AN EFFICIENT TEXTURE GENERATION TECHNIQUE FOR HUMAN HEAD CLONING AND MORPHING

Yu Zhang
Department of Computer Science, School of Computing
National University of Singapore, Singapore 117543

Keywords: Head modeling, texture mapping, real-time rendering, morphing, model adaptation.

Abstract: This paper presents a technique to efficiently generate a parameterized full-head texture from a single face image of the scanned data for modeling photorealistic 3D heads. We automatically register face scans in a database by deforming a generic head mesh to fit the specific person's face geometry with a volume morphing approach. After the face texture is transferred onto it, the 3D generic mesh is parameterized over a 2D texture domain to establish a correspondence between all the scanned textures. After having performed a vertex-to-image binding for all vertices of the head mesh, we automatically generate a full-head texture using color interpolation for unbound regions and weighted average splines for visual boundary removal. We also use a deformation method to extract ear textures from the input image for texturing individual ears. With the exception of the initial feature point selection, our method is fully automated. We show photorealistic and real-time head rendering and morphing with the resulting texture.

1 INTRODUCTION

Among the tasks of reproducing the complexity of the face in the computer, rendering is crucial for generating visually convincing appearance of a particular person's face. Basic rendering algorithms include polygonal shading (flat shading, Gouraud shading and Phong shading), ray tracing and radiosity. However, facial skin has reflectance properties that are not modeled well by these widely-used shading models: the produced facial image appears plastic and cartoon-like with a too-smooth surface.

For sheer photorealism, texture mapping is a popular technique to get a better rendered image. 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 accurately acquire both the face geometry and texture. One limitation of the scanner technology, however, is that the complete head geometry can not 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 or the subject wears a light-color cap, but that destroys the texture. In most cases, only the frontal face can be properly textured. There is no automatic mechanism provided to generate a full-head texture from the acquired single frontal-face image for realistic rendering towards a "cloned" head.

In this paper, we present a technique to efficiently generate a parameterized full-head texture for modeling heads with a high degree of realism. We start with a generic head model with a known topology. It is deformed to fit the face scan of the particular human subject using a volume morphing approach. The facial texture associated with the scanned geometry is then transferred to the original undeformed generic mesh. We automatically construct a parameterization of the 3D head mesh over a 2D texture domain, which gives immediate correspondence between all the scanned textures via a single, prototype layout. After having performed a vertex-to-image binding for vertices of the head mesh, we generate a cylindrical full-head texture from the remaining parameterized texture of the face area. We also address the creation of individual textures for ears. Apart from an initial feature point selection for the texturing, our method works automatically without any user interaction.

Our main contribution is a technique that uses a frontal-face image of the scanned data to generate a full-head texture for photorealistic rendering and morphing with minimal manual intervention. This includes the new algorithms to automatically para-

Zhang Y. (2006).
AN EFFICIENT TEXTURE GENERATION TECHNIQUE FOR HUMAN HEAD CLONING AND MORPHING.
In *Proceedings of the First International Conference on Computer Graphics Theory and Applications*, pages 267-275
DOI: 10.5220/0001352202670275

meterize textures of a set of unregistered face scans to establish the mapping correspondence, to robustly produce individual full-head skin texture, and to efficiently create ear textures from a single input image.

This paper is organized as follows. Section 2 reviews the previous work. Section 3 presents the face data we use. Section 4 describes the model adaptation process. The approaches to mesh parameterization, full-head texture synthesis, and creation of ear textures are elaborated in Section 5. We show experimental results in Section 6. Section 7 concludes by discussing future work.

2 PREVIOUS WORK

Beginning with Parke's pioneering work (Parke, 1972), desire for improved realism has driven researchers to extend geometric models (Parke, 1982; Thalmann et al., 1989) with physically-based models which attempt to model the influence of muscle contraction onto the skin surface by approximating the biomechanical properties of skin (Kahler et al., 2002; Lee et al., 1995; Platt and Badler, 1981; Terzopoulus and Waters, 1990; Waters, 1987). Physically-based models with layered anatomical structure were combined with non-linear finite element methods (Koch et al., 1996) in systems that could be used for planning facial surgeries. Free-form deformations have been employed in (Kalra et al., 1992) to manipulate facial expressions. In parallel, Williams presents a compelling argument (Williams, 1990) in favor of performance-driven facial animation, which anticipated techniques for tracking head motions and facial expressions in video (Essa and Basu, 1996; Pighin et al., 1999). In performance-driven approaches, colored markers painted on the face are extensively used to aid in tracking facial motion (Guenter et al., 1998). A more expensive alternative could use a 3D scanning technique (Zhang et al., 2004a), if the performance can be re-recorded with such a system.

A variational approach is presented in (DeCarlo et al., 1998) to create a range of static face models with realistic proportions. They use anthropometric measurements, which constrain the deformation of a B-spline generic head model. Pighin et al. (Pighin et al., 1998) combine 2D morphing with 3D transformations of the geometric model to produce photorealistic facial animation. In addition, Blanz and Vetter (Blanz and Vetter, 1999) present a process for estimating the shape of a face in a single photograph, and a set of controls for intuitive manipulation of appearance attributes (thin/fat, feminine/masculine).

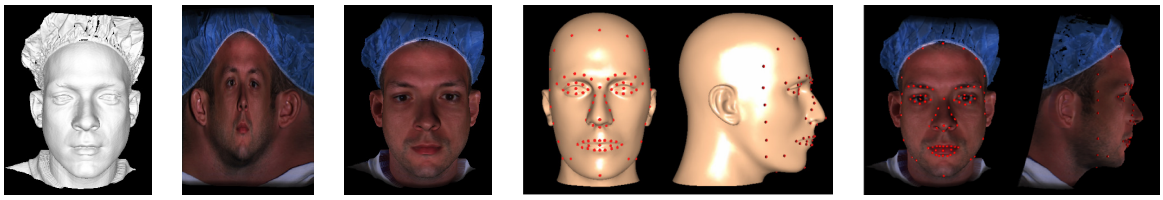
Texturing in the context of face modeling is, however, an often neglected issue. Williams (Williams, 1990) presents an approach to generate and register a texture map from a peripheral photograph. This approach is meanwhile superseded by the ability of laser scanners to acquire geometry and texture in one step (Lee et al., 1995; Waters and Terzopoulos, 1991). The method presented in (Blanz and Vetter, 1999) generates an individual face geometry and texture by linear combination of face geometries and textures from a large database. Marschner et al. (Marschner et al., 2000) describe 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.

Several image-based approaches (Akimoto et al., 1993; Guenter et al., 1998; Lee and Thalmann, 2000; Liu et al., 2001; Pighin et al., 1998) use a small number of input photographs (or video streams) for the reconstruction of both geometry and texture. Although they could potentially yield a higher texture quality compared to the scanned textures, they typically suffer from a less accurate geometry reconstruction and limited animation. Generating high-resolution textures for facial skin and eyes and teeth from several uncalibrated photographs of a person's head has been addressed in (Tarini et al., 2002). In their method, the geometry of the head is acquired using a structured-light range scanner. The photographs are registered and combined into a single texture suitable for mipmapping based on the multiple textures blending technique (Rocchini et al., 1999). In contrast, not requiring separate camera setups, we synthesize a full-head texture from a reflectance image acquired by a laser scanner which captures color of only the face area.

3 FACE DATA

We use a database that contains laser scans of 186 human faces (126 male and 60 female). Each subject is captured wearing a bathing cap and with a neutral expression (see Fig. 2). The laser scans provide face structure data (see Fig. 1 (a)) and RGB-color values that are stored in a $360\times$ 524 image with 8 bit per channel (see Fig. 1 (b)). The image is registered against the range data and can be used for texture-mapping (see Fig. 1 (c)). We use a generic model created with Alias|Wavefront to resemble an average human head. It consists of 2,632 vertices and 5,221 triangles. Prescribed colors are added to each triangle to form a smooth-shaded surface (see Fig. 1 (d)).

We have interactively specified 83 feature points on scanned face geometry. Our generic model is already tagged with the same feature points by default. The feature points are located around the eyes, eyebrows, nose, mouth and chin (see Fig. 1). In the following, the feature points on the generic head model are de-



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

Figure 1: Face data: (a) scanned face geometry; (b) acquired color image; (c) texture-mapped face scan; (d) and (e) feature points specified on the generic model and face scan, respectively.

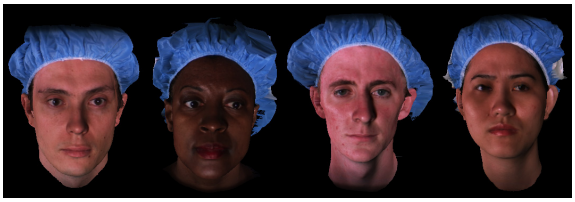


Figure 2: Some exemplar 3D data of the face database.

noted as the *source feature points*, whereas those on the scanned surface are called *target feature points*.

4 MODEL ADAPTATION

4.1 Global Warping

The global warping is based on manually specified corresponding feature point on the generic model and the scanned data. Denote two sets of corresponding 3D points given by $\mathbf{s}_i$ and $\mathbf{t}_i$ $(i = 1, \ldots, n)$ as the source and target feature points, respectively. We need to find a function $\mathbf{f}$ that maps the $\mathbf{s}_i$ to the $\mathbf{t}_i$:

$$\mathbf{t}_i = \mathbf{f}(\mathbf{s}_i) \qquad i = 1, \ldots, n \tag{1}$$

The goal is to construct a smooth interpolating function that expresses the deformation of the non-feature vertices in terms of the changes in the feature points during morphing. This problem is addressed by scattered data interpolation methods. Radial Basis Functions (RBFs) are a popular means of scattered data interpolation. The interpolant using RBFs is a function that utilizes the known displacement of each feature point and returns the displacement value for each non-feature point that takes it from the original position to its position in the target form. Such a mapping can be expressed by a weighted linear combination of n basic functions ϕ_i defined by the source feature points and an explicit affine transformation:

$$\mathbf{f}(\mathbf{s}) = \sum_{i=1}^{n} \mathbf{c}_i \phi_i(\|\mathbf{s} - \mathbf{s}_i\|) + \mathbf{R}\mathbf{s} + \mathbf{t} \tag{2}$$

where $\mathbf{s} \in \mathcal{R}^3$ is a vertex on the generic model, $\mathbf{c}_i \in \mathcal{R}^3$ are (unknown) weights, $\|\cdot\|$ denotes Euclidean distance, $\mathbf{R} \in \mathcal{R}^{3\times3}$ adds rotation, skew, and scaling, and $\mathbf{t} \in \mathcal{R}^3$ is a translation component. Many different functions for $\phi(r)$ have been proposed (Nielson, 1993). We evaluated several functions including the multi-quadric $\phi(r) = \sqrt{r^2 + \rho^2}$, the thin-plate spline $\phi(r) = r^2 log(r)$, the Gaussian $\phi(r) = exp(-\rho r^2)$, and the biharmonic $\phi(r) = r$. We had better results visually with the multi-quadric function.

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

$$\sum_{i=1}^{n} \mathbf{c}_i = 0, \qquad \sum_{i=1}^{n} \mathbf{c}_i^T \mathbf{s}_i = 0 \tag{3}$$

The system of linear equations (Eq. 1 and 3) is solved for the unknowns $\mathbf{R}$, $\mathbf{t}$, and $\mathbf{c}_i$ using a standard LU decomposition with pivoting, to obtain the final warp function $\mathbf{f}$. This function can then be used to transform a vertex $\mathbf{s}$ in the volume spanned by the feature points. Fig. 3 (a) shows the shape of the head model after having interpolated the 3D displacements at 83 feature points and applied them to the entire head.

(a) (b) (c)

Figure 3: Adaptation of the generic model to scanned data: (a) source model after global warping; (b) final mesh geometry after local deformation; (c) textured.

4.2 Local Deformation

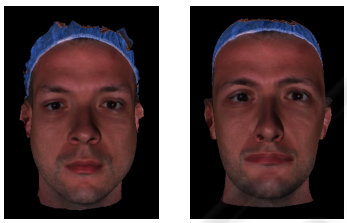
The global warping with a small set of initial correspondences does not produce a perfect surface match. We further improve the shape using a local deformation which ensures that all the generic mesh vertices are truly embedded in the scanned surface. The local deformation is based on the closest points on the surfaces of the generic model and the scanned data. The polygons 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.

After the model fitting, texture from the reflectance image is applied to the deformed generic model. As each vertex on the fitted generic mesh samples a scanned point, it takes the texture coordinates associated with that point. Fig. 3 (b) and (c) show the results of smoothly shading and texture mapping.

5 TEXTURING A HEAD

5.1 Mesh Parameterization

Our head skin texture is generated from the acquired color image, as 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 pairwise 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 shape free face texture images by constructing a parameterization of the 3D generic mesh over a 2D image plane.



(a) (b)

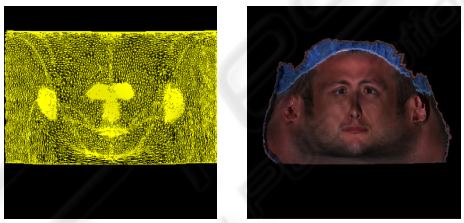
Figure 4: (a) Textured adapted generic model. (b) Texture transferred to the undeformed generic model.

Given the vertex-wise correspondence between the adapted generic head mesh and the original undeformed generic mesh, it is trivial to transfer a texture map between this pair of meshes. Each vertex on the original generic mesh simply takes the texture coordinates of its corresponding vertex on the adapted mesh. Fig. 4 shows the texture transferred from the adapted generic model onto the original undeformed mesh.

We want to parameterize the 3D generic head mesh over a 2D domain $[0, 1]^2$ in order to obtain a single texture map for the whole mesh. We implement a cylindrical projection which is preferable for regions corresponding to a volumetric surface. A cylindrical triangular mesh corresponding to the generic model is constructed on a virtual cylinder enclosing the model. The projection results in a cylindrical face mesh (see Fig. 5 (a)). Each vertex of the 2D cylindrical mesh has cylindrical coordinates with corresponding longitude (0-360 degrees) along the x-axis and vertical height along the y-axis. The mapping of world coordinates $\mathbf{x} = (x, y, z)$ to 2D cylindrical coordinates (u, v) uses

$$u = \tan^{-1}(\frac{x}{z}), \qquad v = y \tag{4}$$

The resulting (u, v) coordinates map to a suitable aspect and resolution image (512×512 in our experiment). We also map the original generic mesh rendered with transferred texture to the image plane using the same cylindrical projection. The result is a 512×512 *cylindrical texture image* in which each pixel value represents the surface color of the texture-mapped face surface in cylindrical coordinates (see Fig. 5 (b)). The generic face mesh can be textured by this cylindrical texture image using normalized 2D cylindrical coordinates as the texture coordinates.



(a) (b)

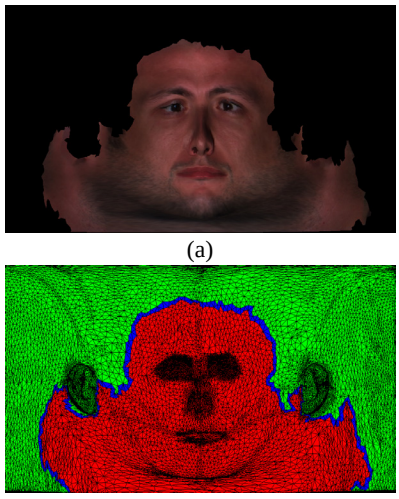
Figure 5: (a) Texture mesh parameterization. (b) Cylindrical texture image.

5.2 Synthesizing A Full-Head Skin Texture

After having created the 2D texture mesh from the 3D generic head mesh, 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 adaptation 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 λ times the standard deviation away from the average. The parameter λ should be chosen within [1.5,3], as it empirically proved to remove the problematic (cap and hair) textures in most cases. Fig. 6 shows the remaining texture with 2D mesh parameterization and its vertex binding visualized with color coding.

Let $\triangle = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ denote a triangle of the face mesh and $\tilde{\triangle} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3)$ be the corresponding triangle in the texture mesh. For each triangle $\triangle$, one of the following situations might occur (see Fig. 6 (b)):

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



(a) (b)

Figure 6: (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.

2. Only one or two vertices of $\triangle$ are bound to the input image (blue triangles).
3. No vertex of $\triangle$ is bound to the input image (green triangles).

In the first case, we rasterize $\tilde{\triangle}$ 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 $\tilde{\triangle}$ that are bound to the input image with the pixel colors of the corresponding pixels simply as in the first case. For each unbound vertex $\tilde{\mathbf{x}}_j$, we check the vertices in its one-ring neighbors that are colored by being bound to the input image. $\tilde{\mathbf{x}}_j$ is then colored by summing up the weighted colors of all the colored vertices $\tilde{\mathbf{x}}_i$ around it.

$$C(\tilde{\mathbf{x}}_j) = \frac{\sum_{i=1}^{n} \cos(\frac{d_i}{d_{max}} \cdot \frac{\pi}{2}) C(\tilde{\mathbf{x}}_i)}{\sum_{i=1}^{n} \cos(\frac{d_i}{d_{max}} \cdot \frac{\pi}{2})} \quad (5)$$

where n is the number of colored neighboring vertices, d_i are lengthes of the edges linking between $\mathbf{x}_i$ and $\mathbf{x}_j$ 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. With the vertex-to-image binding information, the texels of the rasterization of $\tilde{\triangle}$ can be grouped into two sets: $\mathbf{T}_t$ and $\mathbf{T}_c$. Textured texel set $\mathbf{T}_t$ 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 $\mathbf{T}_c$. For each $T_i \in \mathbf{T}_c$, we determine its barycentric coordinates $(\alpha_i, \beta_i, \gamma_i)$ w.r.t. $\tilde{\triangle}$ and compute the corresponding color $C(T_i)$ by interpolating the vertex colors of $\tilde{\triangle}$:

$$C(T_i) = \alpha_i C(\tilde{\mathbf{x}}_1) + \beta_i C(\tilde{\mathbf{x}}_2) + \gamma_i C(\tilde{\mathbf{x}}_3) \quad (6)$$

In the last case, all vertices of $\tilde{\triangle}$ are unbound to the input image and can not be colored by any of the previously described schemes. This problem is addressed in a two-stage process: First, we iteratively assign an interpolated color to each unbound vertex. We then perform the color interpolation scheme for the remaining triangles of $\tilde{\triangle}$ 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 $\tilde{\mathbf{x}}$, we check if the vertices in the one-ring around $\tilde{\mathbf{x}}$ are colored (either by being bound to the input image or by having an interpolated color). If this is true, we assign to $\tilde{\mathbf{x}}$ the weighted sum of colors of all the colored vertices around $\tilde{\mathbf{x}}$ using Eq. 5, 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 $\tilde{\triangle}$ can be colored using the interpolation scheme from the second case.

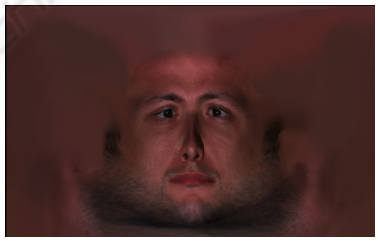


Figure 7: Synthesized cylindrical full-head texture.

Since the input image has been acquired under uncontrolled illumination, the skin color might differ noticeably between the left and right sides of the face. In this case, boundary might appear at the intersection between the sagittal plane (vertical plane cutting through the center of the face) and back of the head. We apply a weighted average spline method (Peleg, 1981) to remove the visual boundary. Fig. 7 shows the generated full-head texture after having synthesized from the remaining face texture and applied the weighted average spline to smoothly blend pixel colors in the leftmost and rightmost texture regions.

5.3 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 automatically and quickly

generate the texture from a single input image where the ears are clearly visible.

We use a deformation technique based on feature points to warp the reference ear model to an individual ear model for obtaining appropriate texture coordinates. We use the acquired 2D image that contains the individual ears as the target model (see Fig. 1 (b)). We identify a small set of feature points in the input image. In practice, we use fourteen feature points for each ear, see Fig. 8 (a). As illustrated in Fig. 8 (b), our generic model is tagged with the same feature points by default. To segment ears, we predefine a bounding box enclosing each ear for the generic mesh (see Fig. 8 (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 onto the 2D image plane of the target ear image (see Fig. 8 (e)). For the target feature points in the input image, they can be easily detected due to their distinct color and their image positions are calculated.

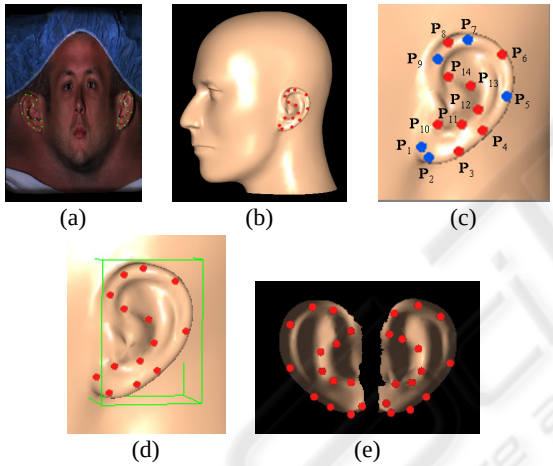


Figure 8: (a) and (b) Position of feature points in the input 2D image and 3D generic model, respectively. (c) Ear feature points. Blue ones are used for the global alignment. (d) Bounding box around a ear. (e) Projected reference ear meshes with feature points.

Given two sets of N corresponding source feature points $\mathbf{p}_i$ and target feature points $\mathbf{p}_i^*$ in the image plane, we fit the projected reference ear mesh $\mathcal{F}$ to the target ear shape in the input image $\mathcal{F}^*$ in a global-to-local fashion. Our ear fitting process consists of global alignment and local adaptation. We use a subset consisting of five features points for the global alignment (see Fig. 8 (c)). The center of $\mathcal{F}$, $\mathbf{p}^c$, 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 $\mathcal{F}^*$, $\mathbf{p}^{*c}$, is calculated in the same way. Let an arbitrary vertex $\mathbf{x} \in R^2$ on $\mathcal{F}$ move to its new position $\mathbf{x}' \in R^2$, $\mathbf{S} \in R^{2\times 2}$ be the scaling matrix, $\mathbf{R} \in R^{2\times 2}$ be the rotation matrix, and $\mathbf{T} \in R^2$ be the translation vector. Eq. 7 computes the transformation:

$$\mathbf{x}' = \mathbf{SR}(\mathbf{x} - \mathbf{p}^c) + \mathbf{T} \tag{7}$$

In principle, five parameters must be estimated, the in-plane rotation angle r, 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. The rotation angle r 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 from the ratio of lengths between a pair of feature points as measured both in $\mathcal{F}$ and $\mathcal{F}^*$:

$$s_u = \frac{\| \mathbf{p}_5^* - \mathbf{p}^{*m} \|}{\| \mathbf{p}_5 - \mathbf{p}^m \|}, \qquad s_v = \frac{\| \mathbf{p}_2^* - \mathbf{p}_7^* \|}{\| \mathbf{p}_2 - \mathbf{p}_7 \|} \tag{8}$$

The translation vector is estimated by matching the model center of $\mathcal{F}$ with that of $\mathcal{F}^*$. The same transformation is applied to the source feature points $\mathbf{p}_i$ to get their new positions $\mathbf{p}_i'$ for further local adaptation. Fig. 9 (a) shows the global alignment results.

Local adaptation involves non-rigid deformation of the transformed 2D reference ear mesh $\mathcal{F}'$ to fit the target ear shape. Once we have computed a set of 2D coordinates for the transformed source feature points $\mathbf{p}_i'$, we use these values to deform the remaining vertices on $\mathcal{F}'$. We construct a smooth interpolation function that gives the displacements between the original point positions and the new adapted positions for every vertex on $\mathcal{F}'$. Let N be the number of feature points, 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. 9 computes the displacement applied to $\mathbf{x}_i'$.

$$\triangle \mathbf{x}_i = \sum_{j=1}^{N} \frac{w_i - l_{ij}}{w_i(N-1)} \triangle \mathbf{p}_j e^{-\mu l_{ij}} \tag{9}$$

where μ is a decay factor determined by the ear size, and $\triangle \mathbf{p}_j$ is the displacement of feature point $\mathbf{p}_j'$:

$$\triangle \mathbf{p}_j = \mathbf{p}_j^* - \mathbf{p}_j', \qquad w_i = \sum_{j=1}^{N} l_{ij} \tag{10}$$

Texture coordinates for all vertices on the reference ear model are obtained by normalizing the newly adapted vertex positions in the image plane to the domain $[0, 1]^2$. Fig. 9 (b) shows the local adaptation results. In the final rendering of the head, the ear parts are textured in a separate process with the assigned individual texture map shown in Fig. 1 (b).

6 RESULTS

Fig. 10 shows the final texture-mapped head, where the texture image shown in Fig. 7 is applied to the adapted generic head model shown in Fig. 3 (a). The scanned data in Fig. 1 has a nice shape, but dose not have good shape and texture for back part and ears. Through our method using deformation and texture

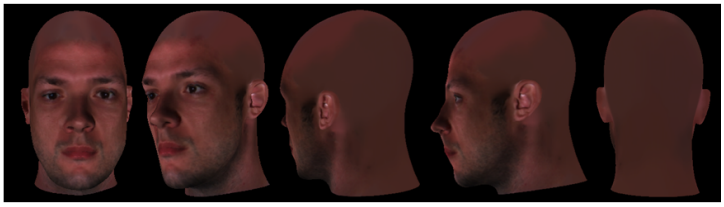


Figure 10: Different views of the head model rendered with the generated full-head texture.

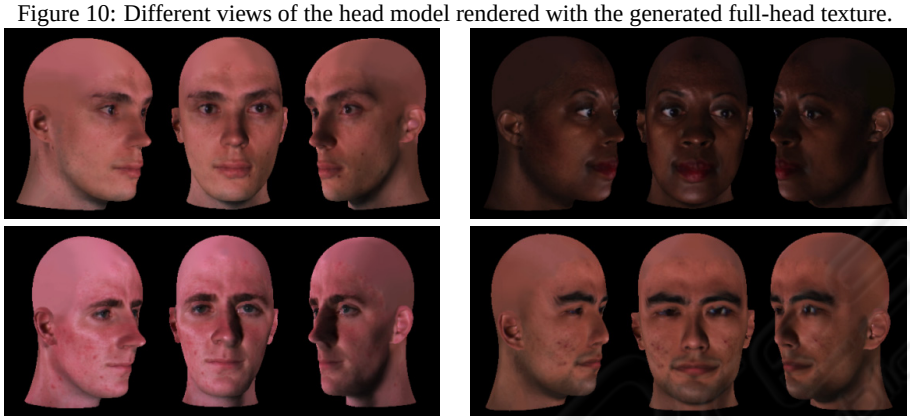


Figure 11: Examples of texture-mapped heads of various people.

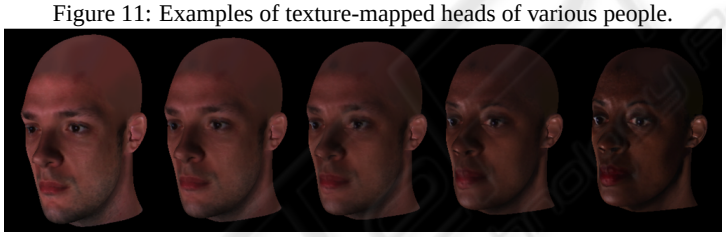


Figure 12: Dynamic morphing between two textured heads.

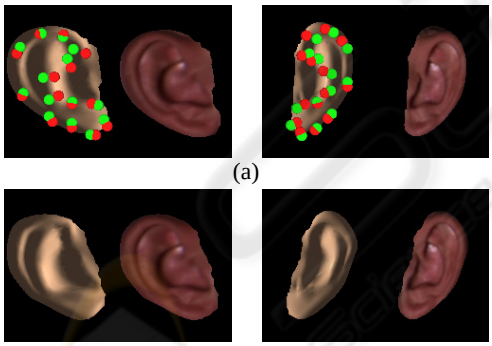


Figure 9: (a) Global alignment of the reference ear meshes with smooth shading and texture mapping. Red and green dots represent the transformed source feature points and detected target feature points, respectively. (b) After the local adaptation.

synthesis with a generic head, it has proper texture on backside and ear part, which makes full rotation of a head. We have used our method to generate full-head textures for various individuals. Female and male in different ethnic groups with different skin color attributes are reconstructed and textured, as shown in Fig. 11. Rendering of head models is performed in real-time using OpenGL hardware (about 120 fps on a 2.4 GHz PC with an ATI FireGL X1 graphics board).

Our method establishes the necessary mapping between different face scans through a single, prototype layout which enables us to morph between any two reconstructed models. Together with the geometric interpolation, we blend the associated textures. The generated cylindrical full-head textures possess the correspondence, enabling 2D texture metamorphosis. We vary the morphing ratios to generate a dynamic morphing between two models, as shown in Fig. 12.

In our method, the only interactive step is the initial identification of corresponding feature points on scanned face surface for model adaptation and in acquired reflectance image for texturing ears. This step takes about 12 minutes. The automated method is then executed to generate a full-head texture. The RBF calculation and the morphing of the generic mesh are performed instantaneously, taking about 2 seconds on a 2.4 GHz Pentium 4. With a dataset of 300k points, the local deformation process runs for about 14 seconds. Computing a parameterization of the head mesh (approx. 5k triangles) takes about 50ms. The texture synthesis process performs 30 iterations in approx. one minute, additional spline blending takes about 40 seconds. Automatic ear fitting process runs fast, taking 0.8 seconds for fitting two

ears. Given the scanned data, the whole process of creating a full-head texture takes within 15 minutes.

7 CONCLUSION

We have presented a new method to effortlessly generate a full-head texture for photorealistic head modeling. We deform a generic head mesh to fit the particular face geometry using a scattered data interpolation approach. We automatically parameterize the 3D generic mesh over the 2D texture domain to establish the mapping correspondence between different face textures. We generate a full-head texture using color interpolation for unbound areas and weighted average splines for visual boundary removal. The individual ear textures are efficiently created from a single input image. Using our technique, we have been able to generate photorealistic head models and natural looking morphing with minimal manual efforts.

We would like to use an anatomy-based model developed in our previous work (Zhang et al., 2004b) as the generic model. As it has been designed to produce real-time physically-based animation, the textured head models with the adapted anatomical structure can be animated immediately with the given muscle parameters. We also plan to address the problem of non-injective mapping under the chin with the cylindrical texture by assigning a separate texture where that part is clearly visible to the region. Instead of current mesh representation, independent generic models of eyes will be used to enable rotations of the eyeball and personal textures extracted from a photograph will be applied to generic models to add individual characteristics. Finally, automated reconstruction of hair texture from images is one of the future challenges.

REFERENCES

Akimoto, T., Suenaga, Y., and Wallace, R. S. (1993). Automatic creation of 3d facial models. In *IEEE Computer Graphics and Application*. 13(5): 16-22.

Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH'99*, pages 187–194.

DeCarlo, D., Metaxas, D., and Stone, M. (1998). An anthropometric face model using variational techniques. In *Proc. SIGGRAPH'98*, pages 67–74.

Essa, I. and Basu, S. (1996). Modeling, tracking and interactive animation of facial expressions and head movements using input from video. In *Proc. Computer Animation'96*, pages 68–79.

Guenter, B., Grimm, C., Wood, D., Malvar, H., and Pighin, F. (1998). Making faces. In *Proc. SIGGRAPH'98*, pages 55–66.

Kahler, K., Haber, J., Yamauchi, H., and Seidel, H. P. (2002). Head shop: Generating animated head models with anatomical structure. In *Proc. ACM SIGGRAPH Symposium on Computer Animation*, pages 55–64.

Kalra, P., Mangili, A., Thalmann, N. M., and Thalmann, D. (1992). Simulation of facial muscle actions based on rational free form deformatios. In *Proc. Proc. EUROGRAPHICS'92*, pages 59–69.

Koch, R., Gross, M., Carls, F., Buren, D., Fankhauser, G., and Parish, Y. (1996). Simulating facial surgery using finite element models. In *Proc. SIGGRAPH'96*, pages 421–428.

Lee, W. S. and Thalmann, N. M. (2000). Fast head modeling for animation. In *Journal Image and Vision Computing*. 18(4): 355-364.

Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. In *Proc. SIGGRAPH'95*, pages 55–62.

Liu, Z., Zhang, Z., Jacobs, C., and Cohen, M. (2001). Rapid modeling of animated faces from video. In *Journal of Visualization and Computer Animation*. 12(4): 227-240.

Marschner, S., Guenter, B., and Raghupathy, S. (2000). Modeling and rendering for realistic facial animation. In *Proc. 11th EG Workshop on Rendering*, pages 231–242.

Nielson, G. M. (1993). Scattered data modeling. In *IEEE Computer Graphics and Application*. 13(1): 60-70.

Parke, F. I. (1972). *Computer Generated Animation of Faces*. Master's thesis, University of Utah.

Parke, F. I. (1982). Parameterized models for facial animation. In *IEEE Computer Graphics and Application*. 2(9): 61-68.

Peleg, S. (1981). Elimination of seams from photomosaics. In *Proc. Conference on Pattern Recognition and Image Processing*, pages 426–429.

Pighin, F., Hecker, J., Lischinski, D., Szeliski, R., and Salesin, D. H. (1998). Synthesizing realistic facial expressions from photographs. In *Proc. SIGGRAPH'98*, pages 75–84.

Pighin, F., Szeliski, R., and Salesin, D. H. (1999). Resynthesizing facial animation through 3d model-based tracking. In *Proc. ICCV'99*, pages 143–150.

Platt, S. and Badler, N. (1981). Animating facial expressions. In *Proc. SIGGRAPH'81*, pages 245–252.

Rocchini, C., Cignoni, P., Montani, C., and Scopigno, R. (1999). Multiple textures stitching and blending on 3d objects. In *Proc. 10th EG Workshop on Rendering*, pages 119–130.

Tarini, M., Yamauchi, H., Haber, J., and Seidel, H.-P. (2002). Texturing faces. In *Proc. Graphics Interface'02*, pages 89–98.

Terzopoulus, D. and Waters, K. (1990). Physically-based facial modeling, analysis and animation. In *Journal of Visualization and Computer Animation*. 1: 73-80.

Thalmann, N. M., Minh, H., deAngelis, M., and Thalmann, D. (1989). Design, transformation and animation of human faces. In *The Visual Computer*. 5: 32-39.

Waters, K. (1987). A muscle model for animating three-dimensional facial expression. In *Proc. SIGGRAPH'87*, pages 17–24.

Waters, K. and Terzopoulos, D. (1991). Modeling and animating faces using scanned data. In *Journal of Visualization and Computer Animation*. 2: 123-128.

Williams, L. (1990). Performance-driven facial animation. In *Proc. SIGGRAPH'90*, pages 235–242.

Zhang, L., Snavely, N., Curless, B., and Seitz, S. M. (2004a). Space-time faces: High resolution capture for modeling and animation. In *Proc. SIGGRAPH'04*, pages 548–558.

Zhang, Y., Prakash, E. C., and Sung, E. (2004b). A new physical model with multi-layer architecture for facial expression animation using dynamic adaptive mesh. In *IEEE Transactions on Visualization and Computer Graphics*. 10(3): 339-352.