mathbf{v}}_2) + \tau_i C(\tilde{\mathbf{v}}_3) \qquad (15)$$

We address the coloring problem in the last case in two stages: First, we iteratively assign an interpolated color to each unbound vertex. We then perform the color interpolation scheme for the remaining triangles of $\tilde{\Delta}$ that have not been colored. The first step iteratively loops over all unbound and uncolored vertices of the 2D texture mesh. For each unbound vertex $\tilde{\mathbf{v}}$, we check if the vertices in the one-ring around $\tilde{\mathbf{v}}$ are colored (either by being bound to the input image or by having an interpolated color). If this is true, we assign to $\tilde{\mathbf{v}}$ the weighted sum of colors of all the colored vertices around it using Eq. 14, otherwise we continue with the next unbound vertex. We repeat this procedure until there are no further vertex updates. After this step, the first round of vertices connecting to the vertices in case 2 has been colored. Next, we start the same procedure iteratively. At each iteration we color new round of vertices adjacent to the round of vertices colored in the last iteration. Upon termination of this loop, all vertices of the texture mesh are either bound or colored and the remaining triangles of $\tilde{\Delta}$ can be colored using the interpolation scheme (Eq. 15) from the second case. Fig. 10 shows the generated full-head texture.

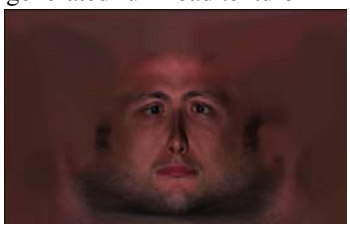


Fig. 10 Synthesized cylindrical full-head texture.

6.3 Texturing Ears

The ears have an intricate geometry with many folds and fail to project without overlap on a cylinder. Nevertheless, it is possible to quickly generate the texture from the input image where the ears are clearly visible.

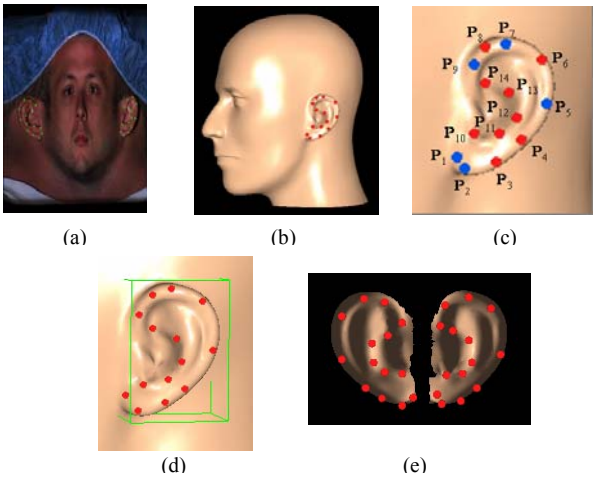


Fig. 11 (a) and (b) Positions of the feature points in the input 2D image and 3D generic model. (c) Ear feature points. Blue ones are used for the global alignment. (d) Bounding box around an ear. (e) Projected reference ear meshes with feature points.

We use a deformation technique based on feature points to warp the reference ear model to an individual ear model for extracting ear textures. We use the input image that contains the individual ears as the target model (see Fig. 1 (b)). We interactively identify a set of fourteen feature points for each ear in the input image, see Fig. 11 (a). As illustrated in Fig. 11 (b), our generic model is tagged with the same feature points by default. To segment ears, we predefine a bounding box enclosing each ear of the generic mesh (see Fig. 11 (d)). Before fitting the reference ears to the target shape, we need to transform positions of the reference ear model into the coordinate system of the target ear image. The segmented ears are

transformed and projected into the 2D image plane of the target ear image (see Fig. 11 (e)). For the target feature points in the input image, they can be easily detected and their image positions are calculated.

Given two sets of N corresponding feature points $\mathbf{p}_i$ in the projected reference ear mesh $\mathcal{E}$ and feature points $\mathbf{p}_i^*$ in the target 2D ear shape $\mathcal{E}^*$, we fit $\mathcal{E}$ to $\mathcal{E}^*$ in two steps: *global alignment* and *local adaptation*. We use five features points for the global alignment (see Fig. 11 (c)). The center of $\mathcal{E}$, $\mathbf{p}^c$, is defined as the midpoint between $\mathbf{p}_5$ and $\mathbf{p}^m$ which is the midpoint between $\mathbf{p}_1$ and $\mathbf{p}_9$. The center of $\mathcal{E}^*$, $\mathbf{p}^{*c}$, is calculated in the same way. Let an arbitrary vertex $\mathbf{x} \in \Re^2$ of $\mathcal{E}$ move to its new position $\mathbf{x}' \in \Re^2$, $\mathbf{S} \in \Re^{2\times 2}$ be the scaling matrix, $\mathbf{R} \in \Re^{2\times 2}$ be the rotation matrix, and $\mathbf{T} \in \Re^2$ be the translation vector. Eq. 16 computes the transformation for global alignment:

$$\mathbf{x}' = \mathbf{SR}\,(\mathbf{x} - \mathbf{p}^c) + \mathbf{T} \qquad (16)$$

Five parameters must be estimated, the in-plane rotation angle Θ, two scaling factors (s_u and s_v) and two translation components (t_u and t_v) along the u and v texture coordinate axes. Θ is estimated as the angle between vectors $\overrightarrow{\mathbf{P}_2\mathbf{P}_7}$ and $\overrightarrow{\mathbf{P}_2^*\mathbf{P}_7^*}$. The scaling factors are estimated as:

$$s_u = \frac{\left\|\mathbf{P}_5^* - \mathbf{P}^{*\mathrm{m}}\right\|}{\left\|\mathbf{P}_5 - \mathbf{P}^{\mathrm{m}}\right\|}, \quad s_v = \frac{\left\|\mathbf{P}_2^* - \mathbf{P}_7^*\right\|}{\left\|\mathbf{P}_2 - \mathbf{P}_7\right\|} \qquad (17)$$

The translation vector is estimated by matching the model center of $\mathcal{E}$ with that of $\mathcal{E}^*$. The same transformation is applied to the feature points $\mathbf{p}_i$ to get their new positions $\mathbf{p}'_i$. Fig. 12 (a) shows the global alignment results.

In the local adaptation we construct a smooth interpolation function that gives the displacements between the original point positions and the new adapted positions for every vertex of the transformed reference ear mesh $\mathcal{E}'$. Let w_i be the sum of weights from all feature points contributed to a ear mesh vertex $\mathbf{x}'_i$, and l_{ij} be the distance between the vertex $\mathbf{x}'_i$ and a feature point $\mathbf{p}'_j$. Eq. 18 computes the displacement applied to $\mathbf{x}'_i$:

$$\Delta\mathbf{x}_i = \sum_{j=1}^{N} \frac{w_i - l_{ij}}{w_i(N-1)} \Delta\mathbf{P}_j e^{-\frac{l_{ij}}{\mu}} \qquad (18)$$

where $\Delta\mathbf{P}_j$ is the displacement of feature point $\mathbf{p}'_j$:

$$\Delta\mathbf{P}_j = \mathbf{P}_j^* - \mathbf{P}_j', \qquad w_i = \sum_{j=1}^{N} l_{ij} \qquad (19)$$

The decay factor μ is determined by the ear size. We set it to 1/20 of the diagonal length of the bounding box of the reference ear model.

After fitting the ear, texture coordinates for all vertices of the reference ear mesh are obtained by normalizing the final vertex positions to the domain $[0, 1]^2$. Fig. 12 (b) shows the local adaptation results. In the final head rendering, the ear parts are texture-mapped in a separate process using the input image shown in Fig. 1 (b).

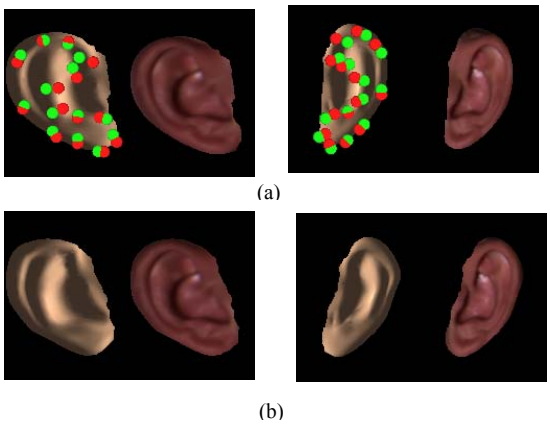


Fig. 12 (a) Globally aligned reference ear meshes with smooth shading and texture mapping. Red and green dots represent the transformed source feature points and detected target feature points, respectively. (b) After the local adaptation.

7. Results

The final texture-mapped head is shown in Fig. 13, where the texture image shown in Fig. 10 is applied to the adapted generic head model shown in Fig. 4 (a). The scanned data in Fig. 1 has a nice shape, but dose not have good shape and texture for the top and back of the head. Through our method using deformation and texture synthesis with the generic model, it has texture on these parts, which makes full rotation of a head. We have used our system to generate full-head textures for subjects whose face scans are in our database. Female and male in different races with different skin color attributes are reconstructed and textured, as shown in Fig. 14. Rendering of a head model is performed in real-time using OpenGL hardware (about 120 fps on a 2.4 GHz PC with an ATI FireGL X1 graphics board).

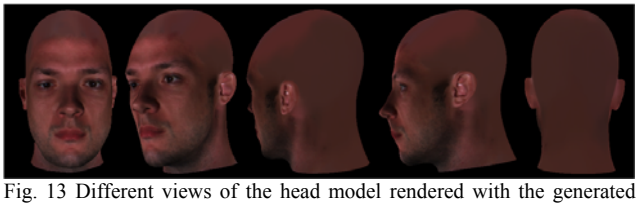


Fig. 13 Different views of the head model rendered with the generated full-head texture.

Our method establishes the necessary mapping between different face scans through a generic model which enables us to morph between any two reconstructed models. Together with the geometry morphing, we blend the textures. The generated full-head textures possess the correspondence, enabling 2D texture metamorphosis. Fig. 15 shows a dynamic morphing between different models.

In our method, the only interactive step is the initial tuning of the 2D feature mask. This process takes about 2

minutes. The automated method is then executed to generate a full-head texture image. The RBF calculation and warping of the generic model take about 2 seconds on a 2.4 GHz Pentium 4. With a data set of 140k points, the local deformation process runs for about 14 seconds. Computing a parameterization of the generic mesh (approx. 12k triangles) takes about 0.2 seconds. The texture synthesis process performs 30 iterations in approx. one minute. Ear fitting process runs for 0.8 seconds to fit two ears. Given the scanned data, the whole process of creating a full-head texture takes about 4 minutes.

Table 2: Number of eigenmeshes and anthropometric control parameters used for face shape synthesis

	Eyes	Nose	Mouth	Chin
Number of eigenmeshes used for shape synthesis	23	26	20	18
Number of anthropometric control parameters	13	20	12	7

For each facial feature, the original full set of 186 eigenmeshes is reduced to a small set which explains 98% of the shape variation in the database. Table 2 shows the number of eigenmeshes and number of input anthropometric controls used for our face shape synthesis. The user can select the facial feature to work on using a windows GUI (see Fig. 16). Using a mouse he can modify the position of a set of sliders, each one related to one anthropometric control of a facial feature. The anthropometric parameter values are chosen within [0,1] to generate realistic face shapes.

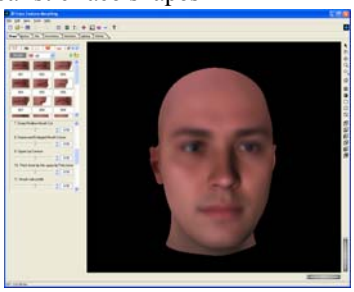


Fig. 16 GUI of our system.

Fig. 17 shows a number of synthesized facial shapes on the average model which is the average shape of reconstructed 186 head models and textured with the mean cylindrical full-head texture image. A wide range of variations are exhibited across the generated faces; clear differences are found in the width of the nose alar wings, the straightness of the nose bridge, the inclination of the nose tip, the roundness of eyes, the distance between eyebrows and eyes, the thickness of mouth lips, the shape of the lip line, the sharpness of the chin, etc. Note that it is not necessary to begin with the average model. We can start with any face model of a specific person and edit various aspects of its shape. The editing focuses on the characteristics of the face interesting to the user while preserving the identity of the subject. Fig. 18 illustrates face editing results on the models of two individuals for various user-intended characteristics. At runtime, our scheme spends about 0.08 seconds in generating a new face shape upon receiving the input anthropometric parameters, corresponding to a rate of 12 fps.

8. Conclusion

We have presented new techniques for creating photorealistic 3D head models rendered with a full-head texture, and for generating natural looking face shapes by synthesizing facial features according to anthropometric parameters. We make use of the scanned face data of real people, which provides the best available resource to regulate the naturalness of modelled faces. In order to establish correspondence among scanned data, we use a two-step model fitting approach to conform a generic head mesh to the particular geometry of the subject's face. We transform the obtained data sets of facial feature shapes into vector space representations by applying the PCA. Using the PCA coefficients as a compact shape representation, we approach the shape synthesis problem by forming scattered data interpolation functions that are devoted to the generation of desired shape by taking the anthropometric parameters as input. At runtime, the interpolation functions are evaluated at the input parameter values to produce new face shapes at an interactive rate. We automatically generate a full-head texture from parameterized texture of the face region. In particular, we address the creation of individual ear textures. We have demonstrated personalized head models rendered with the generated textures and various realistic face shapes generated through anthropometric controls.

We envisage several further developments from our current results. We would like to incorporate more face examples into the existing database, including more diversity of age and race. In order to generate more realistic faces, we would also like to increase the number of facial features to choose from. The possible candidates are the cheeks, forehead and upper jaw. Moreover, we plan to extend our framework on synthesizing local textures of facial features. As anthropometric measurements are often correlated, we plan to improve the shape control by using the facial proportion statistics [28] to model the correlations between measurements. Finally, automatic reconstruction of hair texture from images is one of the future challenges.

Fig. 14 Examples of texture-mapped head models of various people.

Fig. 15 Dynamic morphing between different texture-mapped head models (circled).

Fig. 17 Automatically generated face models by synthesizing the shapes of four facial features on the average model (outlined) according to the input anthropometric parameters.

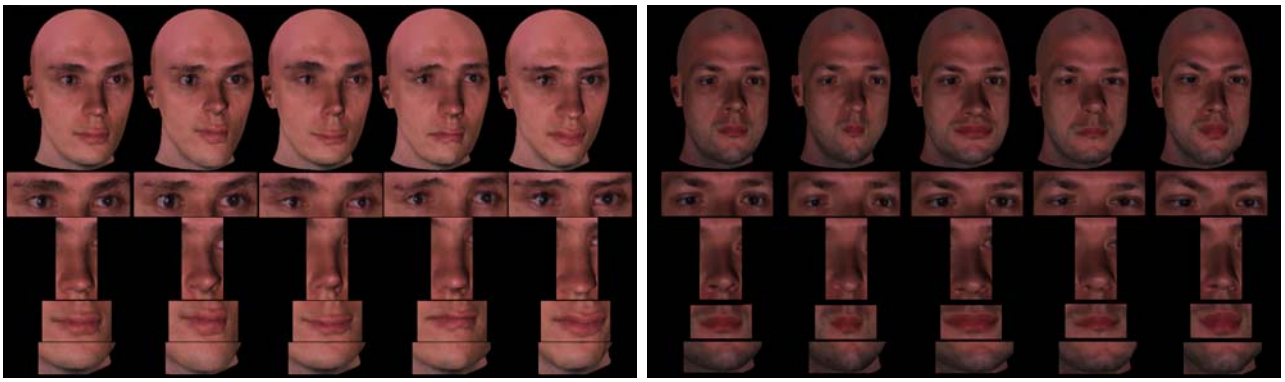


Fig. 18 Feature-based face editing on the models of two individuals. The original model is shown in the first image of each example.

References

[1] A. Young and D. Hay, Configurational Information in Face Perception, Experimental Psychology Society, 1986.

[2] F. I. Parke, "Computer generated animation of faces," Master's thesis, University of Utah, Salt Lake City, 1972.

[3] F. I. Parke and K. Waters, Computer Facial Animation, AK Peters, Wellesley, MA, 1996.

[4] S. DiPaola, "Extending the range of facial types," Journal of Visualization and Computer Animation, vol.2, pp.129-131, 1991.

[5] F. I. Parke, "Parameterized models for facial animation," IEEE Computer Graphics and Application, vol.2, pp.61-68, 1982.

[6] M. Patel and P. Willis, "FACES: the facial animation, construction and editing system," Proc. Eurographics'91, pp.33-45, 1991.

[7] T. Akimoto, Y. Suenaga and R. S. Wallace, "Automatic creation of 3D facial models," IEEE Computer Graphics and Application, vol.13, pp.16-22, 1993.

[8] H. H. S. Ip and L. Yin, "Constructing a 3D individualized head model from two orthogonal views," The Visual Computer, vol.12, pp.254-266, 1996.

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

[10] Z. Liu, Z. Zhang, C. Jacobs and M. Cohen, "Rapid modeling of animated faces from video," Journal of Visualization and Computer Animation, vol.12, pp.227-240, 2001.

[11] C. Kuo, R. Huang and T. Lin, "3-D facial model estimation from single front-view facial image," IEEE Trans. on Circuits and Systems for Video Technology, vol.12, pp.183-192, 2002.

[12] I. K. Park, H. Zhang, V. Vezhnevets and H. K. Choh, "Image-based photorealistic 3D face modeling," Proc. IEEE Automatic Face and Gesture Recognition, pp. 49-54, 2004.

[13] R. Enciso, J. Li, D. Fidaleo, T-Y. Kim, J-Y. Noh and U. Neumann, "Synthesis of 3D faces," International Workshop on Digital and Computational Video, pp.146-151, 1999.

[14] Y. Lee, D. Terzopoulos and K. Waters, "Realistic modeling for facial animation," Proc. SIGGRAPH'95, pp.55-62, 1995.

[15] K. Kahler, J. Haber, H. Yamauchi and H. P. Seidel, "Head shop: Generating animated head models with anatomical structure," Proc. ACM SIGGRAPH Symp. on Comput. Anim., pp.55-64, 2002.

[16] Y. Zhang, E. C. Prakash and E. Sung, "Constructing a realistic face model of an individual for expression animation," International Journal of Information Technology, vol.8, pp.10-25, 2002.

[17] D. DeCarlo, D. Metaxas and M. Stone, "An anthropometric face model using variational techniques," Proc. SIGGRAPH'98, pp.67-74, 1998.

[18] D. Vlasic, M. Brand, H. Pfister and J. Popovic, "Face transfer with multilinear models," Proc. SIGGRAPH'05, pp.426-433, 2005.

[19] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," Proc. SIGGRAPH'99, pp. 187-194, 1999.

[20] L. Williams, "Performance-driven facial animation," Proc. SIGGRAPH'90, pp.235-242, 1990.

[21] S. Marschner, B. Guenter and S. Raghupathy, "Modeling and rendering for realistic facial animation," Proc. 11th EG Workshop on Rendering, pp.231-242, 2000.

[22] C. Rocchini, P. Cignoni, C. Montani and R. Scopigno, "Acquiring, stitching and blending diffuse appearance attributes on 3D models," The Visual Computer, vol.18, pp.186-204, 2002.

[23] X. Li and H. Zha, "Realistic human head modeling with multi-view hairstyle reconstruction," Proc. 5th International Conference on 3D Digital Imaging and Modeling, pp. 432-438, 2005.

[24] USF DARPA HumanID 3D Face Database, Courtesy of Prof. Sudeep Sarkar, Univ. of South Florida, Tampa, FL.

[25] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham, "Active shape models: Their training and applications," Computer Vision and Image Understanding, vol.16, pp.38-59, 1995.

[26] M. J. D. Powell, "Radial basis functions for multivariate interpolation: A review," in Algorithms for Approximation, J. C. Mason and M. G. Cox, Eds. Clarendon Press, 1987.

[27] L. G. Farkas, Anthropometry of the Head and Face, Raven Press, 1994.

[28] L. G. Farkas, Anthropometric Facial Proportions in Medicine, Thomas Books, 1987.

Yu Zhang received the BE and ME degrees from Northwestern Polytechnical Univ., Xi'an, China, in 1997 and 1999, respectively. He received the PhD degree from Nanyang Technological Univ., Singapore, in 2004. From 2003 to 2005, he was a research fellow in the School of Computing, National Univ. of Singapore. In 2005, he worked as a research scientist at the Genex Technologies, Inc., U.S.A. He has been a postdoctoral researcher in the Computer and Information Science Department at Univ. of Pennsylvania since 2006. His research interests are in the areas of computer graphics, computer animation, physically-based modeling, visualization, and virtual reality. He is a member of IEEE Computer Society and ACM SIGGRAPH.

Chew Lim Tan received the BS (Hons) degree in 1971 from the Univ. of Singapore, the MS degree in 1973 from the Univ. of Surrey, UK, and the PhD degree in 1986 from the Univ. of Virginia, U.S.A. He is an Associate Professor in the Department of Computer Science, School of Computing, National Univ. of Singapore. His research interests include document image and text processing, neural networks and genetic programming. He is an associate editor of Pattern Recognition and the current President of the Pattern Recognition and Machine Intelligence Association (PREMIA) in Singapore.