# Real-time, Photo-realistic, Physically Based Rendering of Fine Scale Human Skin Structure

Antonio Haro† Brian Guenter‡ Irfan Essa†

†GVU Center / College of Computing
Georgia Institute of Technology

‡Microsoft Research Graphics Group
Microsoft Corporation

**Abstract.** Skin is noticeably bumpy in character, which is clearly visible in close-up shots in a film or game. Methods that rely on simple texture-mapping of faces lack such high frequency shape detail, which makes them look non-realistic. More specifically, this detail is usually ignored in real-time applications, or is drawn in manually by an artist. In this paper, we present techniques for capturing and rendering the fine scale structure of human skin. First, we present a method for creating normal maps of skin with a high degree of accuracy from physical data. We also present techniques inspired by texture synthesis to "grow" skin normal maps to cover the face. Finally, we demonstrate how such skin models can be rendered in real-time on consumer-end graphics hardware.

## 1 Introduction

Photo-realistic skin is very important in a film or a video game and is hard to render for various reasons. One reason is that humans are experts when it comes to skin realism. Another is that skin has a complicated BRDF, one that depends on many factors such as pigmentation, oiliness and dryness. Skin also possesses very fine and complicated detail. The appearance of fine scale skin structure varies smoothly across the face yet each region of skin on the human body has a very distinct appearance unique to its location. For example, forehead fine scale skin structure is distinctly different from nose fine scale structure in Figure 3. Skin that lacks fine scale structure looks unrealistic since fine scale structure is such an integral part of its appearance. In this paper, we capture fine scale structure samples, build models of fine scale structure production, and then render this detail using a measured skin BRDF. Our results show that the addition of fine scale structure adds significantly to photo-realism of facial models.

We address several problems in this paper: (1) capture of fine scale skin structure from actual skin samples, (2) approximation of the stochastic processes that generate fine scale skin structure patterns, and (3) rendering of photo-realistic skin in real-time. To capture the fine scale skin structure, we employ a material[1] used in cosmetological science to create skin imprints and measure pore size. In that field, the imprints are laser-scanned to create a range map. The resulting range map is very noisy because the samples are very small (about the size of a nickel), and hence, unsuitable for rendering, but not for the type of analysis they do. We use shape from shading [8] to get a much more accurate range map that is significantly less noisy than laser-scanning can provide. Once we produce range maps for imprints from different areas of the face,

[1]Silflo, manufactured by Flexico

we approximate the stochastic processes that “generated” each sample. We encode the range maps as images so that this problem reduces to texture synthesis. Since the range map for each skin sample comes from a different area of the face, and hence, a different stochastic process, separate instances of texture synthesis are started on several points on the face. Fine scale structure is “grown” in 3D until full coverage is attained. We use the result as a normal map and use it to do per-pixel bump mapping. We also approximate the BRDF of the skin, so the result approaches photo-realism.

Section 3 discusses our normal map capturing process. In Section 4, we describe how we “grow” captured normal maps. Section 5.1 discusses how we use our measured reflectance model to render fine scale structure and Section 5.2 describes our lighting model.

## 2 Previous Work

Traditionally, when fine scale structure is rendered, it is either drawn/produced by artists by hand, or is rendered by using layers of specialized shaders. While realistic fine scale structure can be attained by these methods, the creation of additional fine scale structure is time intensive since an artist has to manually create new fine scale structure for each desired face. If specialized shaders are used, it might not be possible to render in real-time. In both cases, some underlying properties of the stochastic process are lost since the fine scale structure might only be visually realistic at certain distances/orientations. Since we compute models of the stochastic processes that generate the skin directly from actual skin data, we can render skin that looks realistic in different lighting conditions and at different scales, without any recomputation/resynthesis.

There is much work that has been done in the area of realistic facial rendering, however, very little work has been published on capturing or synthesizing fine scale skin structure. The closest work to ours is that of Wu *et al.* [23]. In that work, fine scale skin structure is dependent on its underlying geometry. A Delaunay triangulation is done on each desired skin region to generate triangles in texture space. The edges of the triangles are then raised/lowered depending on user parameters to create a height-field which is then used as a bump map. The resulting bump map does not look very realistic because some Delaunay triangulations will not yield realistic fine scale structure. Also, a user would have to spend a lot of time selecting proper basis functions, rejecting bad triangulations, and ensuring proper fine scale structure blending from one region of the face to another in order to get fine scale structure that approaches realism. In contrast, our method implicitly takes the stochastic properties and variance across the face into account, so we achieve more realistic results with minimal user input. Nahas *et al.* [15] captured skin detail with a laser range scanner, but at a low resolution (256x256 samples). However, this resolution is too low to capture the rich detail we are able to capture using skin molds and photometric stereo.

Recent work done to capture skin reflectance properties is also worth noting. Hanrahan *et al.* [6] simulated sub-surface scattering using Monte Carlo simulations to render skin. Marschner *et al.* have done work on capturing BRDFs from images that come from a calibrated camera with subjects wearing a pattern ([13], [14]). Debevec *et al.* [3] use a more complex system to measure skin reflectance as well as surface normals and are also able to incorporate ambient lighting from different environments. All of these approaches yield high-quality renderings, however, none incorporate fine scale skin structure. Recent facial animation work has also resulted in realistic results ([5], [16]). However, since the face texture comes from images that are texture blended, high-frequency detail such as fine scale structure is lost. We use measured skin reflectance

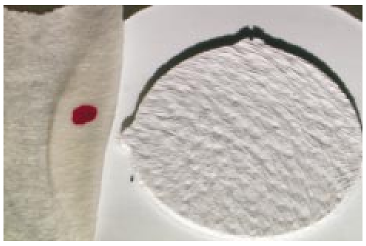
**Fig. 1.** Silicone mold of skin (about the size of a nickel)

models in conjunction with fine scale structure for added visual realism.

## 3 Normal Map Capture

We use shape from shading to capture convincing fine scale skin structure. First we make several samples of real skin texture (Figure 1) from various regions of the face using a silicone mold material. Then we apply shape from shading [8] to the silicone mold to recover the normals of the surface. Our technique is similar to the work done by others ([20], [26], [18], [19]) except that we do not require any camera calibration or structured light. In addition, we can deal with non-lambertian surfaces while using all of the pixels from each captured photograph of each sample. Throwing away pixels due to specular reflections would yield suboptimal maps due to our very simple experimental setup.

The shape from shading algorithm requires the BRDF of the surface being measured to be lambertian. We approximate this by placing polarizing filters in front of the light source and the camera and rotating them with respect to each other to eliminate all specular reflections. The remaining reflection is approximately lambertian. The illuminator we used to capture all our skin normal data, consisting of a halogen light mounted on a lazy susan, was built for $40 using materials available at any hardware store.

We take 8 photographs of the silicone mold illuminated by a point light source. The mold remains stationary, but the light position is changed in every frame by rotating the light on the lazy susan. The resulting set of images yields a set of simultaneous linear equations at each $(x, y)$ pixel location that can be solved for the surface normal at that pixel:

$$\rho_{x,y} n_{x,y} \cdot L_i = I_{x,y} \tag{1}$$

$\rho_{x,y}$ is the diffuse albedo of the surface at pixel $(x, y)$, $n_{x,y}$ is the normal at the pixel, and $I_{x,y}$ is the measured intensity at the pixel. $L_i$ is the light vector for image $i$. $\rho_{x,y}$ is nearly constant across the silicone mold so it needs to be measured only once per image rather than at each pixel. The light intensity and position with respect to the surface is also approximately uniform over the sample so it also needs to be measured only once. This set of equations is solved at each pixel using least squares to give a normal map.

The silicone molds are not always flat, especially those taken in the nose region. This introduces a low frequency change in the surface normals that is inappropriate for bump mapping, where only the high frequency details are of interest. To eliminate the low frequency component we compute the average normal over a 50 by 50 pixel block centered at each pixel, compute the rotation that maps this normal back into

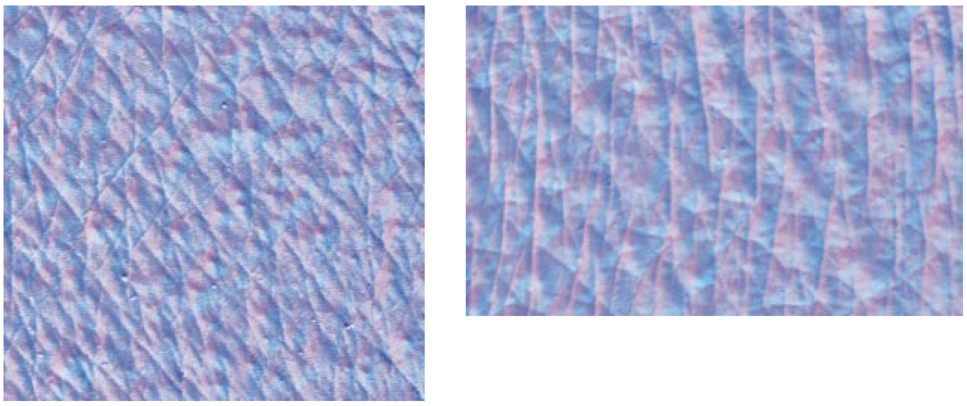

**Fig. 2.** *Left*: Cheek normal map, *Right*: Edge of forehead normal map

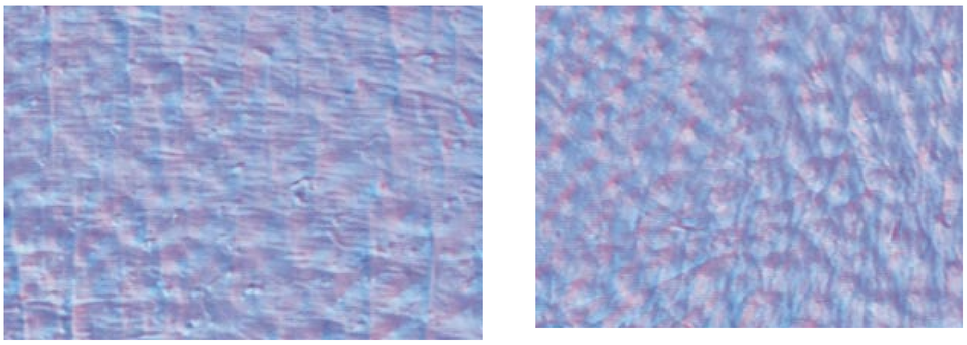

**Fig. 3.** *Left*: Middle of forehead normal map, *Right*: Nose normal map

the perpendicular to the normal map plane, and then apply this rotation to the normal computed from shape from shading.

Figures 2 and 3 show some example skin textures that we have captured from different regions of the face. The normal maps are encoded versions of the normals where the $X, Y, Z$ axes map to the $R, G, B$ channels, respectively, and the range $< -1.0, 1.0 >$ maps to $< 0, 255 >$.

## 4 Skin Growing

The normal maps we capture in Section 3 are of high detail, but are quite small. The molds are about 21mm in diameter. One simple way to get complete facial coverage would be to take a large amount of these molds and then to interpolate between the gaps. Samples taken from curved areas would be distorted and the process would be very inconvenient.

Another approach involves taking a small number of samples and then learning how to produce more. Fine scale skin structure varies significantly across the face (Figures 2 and 3), which is one reason it is hard to model. Since fine scale skin structure can be thought of as a stochastic process, some ideas from texture synthesis can be applied. This does not address the issue of gaps between different skin types; since skin varies

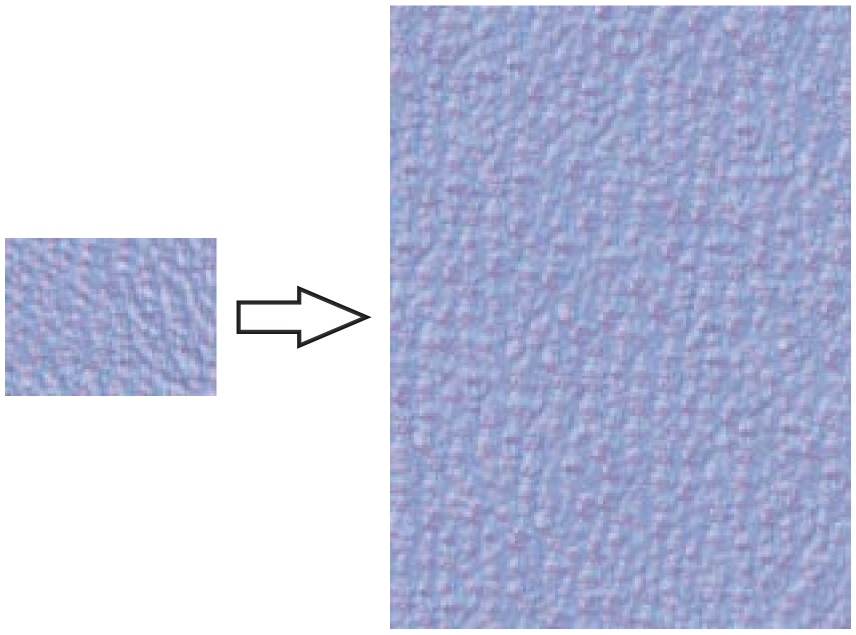

**Fig. 4.** Initial normal map captured from mold and synthesized map

very significantly across the face, it is not adequate to interpolate or blur the edges. Doing so would result in high-frequency discontinuities, which would be very obvious when rendered. We present our solution to this problem in Section 4.2.

### 4.1 Normal map synthesis

Since skin can be thought of as a stochastic process, we can apply ideas from texture synthesis to create larger regions than what we have. To create larger patches of skin, we take our normal maps that we acquired through our capture process, and encode them as images as described in Section 3. These encoded normal maps can then be thought of as small pieces of texture that we would like to synthesize.

We use the technique by Wei and Levoy[22] to synthesize larger patches. The main idea in that work was to treat texture as a Markov random field. That is, as a stochastic process that is both local and stationary, meaning that a pixel's neighborhood characterizes the process and that this characterization is the same for all pixels in the texture.

First, a histogram equalized (to the input texture) color noise image is generated for the desired dimensions needed for the skin region. Then, for each pixel in the noise image in scanline order, we assign the color of the pixel from the input texture with the most similar causal neighborhood. This is done on each level of a Gaussian pyramid starting with the topmost to ensure that the pixel that is chosen from the input texture is similar at the current and previous frequency bands. Finally, the bottom level of the pyramid is copied into the desired facial region when synthesis is complete.

Results of "growing" skin can be found in Figure 4. Recently, several texture synthesis papers ([1], [25]) have proposed copying from the source texture for some classes of texture. These approaches could yield even higher quality skin patches since larger contiguous regions from the input normal maps will exist in the synthesized skin.

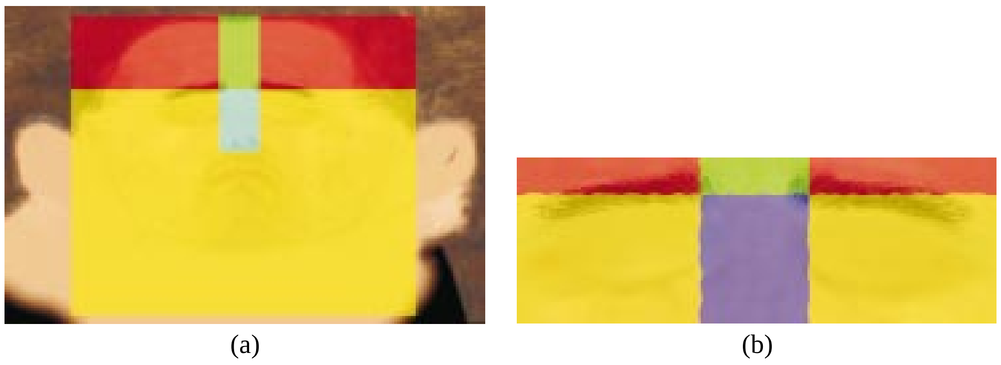

**Fig. 5.** (a) Each color represents a different type of fine scale skin structure to synthesize, (b) Multi-resolution splining along curves hides the boundaries between different patches

### 4.2 Skin stitching

Since fine scale skin structure varies across the face, we start a separate instance of texture synthesis at each different region. The regions are defined by a user by selecting a few points (7) on the albedo map (Section 5.2), specifying the forehead, nose, and cheeks as in Figure 5(a). This step is the only manual step in our pipeline, and can be done quickly.

A separate instance of texture synthesis is started at each region. When the skin in each region has grown too large for its region, the boundaries between it and its neighbors are hidden by using a modified version of multi-resolution splines [2]. Instead of performing the multi-resolution splining along a straight line along the boundary, we compute a random curve through the boundary instead, seen in Figure 5(b). We have found this looks more like the natural transitions between one type of fine scale structure to another on human skin. One major reason for this is that the boundaries are hidden more effectively along the curve; curvaceous boundaries throw the human eye off and hide seams well ([17]). We are able to generate fine scale structure for different faces rapidly with minimal user interaction using this approach.

## 5 Skin Rendering

To render skin realistically, we perform per-pixel bump mapping using the normal maps we have “grown”. Since our mesh topologies are low ($< 15k$ vertices), storing only the normals at vertex locations loses the majority of the detail that we gain through our capture step. Instead, we store our normals in a normal map and access them on a per-pixel basis as we render the mesh. In addition, we approximate the BRDF of skin to achieve greater realism. We will discuss that further in Section 5.2.

### 5.1 Per-pixel Bump mapping

Our bumpmapping technique [4] is similar to the work of others ([21], [7]). The major difference between the techniques is that the per-pixel lighting calculations are done in texture space instead of object space. We perform the per-pixel lighting calculations in texture space since we would have to warp our normal map so that each normal in our map is in object space otherwise. Warping is undesirable because distortions can occur

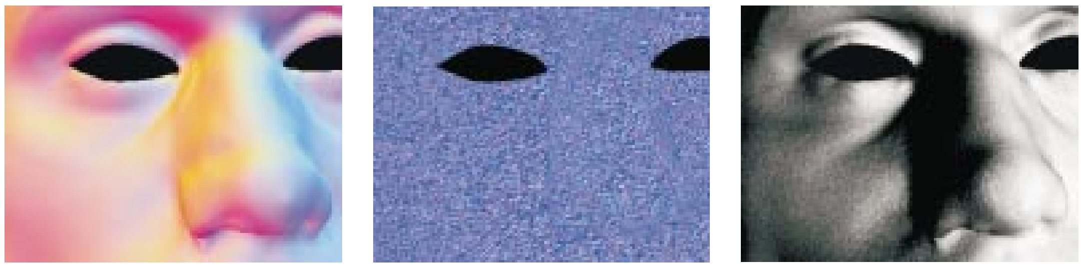

**Fig. 6.** (a) Encoded and transformed light vector (interpolated), (b) Normal map treated as texture (contrast enhanced), (c) $(N \cdot L')$ computed per-pixel

in the warped normal map, especially with lower resolution meshes. Also, if there are hard edges, the interpolated texture to object space transformation could be incorrect in areas between vertices.

Furthermore, if the mesh is animated, the normal map has to be re-warped each frame, which is very expensive. Working in texture space is more appealing since we only have to update the object to texture space transformation once per frame at each vertex and we have far fewer vertices than normal map pixels.

We compute transformations from object space into each vertex's texture space. At each vertex $v$ with position $(v(x), v(y), v(z))$ and texture coordinates $(v(u), v(v))$ we form a coordinate system to perform per-pixel lighting in texture space. This coordinate system is comprised of $\delta U = (\delta u/\delta x, \delta u/\delta y, \delta u/\delta z)$, $\delta V = (\delta v/\delta x, \delta v/\delta y, \delta v/\delta z)$, and $N_{uv} = \delta U \times \delta V$, the texture space normal. The partials and $N_{uv}$ are summed at each vertex, for all triangles, and then normalized. For a triangle $T$, with vertices $v_0, v_1, v_2$ the partials are:

$$\begin{array}{rclrcl} \delta u/\delta x & = & -B_x v_0(u)/(A_x v_0(x)), & \delta v/\delta x & = & -C_x v_0(v)/(A_x v_0(x)) \\ \delta u/\delta y & = & -B_y v_0(u)/(A_y v_0(y)), & \delta v/\delta y & = & -C_y v_0(v)/(A_y v_0(y)) \\ \delta u/\delta z & = & -B_z v_0(u)/(A_z v_0(z)), & \delta v/\delta z & = & -C_z v_0(v)/(A_z v_0(z)) \end{array} \tag{2}$$

where $\langle A_x, B_x, C_x \rangle$, $\langle A_y, B_y, C_y \rangle$, $\langle A_z, B_z, C_z \rangle$ are the normals to the $xuv$, $yuv$, and $zuv$ planes at vertex $v$, respectively, and are computed as:

$$\begin{array}{lcl} v1_x & = & (v1(x) - v0(x), v1(u) - v0(u), v1(v) - v0(v)) \\ v2_x & = & (v2(x) - v0(x), v2(u) - v0(u), v2(v) - v0(v)) \\ \langle A_x, B_x, C_x \rangle & = & v1_x \times v2_x \end{array} \tag{3}$$

$$\begin{array}{lcl} v1_y & = & (v1(y) - v0(y), v1(u) - v0(u), v1(v) - v0(v)) \\ v2_y & = & (v2(y) - v0(y), v2(u) - v0(u), v2(v) - v0(v)) \\ \langle A_y, B_y, C_y \rangle & = & v1_y \times v2_y \end{array} \tag{4}$$

$$\begin{array}{lcl} v1_z & = & (v1(z) - v0(z), v1(u) - v0(u), v1(v) - v0(v)) \\ v2_z & = & (v2(z) - v0(z), v2(u) - v0(u), v2(v) - v0(v)) \\ \langle A_z, B_z, C_z \rangle & = & v1_z \times v2_z \end{array} \tag{5}$$

The $\delta U, \delta V$, and $N_{uv}$ vectors form a transformation matrix at each vertex from object space into texture space:

$$M_T = \begin{bmatrix} \delta u/\delta x & \delta v/\delta x & N_{uv}(x) \\ \delta u/\delta y & \delta v/\delta y & N_{uv}(y) \\ \delta u/\delta z & \delta v/\delta z & N_{uv}(z) \end{bmatrix} \quad (6)$$

The current light position is multiplied by the inverse of the current object to world transformation matrix to bring it into object space. The light vector is then computed at each vertex in object space. It is then transformed into each vertex's texture space by using the $M_T$ for that vertex. After the light vector has been put into texture space at a vertex, it is encoded as a color the same way normals are encoded in the normal map (Section 3) and stored as the vertex's diffuse color (Figure 6(a)).

The result of doing this is that light vectors will be linearly interpolated at all points on the mesh by the graphics card. The normal map can then be treated like a texture map (Figure 6(b)) and dotted with the encoded and interpolated light vectors, yielding per-pixel bump mapping (Figure 6(c)). This operation is equivalent to performing a per-pixel $(N \cdot L')$ where $N$ comes from the normal map and is applied on a per-pixel instead of a per-vertex basis, and $L'$ is the light vector in texture space at that vertex.

While the light vectors that result from this operation are normalized, the interpolated light vectors inside the triangle might not be. A solution to this problem is to use the interpolated light vectors as texture coordinates into a cube map, where each entry in the cube map is the normalized version of the index ([10]).

### 5.2 Lafortune Shading of Skin

Skin has a complicated BRDF that simple Gouraud or Phong shading cannot capture. We approximate the BRDF of skin with a Lafortune [11] shading model. The Lafortune model approximates the BRDF of a surface as a weighted sum of generalized cosine lobes. Each cosine lobe is parameterized by three parameters that control scaling of the dot product of the incident and exitant direction vectors along the $x, y$, and $z$ directions. These parameters and the weight on each lobe comprise a non-linear approximation to the reflectance function. We use three lobes in our renderer.

We measure the parameters of each of the three specular lobes using the technique proposed by Marschner *et al.* [14]. This technique results in the parameters for each lobe as well as the albedo map, that is, the diffuse component of the skin reflectance. In practice, we use a modified version of the Lafortune model. We use the normals from the normal map and the texture space light vectors to compute the diffuse component of the model, and the normals at the vertices to compute the specular component of the model. We use the original normals instead because we do not have access to the interpolated normals at each pixel. Since most graphics cards do not support pixel shaders, we would have to render each specular lobe off-screen as many times as its corresponding exponent if we wanted to compute this per-pixel. This operation is costly since one of the lobes has a very high exponent. We have found that interpolating the specular highlights across triangles looks realistic, but with more complicated lighting, the highlights may appear slightly triangulated. There are techniques for calculating specular reflections from approximated BRDFs efficiently ([24]), and for approximating the BRDF calculations with hardware ([9], [24]), which we did not do. However, the imminent adoption of pixel shaders in upcoming consumer-end graphics cards will allow BRDFs to be calculated directly.

## 6 Results

Our renderer runs in real-time on a Pentium III 1GHZ with a Geforce II graphics card at 13-14 frames per second. Figure 7 shows pairs of heads shaded using the Lafortune model discussed in Section 5.2 with and without fine scale structure rendering.

The faces with fine scale structure look considerably more realistic than those without. Specular highlights such as those on the forehead and left cheek look more realistic when there is fine detail under the highlight. The skin on the faces with no fine structure simply looks too smooth, although it is shaded realistically.

All results were rendered with a high-resolution (4096x2048 pixel) bumpmap. We found that detail was still retained at 2048x1024 pixels, while most was lost at 1024x512. However, the bumpmap can be quantized as in [21] or indexed more efficiently to lower the memory requirement.

## 7 Conclusion

We have presented techniques to model and render realistic fine scale skin structure. Shape from shading is computed on a series of images taken of several molds of human skin under different lighting conditions. The derived normal maps are encoded as images and "grown" using texture synthesis techniques. Boundaries are hidden using a form of multi-resolution splining. Finally, an object to normal space transformation matrix is computed at each vertex so that per-pixel bump mapping can be done.

We presented results showing that fine scale structure adds significantly to the realism of rendered faces, even when the BRDF is approximated using measured lobes in a Lafortune shading model. Skin rendered without an approximated BRDF tends to look like plastic, but even with a BRDF, it looks unrealistic. Fine scale structure is something we take for granted when looking at faces, yet is extremely important for visual realism.

### 7.1 Future work

Rendering fine scale structure is a good first step in improving the realism of skin, but there are more things that can be done. We are interested in creating a framework to synthesize and render other types of 3D skin appearance. The addition of photorealistic 3D moles or pimples is interesting. Rendering rashes, scars, or other types of skin appearance are also intriguing. Adding hair to our fine scale structured skin via image based rendering techniques or other techniques [12] is also a promising avenue of future research. Reducing the amount of texture memory required to render fine scale structure when zoomed close to the skin is also interesting.

*Acknowledgments:* The authors wish to thank Sing Bing Kang for helpful discussions. We would also like to thank Greg Turk and the paper reviewers for useful comments on the paper.

## References

1. M. Ashikhmin. Synthesizing natural textures. In *Symposium on Interactive 3D Graphics*, 2001.
2. P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. In *IEEE Transactions on Communications*, 1983.

3. P. Debevec, T. Hawkins, C. Tchou, H. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In *Computer Graphics (Proceedings of SIGGRAPH 2000)*, 145-156.
4. S. Dietrich. Texture space bumpmaps. Available from http://www.nvidia.com/.
5. B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Computer Graphics (Proceedings of SIGGRAPH 1998)*, 55-66.
6. P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. In *Computer Graphics (Proceedings of SIGGRAPH 1993)*, 165-174.
7. W. Heidrich and H. P. Seidel. Realistic, hardware-accellerated shading and lighting. In *Computer Graphics (Proceedings of SIGGRAPH 1999)*, 171-178.
8. B. K. P. Horn and M. J. Brooks. *Shape from Shading*. MIT Press, 1989.
9. J. Kautz and M. D. McCool. Interactive rendering with arbitrary brdfs using separable approximations. In *Proceedings of 10th Eurographics Workshop on Rendering*, 1999.
10. M. Kilgard. A practical and robust bump-mapping technique for today's gpus. Available from http://www.nvidia.com/.
11. E.P.F. Lafortune, S. Foo, K.E. Torrance, and D.P. Greenberg. Non-linear approximation of reflectance functions. In *Computer Graphics (Proceedings of SIGGRAPH 1997)*, 117-126.
12. J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *Symposium on Interactive 3D Graphics*, 145-156, 2001.
13. S. Marschner. *Inverse Rendering for Computer Graphics*. PhD thesis, Cornell University, 1998.
14. S. Marschner, B. Guenter, and S. Raghupathy. Modeling and rendering for realistic facial animation. In *Proceedings of 11th Eurographics Workshop on Rendering*, 2000.
15. M. Nahas, H. Huitric, M. Rioux, and J. Domey. Facial image synthesis using skin texture recording. *The Visual Computer*, 6(6):337–343, 1990.
16. F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin. Synthesizing realistic facial expressions from photographs. In *Computer Graphics (Proceedings of SIGGRAPH 1998)*, 75-84.
17. E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Computer Graphics (Proceedings of SIGGRAPH 1998)*, 465-470.
18. H. Rushmeier and F. Bernardini. Computing consistent normals and colors from photometric data. In *2nd International Conference on 3D Digital Imaging and Modeling (3DIM)*, 1999.
19. H. Rushmeier, F. Bernardini, J. Mittleman, and G. Taubin. Acquiring input for rendering at appropriate levels of detail: Digitizing a pieta. In *Rendering Techniques*, 81-92, 1998.
20. R. Scopigno, P. Pingi, C. Rocchini, P. Cignoni, and C. Montani. 3d scanning and rendering cultural heritage artifacts on a low budget. In *European Workshop on 'High Performance Graphics Systems and Applications'*, 16-17, 2000.
21. M. Tarini, P. Cignoni, C. Rocchini, and R. Scopigno. Real time, accurate, multi-featured rendering of bump mapped surfaces. In *Eurographics*, 2000.
22. L. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Computer Graphics (Proceedings of SIGGRAPH 2000)*, 479-488.
23. Y. Wu, P. Kalra, and N. Magnenat-Thalmann. Physically-based wrinkle simulation and skin rendering. In *Eurographics Workshop on Computer Animation and Simulation*, 1997.
24. C. Wynn. Real-time brdf-based lighting using cube-maps. Available from http://www.nvidia.com/.
25. Y. Xu., B. Guo, and H. Shum. Chaos mosaic: Fast and memory efficient texture synthesis. Technical Report MSR-TR-2000-32, Microsoft Research, 2000.
26. Yizhou Yu and Jitendra Malik. Recovering photometric properties of architectural scenes from photographs. In *Computer Graphics (Proceedings of SIGGRAPH 1998)*, 207-217.

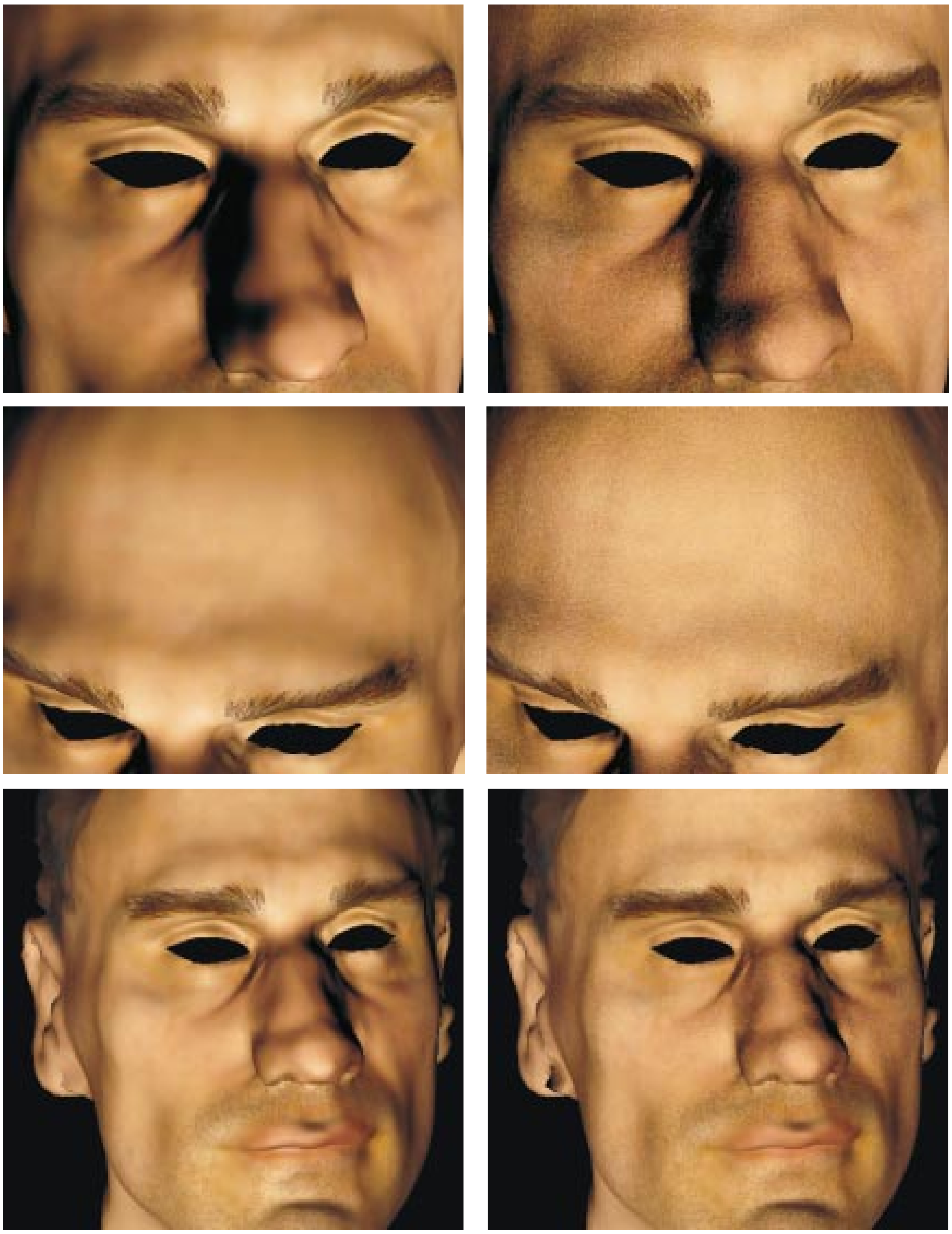
**Fig. 7.** *Left*: Skin rendered without fine scale structure, *Right*: Skin rendered with fine scale structure

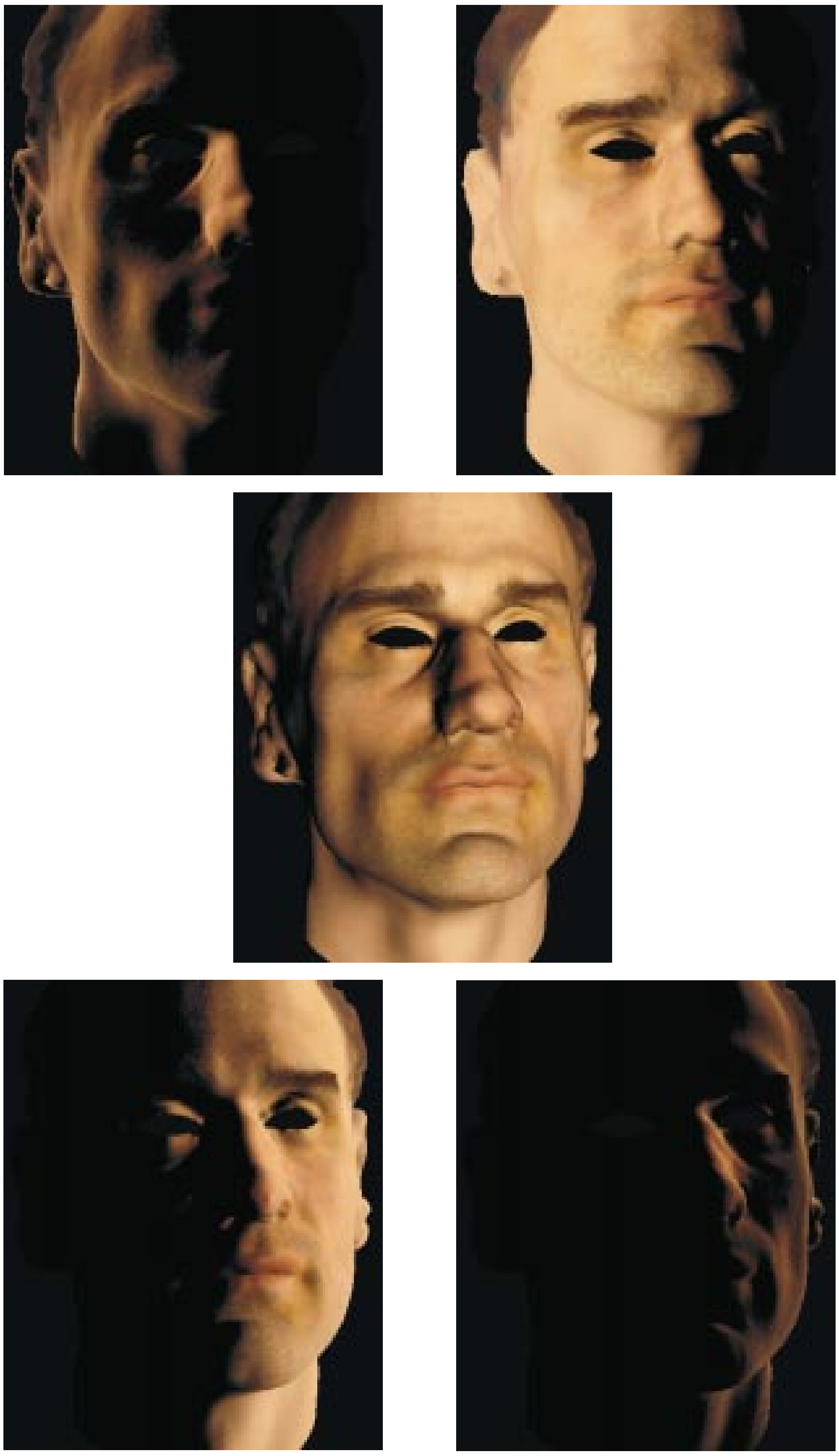

**Fig. 8.** Effects of light position on fine scale structure