

Texturing Faces

Marco Tarini^{1,2}

Hitoshi Yamauchi¹

Jörg Haber¹

Hans-Peter Seidel¹

¹ Max-Planck-Institut für Informatik, Saarbrücken, Germany

² Visual Computing Group, IEI, CNR Pisa, Italy

mtarini@di.unipi.it, {hitoshi, haberj, hpseidel}@mpi-sb.mpg.de

Abstract

We present a number of techniques to facilitate the generation of textures for facial modeling. In particular, we address the generation of facial skin textures from uncalibrated input photographs as well as the creation of individual textures for facial components such as eyes or teeth. Apart from an initial feature point selection for the skin texturing, all our methods work fully automatically without any user interaction. The resulting textures show a high quality and are suitable for both photo-realistic and real-time facial animation.

Key words: texture mapping, texture synthesis, mesh parameterization, facial modeling, real-time rendering

1 Introduction

Over the past decades, facial modeling and animation has achieved a degree of realism close to photo-realism. Although the trained viewer is still able to detect minor flaws in both animation and rendering of recent full-feature movies such as *Final Fantasy*, the overall quality and especially the modeling and texturing are quite impressive. However, several man-years went into the modeling of each individual character from that movie. Trying to model a real person becomes even more tricky: the artistic licence to create geometry and textures that “look good” is replaced by the demand to create models that “look real”.

A common approach towards creating models of real persons for facial animation uses range scanners such as, for instance, Cyberware scanners to acquire both the head geometry and texture. Unfortunately, the texture resolution of such range scanning devices is often low compared to the resolution of digital cameras. In addition, the textures are typically created using a cylindrical projection. Such cylindrical textures have the drawback to introduce visual artifacts, for instance on top of the head, behind the ears, or under the chin. Finally, there is no automatic mechanism provided to generate textures for individual facial components such as eyes and teeth.

In this paper, we present an approach to generate high-resolution textures for both facial skin and facial compo-

Figure 1: Overview of our skin texture generation process: the 3D face mesh is parameterized over a 2D domain and the texture is resampled from several input photographs.

nents from several uncalibrated photographs. The generation of these textures is automated to a large extent, and the resulting textures do not exhibit any patch structures, i.e. they can be used for mip-mapping. Our approach combines several standard techniques from texture mapping and texture synthesis. In addition, we introduce the following contributions:

- a view-dependent parameterization of the 2D texture domain to enhance the visual quality of textures with a fixed resolution;
- a texture resampling method that includes color interpolation for non-textured regions and visual boundary removal using multiresolution splines with a fully automatic mask generation;
- a radial texture synthesis approach with automatic center finding, which robustly produces individual eyeball textures from a single input photograph;
- a technique that uses a single natural teeth photograph to generate a teeth texture, which is applied to an appropriate 3D model to resemble the appearance of the subject’s mouth.

All of these techniques are fully automated to minimize the construction time for creating textures for facial modeling. However, we do not address the topic of facial modeling itself in this paper. We apply the textures generated by the techniques presented in this paper in our facial animation system [12], which has been designed to produce physically based facial animations that perform in real-time on common PC hardware. Thus the focus of our texture generation methods is primarily on the applicability of the textures for OpenGL rendering and a simple but efficient acquisition step, which does not require sophisticated camera setups and calibration steps.

2 Previous and Related Work

Research on either texturing or facial animation has provided a large number of techniques and insights over the years, see the surveys and textbooks in [13, 6] and [25] for an overview. Texturing in the context of facial animation is, however, an often neglected issue. Many sophisticated facial animation approaches, e.g. [32, 18, 19], simply use the textures generated by Cyberware scanners. In [35], Williams presents an approach to generate and register a cylindrical texture map from a peripheral photograph. This approach is meanwhile superseded by the ability of Cyberware scanners to acquire geometry and texture in one step. The method presented in [1] generates an individual head geometry and texture by linear combination of head geometries and textures from a large database that has been acquired using a Cyberware scanner in a costly preprocessing step. Marschner *et al.* 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 [23]. Several other approaches such as [26, 11, 16, 17] are image-based and use a small number of input photographs (or video streams) for the reconstruction of both geometry and texture. Although these approaches could potentially yield a higher texture quality compared to the Cyberware textures, they typically suffer from a less accurate geometry reconstruction, limited animation, and reduced texture quality by using cylindrical texture mapping.

Creating textures from multiple, unregistered photographs has been addressed in the literature by several authors [28, 3, 24]. First, they perform a camera calibration for each input photograph based on corresponding feature points. Next, a texture patch is created for each triangle of the input mesh. The approaches differ in the way these texture patches are created, blended, and combined into a common texture. However, the resulting textures always exhibit some patch structure, which makes it impossible to generate mip-maps from these textures. Creating textures that can be mip-mapped requires to construct a parameterization of the mesh over a two-dimensional domain. To this end, generic techniques based on spring meshes have been presented in [10, 15, 7]. Special parameterizations that minimize distortion during texture mapping for different kinds of surfaces have been investigated by several authors, see for instance [27, 29, 22, 21].

Texture synthesis [9, 33] has become an active area of research in the last few years. Recent publications focus on texture synthesis on surfaces [34, 31, 36] or on texture transfer [8, 14]. All of the methods presented so far use a

Euclidean coordinate system for the synthesis of textures. In contrast, we use a polar coordinate system to synthesize textures that exhibit some kind of radial similarity.

3 Texturing Facial Skin

To generate a skin texture for a head model, we first take about three to five photographs of the person's head from different, uncalibrated camera positions. All photographs are taken with a high-resolution digital camera (3040×2008 pixels). The camera positions should be chosen in such a way that the resulting images roughly cover the whole head. During the acquisition, no special illumination is necessary. However, the quality of the final texture will benefit from a uniform, diffuse illumination. In addition, we acquire the geometry of the head using a structured-light range scanner. As a result, we obtain a triangle mesh that consists of up to a few hundred thousand triangles. After the texture registration step, this triangle mesh is reduced to about 1.5k triangles for real-time rendering using a standard mesh simplification technique. Each photograph is registered with the high-resolution triangle mesh using the camera calibration technique developed by Tsai [30]. Since the intrinsic parameters of our camera/lens have been determined with sub-pixel accuracy in a preprocessing step, we need to identify about 12–15 corresponding feature points on the mesh and in the image to robustly compute the extrinsic camera parameters for each image. This manual selection of feature points is the only step during our texture generation process that requires user interaction.

Next, we automatically construct a parameterization of the 3D input mesh over the unit square $[0, 1]^2$. This step is described in detail in the following Section 3.1. Finally, every triangle of the 2D texture mesh is resampled from the input photographs. A multiresolution spline method is employed to remove visual boundaries that might arise from uncontrolled illumination conditions during the photo session. Details about this resampling and blending step are given in Section 3.2. Figure 1 shows an overview of our texture generation process.

3.1 Mesh Parameterization

We want to parameterize the 3D input mesh over the 2D domain $[0, 1]^2$ in order to obtain a single texture map for the whole mesh. To obtain a mip-mappable texture, the texture should not contain individual patches (*texture atlases*) but rather consist of a single patch. Clearly, this goal cannot be achieved for arbitrary meshes. In our case, the face mesh is topologically equivalent to a part of a plane, since it has a boundary around the neck and does not contain any handles. Thus we can “flatten” the face mesh to a part of a plane that is bounded by its boundary curve around the neck. We represent the original face mesh

by a spring mesh and use the L^2 stretch norm presented in [29] to minimize texture stretch. In our simulations, this L^2 norm performs better than the L^∞ norm that is recommended by the authors of [29].

By applying the texture stretch norm, texture stretch is minimized over the whole mesh. In the following step, we introduce some controlled texture stretch again. Since the size of textures that can be handled by graphics hardware is typically limited, we would like to use as much texture space as possible for the “important” regions of a head model while minimizing the texture space allocated to “unimportant” regions. Obviously, the face is more important for the viewer than the ears or even the back of the head. To accomplish some biased texture stretch, we have introduced an additional weighting function ω into the L^2 stretch norm presented in [29]:

$$L^2(M) := \sqrt{\frac{\sum_{T_i \in M} (L^2(T_i))^2 \omega(T_i) A'(T_i)}{\sum_{T_i \in M} \omega(T_i) A'(T_i)}}$$

with

$$\omega(T_i) := \frac{1}{\langle N(T_i), V \rangle + k},$$

where $M = \{T_i\}$ denotes the triangle mesh, $A'(T_i)$ is the surface area of triangle T_i in 3D, $N(T_i)$ is the triangle normal of T_i , V is the direction into which the head model looks, and $k > 1$ is a weighting parameter. The weighting function ω thus favors the triangles on the face by diminishing their error while penalizing the triangles on the back of the head by amplifying their error. As a consequence, triangles on the face become larger in the texture mesh while backfacing triangles become smaller. Useful values for k are from within [1.01, 2]. Figure 2 shows a view-independent texture mesh parameterization obtained with the original L^2 stretch norm as well as a view-dependent parameterization with our modified stretch norm for $k = 1.2$.

The difference between our *view-dependent texture mesh parameterization* and the *view-dependent texture mapping* proposed in [5, 26] is the following: the latter performs an adaptive blending of several photographs for each novel view, whereas we create a static texture that has its texture space adaptively allocated to regions of different visual importance.

3.2 Texture Resampling

After having created the 2D texture mesh from the 3D face mesh, we resample the texture mesh from the input photographs that have been registered with the face mesh. First, we perform a vertex-to-image binding for all

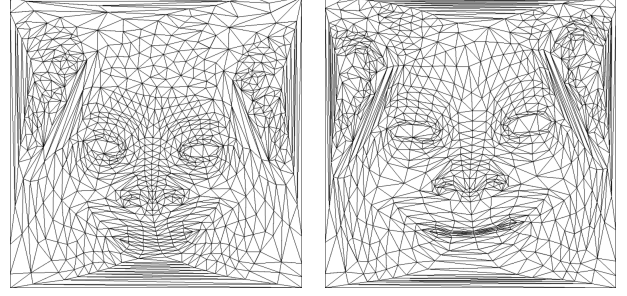


Figure 2: Comparison between a view-independent texture mesh parameterization according to [29] (left) and our view-dependent parameterization (right).

vertices of the 3D face mesh. This step is carried out as suggested in [28]: Each mesh vertex v is assigned a set of *valid photographs*, which is defined as that subset of the input photographs such that v is visible in each photograph and v is a non-silhouette vertex. A vertex v is visible in a photograph, if the projection of v on the image plane is contained in the photograph **and** the normal vector of v is directed towards the viewpoint **and** there are no other intersections of the face mesh with the line that connects v and the viewpoint. A vertex v is called a silhouette vertex, if at least one of the triangles in the fan around v is oriented opposite to the viewpoint. For further details see [28]. In contrast to the approach in [28], we do not require that all vertices of the face mesh are actually bound to at least one photograph, i.e. the set of valid photographs for a vertex may be empty.

Let $\Delta = \{v_1, v_2, v_3\}$ denote a triangle of the face mesh and $\tilde{\Delta} = \{\tilde{v}_1, \tilde{v}_2, \tilde{v}_3\}$ be the corresponding triangle in the texture mesh. For each triangle Δ , exactly one of the following situations might occur (see also Figure 3):

1. There exists at least one common photograph in the sets of valid photographs of the three vertices v_1, v_2, v_3 of Δ (green triangles).
2. All of the vertices of Δ are bound to at least one photograph, but no common photograph can be found for all three vertices (blue triangles).
3. At least one vertex of Δ is not bound to any photograph (red triangles).

In the first case, we rasterize $\tilde{\Delta}$ in texture space. For each texel T , we determine its barycentric coordinates ρ, σ, τ w.r.t. $\tilde{\Delta}$ and compute the corresponding normal N by interpolating the vertex normals of Δ : $N = \rho N(v_1) + \sigma N(v_2) + \tau N(v_3)$. For each common photograph i in the sets of valid photographs of all vertices of Δ , we compute the dot product between N and the viewing direction V_i

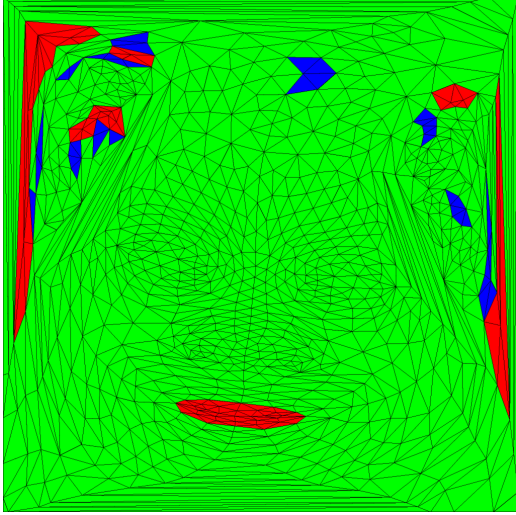


Figure 3: Color-coded triangles of the texture mesh: each green triangle has at least one common photograph to which all of its vertices are bound; the vertices of blue triangles don't have a common photograph, but they are all bound; red triangles have at least one unbound vertex.

for the pixel P_i that corresponds to T . Finally, we color T with the color obtained by the weighted sum of pixel colors $\sum_i \langle N, V_i \rangle \cdot \text{Color}(P_i) / \sum_i \langle N, V_i \rangle$.

In the second case, we color each vertex \tilde{v}_j of $\tilde{\Delta}$ individually by summing up the weighted pixel colors of the corresponding pixels in all valid photographs i of \tilde{v}_j similarly as in the first case: $\text{Color}(\tilde{v}_j) := \sum_i \langle N(\tilde{v}_j), V_i \rangle \cdot \text{Color}(P_i) / \sum_i \langle N(\tilde{v}_j), V_i \rangle$. The texels of the rasterization of $\tilde{\Delta}$ are then colored by barycentric interpolation of the colors of the vertices $\tilde{v}_1, \tilde{v}_2, \tilde{v}_3$. Alternatively, we tried to use as much information as possible from the input photographs if, for instance, the vertices v_1, v_2 of Δ share a photograph and the vertices v_2, v_3 share another photograph. However, we found that this second case does not occur very often (cf. Figure 3) and that the difference between plain color interpolation and a more sophisticated approach is almost invisible.

Since we do not require that each vertex of the face mesh is bound to at least one photograph, there might exist some vertices that cannot be colored by any of the previously described schemes. We address this problem in a two-stage process: First, we iteratively assign an interpolated color to each unbound vertex. Next, we perform the color interpolation scheme from the second case for the remaining triangles of $\tilde{\Delta}$ that have not yet been colored. The first step iteratively loops over all unbound and uncolored vertices of the face mesh. For each unbound vertex v , we check if at least $p = 80\%$ of the vertices in the

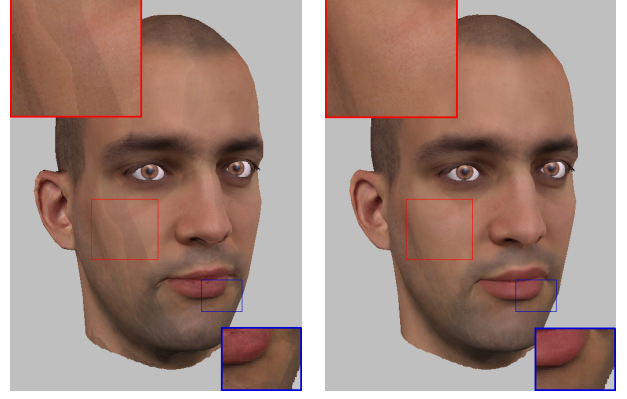


Figure 4: Boundaries in the skin texture (left) are removed using multiresolution spline techniques (right).

one-ring around v are colored (either by being bound to a photograph or by having an interpolated color). If this is true, we assign to v the average color of all the colored vertices around v , otherwise we continue with the next unbound vertex. We repeat this procedure until there are no further vertex updates. Next, we start the same procedure again, but this time we only require $p = 60\%$ of the vertices in the one-ring around v to be colored. As soon as there are no more updates, we repeat this step twice again with $p = 40\%$ and $p = 20\%$. Finally, we update each unbound vertex that has at least one colored neighbor. Upon termination of this last step, all vertices of the face mesh are either bound or colored and the remaining triangles of $\tilde{\Delta}$ can be colored.

If the input photographs have been taken under uncontrolled illumination, the skin color might differ noticeably between the images. In this case, boundaries might appear in the resampled texture. We then apply a multiresolution spline method as proposed in [2, 17] to remove visual boundaries. Figure 4 shows a comparison between a textured head model with and without multiresolution spline method applied. To smoothly combine texture regions that have been resampled from different input photographs, we automatically compute a mask for each region by removing the outmost ring of triangles around the region, see Figure 5. Such a shrinking is necessary to ensure that there is still some valid color information on the outside of the mask boundary, because these adjacent pixels might contribute to the color of the boundary pixels during the construction of Gaussian and Laplacian pyramids. In addition to the masks for each input photograph, we create one more mask that is defined as the complement of the sum of all the other masks. This mask is used together with the resampled texture to provide some color information in those regions that are not covered by



Figure 5: Multiresolution spline masks: three different regions in the texture mesh resampled from different input photographs (top) and their corresponding masks shown in red (bottom).

any input photograph (e.g. the inner part of the lips). As described above, these regions have been filled by color interpolation in the resampled texture. By blending all of the masked input photographs and the masked resampled texture with a multiresolution spline, we obtain a final texture with no visual boundaries and crispy detail.

4 Texturing Facial Components

Both human eyes and teeth are important for realistic facial animation while, at the same time, it is difficult to acquire data from a human being to precisely model these facial components. Thus we use generic models of these components as shown in Figure 8. The design of our generic models has been chosen such that they look convincingly realistic when inserted into a face mesh while still being rendered efficiently using OpenGL hardware.

On the other hand, both eyes and teeth (especially the more visible middle ones) are crucial features to visually differentiate one individual from another. Hence, it would be very desirable to use individual models for each person. Luckily, texturing can do the trick alone: indeed it is sufficient to apply a personal texture to a generic model to get the desired effect. Moreover, it is possible to automatically and quickly generate these textures each from a single input photograph of the subject’s eye and teeth, respectively. Details about this process will be given in the next two subsections.

4.1 Texturing Eyes

In order to realistically animate our head model, we must be able to perform rotations of the eyeball and dilation of the pupil. While the latter can be achieved by transforming the texture coordinates, we need an eye texture

that covers the whole frontal hemisphere of the eyeball for the rotations.

Our goal to generate such an eyeball texture from a single input photograph is complicated by several factors such as the presence of occluding eyelids, shadows of eyelashes, highlights, etc. Still, all these factors are local and can be detected and removed. A new texture can then be synthesized from an input image consisting of the surviving pixels. In our current approach, we focus our effort on the iris, since it is obviously the most characteristic part of the eye.

Both the detection and the synthesis phase rely on the simplicity of the eye structure, i.e. an almost perfect point symmetry about the center, assuming our photograph represents an eye looking at the camera. To take advantage of this symmetry, we must first know precisely where the center of the eye is located. Since this would encumber the user, the center finding is done automatically by refining a rough estimation to sub-pixel precision using the following heuristic: we progressively enlarge an initially point-sized circle while checking the pixels on the circle at every iteration. If these pixels are too bright, they are assumed to be outside the iris and we thus move the center of the circle away from them. When most of the circle is composed by too bright pixels, we assume its center is the eye center and its radius is the iris radius. This approach runs robustly as long as the initial estimation is inside the pupil or the iris.

At this point, removal of occluded, shadowed, and highlighted pixels is done by:

- removing pixels with a color too similar to the skin;
- removing pixels with a color too dissimilar to the pixels at the same radial distance from the center.

For the second case, we compute the average color and standard deviation of the pixels at the same radial distance and remove those pixels that are at least α times the standard deviation away from the average. The parameter α should be chosen within [2, 3]. We typically use a rather small value of $\alpha = 2.3$, as it empirically proved to remove the problematic (occluded, shadowed, highlighted, etc.) pixels in most cases. In addition, we remove pixels too close to the skin to better take into account small shadows cast by eyelids. Actually, the decision of which pixel to remove does not need excessively fine tuning: due to the regularity of the eye, we can be pretty conservative and remove many pixels, since the reconstruction phase requires only a small zone of pixels in order to synthesize more. Figure 6 shows the remaining set of pixels for two different input photographs.

For the reconstruction phase it is natural to resort to some texture synthesis from samples approach like

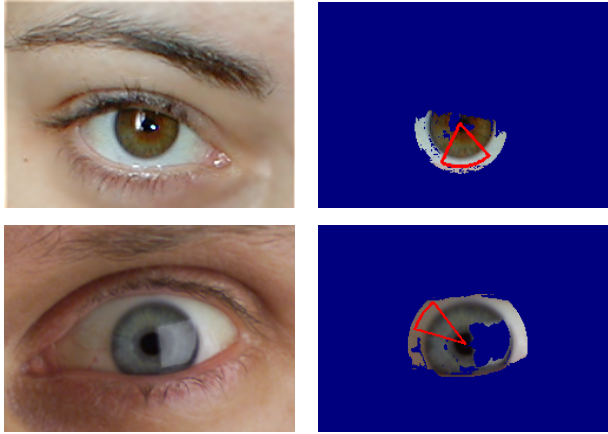


Figure 6: Two input photographs (left) and the resulting reference patches outlined by red sectors (right). Occluded, shadowed, highlighted, and skin-colored pixels (shown in blue) have been removed automatically.

e.g. [33]. In our case, we need to work in polar coordinates, because the eyeball texture behaves like a texture as defined in [33] only along the *angle* axis. This means that subregions of the eyeball texture are perceived to be similar if their *radius* coordinates are the same, cf. Figure 10. To take this into account, when choosing a candidate pixel p in the input image for filling a pixel p' in the output texture, we constrain the radius coordinate of p to be within a small threshold of the radius coordinate of p' .

A robust approach for texture synthesis is to use only a small patch of the original input image as the reference image and synthesize the texture from scratch. Although larger reference images theoretically result in more faithful textures, we obtained very good results with small reference patches covering a sector of about 30 degrees around the pupil. Small reference patches have the advantage of being more uniform and thus bypassing problems related to uneven lighting in the original photograph. In our approach, we simply use the largest sector of valid pixels of at most 60 degrees as the reference patch. In the rare cases where the largest sector is too small, e.g. spanning less than 20 degrees, the entire set of valid pixels with a valid neighborhood is used as the reference image.

Since the detail frequencies of human irises are roughly the same, it is sufficient to use a texture synthesis scheme with a fixed neighborhood size rather than a multiresolution approach. In our case, the size of the neighborhood mask depends only on the resolution of the input image. For instance, for an image of an iris with a diameter of approximately 80 pixels, we use a 3×6 pixel mask (radius \times angle). For other iris diameters, the pixel mask is set proportionally. Depending on the value of the ra-

dius coordinate, a neighborhood with a fixed size in polar coordinates covers areas of different sizes in the input image. Our simulations showed, however, that no correction is needed, since the human iris usually exhibits higher frequency detail towards the center. Thus an iris resampled in polar coordinates shows quite uniform frequency distribution. Figure 9 shows several input photographs together with the resulting eye textures for various individuals.

To speed-up the reconstruction step, we use a one-dimensional texture synthesis approach along the angle axis alone, modeling the texture as a Markov chain rather than a Markov random field. Each symbol of the chain is an entire row of texels at a given angle coordinate. We output each new row accordingly to the previous rows. This approach gives similar results (even if it requires slightly larger reference textures) and is much faster, not even requiring any vector quantization for finding the best neighborhood row. If, however, the size of the reference patch is very small, we apply a two-dimensional texture synthesis approach as described earlier in this section.

4.2 Texturing Teeth

Geometry and color of teeth are difficult to capture and, at the same time, crucial to reflect personal appearance. We address this problem by distinguishing between

- the six middle teeth (incisors and canines) and
- the rest of the teeth (4–5 on each side).

The middle teeth are much more visible than the other teeth. This means that they account for most of the visual appearance of an individual person, but also that it is much easier to reconstruct them from a photograph. In addition, the middle teeth have an almost two-dimensional structure: they are shaped to have the function of a blade. Their small width allows us to model them using a billboard (impostor). Being a 2D data structure, the billboard can be easily extracted directly from a normal photograph of the subject exposing the teeth in a similar way as shown in Figure 11 (left). Using local transparency, it is straightforward to make the texture embed the teeth shape and size including gaps between teeth. This approach allows us to use the same (billboarded) 3D model for every face model and just change the texture from person to person.

The rest of the teeth, while being more voluminous and less accessible and visible, do not allow this useful shortcut. But, for the same reason, it is also less important to model them faithfully and individually for each single person. Thus it seems reasonable to use a standard 3D model and a standard texture (up to recoloring, see below) for this part of the teeth arch.

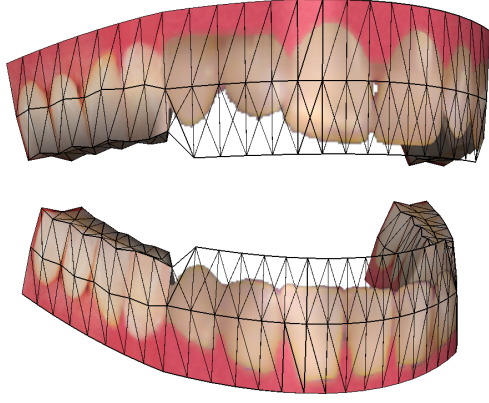


Figure 7: Teeth arch model using the texture shown in Figure 11. The wireframe shows the geometry of the teeth model, which consists of 384 triangles.

Following these considerations, we have built a generic 3D model for the teeth, which is non-uniformly scaled according to the individual skull and jaw geometry to fit into every head model. For each individual head model, we only need to vary the texture (including the billboard), which is created fully automatically. The generic teeth model is constructed such that the transition between the billboard (in the middle) and the 3D structure (left and right) is smooth, see Figure 7. The billboard, which is bent for better realism, could cause undesired artifacts when seen from above. To avoid this, only the upper part of the lower teeth and the lower part of the upper ones is actually modeled as a billboard. The remaining parts of the upper and lower middle teeth smoothly gain some width as they go up and down, respectively.

To automatically create a texture for the teeth, we start from a normal photograph of the subject showing his/her teeth. Several stages of the whole process of generating a teeth texture are shown in Figure 11. We color-code dark parts that represent voids with a blue color, which is replaced by a transparent alpha value during rendering. Similarly, we identify and remove gums, lips, and skin, recoloring it with some standard gums color. To make this color-coding more robust, we identify the different regions using threshold values, which are obtained by finding the biggest jumps in the histograms of the color distances to the target color (red for gums and black for voids). In addition, we expand teeth into those parts of the gums that have been covered by the lips in the input photograph. We use some simple heuristics to include the missing part of the tooth roots, cf. Figure 11.

During rendering, our teeth model is shaded using a Phong shading model, which means that we have to desaturate our teeth texture. In order to do so for uncontrolled

illumination, we equalize the color of the teeth, supposing they have approximately the same albedo. First, we define a target color by computing the average color of all teeth pixels and setting its brightness (but not the hue) to a predefined value. Next, we subdivide the texture in six vertical stripes and compute the average color of each stripe. We then add to the pixels in each column the difference between the target color and the stripe average, taking care of enforcing continuity in this correction by using a piecewise linear function. Similarly, we use the target color to correct the color of the “generic” part of the texture, which is applied to the side teeth. Finally, we composite the middle teeth texture into our generic texture using a curved boundary that follows the silhouettes of the canines.

5 Results

We have created facial textures for several individuals who have also been range-scanned to acquire their head geometry. Rendering of our head model is performed in real-time using OpenGL hardware (about 100 fps on a 1.7 GHz PC with a GeForce3 graphics board). A physics-based simulation is used to control the facial animation. Several images of our head models are distributed over this paper, see for instance Figures 1, 4, 8, and especially Figure 12. For each skin texture, the only interactive step is the initial identification of corresponding feature points. This step takes about five minutes per input photograph, which sums up to about 15–25 minutes spent interactively for three to five photographs. Computing an optimized parameterization of the face mesh (approx. 1600 triangles) takes about 80 minutes on a fast PC (1.7 GHz Pentium 4). Resampling a 2048×2048 texture from five input photographs takes about one minute, additional multiresolution spline blending (if necessary) takes about ten minutes. Currently, our algorithms are optimized with respect to robustness but not to speed.

Generating the teeth and eye textures takes only a few seconds even for large textures using the 1D Markov chain method for the texture synthesis. If a full Markov field is used, construction time may go up to several minutes, depending on the size of the texture being created.

6 Conclusion and Future Work

We have introduced a number of techniques that help to minimize the time and effort that goes into the creation of textures for facial modeling. With the exception of the initial feature point selection for the skin texturing, our methods are fully automated and do not require any user interaction.

For the generation of skin textures from uncalibrated input photographs, we propose a view-dependent param-

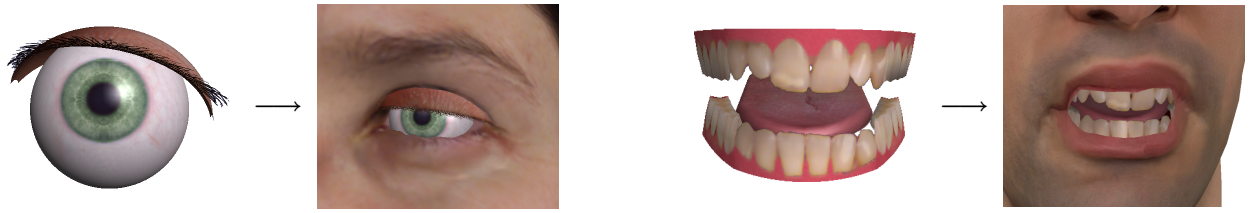


Figure 8: Generic models of eyes, teeth, and tongue are fitted into individual face meshes.

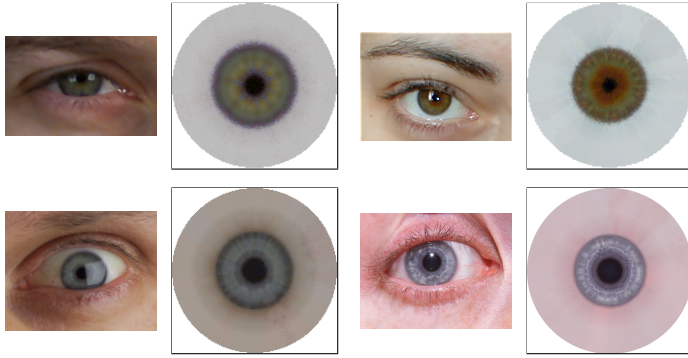


Figure 9: Input photographs and resulting eye textures: the input images have been taken under various illumination conditions with different resolutions. The size of the resulting textures changes from 128×128 (top left) to 1024×1024 (bottom right).

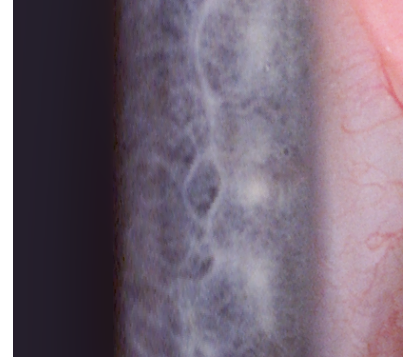


Figure 10: A detail of the texture from Figure 9 (bottom right) shown in polar coordinates. The abscissa represents the radius axis and the ordinate represents the angle axis.

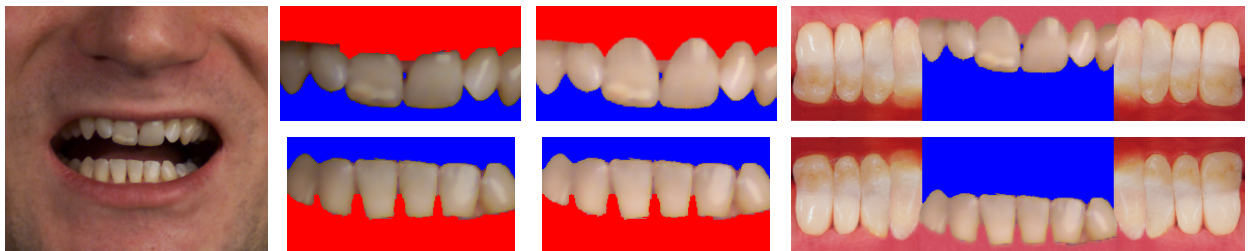


Figure 11: Teeth texture generation. Left to right: starting from an input photograph, we extract the upper and lower middle teeth, fill in missing parts and adjust the color, and composite the new image with a generic teeth texture. The blue pixels in the final texture (right) will be rendered transparently.

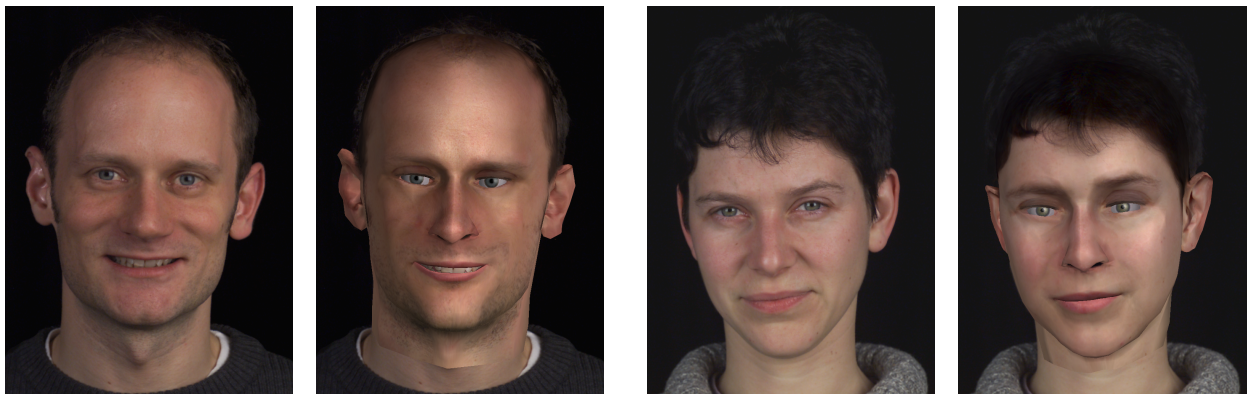


Figure 12: Side-by-side comparison of photographs (left) and head models (right) for plain OpenGL rendering.

eterization of the texture domain and a texture resampling method including color interpolation for non-textured regions and multiresolution splining for the removal of visual boundaries. Using our methods, both eye and teeth textures can be created fully automatically from single input photographs, adding greatly to a realistic appearance of individual subjects during facial animation.

One of the main goals of ongoing research is to get rid of the interactive camera calibration step for skin texturing. Given that the resulting texture should contain fine detail, this is a tough problem, indeed. Automatic approaches such as [20] fail simply due to the fact that the silhouette of a human head looks more or less identical when viewed from within a cone of viewing directions from the front or the back. Furthermore, it would be desirable to account for lighting artifacts in the input photographs. Although a uniform, diffuse illumination during the photo session helps a lot, there are still contributions from diffuse and specular lighting in the photographs. Approaches to overcome these problems have been suggested [4, 23], but they require sophisticated camera setups and calibration steps. Finally, it would be very helpful to speed-up the computation time of the current bottleneck, namely the mesh parameterization, using a hierarchical coarse-to-fine approach.

Acknowledgments

The authors would like to thank their models Letizia, Claudia, and Kolja for all the smiles during the photo sessions. Many thanks also to our colleagues, who gave helpful comments during the development of our techniques, and to the anonymous reviewers for their suggestions.

References

- [1] V. Blanz and T. Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Computer Graphics (SIGGRAPH '99 Conf. Proc.)*, pages 187–194, August 1999.
- [2] P. J. Burt and E. H. Adelson. A Multiresolution Spline with Application to Image Mosaics. *ACM Transactions on Graphics*, 2(4):217–236, October 1983.
- [3] P. Cignoni, C. Montani, C. Rocchini, R. Scopigno, and M. Tarini. Preserving Attribute Values on Simplified Meshes by Resampling Detail Textures. *The Visual Computer*, 15(10):519–539, 1999.
- [4] P. E. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the Reflectance Field of a Human Face. In *Computer Graphics (SIGGRAPH '00 Conf. Proc.)*, pages 145–156, July 2000.
- [5] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach. In *Computer Graphics (SIGGRAPH '96 Conf. Proc.)*, pages 11–20, August 1996.
- [6] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & Modeling: A Procedural Approach*. Academic Press, London, 2 edition, 1998.
- [7] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In *Computer Graphics (SIGGRAPH '95 Conf. Proc.)*, pages 173–182, August 1995.
- [8] A. A. Efros and W. T. Freeman. Image Quilting for Texture Synthesis and Transfer. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 341–346, August 2001.
- [9] A. A. Efros and T. K. Leung. Texture Synthesis by Non-parametric Sampling. In *IEEE Int'l Conf. Computer Vision*, volume 2, pages 1033–1038, September 1999.
- [10] M. S. Floater. Parametrization and Smooth Approximation of Surface Triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.
- [11] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making Faces. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 55–66, July 1998.
- [12] J. Haber, K. Kähler, I. Albrecht, H. Yamauchi, and H.-P. Seidel. Face to Face: From Real Humans to Realistic Facial Animation. In *Proc. Israel-Korea Binational Conf. on Geometrical Modeling and Computer Graphics*, pages 73–82, October 2001.
- [13] P. S. Heckbert. Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.
- [14] A. Hertzmann, Ch. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image Analogies. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 327–340, August 2001.
- [15] K. Hormann and G. Greiner. MIPS: An Efficient Global Parametrization Method. In *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.
- [16] W.-S. Lee, J. Gu, and N. Magnenat-Thalmann. Generating Animatable 3D Virtual Humans from Photographs. In *Computer Graphics Forum (Proc. EG 2000)*, volume 19, pages C1–C10, August 2000.
- [17] W.-S. Lee and N. Magnenat-Thalmann. Fast Head Modeling for Animation. *Image and Vision Computing*, 18(4):355–364, March 2000.
- [18] Y. Lee, D. Terzopoulos, and K. Waters. Constructing Physics-based Facial Models of Individuals. In *Proc. Graphics Interface '93*, pages 1–8, May 1993.
- [19] Y. Lee, D. Terzopoulos, and K. Waters. Realistic Modeling for Facial Animations. In *Computer Graphics (SIGGRAPH '95 Conf. Proc.)*, pages 55–62, August 1995.
- [20] H. P. A. Lensch, W. Heidrich, and H.-P. Seidel. Automated Texture Registration and Stitching for Real World Models. In *Proc. Pacific Graphics 2000*, pages 317–326, October 2000.
- [21] B. Lévy. Constrained Texture Mapping for Polygonal Meshes. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 417–424, August 2001.

- [22] J. Mailliot, H. Yahia, and A. Verroust. Interactive Texture Mapping. In *Computer Graphics (SIGGRAPH '93 Conf. Proc.)*, pages 27–34, August 1993.
- [23] S. R. Marschner, B. Guenter, and S. Raghupathy. Modeling and Rendering for Realistic Facial Animation. In *Rendering Techniques 2000 (Proc. 11th EG Workshop on Rendering)*, pages 231–242, 2000.
- [24] P. J. Neugebauer and K. Klein. Texturing 3D Models of Real World Objects from Multiple Unregistered Photographic Views. In *Computer Graphics Forum (Proc. EG '99)*, volume 18, pages C245–C256, September 1999.
- [25] F. I. Parke and K. Waters, editors. *Computer Facial Animation*. A K Peters, Wellesley, MA, 1996.
- [26] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing Realistic Facial Expressions from Photographs. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 75–84, July 1998.
- [27] D. Piponi and G. D. Borshukov. Seamless Texture Mapping of Subdivision Surfaces by Model Peltin and Texture Blending. In *Computer Graphics (SIGGRAPH '00 Conf. Proc.)*, pages 471–478, July 2000.
- [28] C. Rocchini, P. Cignoni, C. Montani, and R. Scopigno. Multiple Textures Stitching and Blending on 3D Objects. In *Rendering Techniques '99 (Proc. 10th EG Workshop on Rendering)*, pages 119–130, 1999.
- [29] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture Mapping Progressive Meshes. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 409–416, August 2001.
- [30] R. Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 364–374, June 1986.
- [31] G. Turk. Texture Synthesis on Surfaces. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 347–354, August 2001.
- [32] K. Waters and D. Terzopoulos. Modeling and Animating Faces Using Scanned Data. *J. Visualization and Computer Animation*, 2(4):123–128, October–December 1991.
- [33] L.-Y. Wei and M. Levoy. Fast Texture Synthesis Using Tree-Structured Vector Quantization. In *Computer Graphics (SIGGRAPH '00 Conf. Proc.)*, pages 479–488, July 2000.
- [34] L.-Y. Wei and M. Levoy. Texture Synthesis over Arbitrary Manifold Surfaces. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 355–360, August 2001.
- [35] L. Williams. Performance-Driven Facial Animation. In *Computer Graphics (SIGGRAPH '90 Conf. Proc.)*, volume 24, pages 235–242, August 1990.
- [36] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin. Texture and Shape Synthesis on Surfaces. In *Rendering Techniques 2001 (Proc. 12th EG Workshop on Rendering)*, pages 301–312, 2001.