Image-Based Object Editing

Holly Rushmeier Jose Gomes Laurent Balmelli Fausto Bernardini Gabriel Taubin

IBM T.J. Watson Research
P.O. Box 704, Yorktown Heights, NY 10598
hertjwr,josegome,balmelli,fausto@us.ibm.com

Abstract

We examine the problem of editing complex 3D objects. We convert the problem of editing a 3D object of arbitrary size and surface properties to a problem of editing a 2D image. We allow the user to specify edits in both geometry and surface properties from any view and at any resolution they find convenient, regardless of the interactive rendering capability of their computer. We use specially-constrained shape from shading algorithms to convert a shaded image specified by the user into a 3D geometry.

1 Introduction

Scanning devices can capture models of up to billions of vertices with accompanying surface properties [1]. The editing of such complex 3D objects is a challenging user interface problem. Engineers and technicians are no longer the only population that want to edit complex models. Object editing may be required by historians or archaeologists to study and evaluate conjectures about digitized artifacts [2]. We present an image-based method for editing complex 3D objects. Our approach decouples object complexity and the interactive display capabilities of the user's computer. Most importantly, it also allows the use of familiar image editing tools, making 3D editing easily accessible to a wider range of users.

One reason that 3D object editing is difficult is that it is now common for models to consist of hundreds of megabytes or more. While numerous simplification methods have been developed, many objects overwhelm rendering systems when displayed with full visual quality. In the method presented in this paper, complex objects are represented by triangle meshes of arbitrary size associated with an atlas of images which define surface details and appearance properties. The geometric representation of the object is not exposed to the user. The user specifies edits by positioning a simplified model and generating a detailed image at arbitrary resolution. The user edits the detailed image to specify object edits. The edited image is then used to update the 3D model.

The second reason the problem is difficult is that a user is trying to modify an object in 3D with either a 2D or an imprecise 3D interface device. Large scale general changes in shape are readily indicated, but fine scale editing over a large region is difficult. In 2D, a gesture by the user needs to be interpreted as a 3D operation – such as pushing, pulling or cutting an object. In 3D, the response the user has using common haptic devices is limited compared with true physical modeling such as shaping clay or chiseling stone.

We explore the problem of object editing to achieve a particular visual effect. Our motivating application is virtual restoration of museum artifacts, as illustrated in Fig. 11. The user wants to change the object to look a particular way, rather than trying to achieve a mechanical effect such as fitting into another part. A direct way to express the visual effect is to alter an image of the object to how the user envisions it after modification. Since the object is not updated during the edits, the editing operation is not limited by the user's computer's 3D rendering capabilities, and the user does not need to be aware of the object's numerical representation.

2 Previous Work

In this section we review relevant previous work relating images and geometry that we build on to develop our image-based object editing system.

Systems for 2D image painting and editing have evolved over several decades [3]. While a variety of commercial systems are available, they share a common set of user tools. Commercial products, building on the work of Hanrahan and Haberli [4], have extended these 2D paint tools to painting multiple layers of appearance attributes (color, bumps, shininess etc.) on 3D objects. Some 3-D paint programs (*e.g.* DeepPaint3D ™) offer the option of a "projection paint" mode that allows the user to export 2D images of

an object from an arbitrary view to a 2D image editing program to edit the object texture layers with a full array of 2D imaging tools. The projection paint system then projects the user's texture edits back onto the maps associated with the 3D model.

Recognizing the success of 2D paint programs, researchers in 3D geometric editing have adapted successful tools such as cut-and-paste and other image operations to 3D editing toolkits. However, these systems use the 2D toolkits as inspiration, rather than giving users the ability to edit geometry from within an existing 2D editing system. Some systems allow users to use 2D systems to create 2.5D surfaces by interpreting painted intensity values as heights. These systems are indirect, however, because grayscale coded heights do not correspond to a shaded rendering of an object.

With the interest in recent years in image-based modeling and rendering, many efforts have explored how geometry and images can be beneficially combined. Various researchers have considered how limited geometric information can be used to enhance image editing systems. Oh *et al.* present a system that allows a user to add depth information that is used in an enhanced photo-editing system [5]. Seitz and Kutulakos [6] describe a method that uses a crude intermediate geometric representation to facilitate the simultaneous editing of multiple views of an object. There has also been a lot of interest in refining methods from computer vision for geometric object creation, such as the method of Debevec *et al.* [7]. While methods from computer vision have been successful for *creating* 3D objects, most methods do not lend themselves to *editing* 3D objects.

It is natural to ask the user to edit a single image of the shape as they want to see it. The method from computer vision which extracts an object from a natural image is shape from shading [8]. A wide variety of shape from shading algorithms exist, but have not gained popularity in capture systems because they are not robust in the presence of effects such as spatially varying albedo, uncertainty in light source direction and surfaces with discontinuities. Van Overveld [9] makes compelling arguments for an image-based geometric editing system, but rejects the use of shape from shading as too time consuming. Instead, Van Overveld proposes a specialized system in which a user paints gradients directly and is required to define an integrable surface at each step in the editing process. This constrains intermediate edits, and disallows the use of general 2D paint programs.

We present how the shape from shading problem can be formulated and constrained to be usable for image-based object editing. Our new approach allows 3D object editing to be performed in any 2D image editing program.

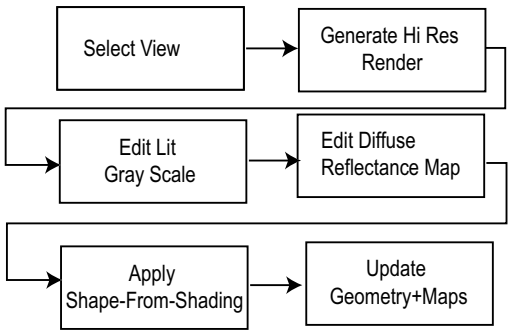

Figure 1. Image-based object editing workflow.

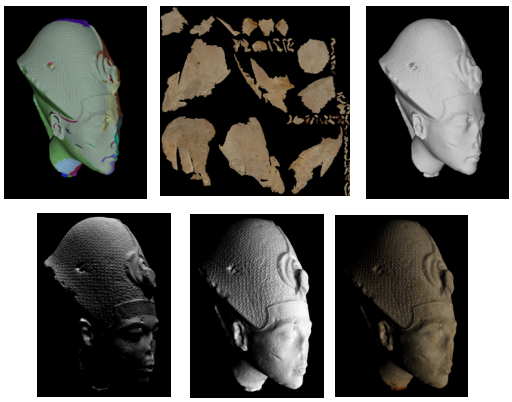

Figure 2. Top row: Partitioned triangle mesh (left), diffuse reflectance(center), simplified geometry (right); Bottom row: Model lit using normal maps for two lighting conditions (left and center), a complete object rendering (right.)

3 System Overview

Figure 1 shows the process of defining and processing an object edit in our system. We describe the system for a form of surface representation produced by a particular 3D scanning system, although the approach can be applied to the geometric output of any scanning system.

3.1 Object Representation and Viewing

We consider models that are generated by scanning systems similar to the system described in [10]. The geometry of the object is represented as a triangle mesh. The mesh is partitioned into charts that are each a height field. An atlas of surface maps representing diffuse surface reflectance and high spatial resolution normals are associated with the charts using a standard texture coordinate scheme.

An example of a partitioned surface and an associated atlas is shown in the top row of Fig. 2. We show the colored map that gives the diffuse (*i.e.* Lambertian) surface reflectance. There are also maps with the same structure that store the surface normals at a higher resolution than the base geometry. These high resolution normals were obtained by a separate photometric system attached to the shape-scanning device used to capture the base triangle mesh. On the far right of the first row, we show a simplified version of the geometry that is provided with the full model. In the second row of Fig. 2 renderings of the detailed geometry represented in the normal maps are shown for two lighting conditions (left and center images), and the right most image shows the result of rendering using the geometry, normal maps and colored diffuse reflectance map.

The objects we consider may be too large to render interactively with all of the associated surface texture maps. We separate the selection of the view and the lighting of the image to be edited. First, to inspect the object we use a 3D viewer to position the geometrically simplified version of the object, and save the camera parameters for a view we wish to inspect in greater detail. An off-line renderer is then used to generate a high resolution multi-channel image for the camera view. The multi-channel image includes three color channels for the reflectance and additional channels for surface normals and surface z-depth from the camera. These images are not exposed directly to the user, but are supplied to an image-based relighting system [10] that the user employs to determine a light direction for performing the object edits. The images above the horizontal rule in Fig. 3 show a small region on the object being targeted, and the shaded image of the geometry (using both the full geometry and normal map details) and colored diffuse reflectance map generated for the view and lighting direction selected by the user.

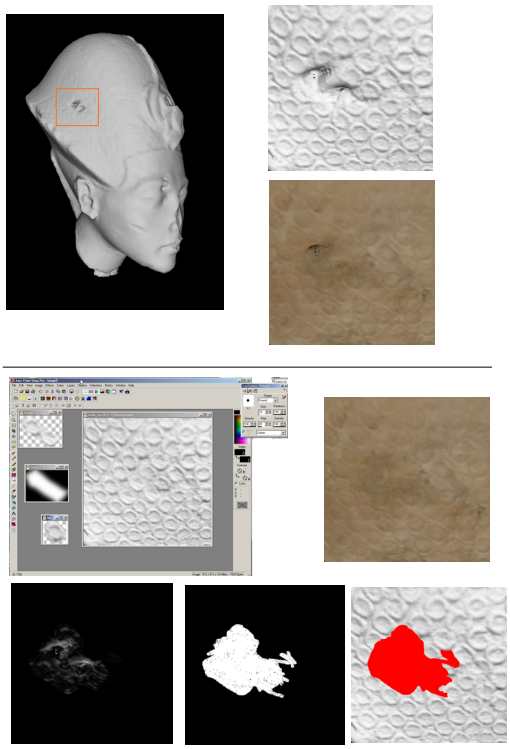

Figure 3. Above horizontal rule: simplified geometry, shaded geometry and diffuse reflectance to be edited. Below horizontal rule: editing shaded geometry with a 2D paint program, and edited diffuse reflectance. Bottom row: the difference of original and edited images of lit geometry (left), thresholded difference image (center) and section of the geometry to be updated (right).

3.2 Editing Scenarios

Given a rendering for a particular viewpoint and light source, the user performs edits to geometry and diffuse reflectance in two phases. First the user edits the gray-scale diffuse rendering of the geometry. The image is generated by our relighting system, so we can ensure that the pixel values are linear with the luminance. In the section of Fig. 3 below the horizontal rule the editing of the image of the lit geometry using a 2D paint program with tools such as cut and paste and image blur is shown. We are filling a chip in the head of the sculpture, and covering the filled area with the ringed pattern that appears in the surrounding region. Next, the diffuse reflectance map is updated by the user. The result of editing the diffuse reflectance map to delete the black area associated with the chip is also shown in Fig. 3.

It is possible that the luminance of some pixels will not change, even though a change in the geometry at that pixel is intended. In the bottom row of Fig. 3 on the left we show the difference between the original and edited geometry images. The center image shows the results of thresholding the difference image. We use image dilation and painting to fill in the full area in which we would like to have the geometry change. This is shown in red on the far right. This"hints" image will indicate to the shape from shading solver the area in which changes are to be made.

To ensure that the edited object will remain a manifold we require that edited areas in the image are bounded by fixed depth map values and do not include depth discontinuities. An editing scenario with these considerations is shown in Fig. 4. Areas that cannot be edited for a view are marked in blue, as shown in the shaded geometry image Fig. 4b. Within these constraints, any image operation

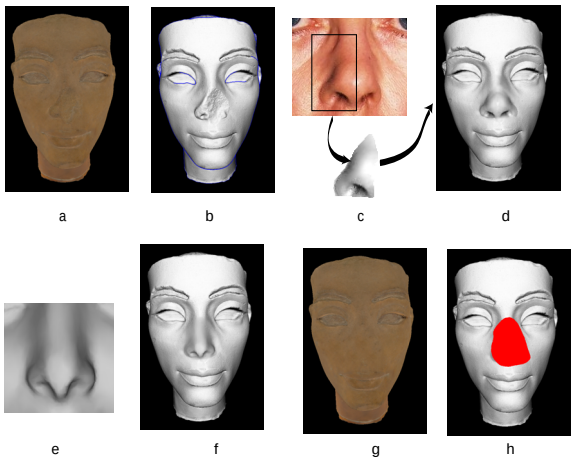

Figure 4. Editing: (a) The diffuse reflectance, (b) shaded geometry, (c)and (d) new nose inserted using photograph of a nose, (e) and (f) new nose using an image of a nose from another model, (g) edited diffuse reflectance map (h) and "hints" image.

is acceptable – blurring, sharpening, image cut-and-paste, painting, etc. In Fig. 4 we show the replacement of the nose of the figure with two types of input. In Fig. 4c and d, we start with an image of a real nose, take one side of it, alter and smooth the shading, and then use it to replace the nose section on the head. *Note that the albedo and lighting of the source image for the nose are unknown.* In Fig. 4e and f we take an example from a shaded image of another model. We alter the shading and change the aspect ratio of the image to fit it into the missing nose area. Figure 4g shows the edited diffuse reflectance map and h shows the extent the edits to apply for these two alternative noses.

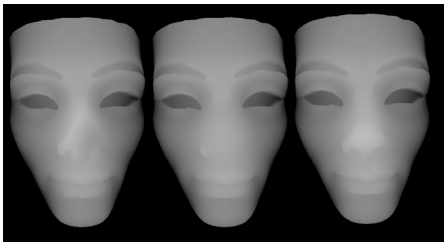

Figure 5. Left: initial heights for the example in Fig. 4. Center: rough estimate of new shape painted by user. Right: the resulting height field for the first edit.

For larger scale edits such as the ones shown in Fig. 4 we can further assist the shape from shading solution by specifying an initial guess for the solution by painting on an image of the area with pixels given gray values proportional to height, as shown in Fig. 5. On the left is an image of the original height field. This image shows clearly why editing grey scale encoded heights directly is not suitable for defining fine edits. The main feature that is visible is the sharp change in height where the nose is broken. The image in the center has been painted to indicate crudely the change in the object by smoothing out this sharp feature. The image on the right shows the height field resulting after applying the edit in Fig. 4c and d using shape from shading.

3.3 Shape from Shading

The diffuse reflectance map edits can be applied directly to update the original object. However, the grayscale edits must be converted into an updated depth map before the edits can be applied. To determine the new depths for the edited region, we solve the classic shape from shading problem. Shading is the variation of brightness across the photograph resulting from the local variation of the orientation of the surface with respect to the light sources and the camera. This question has been explored extensively as described in a textbook [11] and a recent survey [8].

Let us identify the aspects of the theory relevant to image-based geometric editing. The factors accounting for shading are the lighting conditions, the object shape, its material reflectance properties, and the camera properties. Isolating the shape information is too difficult in general and the problem needs to be simplified a great deal. The approach that has been most successful is to assume that the light source is bounded and at an infinite distance (*i.e.* a directional light), that the object has a smooth shape and is Lambertian, that there is no occlusion boundary, that the solution is known on the boundary of the resolution domain, and finally, that the camera performs an orthographic projection.

By design, all these conditions except the last are ideally met in our application. We are not dealing with a natural photograph but with an artificial image. In our retouching application, we use a perspective camera for more realism, but the spatial extent of the edits is relatively small and we approximate the camera locally by an orthographic model. The usual approximate model of shape from shading for real photographs becomes a better model for image-based geometric editing because the only approximation is the camera model.

Let us recall this model. Consider an open set $\Omega \in \mathbb{R}^2$ of image pixels corresponding to an entirely visible and lit part $\mathbf{S}$ of the depicted object surface. The brightness of the rendered image is then given by $I(\mathbf{p}) = \mathbf{n}(\mathbf{x}) \cdot \mathbf{L}$, where the point $\mathbf{x} \in \mathbf{S}$ projects onto the pixel $\mathbf{p} \in \Omega$, $\mathbf{n}(\mathbf{x})$ is a unit normal to $\mathbf{S}$ at $\mathbf{x}$, $\mathbf{L}$ is a unit vector representing the light

direction, and . denotes the scalar product of two vectors.

Note that there are two distinct definitions of $\mathbf{n}$ in our framework. Indeed, we represent $\mathbf{S}$ as the union of a number of triangulated patches and the associated normal maps. The normal maps arise from "photometric stereo", *i.e.* from a set of real photographs. We shall denote by $\mathbf{n}^p$ these "photometric normals". But $\mathbf{n}$ can be computed from the triangulation as well. We denote by $\mathbf{n}^g$ the "geometric normals". The motivation for this distinction is that $\mathbf{n}^p$ is sampled at a higher resolution, typically we have 10 pixels per triangle. Ideally, the brightness equation should be satisfied by both normals, *i.e.* $I(p) = \mathbf{L} \cdot \mathbf{n}^g(\mathbf{x})$ and $I(p) = \mathbf{L} \cdot \mathbf{n}^p(\mathbf{p})$. The first equation should allow the underlying geometry to be recovered, while the second should yield a more precise description of the normals, accounting for details that are smaller than the resolution of the triangulation. We do not solve the second equation alone because photometric normals do not necessarily integrate to a valid surface, and so will not necessarily yield the underlying geometry. Their role is only to describe the small details. That said, the two normals should agree to an extent and the two solutions must be somewhat coupled.

It is convenient to solve both equations in the image grid, and this also makes it more natural to deal with multiple resolutions. We do this by using the depth map z to compute the geometric normals. The photometric normals are readily available on the image grid. It is well known that the shape from shading problem can be ill-posed, depending on the data. In the case of image-based geometric editing, we can expect to encounter situations where no solution exists or multiple solutions exist because a hand-defined shading might be incompatible with any realizable surface or be degenerate. This motivates the use of a variational method to look for the "best" surface, in a sense to be made more precise. In addition, variational methods result in iterative algorithms, making it easy to take advantage of an initial guess provided by the user. This is an essential feature of our method as, in practice, it solves the ill-posedness of the shape from shading. Our contribution consists only of the specific integration of known ideas, so we shall only describe briefly the variational method used and point the reader to previous work for details.

We consider a 2D vector field $\mathbf{u}$ defined on Ω, presumably equal to ∇z, and look for a smooth integrable solution by minimizing

$$\int_\Omega \alpha(I - \mathbf{L} \cdot \mathbf{n}^g(\mathbf{u}))^2 + \beta(\nabla^\perp \cdot \mathbf{u})^2 + \gamma(D\mathbf{u})^2,$$

where $\mathbf{n}^g(\mathbf{u}) = (||\mathbf{u}||^2 + 1)^{-\frac{1}{2}}(-\mathbf{u}, 1)$, $\nabla^\perp = \left(\frac{\partial}{\partial y}, -\frac{\partial}{\partial x}\right)$, $D\mathbf{u}$ is the Jacobian of $\mathbf{u}$ and α, β and γ are scalar weights.

The first term accounts for the fidelity to the shading. The Euclidean norm is used for the sake of simplicity. The second term accounts for the integrability of $\mathbf{u}$ and it is crucial to deal with inconsistent shadings. The last term is a regularization penalty which accounts for the smoothness of the solutions. As explained in [11], this term is known to distort the solution in the ideal case of synthetic data but will select a smooth solution in more realistic cases. See [11] and [12] for further details on deriving the Euler-Lagrange equations of this integral, devising a numerical scheme to minimize it and analyzing its stability. See [11] for details on obtaining z from the almost integrable vector field $\mathbf{u}$.

Similarly, the photometric normals $\mathbf{n}^p$ are computed by minimizing the integral

$$\int_\Omega \mu(I - \mathbf{L} \cdot \mathbf{n}^p)^2 + \nu(D\mathbf{n}^p)^2 + \psi(\mathbf{n}^p - \mathbf{n}^g)^2,$$

under the constraint $||\mathbf{n}^p|| = 1$, where μ, ν and ψ are scalar weights. The first term accounts for the fidelity to the shading, the second term for smoothness and the last term couples $\mathbf{n}^p$ to $\mathbf{n}^g$. Another interpretation of the last term is that it penalizes non-integrable photometric normals since $\mathbf{n}^g$ is the normal vector to an (almost) integrable surface. However, one might want to keep this contribution relatively small to allow sharp variations of the photometric normals at the expense of integrability.

As previously, a stable minimizing numerical scheme may be derived from the corresponding Euler-Lagrange equations. The latter are straightforward for the first and last terms. The calculation for the second term is not obvious because of the constraint on $\mathbf{n}^p$ but it is readily available in a paper by Tang *et al.* [13].

As far as the scalar weights are concerned, they are defined up to a scaling factor for each equation. We have chosen a set of parameters experimentally and all the examples shown in this paper use the following: $\alpha = 1$, $\beta = 1$, $\gamma = 0.1$, $\mu = 1$, $\nu = 1$ and $\psi = 0.01$.

3.4 Applying the Edits

After shape from shading has been applied to transform the edited image into a new shape, the new geometry depth map, normal map and diffuse reflectance maps are used to update the original model. The update proceeds in two steps – updating the underlying charts and then the normal and diffuse reflectance maps.

The process of updating the underlying mesh is illustrated in Fig. 6. Each existing vertex in a changed area is moved along the line-of-sight of the edited image's virtual camera so that it lies on the new depth map computed from shape from shading. The original mesh may have a resolution that is either too dense or too sparse to represent the change in geometry. After changing the position of existing vertices, the surface could be refined or decimated to maintain the same level of consistency with the true geometric surface as was represented in the original model.

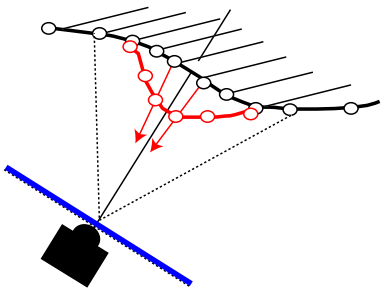

Figure 6. Each mesh vertex is moved along a ray to the camera to update the old surface to new surface.

We need to ensure that the updates produced by the shape from shading solution will not result in a self-intersecting surface. Checking for self-intersections is shown in Fig. 7. For changes in the direction of the virtual camera, as long as the surface stays in front of the camera, no surface self-intersection is possible. For changes in the direction away from the camera, a check is needed against a second depth buffer that contains the depth to the closest back-facing surface.

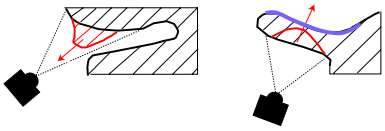

Figure 7. Surface updates in the direction of the camera are along a clear line of sight; updates away from the camera require a depth check.

Edits that result in a valid surface can require a repartitioning of the surface for texture mapping to avoid unacceptable stretching of the texture maps. Figure 8 shows an example. Stretching of the texture map occurs when the maximum angle between surface facets in the chart becomes large. An edit from an arbitrary angle can cause a deformation that greatly increases the maximum angle between chart facets. In these cases the chart is split, as shown in the figure, and a check can be made if the smaller portion of the chart can be joined to one of its neighbors to avoid splintering the model.

The process of updating the diffuse surface reflectance and normal maps is shown in Fig. 9. The patches affected by the edits are identified, and new normals and albedo maps are formed for these patches. As in scanning systems and 3D projection paint systems, the edited region is projected back onto the geometry, and then textures are replaced in the affected areas.

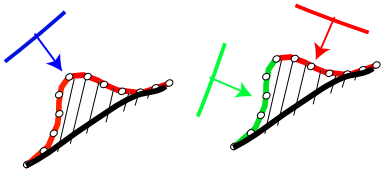

Figure 8. Splitting to avoid texture stretching.

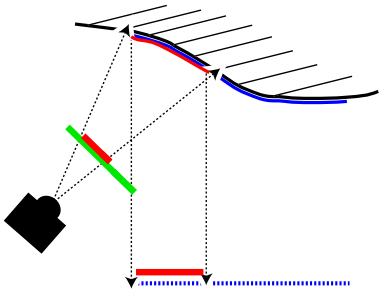

Figure 9. Updating the maps: The updates on the edited image are projected on the geometry and then back into the map associated with that portion of the geometry.

4 Results

We implemented the system described in the previous section in C++ using OpenGL software libraries to perform the rendering to obtain the initial images for editing, and to do the projections required to update the geometry and image maps. All of the editing illustrated in this paper was performed on a computer with a Pentium III 1.2 Ghz processor, 512 Mb of memory and no graphics hardware acceleration.

Figures 10 and 11 show the results for the small edit illustrated in Fig. 3. The full object is composed of 314,246 triangles organized into 92 charts, and maps containing 2.9 million non-zero pixels that specify the diffuse reflectance and normal at a higher spatial resolution (*i.e.* approximately 9 pixels per triangle.) The view to be edited was rendered as a 512 by 512 pixel image, and 15.7% of the pixels (*i.e.* 41,174) were in the edited region. The shape from shading solution took 4 minutes and 45 seconds on the Pentium III processor. Figure 10 shows the affected charts before and after the edit. The figure also shows the updated normal map for two lighting conditions. The top two rows of Fig. 11 show before and after views of the edit in Fig. 3 The original detailed close-up view is shown in the leftmost images, and the center and right images show the full model before (top) and after (second row) with two new views and lighting conditions.

Figure 11 also shows the results for the larger edits il-

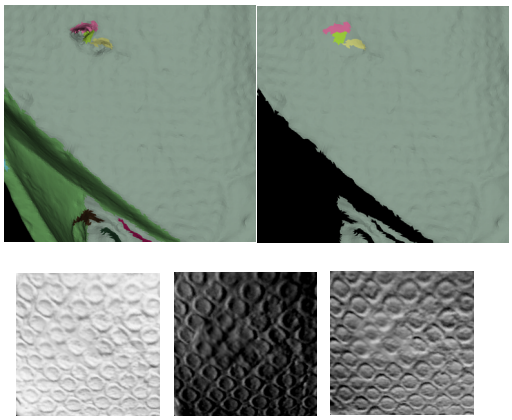

Figure 10. Top row: the affected charts before and after the edit specified in Fig. 3. Bottom row: the normal map for the edited view for various lighting conditions.

lustrated in Fig. 4. The full object is composed of 289,698 triangles organized into 70 charts, and maps containing 2.9 million non-zero pixels representing the diffuse reflectance and normals at a higher resolution. The view to be edited was rendered as a 512 by 769 pixel image and 6% of the pixels were in the edited region. The shape from shading solution was run for each alternative edit for the nose with the same initial conditions and intial guess. In each case the solution required approximately 2 minutes and 49 seconds on the same Pentium III. Before and after views of the edits are shown under different views and lighting conditions in the lower three rows.

5 Conclusion

We have demonstrated an image-based geometric editing system. The user may use any image editor to retouch a picture of a shape, thus specifying a shape alteration. A special shape from shading algorithm is used to update the initial geometric model accordingly. At the same time, the reflectance information may be retouched by image painting. The system is most appropriate for the fine retouching of complex existing models. We have focused on the output of the particular scanning system, but the approach can be applied to the geometric output of any system.

A weakness of the method is that it is limited by the ability of the user to specify a shading that is compatible with a realizable surface. Based on our experiments, we believe that this issue is well addressed by enforcing the integrability and smoothness of the solution and by using a reasonable initial guess. However, the system will not automatically generate a high-quality result from a poorly specified input. Our current implementation of the shape from shading algorithm is slow. A multi-grid approach needs to be used to improve the efficiency.

The strengths of this system are twofold. First, it allows the user to express intent directly, without being exposed to the internal representation or learning a new user interface. Second, the user can comfortably retouch an image rendered with the highest resolution since no 3D rendering is performed during the edit. This feature is particularly valuable in applications where there is a need to edit models of a size that cannot be displayed interactively in 3D.

We would like to acknowledge the Egyptian Center for Cultural and Natural Heritage Documentation for obtaining the scanned models of Akhenaten and the head of a queen used as examples in this paper.

References

[1] M. Levoy et al., "The digital Michelangelo project: 3D scanning of large statues," in *SIGGRAPH 2000*, pp. 131–144.

[2] G. Godin, J.-A. Beraldin, J. Taylor, L. Cournoyer, M. Rioux, S. El-Hakim, R. Baribeau, F. Blais, P. Boulanger, J. Domey, and M. Picard, "Active optical 3D imaging for heritage applications," *IEEE Computer Graphics and Applications*, vol. 22, no. 5, pp. 24–35, 2002.

[3] A. R. Smith, "Digital paint systems: and anecdotal and historical overview," *IEEE Annals of the History of Computing*, vol. 23, no. 2, pp. 4–30, 2001.

[4] P. Hanrahan and P. Haberli, "WYSIWYG painting and texturing on 3D shapes," in *SIGGRAPH 1990*, pp. 215–223.

[5] B. M. Oh, M. Chen, J. Dorsey, and F. Durand, "Image-based modeling and photo editing," in *SIGGRAPH 2001*, pp. 433–442.

[6] S.M. Seitz and K.N. Kutulakos, "Plenoptic image editing," in *6th IEEE Intl. Conf. on Comp. Vision*, 1998, pp. 17–24.

[7] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *SIGGRAPH 1996*, pp. 11–20.

[8] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah, "Shape from shading: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 690–706, 1999.

[9] C. W. A. M. van Overveld, "Painting gradients: Free-form surface design using shading patterns," in *Graphics Interface '96*, 1996, pp. 151–158.

[10] F. Bernardini, H. Rushmeier, I. Martin, J. Mittleman, and G. Taubin, "Building a digital model of Michelangelo's Florentine Pieta'," *IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 59–67, 2002.

[11] B. K. P. Horn and M. J. Brooks, Eds., *Shape from Shading*, MIT Press, Cambridge, MA, 1989.

[12] T. M. Strat, "A numerical method for shape from shading from a single image," M.S. thesis, MIT, 1979.

[13] B. Tang, G. Sapiro, and V. Caselles, "Direction diffusion," in *7th IEEE Intl. Conf. on Comp. Vision*, 1999, pp. 1245–1252.

Figure 11. Top two rows: results of edit shown in Fig. 3; Bottom three rows: results of edits shown in Fig. 4.