Volumetric Modeling with Diffusion Surfaces

Kenshi Takayama
The University of Tokyo

Olga Sorkine
New York University

Andrew Nealen
Rutgers University

Takeo Igarashi
The University of Tokyo / JST ERATO

Abstract

The modeling of volumetric objects is still a difficult problem. Solid texture synthesis methods enable the design of volumes with homogeneous textures, but global features such as smoothly varying colors seen in vegetables and fruits are difficult to model. In this paper, we propose a representation called *diffusion surfaces* (DSs) to enable modeling such objects. DSs consist of 3D surfaces with colors defined on both sides, such that the interior colors in the volume are obtained by diffusing colors from nearby surfaces. A straightforward way to compute color diffusion is to solve a volumetric Poisson equation with the colors of the DSs as boundary conditions, but it requires expensive volumetric meshing which is not appropriate for interactive modeling. We therefore propose to interpolate colors only locally at user-defined cross-sections using a modified version of the positive mean value coordinates algorithm to avoid volumetric meshing. DSs are generally applicable to model many different kinds of objects with internal structures. As a case study, we present a simple sketch-based interface for modeling objects with rotational symmetries that can also generate random variations of models. We demonstrate the effectiveness of our approach through various DSs models with simple non-photorealistic rendering techniques enabled by DSs.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: volumetric modeling, sketching interface, color diffusion.

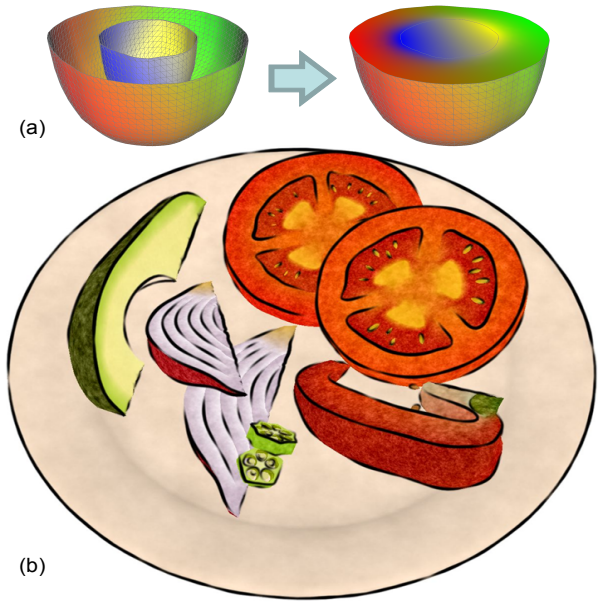

Figure 1: *(a) Diffusion surfaces, a set of 3D surfaces with colors assigned to both sides, define smooth color transitions inside of a volume. (b) A vegetable salad created using our symmetry-aware modeling interface. Simple NPR techniques are employed as a post process for artistic impression.*

1 Introduction

Cutting 3D virtual objects can enrich diverse graphics applications, such as visualization, education, games, and films [Taito 2006; Pixar 2007]. However, although many approaches have been proposed in the literature, volumetric modeling of objects that contain distinct global structures with smooth and sharp color transitions still remains a challenge. In this paper we introduce a volumetric representation called *diffusion surfaces* (DSs) to enable modeling such objects. DSs are, conceptually, an extension of diffusion curves [Orzan et al. 2008] (DCs) to 3D volumes. This representation consists of a set of colored surfaces in 3D, describing global structures of the model's volumetric color distribution. A smooth volumetric color distribution that fills the model is obtained by diffusing colors from these surfaces (Fig. 1a).

In DCs, color diffusion is computed by solving a Poisson equation over a 2D pixel grid, where the colors specified along the curves serve as Dirichlet boundary conditions. Directly extending this method to 3D volumes is costly both in terms of computation time and memory consumption: to achieve high-quality results, high-resolution voxel grids or tetrahedral meshes are required (especially in order to faithfully represent complex volumetric models), and precomputing and storing the resulting color information is expensive. Instead, we propose to interpolate colors only locally at cross-sectional locations using a modified version of the positive mean value coordinates (PMVC) algorithm [Lipman et al. 2007], enabling us to generate high-resolution cross-sections efficiently. Our method produces *consistent* volumetric color distributions comparable to Poisson solutions, and at the same time saves the effort of computing the entire color volume at once, eliminating the necessity of precomputation. The resulting framework is suitable for trial-and-error design processes where volumetric structures can be interactively added, removed, edited, and explored, which is indispensable for creative interactive modeling.

The process of modeling the shapes and colors of the DSs is independent of the representation, such that any general-purpose modeling tools can be used to create DSs for arbitrary objects, e.g., anatomical structures and organs, geological structures, fruits or vegetables. However, modeling and spatially positioning the required number of intricate and nested geometric components using existing tools is often difficult and time-consuming. Thus, we leverage the fact that many natural objects exhibit rotational symmetries in their internal structure, and design a sketch-based interface tailored for this purpose. Fruits, vegetables, and other familiar objects fall into this category of volumetric structures. Our interface assumes and exploits rotational symmetries, which greatly simplifies the modeling process. The interface also enables generating random variations of models, which is particularly useful for scenes containing a large variety of similar objects.

2 Related Work

Solid texture synthesis [Perlin 1985; Dong et al. 2008] is one of the most common approaches to modeling volumetric objects. However, its main focus is the synthesis of solid textures of single homogeneous materials such as marble, wood, or piles of stones. On the other hand, material properties in many objects vary spatially (e.g., fleshy and juicy parts in tomatoes), so this approach alone is not sufficient for such objects.

Cutler et al. [2002] proposed a procedural approach in which the user specifies material information at certain layer depths for given 3D surfaces using a scripting language. Their method is limited to simple layered structures, and it is difficult to create more complex structures that are not layered. Additionally, we believe that writing a script is not necessarily an intuitive modeling interface.

The illusion of volumetric models can be created by generating plausible cross-sectional images based on user-specified associations between 2D photographs and cross-sections of 3D models [Owada et al. 2004; Pietroni et al. 2007]. This approach offers intuitive modeling interfaces, but the lack of consistent 3D structures often leads to visible artifacts such as at an intersection of two cross-sections.

Takayama et al. [2008] manually created solid texture exemplars for several objects (e.g., carrots and trees), and proposed a method to paste patches of exemplars over tetrahedral meshes while aligning their orientation to user-specified volumetric tensor fields. Their method does not work well for objects with distinct global structures (such as inner chambers in tomatoes). Moreover, it is often difficult to design a single representative solid texture exemplar that fully describes the volumetric structure of an entire object.

Owada et al. [2008] proposed an interface for directly painting voxels by distributing particles filled with constant color along guide surfaces. Unfortunately, their method ignores smooth color transitions, which are often essential for plausible appearance.

Wang et al. [2010] recently proposed the first attempt at vector representation for solid textures. Their method can convert a bitmap solid texture to a vector solid texture that is resolution independent, compact, and fast to evaluate. However, their method is limited to features that are boundaries of separate closed regions, and features have to be larger than the grid cell size since they are defined implicitly by the signed distance field. In addition, they did not deal with the creation of volumetric structures from scratch.

3 Diffusion Surfaces

In the following, we describe our basic primitive for volumetric modeling, called a *diffusion surface* (DS), and describe an efficient algorithm for computing and rendering cross-sections of volumetric objects represented by DSs.

3.1 Definition

DSs extend diffusion curves (DCs) to 3D volumes by replacing 2D curves with 3D surfaces. A DS is a triangle mesh in which each mesh vertex has the following attributes: colors on its front and back sides and a blur value. A DS diffuses its colors on either side, and the softness of the color transitions between the front and the back sides of the surface is controlled by the blur values (Fig. 2).

Unlike DCs in which the blur is applied after diffusion as a post process, the blur in DSs is achieved by duplicating each DS into two surface sheets (one for the front and one for the back side), and moving the sheets apart along the surface normal directions by the distances explicitly specified as the blur values (Fig. 2b). This is because DSs are not a voxel-based representation, and the post-process blur of DCs cannot be applied to DSs. Our blur may be less smooth, but we found it sufficiently smooth in practice.

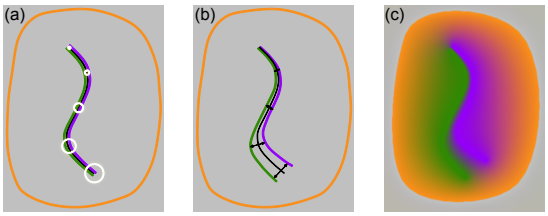

Figure 2: *A DS is a surface with colors on either side and blur values (a). The blur radii are depicted as white circles. (b) The surface is duplicated into two sheets, which are moved apart along the normals by the blur radius distance. (c) Colors from both sheets are smoothly diffused in space.*

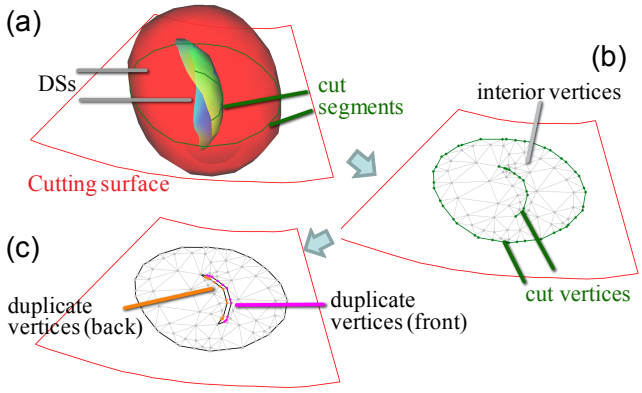

Figure 3: *Cross-sectional mesh generation. Given DSs and a cutting surface (a), the system triangulates the cutting surface (b) and splits the mesh connectivities (c).*

3.2 Generating cross-sections of DSs

Our goal is to generate a cross-section with colors diffused from DSs when the user cuts the model at an arbitrary location. A straightforward way to achieve this is to compute the entire volumetric color diffusion by solving a Poisson equation over a voxel grid or a tetrahedral mesh, similar to DCs. However, this approach lacks scalability and becomes too expensive for complex models, as discussed in detail in Section 3.3. Thus, we propose to compute color diffusion locally at cross-sectional locations using positive mean value coordinates [Lipman et al. 2007] (PMVC). This is inspired by the work of Farbman et al. [2009] where the Poisson solution was replaced by mean value coordinates [Floater 2003] to accelerate various image editing tasks. The actual process of our algorithm consists of two steps: cross-sectional mesh generation and color diffusion using PMVC.

Cross-sectional mesh generation. The system first generates a cross sectional mesh when the user cuts the model. Given a 3D cutting surface, all of its intersections with DSs are computed, forming a set of line segments called *cut segments* (Fig 3a). The cutting surface is then tessellated while preserving these cut segments using conforming 2D Delaunay triangulation, which nicely adapts the mesh density to the size and placement of the DSs to alleviate discretization artifacts. The cross-sectional mesh vertices that are not part of the cut segments are called *interior vertices* (Fig. 3b); their colors are computed by diffusing colors from the DSs using

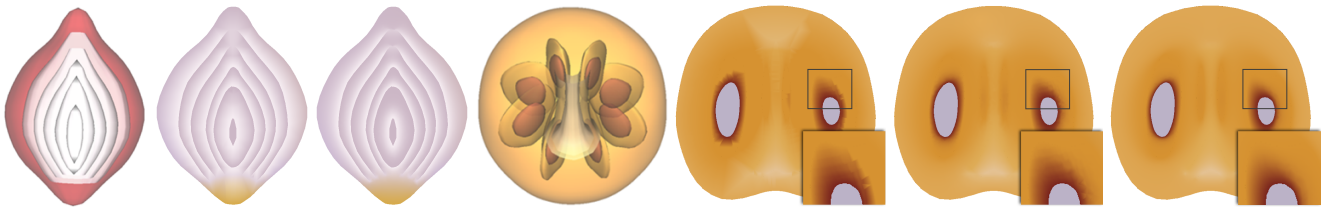

Figure 4: *Comparisons of our PMVC diffusion and Poisson diffusion. The precomputation time (denoted "pre:") includes the tetrahedralization and solving the Poisson equation. The storage sizes refer to the precomputed tet mesh with colors for Poisson and the cross-section mesh with colors for our approach.*

PMVC, as explained later. Mesh vertices that lie on the cut segments are called *cut vertices*; their colors are sampled directly from the respective DSs. A cut vertex has two different colors (front and back) at the same location, so it is necessary to split each cut vertex into *duplicate vertices* (Fig. 3c), divide the mesh connectivity along the cut segment, and assign the front and back colors of the corresponding DS to each duplicate vertex appropriately. The rendering is done simply by drawing mesh triangles with associated colors.

Color diffusion using PMVC. Once the mesh is generated on the cross section, the system diffuses colors on the mesh using PMVC. Given an interior vertex, our goal is to compute its color by interpolating the colors of the DSs using PMVC. To do so, we first render all the DSs viewed from the interior vertex to a cube map. This efficiently determines which parts of the DSs are visible and how far they are from the interior vertex. The system then sums up all the colors rendered on the cube map pixels, weighted by the inverse of the distances obtained from the depth buffer. Note that the diffusion is *consistent* for any cut surface, since the color information is obtained by integrating over the entire set of 3D DSs.

While the original method [Lipman et al. 2007] renders triangles into the cube map using carefully tailored colors to compute coordinate values, our method renders the actual colors of the DSs directly to the cube map since we only need interpolated colors instead of coordinate values. This allows the integration of the cube map pixels to be performed entirely on the GPU, significantly reducing the cost of both reading the GPU memory back to the CPU and integrating cube map pixels on the CPU. Our modification means that the cost of integrating cube map pixels is negligible and the bottleneck becomes the cost of rendering DSs (roughly 99% of the total). Our current unoptimized implementation renders all the mesh triangles of the DSs naïvely per each interior vertex, but it could be greatly accelerated by parallelizing processes for all interior vertices and using efficient spatial data structures [Ritschel et al. 2009].

3.3 Comparison of PMVC and Poisson diffusion

We compared our PMVC diffusion to a standard Poisson diffusion to demonstrate the effectiveness of our approach. For the discretization of the Poisson equation, we used unstructured tetrahedral meshes instead of regular voxel grids, since tetrahedra can easily conform to DSs while voxels lead to rasterization artifacts.

Figure 4 shows some comparison results. The quality of the Poisson diffusion heavily depends on the resolution of the tetrahedral mesh. For relatively simple models (e.g., onions), low resolution meshes are sufficient to produce smooth diffusions, but this does not hold for more complex models (e.g., persimmons). For such models we need to use denser volumetric meshes, which lead to a rapid increase in computational cost. In contrast, PMVC diffusion involves no volumetric meshing and no large systems of equations, and thus scales well with the complexity of the model. The visual differences between PMVC diffusion and Poisson diffusion are noticeable but not significant. The run-time cutting using PMVC diffusion takes a bit longer than that with Poisson diffusion, but this is mainly due to our unoptimized implementation which can be further accelerated. PMVC diffusion also allows frequent switching between designing and browsing of DSs since it requires no precomputations for browsing, which is highly desirable for trial-and-error modeling processes. This way, surfaces can be interactively added, removed, and edited, and the results can be viewed without delay. Additionally, Poisson diffusion has high memory consumption, since the entire precomputed color volume needs to be stored, whereas PMVC diffusion does not suffer from this problem. In summary, Poisson diffusion has some benefits but lacks scalability, and therefore our PMVC diffusion process is better suited for interactive modeling.

4 Creating Diffusion Surfaces

We focus on creating DSs from scratch rather than converting from scanned color volume data, since such data is not yet abundant. DSs are simple and basic primitives, therefore any existing 3D modeling tools can be used to create DSs representing arbitrary objects. However, modeling volumetric structures composed of many nested surfaces using traditional tools can be difficult, time-consuming, and unintuitive. Thus, we chose to focus on a case study of objects with rotational symmetries, and designed a sketch-based modeling interface especially tailored to such objects. Particularly important classes of this kind are fruits and vegetables, since they are one of the most common volumetric objects in our daily lives, often possess intricate volumetric structure and both sharp and smooth color transitions. Our assumption of rotational symmetry additionally enables synthesis of random variations of models, which is useful for populating scenes with many similar objects, such as the salad examples shown in Figures 1b and 11.

We assume that representative horizontal and vertical cross-sections of the object to be modeled with DSs are available as photographs or illustrations, to make it easier on the user by providing a rough reference of the object's shape and structure. The user provides the system with a few guidance sketches that delineate the salient structures of the object, and 3D DSs are then generated from these sketches. The entire modeling process with our interface works as follows (Fig. 5):

1. The user first sketches several 2D curves on both vertical and horizontal cross-sectional images, and the system generates 3D surfaces by sweeping.

2. Optionally, the user can distribute predefined small grains over these sweep surfaces by specifying distribution parameters (the region to be populated by grains and their den-

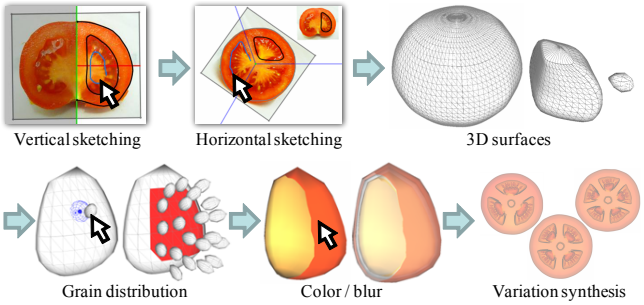

Figure 5: *Overview of our modeling interface for creating DSs.*

sity). The distribution proceeds automatically by dart throwing while respecting a user-defined Poisson disc radius.

3. Next, the user paints colors and blur values on surface mesh vertices in the 3D view. Colors can be sampled from the reference images.

4. Finally, the system instantly synthesizes random variations of the DSs model on request.

Since steps 2 and 3 are mostly manual and straightforward, we only explain steps 1 and 4 in detail below.

4.1 Symmetry-aware sketching interface

Symmetry types. We consider two types of rotational symmetry: *cylindrical symmetry* (Fig. 6) that refers to objects whose structures remain mostly the same after any amount of rotation around the axis of symmetry (i.e., objects with continuous rotational symmetry such as strawberries), and *N-fold symmetry* (Fig. 7) that refers to objects having N repetitive structures around the axis of symmetry (e.g., tomatoes). Modeling processes for each type are explained below.

Cylindrical symmetry. The user first draws several curves on an image of a vertical cross-section (i.e., in the plane parallel to the axis of symmetry). The axis of symmetry is assumed to be straight, and the user sketches only on the right side of the axis (Fig. 6a). Then, the user draws curves on an image of a horizontal cross-section (i.e., on the plane perpendicular to the axis of symmetry) corresponding to the vertical curves drawn previously (Fig. 6b). The axis of symmetry is shown as a point, and the user must draw closed loops surrounding this point. Given pairs of vertical and horizontal curves, the system generates 3D surfaces by sweeping [Snyder and Kajiya 1992].

N-fold symmetry. In this case, we classify features into two types: *Type I*, which surrounds the axis of symmetry (e.g., the outer skin of a tomato); and *Type II*, the geometry of which is included entirely in a single fold (e.g., an inner chamber of a tomato), as depicted in Figure 7. The modeling process for Type I geometry is almost the same as for the case of cylindrical symmetry described above, except that the user must specify the number of folds, N, and adjust the fold angles appropriately when drawing horizontal curves (Fig. 7b).

For the Type II geometry, the user draws a closed loop in the vertical cross-section, as well as a set of N closed loops, each of which is entirely included in a single fold in the horizontal cross-section. In this case, the system generates 3D surfaces by first dividing each vertical curve into two at its top and bottom, and then sweeping the corresponding horizontal curve along these two curves (Fig. 7c).

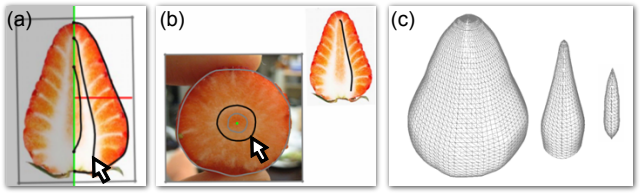

Figure 6: *Sketching interface for the case of cylindrical symmetry.*

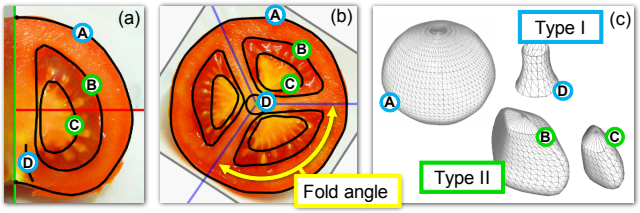

Figure 7: *Sketching interface for the case of N-fold symmetry.*

Note that we chose to let the user extract the image features that define the volumetric structures manually, instead of relying on vision algorithms, because extracting features appropriately from casual images of often glossy and transparent objects can be difficult. In addition, we regard this task as part of the creative design process rather than an automatic reconstruction problem.

4.2 Synthesis of random variations

The system synthesizes the horizontal curve geometries, while other information, such as the vertical profile curves, remains fixed. We analyze and synthesize the horizontal curves in a 2D polar coordinate system whose origin is the axis of symmetry. The process differs depending on the type of symmetry.

In the case of cylindrical symmetry (Fig. 8 top), the curve geometry is first uniformly resampled. Then for each curve point, we sample its radius along with the angular difference from its next curve point. This forms a 1D cyclic array each element of which is a pair of radius and angular difference. This array can be regarded as a 1D texture sample. We thus simply apply a texture synthesis algorithm [De Bonet 1997] on this sample to generate a new randomized 1D array, resulting in a new randomized curve geometry.

In the case of N-fold symmetry (Fig. 8, bottom), the user can first specify the desired N (which can differ from the original). To create variation, we use an approach similar to the morphable face model [Blanz and Vetter 1999]. The system first takes N samples of feature vectors, each representing geometries within a fold, then performs a principal component analysis over these samples, and finally blends the principal vectors linearly with random coefficients.

Figure 8: *Variation synthesis. Top: cylindrical symmetry (onion); bottom: N-fold symmetry (tomato).*

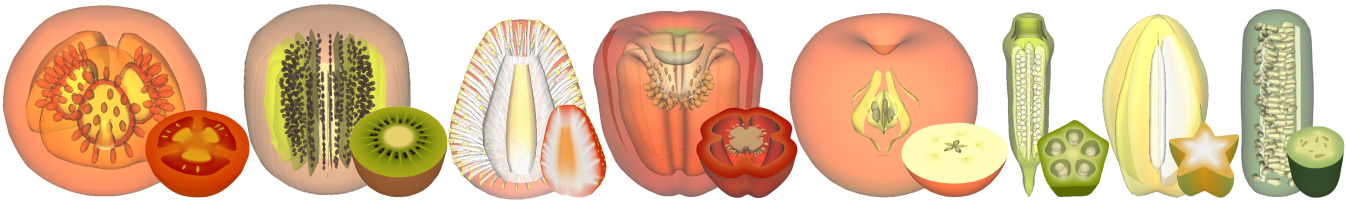

Figure 9: *DSs models created using our user interface. Surfaces are rendered with alpha blending to visualize internal structures.*

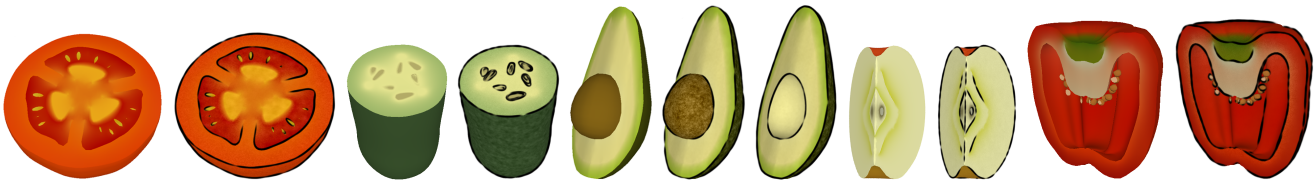

Figure 10: *Increasing the expressivity of plain color rendering by applying simple NPR techniques.*

The feature vectors are computed as follows. For the Type I geometry, the curve is split at every fold line, forming a set of N open curves. These open curves are then uniformly resampled with the same number of points and transformed into a canonical space (i.e., a pie-shaped space with the angle of $2\pi/N$). Finally, the curve points are concatenated into the feature vectors. For the Type II geometry, the set of N curves is evenly resampled and then transformed into the canonical space. Then the correspondences among curve points are obtained based on the sums of Euclidean distances. Finally, the curve points are concatenated into the feature vectors. Note that after synthesizing feature vectors, the Type I geometries (open curves) are blended linearly across each fold line to ensure the continuity among adjacent folds.

All synthesized geometries share the same mesh connectivity with the original so that they can use the same attribute data, such as colors and grain distribution parameters associated with surface meshes.

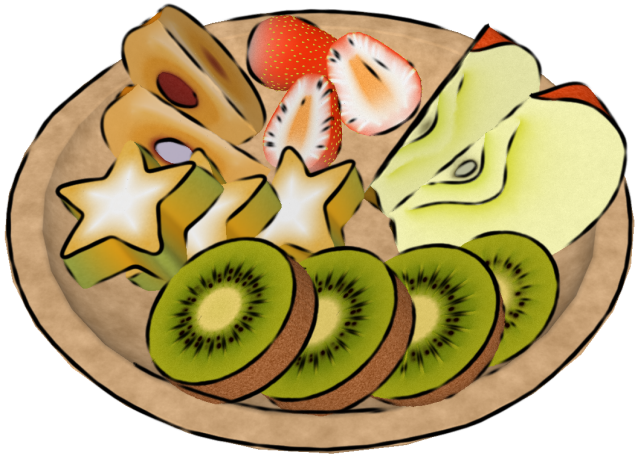

Figure 11: *A scene of assorted fruits displayed using NPR.*

5 Results and Discussion

We have created numerous DSs models using the proposed interface (Fig. 9). Photorealistic rendering of real-world objects using DSs is challenging, since the current representation lacks texture detail information. That said, since DSs explicitly represent volumetric structures via 3D surfaces and blur values, a variety of rendering styles can be easily applied. We have experimented with simple non-photorealistic rendering (NPR) techniques (Fig. 10) that highlight the internal structures and are more illustrative and expressive than the plain color rendering. We have employed artistic silhouettes [Northrup and Markosian 2000] and color modulation based on 3D Perlin noise [Perlin 1985] with manually adjusted weights on different frequency bands based on the unmodulated color. Note also that the readily available structure representation allows rendering cross-sections with concavities or hollow spaces, as visible in the okra and pepper models, for example. This is simply done by tagging the respective side of the DSs representing those regions as invisible and hiding them during rendering.

Figures 1b and 11 show scenes consisting of many cut pieces of objects displayed using NPR. In such scenes, our algorithm for synthesizing random variations of models proved very effective. All objects contain global distinct structures, spatially-varying materials, and smooth color transitions, which would be difficult to handle with previous methods [Takayama et al. 2008; Dong et al. 2008; Owada et al. 2008]. Also note that most examples (e.g., tomatoes, strawberries, star fruits, and more) make use of open surfaces, which cannot be represented by isosurfaces of signed distance fields [Wang et al. 2010]. We emphasize that the DSs representation is rather general and can be used to model a variety of volumetric objects. For instance, we have successfully created a geological model (Fig. 12); our user interface was instrumental in designing the different layers and veins in this case, since they generally follow symmetry around the main lava axis. The kidney model (Fig. 13) is another example of a natural object whose inner structure is modeled with DSs (its shape was modeled manually).

Our prototype system is implemented using C++, OpenGL and GLSL on a laptop with a 2.6 GHz CPU, 3.0 GB of RAM, and an NVIDIA Quadro FX 570M GPU. Table 1 shows our result statistics, indicating the efficiency of the DSs representation in terms of both storage and computation, as well as the usefulness of our user interface for creating DSs, which leads to fairly low design times. All results were designed by the authors, and the design times typically took several minutes. Some of the more complex models were created through a lot of experimentation; the creative exploration took several hours in those cases. Most of the modeling processes required much trial and error, and our simple sketching interface was effective for this purpose. A formal user study is a subject for future work.

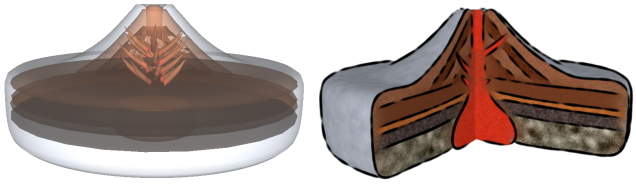

Figure 12: *A DSs volcano model created with our interface.*

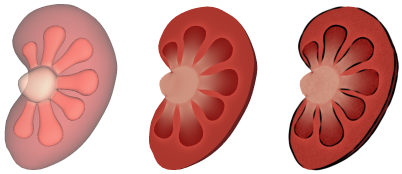

Figure 13: *A kidney modeled using the DSs representation.*

Title	Type	Design (min)	Cut (sec)	Vtx #	Face #	Size (KB)
Tomato	N-fold	25	6.1	10619	20412	669
Onion	Cylndr	10	1.9	4184	8110	264
Persimmon	N-fold	16	8.1	9162	18208	584
Pepper	N-fold	20	6.8	9317	18328	592
Avocado	Cylndr	8	0.8	2699	5341	172
Apple	N-fold	25	3.3	4074	7685	255
Okra	N-fold	17	3.7	9400	18480	597
Star fruit	N-fold	15	3.2	6357	12190	400
Strawberry	Cylndr	141	7.7	18158	32049	1110
Cucumber	N-fold	184	7.9	14984	28592	942
Kiwi	N-fold	162	27.7	21438	38688	1321
Volcano	Cylndr	187	10.9	11057	20673	690
Kidney	–	362	3.9	2939	5694	185

Table 1: *Result statistics. We report the type of rotational symmetry in column 2;* Design *refers to the total design time including sketching and painting;* Cut *is the typical time for cross-sectioning;* Size *is the total data size of the DSs model.*

6 Limitations and Future Work

The main current shortcoming of DSs is the lack of information about texture details. Combining DSs with texture synthesis, in which smooth color distributions defined by DSs guide the synthesis of spatially-varying textures, is an interesting future direction. Additionally, we currently ignore translucency which plays an essential role in photorealistic rendering of volumetric objects. Translucency could be introduced by assigning translucent material information to DSs in addition to colors, but realistic rendering of such highly translucent and heterogeneous materials is still a challenge.

In this paper we focused on rotationally-symmetrical structures such as those frequently found in fruits and vegetables, as a case study of designing user interfaces for creating DSs. We plan to investigate different interface designs for different classes of objects such as organs, geological models, cookings and mechanical parts, by exploiting knowledge specific to each class. In addition to creating DSs from scratch, converting given volume data to DSs is important and deserves further research.

Acknowledgments. We thank Wilmot Li and Doug DeCarlo for their helpful suggestions on illustrative renderings, Murphy Stein for his narration of the accompanying video, and the anonymous reviewers for their valuable comments and feedback. This work was supported in part by the National Science Foundation (grants IIS-0905502 and IIS-0916845), JSPS, JST ERATO, and NYU URCF.

References

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3D faces. In *Proc. SIGGRAPH 99*, 187–194.

CUTLER, B., DORSEY, J., MCMILLAN, L., MÜLLER, M., AND JAGNOW, R. 2002. A procedural approach to authoring solid models. *ACM Trans. Graph. 21*, 3, 302–311.

DE BONET, J. S. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proc. SIGGRAPH 97*, 361–368.

DONG, Y., LEFEBVRE, S., TONG, X., AND DRETTAKIS, G. 2008. Lazy solid texture synthesis. *Computer Graphics Forum 27*, 4, 1165–1174.

FARBMAN, Z., HOFFER, G., LIPMAN, Y., COHEN-OR, D., AND LISCHINSKI, D. 2009. Coordinates for instant image cloning. *ACM Trans. Graph. 28*, 3, 67:1–67:9.

FLOATER, M. S. 2003. Mean value coordinates. *Comput. Aided Geom. Des. 20*, 1, 19–27.

LIPMAN, Y., KOPF, J., COHEN-OR, D., AND LEVIN, D. 2007. GPU-assisted positive mean value coordinates for mesh deformations. In *Proc. SGP 2007*, 117–123.

NORTHRUP, J. D., AND MARKOSIAN, L. 2000. Artistic silhouettes: a hybrid approach. In *Proc. NPAR 2000*, 31–37.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3, 92:1–92:8.

OWADA, S., NIELSEN, F., OKABE, M., AND IGARASHI, T. 2004. Volumetric illustration: designing 3D models with internal textures. *ACM Trans. Graph. 23*, 3, 322–328.

OWADA, S., HARADA, T., HOLZER, P., AND IGARASHI, T. 2008. Volume painter: Geometry-guided volume modeling by sketching on the cross-section. In *Proc. SBIM 2008*, 9–16.

PERLIN, K. 1985. An image synthesizer. In *Proc. SIGGRAPH 85*, 287–296.

PIETRONI, N., OTADUY, M. A., BICKEL, B., GANOVELLI, F., AND GROSS, M. 2007. Texturing internal surfaces from a few cross sections. *Computer Graphics Forum 26*, 3, 637–644.

PIXAR, 2007. Ratatouille (motion picture).

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J., AND DACHSBACHER, C. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. 28*, 5, 132:1–132:8.

SNYDER, J. M., AND KAJIYA, J. T. 1992. Generative modeling: a symbolic system for geometric modeling. In *Proc. SIGGRAPH 92*, 369–378.

TAITO, 2006. Cooking mama (video game).

TAKAYAMA, K., OKABE, M., IJIRI, T., AND IGARASHI, T. 2008. Lapped solid textures: filling a model with anisotropic textures. *ACM Trans. Graph. 27*, 3, 53:1–53:9.

WANG, L., ZHOU, K., YU, Y., AND GUO, B. 2010. Vector solid textures. *ACM Trans. Graph. 29*, 4, 86:1–86:8.