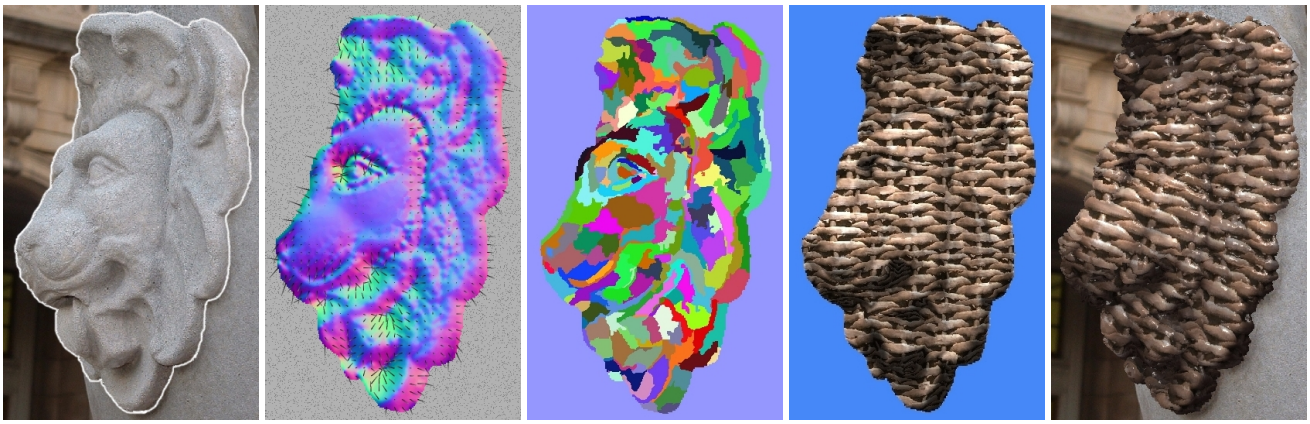
Textureshop: Texture Synthesis as a Photograph Editing Tool

Hui Fang John C. Hart

University of Illinois, Urbana-Champaign



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

Figure 1: (a) An ordinary photo is scanned in. (b) Shape-from-shading applied to an object in the image to estimate its normals. (c) Pixel patches formed by clustering normals. (d) Texture synthesized on these patches and aligned with neighboring patches. (e) Final result with texture orientation distortion, displacement mapping and environment mapping.

Abstract

We combine existing techniques for shape-from-shading and texture synthesis to create a new tool for texturing objects in photographs. Our approach clusters pixels with similar recovered normals into patches on which texture is synthesized. Distorting the texture based on the recovered normals creates the illusion that the texture adheres to the undulations of the photographed surface. Inconsistencies in the recovered surface are disguised by the graphcut blending of the individually textured patches. Further applications include the generation of detail on manually-shaded painting, extracting and synthesizing a displacement map from a texture swatch, and the embossed transfer of normals from one image to another, which would be difficult to create with current image processing packages.

1 Introduction

Texture synthesis has revolutionized the construction of texture maps and the application of texture to surfaces. Given a texture swatch, it uses a machine learning process to plausibly extrapolate its pattern, extending the texture features across an image plane or the surface of a geometric model [Turk 2001; Wei and Levoy 2001].

This paper suggests a novel application of texture synthesis in photograph editing. Various commercial software packages for photograph editing exist, but they work in the image space and do not offer a direct method to apply texture to the surface of a photographed object, and using these tools for the convincing texturing of an undulating surface requires both time and skill.

Shape-from-shading techniques can recover a height field from the image of a shaded surface, and existing texture synthesis techniques could be applied to a recovered mesh. However, such reconstruction is complex, expensive and inaccurate, particularly in the presence of noise or an unknown reflectance map.

We overcome these limitations by creating small pixel patches, clustered by similar recovered normals. We perform texture synthesis on these patches, distorting pixel positions into a local parameterization to account for patch orientation and displacement. These patches are further distorted to match the features of neighboring patches. This patchwork of feature-aligned foreshortened textured pixel clusters gives the illusion that the texture is applied to the photographed surface. We also apply shape-from-shading to the texture to support displacement mapping and normal transfer (embossing).

2 Previous Work

Of the large body of recent papers on texture synthesis, those most germane to our application focus on finding seams in a texture swatch that allow low-frequency features to be plausibly reproduced, such as image quilting [Efros and Freeman 2001] and graphcut textures [Kwatra et al. 2003]. Our approach segments the image of an object into patches of similarly oriented pixels, and performs a similar seam determination on these patches.

Any of the existing methods for reconstructing a full 3-D meshed model, usually from a series of images from different viewpoints, would support existing surface texture synthesis methods [Wei and Levoy 2001; Turk 2001]. Our application of graphcut textures avoids the need for reconstructing global 3-D mesh, and synthesize the texture instead on a network of individually parameterized surface patches.

Our method is designed to operate on a single image. Inpainting is a technique for extending an image across the pixel regions left blank from removing unwanted objects from the picture [Bertalmio et al. 2000]. Image inpainting performs texture synthesis on an image, but draws its texture samples from the original image instead of a separate texture swatch. Poisson image editing performs inpainting with a texture synthesized from a separate source, but this source is a second image instead of a texture swatch [Perez et al. 2003]. Furthermore, neither method was designed to utilize the 3-D shading cues of the removed object. Our normal transfer application instead applies Poisson image editing to blend surface normals recovered from one image onto another.

Oh *et al.* [2001] describe a user-assisted system for constructing an image-based model from a single photograph. Like our method, they too segment an image, though into a layered depth image [Shade et al. 1998]. They depend on a computer-assisted user to infer depth from parallel and orthogonal configurations in the scene and to paint the portions of the scene occluded in the original image. Texture synthesis could be implemented on the resulting layered depth image, but coding such a complex system to achieve just texture synthesis is an overkill, and we describe a much simpler method that adequately extracts the normals of an object in a photograph.

Ismert *et al.* [2003] work on the related problem of image-based texturing, where a texture is extracted from input images used for image-based modeling, and texture synthesis is used to recover the high-frequency detail lost in the resampling process of the image-based rendering system.

Graphcut textures segment an image into overlapping patches and a max-flow algorithm is used to find a portion of a source image to lay down in the current patch position, blending features through overlapping patch boundaries [Kwatra et al. 2003]. Our application augments graphcut textures with deformations that create the illusion that the textured patches lie on the photographed surface.

3 Shape from Shading

A wide variety of sophisticated shape-from-shading and photoclinometric algorithms exist for reconstructing a surface from an image [Horn 1990]. These methods pose shape-from-shading as an optimization problem and employ iterative methods to solve the resulting partial differential equations. The results are gradient and height fields consistent with the surface portrayed in the image.

For visually plausible texture synthesis, we need not find a strictly consistent height field nor construct a full surface representation. We will instead segment the surface into oriented patches, and perform texture synthesis independently on these patches.

3.1 Normal Recovery

Horn [1990] gives the formulae for recovering the surface normals from an image for a wide variety of reflectance functions, but we find the following simple Lambertian reflectance model works well for our purposes. Let S be the unit vector from the center of the object toward a sufficiently distant point light source. We further assume the point on the surface with largest intensity I_{max} faces the light source. The darkest point is shadowed and its intensity I_{min} indicates the ambient light in the scene. The function $c(x,y) = (I(x,y) - I_{\min})/(I_{\max} - I_{\min})$ estimates the cosine of the angle of incidence, and $s(x,y) = \sqrt{1 - c(x,y)^2}$ its sine, which leads to the recovered normal $N(x,y)$ as

$$G(x,y) = \nabla I(x,y) - (\nabla I(x,y) \cdot S)S, \quad (1)$$

$$N(x,y) = c(x,y)\, S + s(x,y)\, G(x,y)/||G(x,y)|| \quad (2)$$

where $\nabla I(x,y) = (\partial I/\partial x, \partial I/\partial y, 0)$ is the image gradient.

We estimate the vector to the light S from the intensity of pixels (x_i, y_i) on the boundary of the object's projection. For such pixels the normal $N(x_i, y_i)$ is in the direction of the strong edge gradient. The source vector S is then the least-squares solution to the over-constrained linear system

$$N(x_i, y_i) \cdot S = \frac{I(x_i, y_i) - I_{\min}}{I_{\max} - I_{\min}}. \quad (3)$$

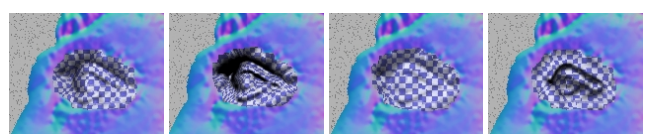
The user can adjust the light source direction manually if the inferred result is incorrect.

Such an estimated normal field is generally inaccurate, but it captures the undulations of a surface well enough to support texture synthesis. It is also very fast, thus suitable for an interactive photograph editing tool.

3.2 Interactive Normal Editing

The normal field recovered from the brightness of an image can be unsatisfying due to various reasons: multiple light sources, non-Lambertian materials and textured materials. Moreover, shading information is lost in shadowed areas. Rather than attempting to handle these complications automatically, we instead implemented several intuitive methods for tuning the result interactively.

We display the normal field as an RGB-coded normal image, a vector field, or with a quickly synthesized texture. The user can manipulate the reflectance model of the surface with a spline curve initialized to the Lambertian reflection model. The influence of surface texture and noise can be reduced by smoothing the image intensity and/or the recovered normal field. The user can also rotate selected normals to compensate for the effect of a second light source. Finally, the variation of normals over a region can be manipulated, as demonstrated in Fig. 2.



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

Figure 2: Variation of the normal field (a) is enhanced (b), reduced (c) and reversed (d).

3.3 Surface segmentation

The surface pixels are grouped into patches with similar normal directions using a bottom-up scheme. The segmentation process is initialized by assigning each pixel to its own patch. For each patch P_i, let N_i, C_i and $|P_i|$ denote the patch's mean normal, centroid pixel and number of pixels, respectively. Then two neighboring patches P_1, P_2 are merged if the error metric

$$E(P_1, P_2) = k_1 \sqrt{1 - N_1 \cdot N_2} + k_2 ||C_1 - C_2|| + k_3(|P_1| + |P_2|) \quad (4)$$

falls below a given threshold. Appropriate settings for the constants $k_{1,2,3}$ will yield moderate-sized round patches of similarly oriented pixels. In most cases we used $k_1 = 187$, $k_2 = 20$, $k_3 = 1$ except for Fig. 1(c) which used a larger k_1 to further emphasize orientation clustering. The patches are then expanded by a fixed-width boundary (16 pixels in our examples) so they overlap each other.

4 Patch Distortion

Though the patches have been formed, simply applying the graphcut texturing algorithm [Kwatra et al. 2003] on them will yield a flat

texture. We describe several patch distortions that result in a more realistic surface texturing appearance.

4.1 Patch Orientation

To achieve the illusion that the texture follows the underlying surface, a patch orientation distortion algorithm will assign each pixel $P(x,y)$ a new position in its texture coordinates $U(x,y) = (u,v)$ to capture the foreshortening distortion due to its recovered normal.

The algorithm starts at the center pixel $P(0,0)$ of the patch, setting its texture coordinates $U(0,0) = (0,0)$, and propagates the distortion to the rest of the patch in a width-first floodfill order.

Let $P(x,y)$ indicate the pixel at (x,y) with distorted position $U(x,y)$ and recovered (unitized) normal $N(x,y) = (N_x, N_y, N_z)$. Given $P(x,y)$ we compute the foreshortening distortion of the next pixel to its right $P(x+1,y)$ by projecting this pixel's position $(x+1,y,0)$ onto the recovered tangent plane of pixel $P(x,y)$ and then rotating this projection back into the image plane, as illustrated in Fig. 3. The distortion is cumulative and propagates by adding the resulting offset to the current distortion $U(x,y)$ and storing the result in $U(x+1,y)$.

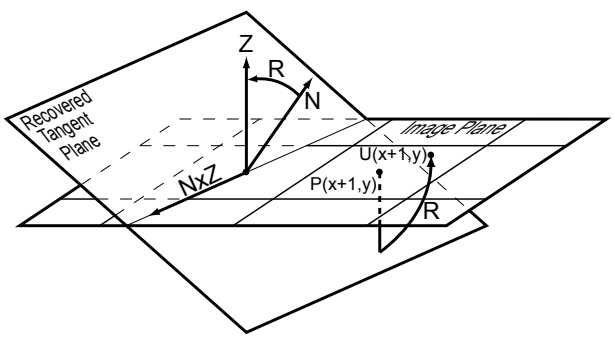


Figure 3: Texture distortion according to recovered patch orientation.

The projection of the point $(x+1,y,0)$ onto the plane with normal $N(x,y)$ passing through $(x,y,0)$ is $(x+1,y,-N_x/N_z)$. Let θ be the angle between N and $Z = (0,0,1)$ and abbreviate $c = \cos\theta = N_z$ and $s = \sin\theta = \sqrt{N_x^2 + N_y^2}$. The unitized axis of rotation is $(N \times Z)/||N \times Z|| = (N_y/s, -N_x/s, 0)$ which leads to the rotation matrix

$$R = \begin{bmatrix} c + (1-c)N_y^2/s^2 & -(1-c)N_xN_y/s^2 & -N_x \\ -(1-c)N_xN_y/s^2 & c + (1-c)N_x^2/s^2 & -N_y \\ N_x & N_y & N_z \end{bmatrix}. \quad (5)$$

The product $R(1,0,-N_x/N_z)$ yields the new position of pixel $P(x+1,y)$, leading to the propagation rules

$$U(x \pm 1, y) = U(x,y) \pm \frac{(1 + N_z - N_y^2, N_xN_y)}{(1+N_z)N_z}, \quad (6)$$

$$U(x, y \pm 1) = U(x,y) \pm \frac{(N_xN_y, 1 + N_z - N_x^2)}{(1+N_z)N_z}. \quad (7)$$

We clamp N_z to a minimum of 0.1 and renormalize N_x, N_y to avoid outrageous distortions. If the distortions of more than one neighboring pixel are available for propagation, then the final orientation distortion is the mean of the distortions computed from each of these neighbors. This averaging reveals that this scheme generates an inconsistent parameterization, and these inconsistencies increase in severity with distance from the centroid, but our segmentation heuristic is designed to produce small round patches that reduce the variance of their normals to keep these internal inconsistencies small, and the inconsistencies between patches are later camouflaged by the feature-sensitive seams cut through overlapping areas by the graphcut algorithm.

4.2 Texture Orientation

Texture orientation can also be defined over the image, to more consistently align anisotropic features of the synthesized texture. Vector field orientation can be modified by dragging the mouse over the photo. The patch parameterization is then rotated about its centroid (conveniently the origin of the parameterization) to align the preferred texture direction vector with the appropriate axis of the texture swatch. The rotation of the parameterization effectively rotates the patch about its average normal.

4.3 Displacement Mapping

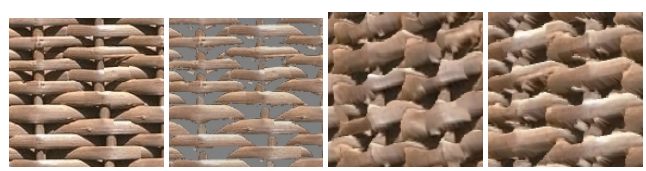
We have thus far applied shape-from-shading techniques to the photograph to recover a local surface on which to synthesize a texture. We can also apply shape-from-shading to the texture swatch used as a source for texture synthesis. This will allow us to do displacement mapping on surface.

We predict the normals $\widehat{N}(x,y)$ of the texture swatch using the same method from the previous section. But whereas the photographed object surface was reconstructed locally, the texture swatch will require a global surface reconstruction. Assuming the input texture color variation is caused only by local normal changes, the height field of the texture swatch $h(x,y)$ is determined by the Poisson equation

$$\nabla^2 h(x,y) = \nabla \cdot \widehat{N}(x,y) \quad (8)$$

and solved by conjugate gradients. The user-specified height of a portion of the texture serves as a boundary condition (e.g. the shadowed area of Fig. 4(b) was assigned a height of zero).

Often features reconstructed by (8) will shrink or grow when compared to the original. In Fig. 4(c), the reconstructed wicker is too narrow, but can be interactively corrected by a user-specified nonlinear scale of the height field, yielding Fig. 4(d).



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

Figure 4: Basket texture original (a). Height of shadowed area is set to zero (b). Wicker of the basket is narrow in recovered result (c). Wicker become wider after a non-linear scaling on the height field (d).

During displacement-map texture synthesis, each texture sample is translated in the direction of the photograph's image-projected recovered normal $(N_x, N_y, 0)$ by the recovered texture height $h(x,y)$ foreshortened by the recovered texture normal $\sqrt{1 - \widehat{N}_z^2}$. We upsample both the surface normal and texture height to avoid holes. An example is shown in Fig. 5 (b).

These displacements are significant enough to cause aliases when a texture, such as wicker, contains sharp edges. These artifacts can be sufficiently reduced by blending the edge samples with

(a) (b) (c)

Figure 5: Texture synthesis from a source image texture without displacement mapping is betrayed by its smooth silhouette (a). Applying shape-from-shading to the source texture produces a noisy displacement mapping (b) that is fixed by upsampling and filtering (c).

Painter's algorithm according to the percentage of the pixel they cover, as shown in Fig. 5(c).

4.4 Feature Matching

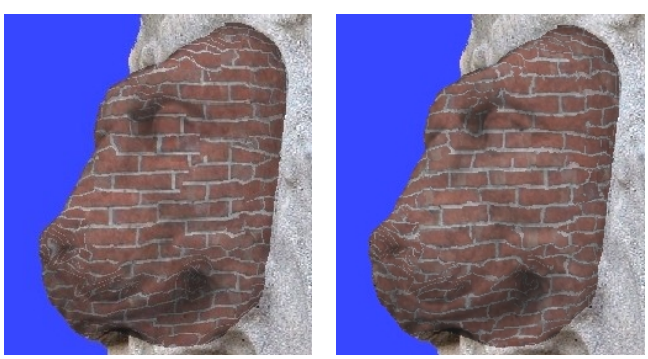
Once the distortions and displacements have been computed, texture synthesis occurs on the deformed patches with samples from the displaced texture swatches. At this point the graphcut algorithm [Kwatra et al. 2003] could cut a seam through the overlapping textured patches, but graphcut may not align all of the features in the synthesized textures.

We align these features with a deformation algorithm that resembles methods used in smoke animation [McNamara et al. 2003]. First we blur the synthesized texture in the overlapping portions of both patches $P_1(x,y)$ and $P_2(x,y)$. For each pixel position $\mathbf{x} = (x,y)$ in the overlapping boundaries of the patches, we define a 2-D deformation vector $U(\mathbf{x})$, initialized to $(0,0)$. We then define an objective function

$$\varphi = k_1 \sum ||P_1(\mathbf{x}) - P_2(\mathbf{x} + U(\mathbf{x}))|| + k_2 \sum |\nabla \cdot U(\mathbf{x})| \qquad (9)$$

to maximize the color match while minimizing the amount of deformation over the patch overlap area. We set $k_1 = 1, k_2 = 9$ and our RGB channels ranged from $0 \ldots 255$. Our feature mapping implementation computed $\partial\varphi/\partial U(\mathbf{x})$ and minimized φ using conjugate gradients. We found the deformation vector can be solved on a subset of the overlapping pixels and interpolated on the rest to accelerate convergence and further smooth the deformation, though at the risk of overlooking the matching of smaller features.

Once the deformation vectors have been constructed on the overlapping boundaries, they are blended into to the new patch's interior via Poisson image editing [Perez et al. 2003], and the graphcut algorithm is finally applied to find the optimal seam through the overlapping area. Our cost function for cutting a seam through overlapping areas is a weighted combination of pixel color and recovered surface normal, though color alone suffices in most cases. Our overlapping area is 16 pixels wide. Fig. 6 demonstrates the result.



(a) (b)

Figure 6: Brick texture without (a) and with (b) morphing.

Figure 7: Brick texture follows the surface in the original photo (inset).

5 Results

We demostrate several applications of our algorithm in photograph editing that would be difficult or at least cumbersome to achieve with current software.

Surface Replacement. These techniques were designed for the application of replacing the appearance of a photographed surface with that of a synthesized texture. This works best when the photographed surface is untextured and nearly Lambertian (e.g. skin, clothes, sculptures), illuminated by a single directional source. Errors in the recovered normal field can be rectified by additional user manipulation. Figs. 7, 10 and 11 give three examples of texturing different real world surfaces. Fig. 10 (f) pushed our method to its perceptual limits. The constant size and well-known shape of text characters, more so than the other textures, accentuate the inaccuracy of estimated normal field.

Detail Generation. Hand painting objects into a photo with plausible shading is not difficult, but painting detail into such artificial objects can be time consuming. In Fig. 8, several vases are

Figure 8: Texture synthesis (below) yields detail that follows the surface implied by the hand painted shading (above).

painted into a scene with a plausible approximation of shading, then different textures are applied to create intricate details that follow the surface implied by the painted shading.

Normal Transfer (Embossing). In Fig. 9, the recovered surface normals from one image is applied to another, yielding an embossed result. Poisson image editing is used to seamlessly merge transfered normal and brightness into the target image's normal and brightness respectively. Texture on the original surface in target image is extracted as texture swatch. If available texture is not large enough, a 2D texture synthesis may be applied to generate a larger one. A 256×256 pixel texture is enough for generating all results for this paper.

Lighting. Though the original photographed surface brightness can be used to shade the result, the synthesized texture with the recovered surface normals can be rendered under any desired lighting configuration. The synthesized texture can even be environment mapped to appear more naturally embedded in the scene as demonstrated in Fig. 8. An environment map can be synthesized from the background of the photograph through inpainting and extrapolation [Drori et al. 2003].

Performance. The synthesis speed highly depends on searching effort, patch size and whether displacement mapping and feature matching are enabled. When the texture has a large regular pattern like brick, searching a larger area is required to find a match. Some typical speed on a 1.2GHz Athlon is shown in Table 1.

Image	Displacement Mapping	Feature Matching	Time
Fig. 10(d)	Yes	No	55s
(Fig. 10(d))	No	No	24s
(Fig. 10(d))	Yes	Yes	140s
Fig. 7	No	Yes	90s
Fig. 11(c)	No	No	12s
Fig. 8 (all vases)	No	Yes	63s
Fig. 9(b)	No	No	18s

Table 1: Features and running times of figures in this paper.

6 Conclusion

The sophisticated tools of shape from shading and distorted graphcut textures combine to form a new tool that conveniently and robustly allows a user to texture an object in a photograph. The objects are limited to smooth diffuse surfaces illuminated by simple lighting conditions, but the method has worked well for us on faces and t-shirts, as well as sculptures and architectural embelishments. Implementation of more sophisticated shape from shading could extend these technqiues to a wider variety of surface types, reflectances and lighting conditions. In its present form, Textureshop works well enough to serve as a tool for concept visualization in architecture, art, fashion, design, visual effects and, in general, personal digital content creation.

Acknowledgments. Thanks to Jesse Hall for editing the demo video and Patrick Lacz for proofreading. This work is supported in part by the NSF under CCF-0121288.

References

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Proc. SIGGRAPH 00*, 417–424.

DRORI, I., COHEN-OR, D., AND YESHURUN, H. 2003. Fragment-based image completion. *Proc. SIGGRAPH 03*, 303–312.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. *SIGGRAPH 2001*.

HORN, B. K. 1990. Height and gradient from shading. *International journal of computer vision, 5:1,, 37-75, 1990*.

ISMERT, R. M., BALA, K., AND GREENBERG, D. P. 2003. Detail synthesis for image-based texturing. In *Proc. I3D*, 171–175.

KWATRA, V., SCHOEDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *Proc. SIGGRAPH 2003*.

MCNAMARA, A., TREUILLE, A., POPOVIC, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *Proc. SIGGRAPH 2003*.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proc. SIGGRAPH 2001*, 433–442.

PEREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *Proc. SIGGRAPH 2003*, 313 – 318.

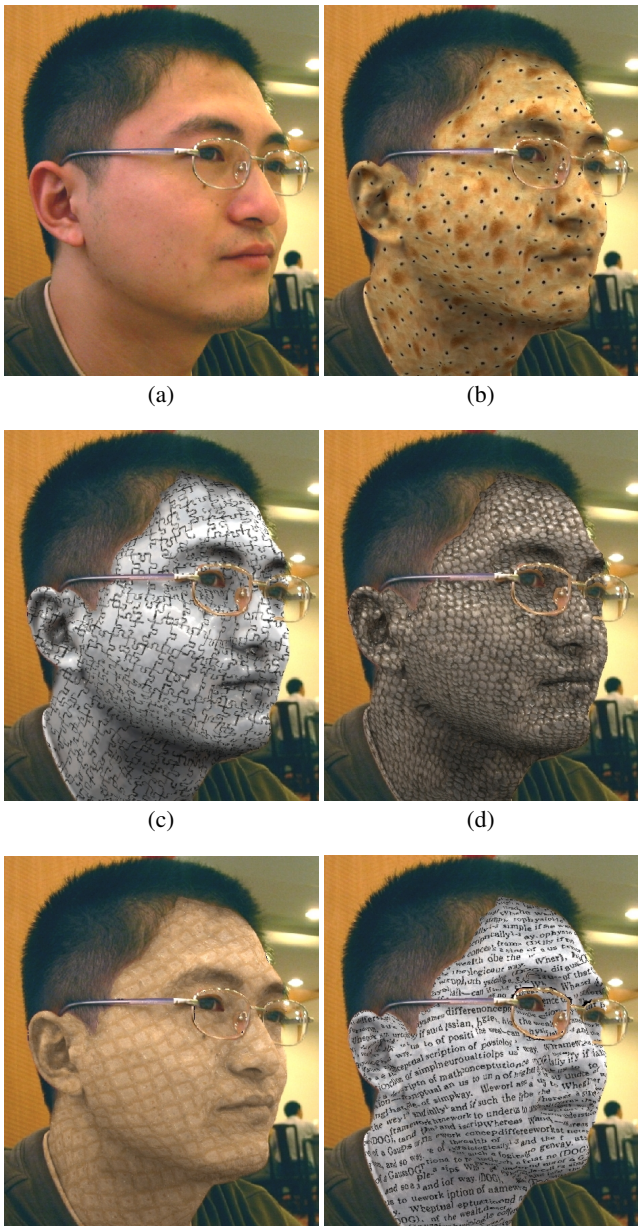
SHADE, J., GORTLER, S., WEI HE, L., AND SZELISKI, R. 1998. Layered depth images. In *Proc. SIGGRAPH 98*, 231–242.

TURK, G. 2001. Texture synthesis on surfaces. *Proc. SIGGRAPH 2001*.

WEI, L.-Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. *SIGGRAPH 2001*.

(a) (b) (c)

Figure 9: The normal field recovered from one image is transferred onto another.



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

Figure 10: Makeup.

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

Figure 11: Fashion.