Volumetric Illustration: Designing 3D Models with Internal Textures

Shigeru Owada[‡†] Frank Nielsen[†] Makoto Okabe[‡†] Takeo Igarashi[‡§]

[‡]The University of Tokyo [†]Sony CS Laboratories Inc. [§]PRESTO/JST

Abstract
This paper presents an interactive system for designing and browsing volumetric illustrations. Volumetric illustrations are 3D models with internal textures that the user can browse by cutting the models at desired locations. To assign internal textures to a surface mesh, the designer cuts the mesh and provides simple guiding information to specify the correspondence between the cross-section and a reference 2D image. The guiding information is stored with the geometry and used during the synthesis of cross-sectional textures. The key idea is to synthesize a plausible cross-sectional image using a 2D texture-synthesis technique, instead of sampling from a complete 3D RGB volumetric representation directly. This simplifies the design interface and reduces the amount of data, making it possible for non-experts to rapidly design and use volumetric illustrations. We believe that our system can enrich human communications in various domains, such as medicine, biology, and geology.

Keywords: Interactive Techniques, Texture Synthesis, Non-Photorealistic Rendering, Volumetric Modeling

1. Introduction

Surfaces are the most popular geometric representation for 3D models. They are widely used because they are compact, making them easy to construct, transmit, and render. However, the application of a surface representation is limited because it lacks internal information. For example, it is impossible to cut the model and inspect the detailed internal structures.

Volumetric representations have complementary advantages and limitations. Since a volumetric representation stores internal information, the user can cut a model and observe internal structures. However, the amount of data required generally far exceeds that of a surface representation, making storage, transmission, modeling, and rendering much more difficult. Modeling is especially problematic, although the other problems can eventually be mitigated by more memory, faster processors, and networks. Since modeling problems are closely related to the limitations of human perception and manipulation, the design of appropriate user interfaces plays a critical role in addressing them.

The main sources of volume data are the capture of real-world objects and procedural design. However, these are not appropriate for drawing volumetric illustrations quickly for communication purposes. Some interactive methods use 3D input devices (*e.g.*, a Phantom), but they are unsuitable for designing detailed internal textures.

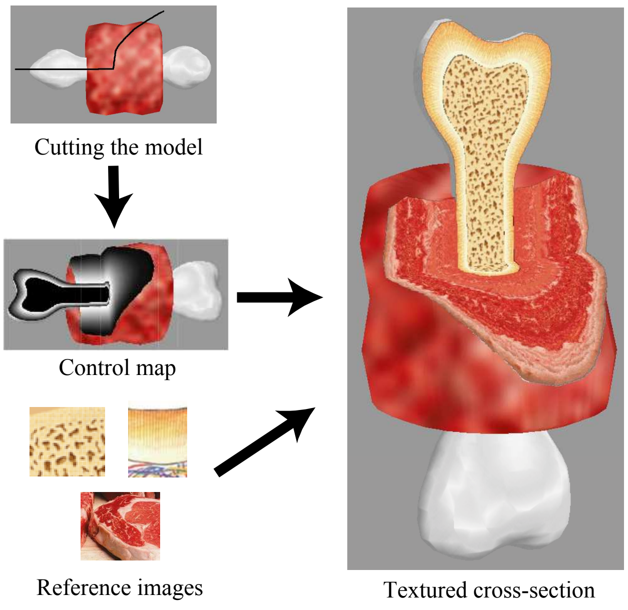


Figure 1 System overview. The user can cut the model anywhere and observe internal textures. Internally, the system first computes a control map using predefined guiding information and then synthesizes the texture using the control map and the reference textures.

Our goal is to develop an interactive designing and browsing system that allows the user to add interesting textures to surface meshes manually by using existing 2D reference images (Figure 1).

We cannot see all of the 3D volumetric information simultaneously; because of occlusion, we can see only one 2D cross-section at a time. For example, illustrations in biology textbooks or scientific magazines often show cross-sections of a volumetric object to explain internal structures (Figure 2). It is also important to note that these illustrations are the result of careful design processes rather than a literal simulation of reality. In Figure 2, for example, nuclei are seen in all the cells in the illustration, while an actual cross-section would contain cells whose nuclei were not visible.

Based on this observation, we propose a new representation for 3D models with internal textures, namely one in which the system synthesizes the internal textures for a cross-section by using 2D reference images instead of maintaining all the 3D volumetric data (Figure 1). To assign internal textures to a model, the designer specifies the correspondence between a geometric cross-section and a reference 2D image by providing guiding information, such as flow orientation. This approach significantly reduces the amount of data that the model requires. It also allows designers to add 3D internal texture to a model without specifying each voxel manually.

Our technical contributions are the interfaces that are used to assign internal textures to a given surface mesh and the algorithms that synthesize textures on a cross-section. On a larger scale, our contribution is a modeling structure in which the specification and viewing of simple volumetrically textured models is easy and convenient, allowing non-experts to create volumetric illustrations rapidly.

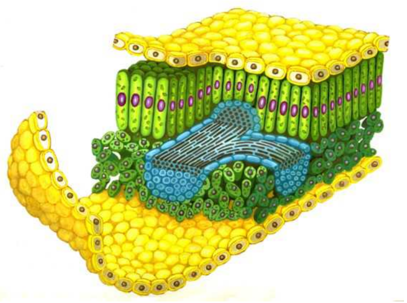


Figure 2 An example of an illustration that reveals internal structures (courtesy of Saeko Sato).

2. Related work

Volumetric modeling. One popular approach to designing volume data is to use procedural methods [Perlin 1985; Kniss *et al*. 2002], which enable the user to design various 3D structures, such as marble, clouds, and fire, using relatively simple programs. Cutler *et al*. [2002] proposed a scripting language for volumetric modeling. However, it is difficult for most people to obtain a desired texture by programming or adjusting parameters.

Another popular source of volumetric data is the capture of existing real objects using special-purpose instruments, such as computed tomography (CT) scanners or cameras to take pictures of the slices [Banvard 2002]. However, the user can capture only existing objects. It is often desirable to design appropriate representations for communication purposes manually.

There are systems that use 3D pointing devices [Galyean and Hughes 1991; Ferley *et al*. 2000] or standard 2D devices [Wang and Kaufman 1995] to create 3D volumes. However, they are designed mainly to convey the overall shape of a model and it is still difficult to design the detailed internal textures of 3D volumes.

Texture synthesis. Texture synthesis algorithms take reference images as input and synthesize new images that appear similar. Texture synthesis algorithms can be categorized into four types: frequency domain, pixel-based, patch-based, and non-periodic tiling. Early systems used frequency domain techniques [Heeger and Bergen 1995], but they can handle only specific types of texture. The pixel-based approach [Efros and Leung 1999; Wei and Levoy 2000] uses a simpler strategy, and various extensions have been developed [Ashikhmin 2001; Hertzmann *et al*. 2001]. The patch-based [Efros and Freeman 2001; Kwatra *et al*. 2003] and non-periodic sampling [Stam 1997; Cohen *et al*. 2003] approaches can generate high-quality images quickly.

Some texture-synthesis techniques have been extended to 3D textures. Heeger and Bergen [1995] extended their frequency domain methods to 3D textures, and Wei [2001] used a pixel-based technique to generate 3D volumes from 2D references. Other extensions synthesize textures on the surface of 3D models; Turk [2001] used a pixel-based technique and Praun *et al*. [2000] used a patch-based technique.

Non-photorealistic modeling and rendering. Our system draws inspiration from various non-photorealistic rendering (NPR) systems that focus on the communication of particular information rather than the simulation of light transport [Gooch and Gooch 2001]. Our system is particularly influenced by the stylized rendering of 3D models that synthesize interesting 2D pictures by adding details to simple 3D geometries on the fly [Lake *et al*. 2000; Kalnins *et al*. 2002]. In a similar spirit, we synthesize detailed textures on cross-sections of simple surface models.

Some systems also address the problem of authoring. Hertzmann *et al*. [2001] introduced a painting interface for directing the texture-synthesis process for various artistic expressions; Kalnins *et al*. [2002] proposed an interface for painting a 3D model to specify rendering styles directly. Like these systems, our system provides a tailored user interface for intuitively designing volumetric illustrations.

3. User interface

3.1 Browsing interface

The system comprises two functions: browsing and modeling. The browsing interface is a subset of the modeling interface. The browsing interface is a standard 3D model viewer with an extension that allows inspection of internal textures using a cut operation. Rotating and translating the model are assigned to the right mouse button, and cutting the model is assigned to the left mouse button. If the right button is pressed on the model, the model rotates. If the right button is pressed elsewhere, the model translates parallel to the screen.

The user can cut the model by drawing a freeform stroke that crosses the model on the screen [Igarashi *et al*. 1999] (Figure 3). The cut object then opens automatically with animation [Owada *et al*. 2003] and the user can see the internal textures on the cross-section. The model closes when the user clicks an empty space.

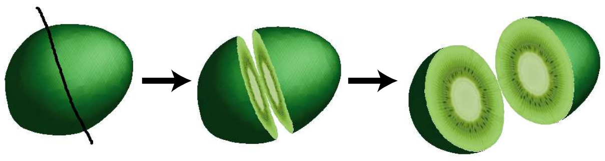


Figure 3 Inspecting internal structures using a freeform cut operation.

Although our current implementation uses a cut operation that is based on a freeform stroke, the framework can easily be extended to support other interfaces for specifying cross-sections, such as 3D magic lenses [Viega *et al*. 1996] and two-handed operations using a prop and a plate [Hinckley *et al*. 1994].

3.2 Modeling interface

The modeling operation starts by loading a predefined surface mesh (*i.e*., a surface mesh that delimits volumetric regions) and predefined 2D images. The goal of the modeling operations is to specify how given images are to be mapped to the interior of the given surface mesh, while existing methods usually specify textures on surfaces [Hanrahan and Haeberli 1990; Pederson 1996]. The modeling process has the following steps.

1. Cut the target 3D model and specify a 3D region to be textured by clicking on the cross-section.
2. Choose one of the three texture types to use for the region.

3. Import a reference 2D image.
4. Establish the correspondence between the 2D reference image and the cross-section of the 3D model by providing the necessary guidance information.
5. Repeat steps 1-4 for each 3D region.

We explain each step in turn.

3.2.1 Specifying a region to be filled

First, the user cuts the target surface mesh using the freeform cut operation. The cross-section reveals the internal structure of the model and can be divided into several closed regions. For example, a model of an egg might consist of a spherical surface mesh representing the yolk enclosed by a larger sphere representing the egg white. In this case, the cross-section has two regions. The user can simply click on the target region on the cross-section to specify the 3D region to be textured.

3.2.2 Selecting a texture type

When the user clicks on the target region on the cross-section, the system opens a dialog box that is used to specify the texture type (Figure 4a). Once the user has specified the texture type, the system opens a separate pane that shows the reference image and a reference cube (Figure 4b). The reference cube is an intermediate representation that visualizes the relationship between the 2D reference image and the 3D region. Steps 3 and 4 differ slightly for each texture type. Therefore, we first introduce the three texture types and then explain the steps for each.

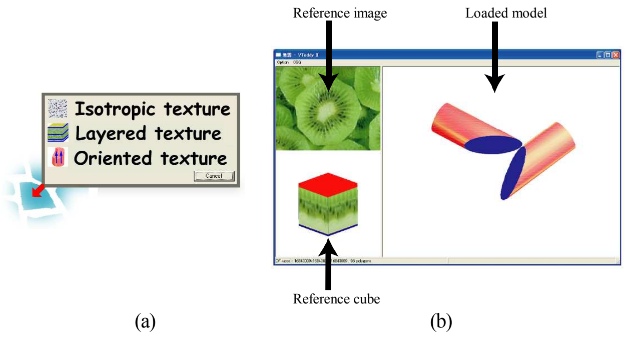


Figure 4 Window layout of the system for assigning textures to a model.

The current system supports three types of textures: isotropic, layered, and oriented (Figure 5). Isotropic textures have a uniform distribution in the 3D space with no dependency on position or orientation. All of the cross-sections of an isotropic texture look similar, regardless of their location or orientation. Examples include a sausage, a sponge, and any other material that consists of isotropic elements. Layered textures have varying appearances according to their position in the axial or radial direction. Examples include kiwi fruits, human skin, tree trunks, and leaves. Layered textures require depth information for the target 3D region. Finally, oriented textures are defined by both a reference image and a flow direction; the appearance of an oriented texture depends on the orientation of the cut-plane relative to the flow-direction (Figure 5c). Examples include muscle, plant stems (of monocotyledonous plants), and any other material that consists of bundled long fibers. The oriented texture requires that the flow orientation in the target 3D region be specified.

We do not claim that these three types cover all possible real-world textures. Some textures can be combinations of layered and oriented textures, and some have more complicated structures. We support these three types in the current implementation because they are relatively easy to understand, they can be specified with a simple interface, and they cover a wide range of interesting textures that are commonly seen in organic materials. Future work will investigate other types of texture filling.

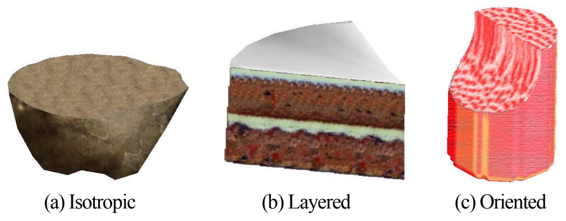


Figure 5 Examples of texture types

3.2.3 Isotropic textures

The user imports a reference image by dragging an image file and dropping it onto the source image area in the main window. The user can choose a specific region of the reference image by rubber banding. The selected region is immediately transferred to all faces of the reference cube and to the cross-section of the surface mesh using a texture-synthesis technique. No guidance information is required in this case (Figure 6).

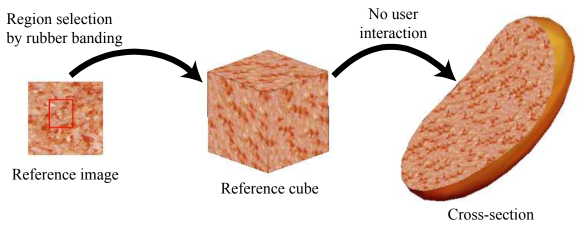


Figure 6 Using an isotropic texture. The user specifies a region to use by rectangular rubber banding, and it is then transferred to all faces of the reference cube and the cross-section.

3.2.4 Layered textures

A layered texture requires additional guidance information in the surface mesh to specify the mapping between the image and the model. The user first draws two freeform strokes that correspond to the upper and lower bounds of the layer on the imported reference image (Figure 7, left). The first and last points of the two strokes are connected by straight lines to carve out a portion of the input image. Then, texture-synthesis techniques fill the side faces of the reference cube using the portion of the image as a reference (Figure 7, middle). The user then specifies two corresponding upper and lower bounds in the surface mesh by clicking a boundary or drawing a stroke on the cross-section (Figure 7, right). Clicking on the boundary selects the associated surface region and the stroke becomes the constraint on the cross-section (Figure 8).

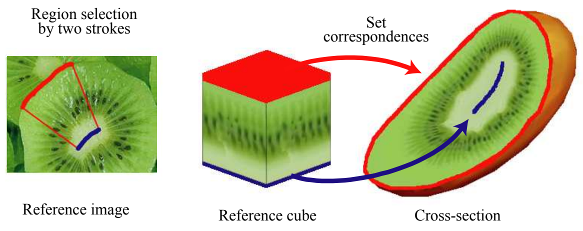


Figure 7 Using a layered texture. The red and blue marks on the reference image define the upper and lower bounds for the algorithm. The texture-synthesis techniques fill the side faces of the reference cube, using the image between the two marks as a reference. Similarly, the red and blue marks on the surface mesh define the upper and lower bounds. The texture on the cross-section is synthesized using the reference image as an example.

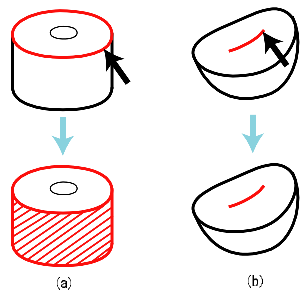


Figure 8 Clicking a boundary on the cross-section selects the associated surface region as the upper or lower bound (a). We assume that the user has predefined the correspondence between a boundary and a surface region (provided as a surface mesh). A stroke drawn in the interior of a cross-section becomes a bound on the cross-section (b).

3.2.5 Oriented textures

An oriented texture has distinct appearances in cross-sections that are perpendicular and parallel to the flow orientation. Our current implementation asks the user to provide a reference image for the cross-sections perpendicular to the flow orientation. The reference image must be isotropic. The user specifies a rectangular region in the reference image using rubber banding. The system then synthesizes the top face of the reference cube using the selected region as a reference. Then, it generates a reference volume by sweeping the image vertically and shows textures for the cross-sections parallel to the flow orientation on the side faces of the reference cube.

We experimented with other strategies for specifying the reference volume. One let the user specify the images for the side faces of the volume, and the other let the user specify the images on both the top and side faces. We used Wei's [2001] volumetric texture-synthesis technique to synthesize the reference volume in these cases. We did not pursue this direction further in our current implementation because it was too slow and the quality of the resulting volume was unsatisfactory. However, it is sometimes desirable to specify the appearance of side faces, and in the future we will investigate efficient supporting strategies.

An oriented texture requires the user to specify the flow field across the target region as guidance information. This is done by drawing short arrows that represent local flow directions on the cross-section and surface of the region (Figure 9, right) [Turk 2001]. Our current implementation does not allow the user to draw arrows that are not parallel to the cross-section. Therefore, the user should cut the model parallel to the desired flow orientation.

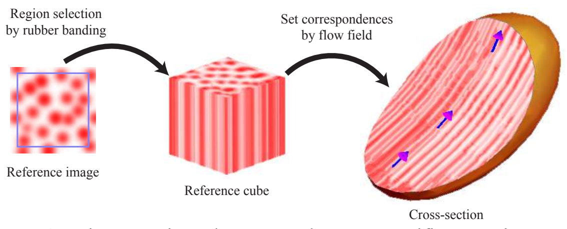


Figure 9 Using an oriented texture. The user specifies a region to use by rectangular rubber banding, and it is then transferred to the top face of the reference cube. The system generates the side faces by sweeping the image on the top face vertically. The user draws short arrows on the cross-section to specify the flow orientation.

4. Algorithms

This section describes the algorithms and implementation details for synthesizing a texture for a given cross-section using the reference images and the associated guidance information. The cutting operation cuts the model by sweeping a user-drawn 2D stroke in the direction perpendicular to the screen (we use orthogonal projection). Using this curved plane, a model is divided by CSG operations [Hoffman 1989]. The parametrization of the cross-section is given as follows: the y-axis is defined along the cutting stroke and the x-axis is defined along the sweeping direction. The starting point of the stroke becomes y=0 and the corner of the model's bounding box nearest the screen becomes x=0. The imported surface mesh is scaled so that the size of the bounding box equals 1, and the pixel size of the synthesized texture is 1/150~1/400 in our current implementation. A cross-sectional bitmap is obtained by synthesizing the pixel color at each grid point on this parameterized cross-section within a rectangular region that covers the model's bounding box. If the user wants to change the scale of the synthesized texture, the original image must be scaled beforehand. We describe our texture-synthesis algorithms for each of the three texture types.

4.1 Isotropic textures

An isotropic texture has no dependency on position or direction. Therefore, we simply use a standard 2D texture-synthesis algorithm to construct a 2D texture image for the cross-section [Wei and Levoy 2000; Cohen *et al.* 2003; Kwatra *et al.* 2003]. The system uses the selected region in the original reference image as the reference for the synthesis directly. The reference cube exists only to give feedback to the user. Note that there is no guarantee of obtaining exactly the same image when a model is cut twice at the same cross-section. However, since our aim is to convey a volumetric impression rather than to generate consistent volumetric data, we believe that this simple approach is sufficient.

4.2 Layered textures

A layered texture has an appearance that varies according to the depth. Therefore, the algorithm needs depth information for each pixel in the reference image and target cross-section. Figure 10 illustrates the overall process. The system generates a reference control map for the reference image and a target control map for the cross-section. A control map is a grayscale 2D image in which the floating-point pixel values indicate associated depth values.

The grayscale values of the reference control map represent a smooth 2D depth field constrained by the two bounds provided in the reference image. The mark for the upper bound (red curve in Figure 10) is associated with depth value 0 and that for the lower bound (blue curve in Figure 10) is associated with depth value 1. We use a 2D thin-plate interpolation technique [Turk and O'Brien 1999] to compute this smooth 2D depth field (reference control map in Figure 10). The depth field is given as a continuous function that returns a scalar value for a given 2D position. This function is sampled on each pixel location on the reference control map, which has the same resolution as the reference image.

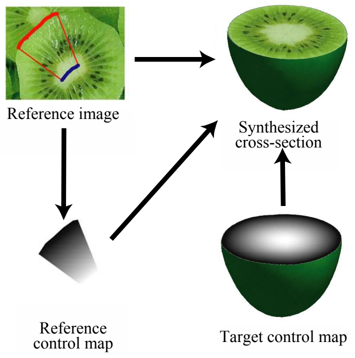


Figure 10 Overview of texture synthesis for layered textures. The system synthesizes the texture bitmap for a cross-section by using the reference image, the reference control map associated with the reference image, and the target control map associated with the target texture.

To construct the target control map, the system first computes a 3D scalar field using the user-defined upper and lower bound 3D geometries as constraints. The upper-bound geometry (a surface region or a line in 3D space) is associated with depth value 0 and the lower-bound geometry is associated with depth value 1. Again, we use Turk and O'Brien's [1999] 3D thin-plate interpolation technique to construct this smooth 3D scalar field. When the user cuts the model, the system generates a 2D target control map by sampling the aforementioned 3D depth field on the cross-section.

Given the reference image, reference control map, and target control map, it is now possible to start the texture synthesis process using a pixel-based technique. This synthesis process is similar to field distortion synthesis, which is used for texture synthesis on surfaces [Turk 2001; Zhang *et al.* 2003]. The differences are as follows (also see Figure 11):

1. An orientation field is computed from the target control map as the gradient direction in the target control map.
2. Each pixel in the reference image also has an orientation computed as the gradient direction in the reference control map. The neighboring structures are computed according to this orientation.
3. The order of synthesis is determined using the depth value of pixels.
4. When synthesizing a pixel in the target image, the search space in the reference image is restricted by the depth value of the pixel being synthesized; pixels whose depth value equals that of the synthesizing pixel are subject to the search. In our implementation, real-valued depth values are discretized into 32 levels and the pixels are indexed according to the discretized scalar level during preprocessing.

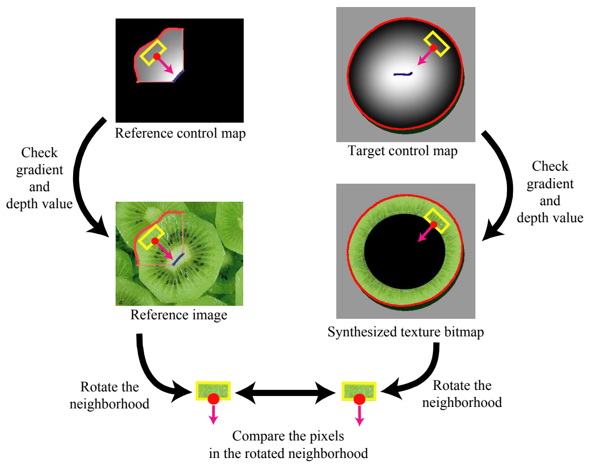

Reference control map
Target control map
Check gradient and depth value
Check gradient and depth value
Reference image
Synthesized texture bitmap
Rotate the neighborhood
Rotate the neighborhood
Compare the pixels in the rotated neighborhood

Figure 11 Computing the pixel color in the synthesized texture. First, the system rotates the neighborhood of the target pixel so that the gradient of the target control map matches that of the reference control map. Then, the system compares the rotated neighborhood of the target pixel with that of the reference image around a pixel whose gray-scale value in the reference control map is identical to the gray-scale value in the target control map.

4.3 Oriented textures

The synthesis process for an oriented texture requires a 3D reference volume and a flow field defined in the 3D region in the surface mesh. As already discussed, the 3D reference volume is obtained by sweeping the top face of the reference cube vertically. The top face is synthesized using the selected region in the reference image as the example. The reference volume is oriented vertically (each pixel in the reference volume is associated with a vertical flow vector) and the size of the reference volume is 64×64×64.

The construction of the 3D flow field again uses the thin-plate interpolation technique [Turk and O'Brien 1999], employing user-defined arrows as constraints. A smooth scalar-valued interpolation function is constructed for each xyz component. The flow field is constructed by combining them and normalizing the resulting flow vectors. This approach can produce singular points where no flow is defined. However, singular points rarely appear on the cross-section in our system because the user cuts the model using a freeform stroke. When singular points do appear, we assign a random orientation to the pixel. After obtaining the 3D flow field, the system generates a 2D target control map for the given cross-section (each pixel is associated with a flow orientation) by sampling the flow vectors along the cross-section.

At this point, we have the 3D reference volume (associated with vertical flow orientation) and the 2D control map for the cross-section that contains the flow vectors for each pixel. Given this information, the system computes the color for each pixel in the cross-section by finding a pixel in the reference volume that has a similar neighborhood [Wei and Levoy 2000]. The similarity of neighbors is computed as follows. The neighborhood of the pixel on the cross-section is approximated by a small flat rectangle. For each pixel in the reference volume, the system samples the neighborhood in a corresponding small flat rectangle whose slant angle (angle between the rectangle and the vertical flow vector) equals that of the rectangle on the cross-section (Figure 12). Rotation about the flow vector does not matter, because we assume that the reference volume has an isotropic structure on cross-sections perpendicular to the flow orientation.

Given the rectangle on the cross-section and that in the reference volume, the system can now compare the similarity between the two.

Ideally, the system should sample every possible small rectangle in the entire reference volume. For performance reasons, however, the system samples pixels in a slanted 2D square region at the center of the reference volume and uses the resulting image as a reference for standard 2D texture synthesis. In our current implementation, the size of the square is 45×45 (to fit within the reference volume completely). We further reduce the computation time by caching the sampled image using discretized slant angles as keys (a discretization step is $\pi/32$).

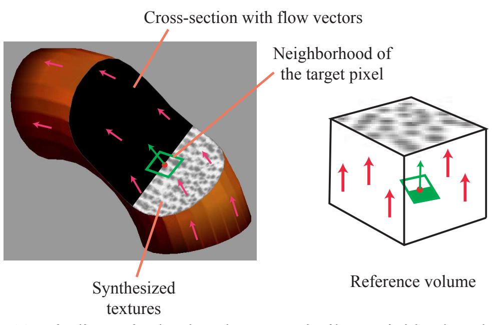
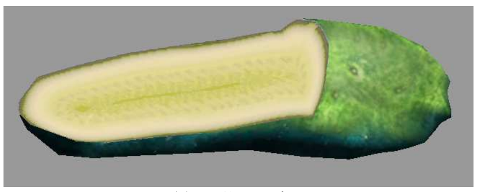
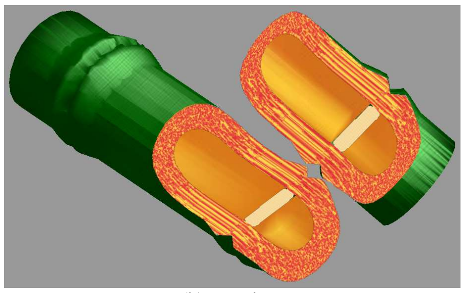

Cross-section with flow vectors
Neighborhood of the target pixel
Synthesized textures
Reference volume

Figure 12 Finding pixels that have a similar neighborhood in the reference volume for each pixel in the cross-section.

5. Results

Figure 13 shows some volumetric illustrations that were created using our system. The amount of data in the models, time for design, and time for synthesizing the cross-section are summarized in Table 1. For isotropic and oriented textures, we used a three-level multi-resolution pyramid with a 3×3 square neighbor for lower resolution and a 5×5 L-shaped neighbor for higher resolution. For layered textures, we used a 3×3 square neighbor for lower resolution and a 5×3 rectangular neighbor for higher resolution. We used a laptop computer with a Pentium M 1.6-GHz processor and 1 GB of RAM. Although in some cases it took more than 10 seconds to obtain the result for the finest resolution, this did not impede the interactive modeling process because progressive synthesis of cross-section images frees the user from waiting for the final result each time. Since the resolution of the cross-section is approximately 300×300, the quality is comparable to that of 300^3 colored voxels, which would require approximately 80 MB of storage.


(a) Cucumber


(b) Bamboo

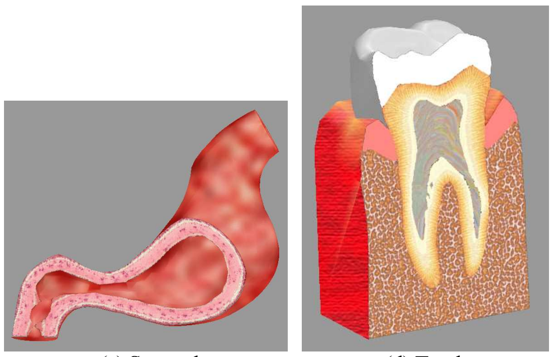

(c) Stomach (d) Tooth

Figure 13 Results

Title	Amount of data (without compression)	Modeling time (excluding mesh editing)	Synthesis time
Meat (Fig. 1)	622 kb	90 sec	22 sec
Cucumber	53 kb	40 sec	4 sec
Bamboo	291 kb	30 sec	14 sec
Stomach	402 kb	15 sec	18 sec
Tooth	307 kb	120 sec	32 sec

Table 1 Statistics for the example models

6. Conclusions and future work

We described a modeling and browsing system that adds internal textures to a surface mesh. The user provides 2D reference images and a surface mesh with simple guidance information that specifies the correspondences between them. When the user cuts the model, the system synthesizes cross-sectional images using 2D texture-synthesis techniques. This system would be useful for conveying volumetric information, such as between a teacher and students, a doctor and a patient, or a virtual-reality content-provider and consumers. Although the lack of real volumetric data makes some applications impossible, such as translucent rendering or volumetric simulation, our lightweight data representation may well be useful in many applications.

Nevertheless, our system has several limitations. One is the computational cost. A layered texture rotates the neighbors so that the gradient of the target control map matches that of the reference control map, while an oriented texture generates a 2D reference image by slicing the reference volume at an appropriate angle for each pixel. These processes require more computation than standard texture synthesis. The quality of the synthesized image also requires improvement, in part because of the cascading resampling and resulting distortion. In future work, we will improve both the performance and quality of the overall process.

One interesting avenue for future research is to enhance images of the cross-section by using the information that is embedded in the pixels of the reference map. We are currently exploring text annotation and displacement mapping. The pixels of the reference image are associated with annotations and displacements, and the system adds them to the corresponding pixels on the cross-section. Text annotation allows designers to add textual explanations to the internal material of 3D models, which would be useful for educational and communication applications. Displacement mapping can add realism to a cross-section, as cross-sections of real objects cannot be perfectly flat. Adding such additional information to 3D regions directly can be very difficult, but is straightforward in our framework.

We are also interested in designing other interaction techniques to explore the internal textures of 3D models, in addition to the current freeform cutting. As McGuffin *et al*. [2003] suggested, active interaction can facilitate the understanding of volumetric structures as compared to passive browsing of static images.

Acknowledgments

We thank Professor John F. Hughes, Professor Issei Fujishiro, and Dr. Kazuo Nakazawa for their detailed comments and advice based on their experience and deep understanding. Thanks also to Ms. Katrina Avery and Textcheck service for their quick and accurate proofreading. This work was funded in part by grants from IPA (Information-Technology Promotion Agency, Japan).

References

ASHIKHMIN, M. 2001. *Synthesizing Natural Textures*. In *Proceedings of the Symposium on Interactive 3D Graphics 2001,* 217-226.

BANVARD, R.A. 2002. *The Visible Human Project® Image Data Set From Inception to Completion and Beyond,* In *Proceedings CODATA 2002: Frontiers of Scientific and Technical Data* , Track I-D-2: Medical and Health Data, Montréal, Canada, October, 2002.

COHEN, M.F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. *Wang Tiles for Image and Texture Generation. ACM Transactions on Graphics (Proc. Siggraph 2003),* 287-294.

CUTLER, B., DORSEY, J., MCMILLAN, L., MULLER, M., AND JAGNOW, R. 2002, *A Procedural Approach to Authoring Solid Models. ACM Transactions on Graphics (Proc. Siggraph 2002)*, 302-311

EFROS, A.A., AND FREEMAN, W. T. 2001. *Image Quilting for Texture Synthesis and Transfer*. In *Proceedings of Siggraph 2001*, 341-346.

EFROS, A.A., AND LEUNG, T. 1999. *Texture Synthesis by Non-parametric Sampling*. In *Proceedings of International Conference on Computer Vision (ICCV '99)*, 1033-1038.

FERLEY, E., CANI, M.P., AND GASUEL, J.D. 2000. *Practical Volumetric Sculpting*. The Visual Computer, 16(8), (2000) 469-480.

GALYEAN, T.A., AND HUGHES, J.F. 1991. *Sculpting: An Interactive Volumetric Modeling Technique*. In *Proceedings of Siggraph 1991*, 109-116.

GOOCH, B., AND GOOCH, A. 2001. *Non-Photorealistic Rendering*. A. K. Peters.

HANRAHAN, P., AND HAEBERLI, P. 1990. *Direct WYSIWYG Painting and Texturing on 3D Shapes*. In *Proceedings of Siggraph 1990*, 215-223.

HEEGER, D. J., AND BERGEN, J.R. 1995. *Pyramid-based texture analysis and synthesis*. In *Proceedings of Siggraph 1995*, 229-238.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. *Image Analogies*. In *Proceedings of Siggraph 2001*, 327-340.

HINCKLEY, K., PAUSCH, R., GOBLE, J., AND KASSELL, N. 1994. *Passive Real-World Interface Props for Neurosurgical Visualization*. In *Proceedings of CHI '94*, 452-458.

HOFFMAN, C.K. 1989. *Geometric and Solid Modeling*. Morgan Kaufmann Pub.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. *Teddy: A Sketching Interface for 3D Freeform Design*. In *Proceedings of Siggraph 1999*, 409-416.

KALNINS, R. D., MARKOSIAN, L., MEIER, B. J., KOWALSKI, M. A., LEE, J. C., DAVIDSON, P. L., WEBB, M., HUGHES, J. F., AND FINKELSTEIN, A. 2002. *WYSIWYG NPR: drawing strokes directly on 3D models. ACM Transactions on Graphics (Proc. Siggraph 2002)*, 755-762.

KNISS, J., PREMOŽE, S., HANSEN, C., AND EBERT, D. 2002. *Interactive Translucent Volume Rendering and Procedural Modeling*. In *Proceedings of Visualization 2002*, IEEE Computer Society, 109-116.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2002. *Graphcut Textures: Image and Video Synthesis Using Graph Cuts. ACM Transactions on Graphics (Proc. Siggraph 2003)*, 277-286.

LAKE, A., MARSHALL, C., HARRIS, M., AND BLACKSTEIN, M. 2000. *Stylized Rendering Techniques for Scalable Real-Time 3D Animation*. In *Proceedings of NPAR*, 13-20.

MCGUFFIN, M. J., TANCAU, L., AND BALAKRISHNAN, R. 2003. *Using Deformations for Browsing Volumetric Data*. In *Proceedings of Visualization 2003*, 401-408.

OWADA, S., NIELSEN, F., NAKAZAWA, K., AND IGARASHI, T. 2003. *A Sketching Interface for Modeling the Internal Structures of 3D Shapes*. Lecture Notes in Computer Science (LNCS 2733; *Smart Graphics 2003*), Springer-Verlag, 49-57

PEDERSON, H.K. 1996. *A Framework for Interactive Texturing on Curved Surfaces*. In *Proceedings of Siggraph 1996*, 295-302.

PERLIN, K. 1985. *An Image Synthesizer*. In *Proceedings of Siggraph 1985*, 287-296.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. *Lapped Textures*. In *Proceedings of Siggraph 2000*, 465-470.

STAM, J. 1997. *Aperiodic Texture Mapping*. Technical Report R046, European Research Consortium for Informatics and Mathematics (ERCIM).

TURK, G., AND O'BRIEN, J. F. 1999. *Variational Implicit Surfaces*. Technical Report GIT-GVU-99-15, Graphics, Visualization, and Usability Center. Georgia Institute of Technology.

TURK, G. 2001. *Texture Synthesis on Surfaces*. In *Proceedings of Siggraph 2001*, 347-354.

VIEGA, J., CONWAY, M.J., WILLIAMS, G., AND PAUSCH, R. 2002. *3D Magic Lenses*. In *Proceedings of UIST '96*, 51-58.

WANG, S.W., AND KAUFMAN, A.E. 1995. *Volume Sculpting*. In *Proceedings of Symposium on Interactive 3D Graphics 1995*, 151-156

WEI, L-Y., AND LEVOY, M. 2000. *Fast Texture Synthesis using Tree-structured Vector Quantization*. In *Proceedings of Siggraph 2000*, 479-488.

WEI, L-Y. 2001. *Texture Synthesis by Fixed Neighborhood Searching*. Ph.D. Thesis. Stanford University.

ZHANG, J., ZHOU, K., VELHO, L., GUO, B., AND SHUM, H-Y. 2003. *Synthesis of Progressively-Variant Textures on Arbitrary Surfaces. ACM Transactions on Graphics (Proc. Siggraph 2003)*, 295-302.