EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2008), pp. 1–8
C. Alvarado and M.- P. Cani (Editors)

Volume Painter: Geometry-Guided Volume Modeling by Sketching on the Cross-Section

S. Owada[1], T. Harada[2], P. Holzer[1,3], and T. Igarashi[2]

[1]Sony Computer Science Laboratories, Inc., [2]The University of Tokyo, [3]University of Munich

Abstract

We propose a sketch-based system to design volume data from scratch. The user splits a surface model and paints brush strokes to volumetrically fill the 3D space. We extend exisiting sketch-based modeling systems in various ways. First, we use the cross-sectional plane as the interface to design internal structures. Second, the predefined surface geometries are used to guide the synthesis process from 2D cross-sectional information to 3D spatial distribution. Third, the 3D solid modeling process is decomposed into two phases: space division and color particles distribution. The user divides the space into multiple regions, each of which corresponds to an uniform texture. The texture is designed by distributing particles in the regions, guided by the surface geometries that define the boundaries. We performed a survey to examine how people associate 2D cross-sectional information to 3D, and reflected the result in designing the system. Our system is capable of producing a wide variety of non-photorealistic volumetric objects such as foods, anatomic/biological models or furry models.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and TechniquesInteraction techniques; I.3.5 [Computer Graphics]: Computational Geometry and Object ModelingGeometric algorithms, languages, and systems, Object hierarchies;

1. Introduction

Easily generating volume data has been one of the central challenges to make volume graphics a standard tool for contents creation. We believe that volumetric information is very important for contemporary computer graphics in games or real-world simulation, as it allows more interaction with synthetic objects, such as to cut, explode, or simulate them [CSC06]. However, we lack tools to intuitively and easily create volumetric objects. Our experience shows that the most practical tool is a procedural approach that requires scripting or parameter tuning [Per85, Pea85]. This method is very effective for noisy and/or fractal-like patterns but not immediately useful for semi-regular, user-controlled geometric patterns, particularly because of the user interface limitation. Another promising approach is the example-based solid texture generation, which usually inputs one or more 2D sample textures to generate a solid texture pattern. Although there are some remarkable previous work in this domain [HB95, Wei01, KFCO*07], they are basically automatic isotropic solid texture synthesis algorithms from 2D exemplars. Therefore, how to design texture from scratch, how to synthesize anisotropic texture, how to specify the region to synthesize and introduce the anisotropy, and how to achieve interactive performance, are still unsolved problems. Our goal is to design a solid 3D texture from scratch, with intuitive, simple, and interactive user interface. To achieve this goal, we propose the system that has three main novel aspects: (1) our system utilizes the cross-sectional plane as the design interface. Since a cross-sectional plane is a 2D entity, the user can easily provide necessary information by a 2D mouse. (2) Surface geometries are used as guides to propagate cross-sectional information to 3D space. We performed an informal survey to verify that humans can associate 3D structure from 2D information on the cross-sectional plane. We utilized the knowledge to design the algorithm. (3) The user designs volumetric textures by a space division (segmentation), followed by color particle multiplication (texture assignment). This coarse-to-fine editing scheme well matches the typical design procedure. We believe our system is the first that offers interactive and intuitive, sketch-based volumetric texture modeling from scratch (Figure 1). The behavior of our system is supported by our questionnaire result to examine the tendency of humans in associating 2D information to 3D. This is in contrast to previous work where 2D to 3D algorithm is devised based on the researcher's intuition, and our design scheme is another contribution of this paper.

2. Our approach

The active use of the cross-sectional plane is based on the fact that we can easily imagine the 3D volumetric structure from a 2D technical illustration which exhibits only a cross-section of a volumetric object (See Figure 2). It is also very conveneient for a standard setup of a personal computer, which is equipped with a mouse that can provide only 2D geometric information. The idea to use the cross-sectional plane as the interface for volumetric design

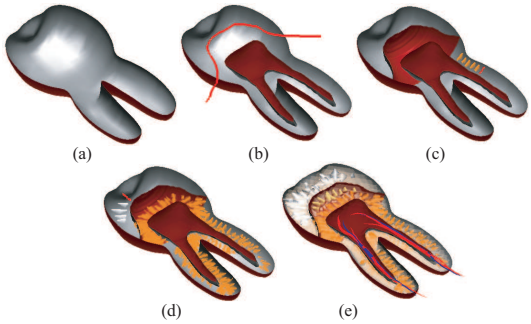


Figure 1: *Overview of our system. The user first creates an outer shape and splits it by a plane, which is then used as the design interface(a). Then they split it into regions, using surface geometries as the guide (b), and assigns textures by color particles, which are multiplied and distributed into 3D space, again using surface geometries as the guide (c,d). The system outputs a set of 3D particles as the volumetric color assignment (e).*

is not new. Owada et al. proposed a sketch-based modeler that can create non-genus-one solid object without textures [ONNI03]. They also proposed the Volumetric Illustration system, which defines internal pseudo solid textures by assigning 2D exemplar texture fragments on the cross-section [ONOI04]. Pietroni et al. recently proposed an image-based system to assign realistic solid texture to the inside of an object by morphing a few number of cross-sectional images [POB*07]. In this paper, we try to unify their notion to use cross-sectional plane as the volumetric design canvas, which allows the user to design a 3D colored texture from scratch, without using any example textures.

Our strategy to interpret 2D cross-sectional information as the definition of the 3D spatial distribution is strongly guided by the geometry of the enclosing surfaces, called guiding geometries. Previous sketch-based modelers also infer 3D information from 2D, but they usually do not consider the contextural information-the enclosing geometry (Figure 4). We performed an informal survey to confirm that the way humans associate 2D cross-sectional information with invisible 3D structure is strongly affected by the enclosing surface geometry (Figure 10, 11,12).

The actual modeling process of our system consists of two phases: space division and color particles multiplication. The modeling domain must be first divided into several regions that have different textural appearances (Figure 1b). Then the each region is filled by multiplication of color particles, considering the geometry of the outmost and the space dividing surfaces (Figure 1c,d,e). It is important to combine these two features, since most meaningful models consist of multiple types of textures in one model, each of whose appearance reflects the boundary surfaces (Figure 2).

In addition, the entire model can be interactively deformed by applying a grid-based 2D deformation technique in the projected space [IMH05]. Existing 3D deformation algorithms usually work only on the surface geometry [SZT*07, SSP07], while our system can interactively deform internal structure that follows the guiding surface, without losing textural appearance and interactive performance. While related works are mainly in the 2D domain [FH07, AS07], we focus on 3D texture re-synthesis.

Our work is an experimental system that tries to mimic the human ability to associate 2D with 3D. Although our system is not capable of designing all possible 3D textures, it can still produce a wide variety of models with little user interaction, especially non-photorealistic styled volume data, such as technical illustrations. In contrast to the photo-realistic modeling that usually requires scanned data from the real world, our focus is on designing volume data from scratch by simple sketch-based user interface and manual painting of color particles.

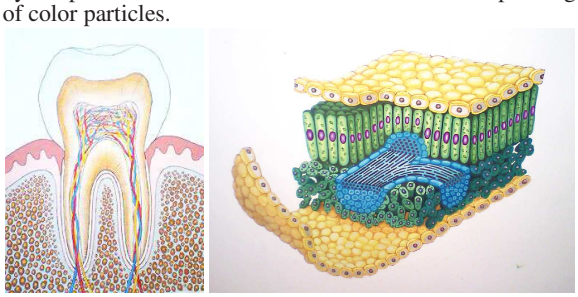


Figure 2: *Examples of technical illustration*

3. Related work

Automatic 2D stroke to 3D algorithms were originally discussed in the context of sketch-based modeling. The SKETCH system utilizes a set of 2D gestural operations to input 3D shapes without altering the viewpoint [ZHH96]. The Teddy system extends the idea of using 2D gestural inputs to define 3D geometry [IMT99] by applying a plausible assumption of the target shape, achieving a highly intuitive response to the 2D input. These systems were refined to produce commercial softwares [Ske] and further extended in recent works [KH06, NISA07]. One limitation in most of previous work is that the mapping from 2D to 3D is fixed. In other words, a given 2D stroke gives always the same object. However, if the input 2D stroke is enclosed by other geometries, the mapping should be strongly affected by them. Our system is novel in this aspect: we actively use the nearby geometry to generate 3D from 2D.

Solid texturing has been an important technique to apply 3D-consistent organic appearance to 3D objects [Per85, Pea85]. The procedural technique is further extended to generate more complicated objects [KPHE02], with possible guidance by an external geometry [CDM*02]. However, procedural techniques usually lack intuitive user interface for modeling. One exception is the system proposed by Phan and Grimm, which adopts a sketching interface to control procedural textures [PG06]. One remarkable feature of their system is the global control of the texture orientation and sizes. Unfortunately, their system is applied only in 2D domain, and the user-controllability is not sufficient. Image-based solid texture synthesis techniques were also developed [HB95, Wei01, KFCO*07]. They used 2D sample images as input and automatically produce 3D solid textures. While such interfaces are easy to use, no interactive systems were proposed. In addition, we focus more on designing solid texture from scratch, without any predefined exmplars.

Automatic multiplication of elemental objects is a key technique to facilitate the 3D modeling process. Geometry synthesis is an approach that tries to achieve this goal [ZHW*06, Mer07]. Barla et al. proposed a new 2D texture synthesis technique applied to vector graphics [BBT*06]. This work shows nice results in synthesizing isotropic 2D textures. Schwarz et al. recently proposed an interactive system that automatically multiplies color particles on a 2D canvas [SIMC07]. Ijiri et al. also proposed another 2D synthesis technique for discrete objects [IMMI08]. These are direct previous work, though these deal only with 2D domain. In addition, some previous techniques offer global control of the arrangement [SIMC07, IMMI08]. However, the information should be supplied by an additional user input that provides guiding field information. Our system infers the global distribution of particles from local examples that are naturally provided by the user.

4. User interface

We first overview the user's experience. Our system works together with the Teddy system [IMT99]. The surface model designed and saved with the Teddy system is automatically loaded in our software.

In our modeler, the user begins in split mode: they first draw a straight line that splits the model into two by a cross-sectional plane. Then they can either divide the inside of the model into multiple regions, or start volumetric painting by switching to paint mode. (We use the term 'split' for the initial cutting operation that splits the outmost surface object into two by a plane, while 'divide' is used for subdividing the internal space into multiple regions, by non-planar surfaces in most cases.) If the user wants to divide the inside of the model, they draw dividing strokes on the cross-sectional plane. The strokes are converted into 3D surface by the algorithm described in Section 5. If the user draws a closed stroke, the system automatically inflates the contour to make a closed, floating region on the cross-sectional surface (Figure 3(a-b)). If the user draws an open stroke that divides the cross-sectional region into two parts, the internal space of the object is divided into two regions, with the newly generated surface touching the outer object (Figure 3(c-d)). The advantage of our system is that the inflated surface geometry usually better matches human intuition than simple geometry-oblivious inflation algorithm. For example in Figure 4, the user is more likely to expect isocontour-like surface generation, although existing algorithms produce small blobby object. This tendency is clearly observed from the result of our questionnaire (See section 5.)

After the space is sufficiently divided, the user can change the mode to volume painting mode by pressing a toolbar button. Three tools are available in the paint mode: a smooth brush (Figure 5 left), a pencil brush (Figure 5 center), and a flood fill tool (Figure 5 right). The smooth brush is a brush with a transparent edge. The pencil brush presents a hard edge on the boundary. These brushes are visualized by billboard meshes with alpha blending information. The flood fill tool fills a connected region with an uniform color.

Smooth/pencil brushes on the cross-sectional plane define example color particles that should eventually fill the whole 3D region. When the user triggers the multiplication by a mouse click, the system automatically finds (a set of) boundary surfaces that are most closely correlated to the example particles. We assume that the particles are arranged as a layer, whose geometry is guided by the boundary surfaces [CDM*02]. Then the system automatically multiplies the example particles and distributes them into 3D space using the boundary surfaces as a guide (Figure 6). If the object is split into multiple regions, our system tries to find the (combination of) surfaces that most closely guide the provided particles (Figure 7).

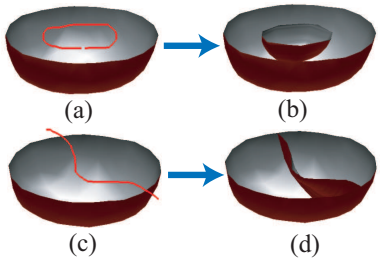


Figure 3: *Drawing space-division curve on the cross section.*

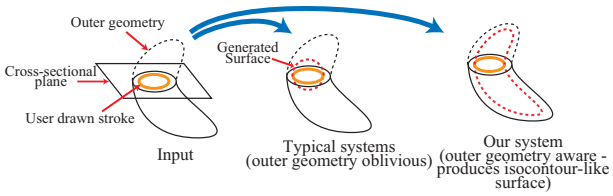


Figure 4: *Advantage of our system. The closed contour drawn on the cross-section, which looks like an offset boundary of the outer surface, tends to be thought of as the cross-section of the isosurface. Existing algorithms do not necessarily produce isocontour because they do not consider the outer geometry.*

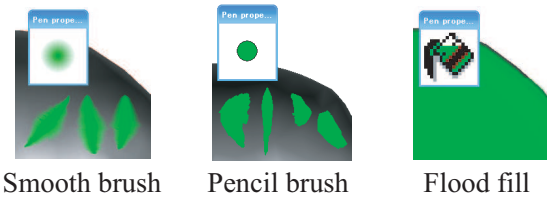


Figure 5: *Brush types*

One novel feature of our system is the deformation of textured objects. When the user deforms the object, our system first applies the deformation only to the external and splitting surfaces and then the internal texture is re-synthesized, instead of only deforming the textured domain (Figure 8). The information specified by the user when creating the object is always stored in the system and used for this re-synthesis process. We consider that re-synthesis is very important because just applying deformation and displacement of the particles can produce undesired local density variations, thereby changes the appearance.

The defined set of particles and background colors are previewed by OpenGL alpha blended polygons. The user can

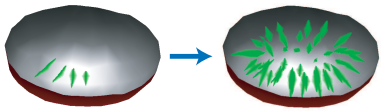


Figure 6: *The user-drawn cross-sectional color particles are automatically multiplied and distributed in 3D, using the geometry of the enclosing surface.*

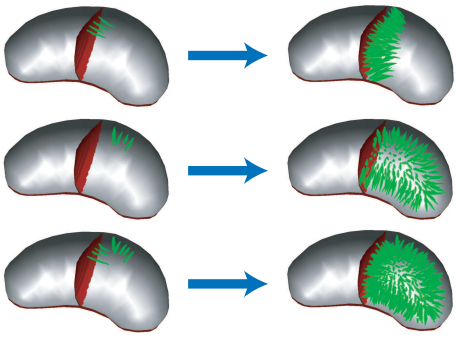


Figure 7: *The system determines which geometric object should guide the multiplication of the color particles.*

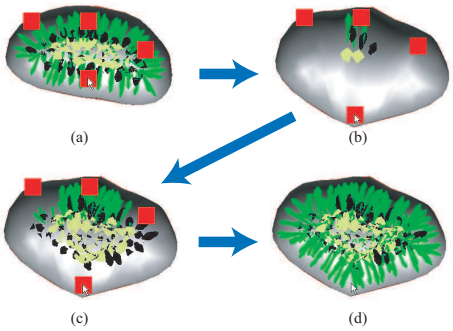


Figure 8: *The user can deform the model after the automatic particle distribution (red squares indicate anchor points which the user can move.) The particles are re-synthesized to retain the appearance, following the new geometry.*

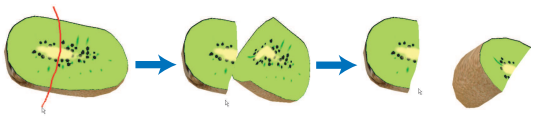


Figure 9: *Interaction mode where the user can interactively cut the model and explore the spatial distribution of the generated solid texture*

optionally generate a regular grid of colored voxels and use it as a solid texture for the out most surface object. The user can interactively cut and explore the 3D texture distribution (Figure 9).

5. Preliminary Study

We first performed an informal survey on how people infer invisible 3D geometry from 2D information on a cross-section. Figure 10 is the questionnaire sheets we distributed. Each question shows a 3D object that is split into two parts by a plane. On the cross-section, one or more contours are drawn. The subjects were asked to extend the cross-sectional contours into 3D space and draw the outline on the sheet. Figure 11,12 shows the summary of the result. The number below each answer shows the number of subjects that gave a similar answer. Some subjects drew nonsense lines, which are not counted.

For closed strokes, the expected shapes are clearly guided by the enclosing object (question a-h). If the user-drawn contour line closely follows the cross-sectional shape, then the extende surface also follows the enclosing surface geometry (question a,c,e,f). Otherwise the contour tends to be swept along the axis of the enclosing object (question b,d,e,g,h). If no distinct axis is found in the enclosing object, or the axis strongly curves, the sweep may be terminated, possibly with a smooth end, like a simple geometry-oblivious inflation in existing sketch-based modeling system (question a,b,d,h).

Open strokes also generate the surface which tends to follow the surface (question i,p,q,r). However, simple sweeping of the stroke along the direction perpendicular to the cut-plane is also distinct (question m,o,s,t).

6. Space division

Based on the preliminary study, we categorize the splitting strokes into three types: (1) a closed stroke that is entirely enclosed in the area to be split, and all sample points of the stroke are located almost at the same distance from the boundary (as in question a,c,e,f), (2) a closed stroke that is entirely enclosed in the area to be split, but the distance between sample points of the stroke and the boundary surface has large variance (as in b,d,g,h), and (3) an open stroke that starts and ends outside of the region and intersects the region (as in i-t).

Differenciating types (1) or (2) is performed by checking if the variance of the distance between the sample points of the stroke and the considered surface is over a threshold value. To cancel the effect of scaling, we normalize the distance values on the stroke by dividing it by the distance value at the deepest position on the cross-section.

(1) Offset division Offset division happens if the closed stroke is entirely enclosed in the area to be split, and all sample points of the stroke are located almost at the same distance from the boundary (See Figure 4, for example.) The resulting 3D surface should be the isosurface of the outer surface at the given distance. This is simply done by Marching Cubes algorithm for the distance function of the boundary [LC87]. If there are multiple connected components in the result, only the closest to the drawn stroke is generated.

(2) Sweep division If the closed stroke is entirely enclosed in the area to be split, but sample points are not on a unique isosurface, the stroke should be swept along the axis perpendicular to the cross-sectional plane. According to the survey, the generated surface should hold the following properties: (1) the sweep axis tends to follow the enclosing geometry, (2) if the axis is too curved, the swept stroke

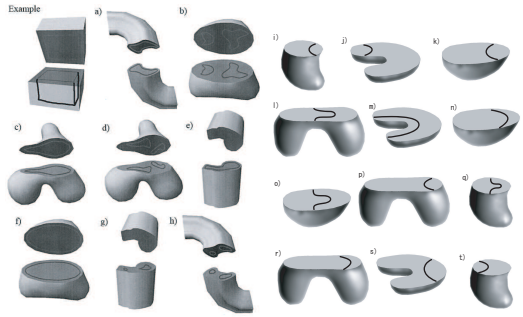


Figure 10: *The questionnaire sheet we distributed. One or more contours are drawn on the cross-section of an object and the subjects are asked to extend the contour into the object.*

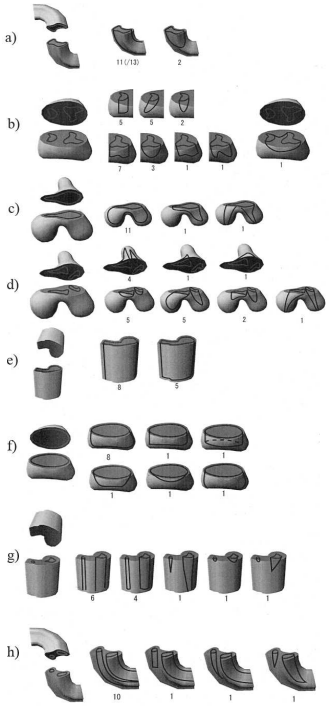


Figure 11: *Answers for the questionnaire. We classified the answers for each question into several typical types. The number below an answer indicates the number of subjects who expect the corresponding result. There is a clear tendency that the extended shape follows the outer geometry.*

should be shrunk, (3) sweep is terminated if the distance function changes too much or self occlusion hass happened, and (4) the stroke is never enlarged during the entire sweep process (Figure 11b,h).

To satisfy these requirements, sweep is performed in the step-by-step manner. Starting from the user-drawn dividing stroke, it is slightly displaced along the normal direction of the cross-section, and deformed by a nonlinear optimization technique, to satisfy the above mentioned requirements. The

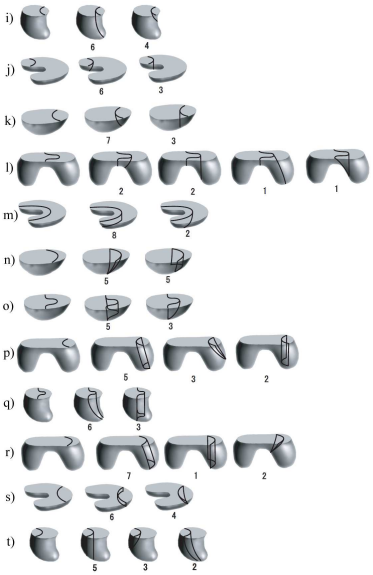


Figure 12: *More answers for the questionnaire*

objective function to be minimized to compute i-th contour is represented as follows:

$$\lambda_c||c_i - c_{i-1}|| \quad +\lambda_s(s_i - s_{i-1}) \\ +\sum_j D(c_i + s_i(p_{i-1,j} - c_{i-1})) - D(p_{i-1,j}),$$

where λ_c, λ_s are scalar constants to suppress the sudden change of the contours, $||\cdot||$ is the root-sum-square of a vector value, a 3D vector c_i and a scalar value s_i are unknowns, representing the gravity center of the new contour and the scaling factor, respectively. c_{i-1} is the gravity center of previous contour, s_{i-1} is the scaling factor of the previous contour, $p_{i-1,\cdot}$ is each sample point on the previous contour, and $D(\cdot)$ is the distance function of the enclosing geometry. The variable c_i only moves on the plane parallel to the cross-sectional plane, while s_i is between 0 and 1.

After the optimization, the new center of the contour (c_i) and the scaling factor (s_i) are determined. If s_i is too small or the new contour does not fit into the enclosing geometry, or, c_i or s_i change too much, the sweep is terminated.

(3) Open-stroke division If the stroke does not start and end inside the object, it is considered as an open division stroke. For this type of splitting stroke, we switch the surface generation algorithm by pressing the modifier key on the keyboard. This is based on the fact that the result of questionnaire shows two different types of generated surfaces (Figure 10). If the shift key is pressed during operation, the dividing stroke is swept along the direction perpendicular to the cutplane (Figure 13(a)). Otherwise, we find start/end points that cross the boundary of the object, and split the object by the plane that contains the points and perpendicular to the cross-sectional plane, producing a closed contour that passes through the points. The system then generates a surface such that its silhouette matches the drawn contour (Figure 13(b)). This algorithm amounts to the 'Extrusion' algorithm in the Teddy system, while in our system, the initial contour to be

swept reflects the outer surface geometry (See [IMT99], section 4.4.) Although this algorithm does not consider the distance function of the enclosing surface geometry, it produces the similar effect as the questionnaire (Figure 11,12).

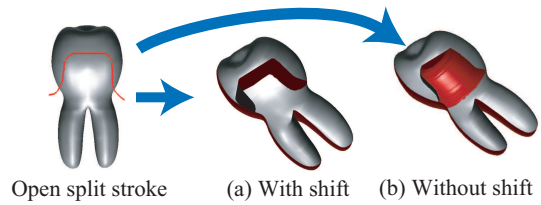


Figure 13: *An example of open stroke division.*

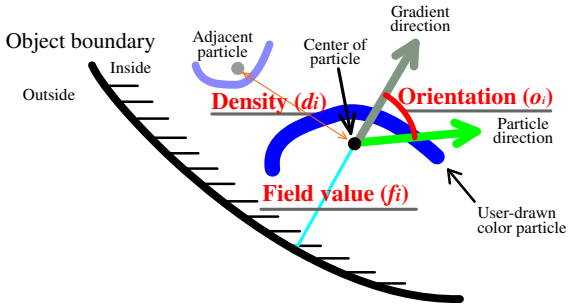


Figure 14: *Variables in analysis phase*

7. Object-guided color particle distribution

After the space is divided into one or more regions, the user assigns textural information to each of them. Our system allows the user to directly paint color particles on the cross-sectional plane and automatically propagates them into the 3D region. This is in contrast to previous systems, which use an example image as input [ONOI04, POB*07]. We are more interested in manual color assignment from scratch, and focus specially on textural patterns that consist of a set of small color particles. The user specifies a few number of example colored particles on the cross-sectional plane, and the system automatically distributes them into the 3D space with consideration on the particles layout and guiding geometry. This process is also similar to some previous systems [BBT*06,SIMC07,IMMI08]. However, we propose the system to convert from 2D domain to 3D, without explicitly supply the global control information.

Our system consists of two phases: analysis and synthesis. The analysis phase gathers information from the color particles manually specified by the user, and associates them with the enclosing geometry. The synthesis phase actively generates and layouts synthetic particles into 3D space. We explain these phases as follows.

7.1. Analysis

In this phase, three properties are assigned for each given particles: density, field value, and orientation, which are represented as d_i, f_i, o_i, respectively (i is the index of the corresponding particle.) In our system, density d_i is defined as the distance to the center of the closest adjacent particle. Field value f_i is the distance from the outer guiding boundary to the center of the particle (Figure 14). However, if multiple boundaries are defined, it is undetermined which distance function to be used. Therefore, the system evaluates all possible combinations of distance functions and take the combination that produces the smallest variance of the field values f_i (see Figure 7). We call this combined reference distance function D_f. Multiple distance functions are combined by taking the minimum absolute value of the functions. Orientation o_i is the angle between the gradient of D_f and the 3D unit vector that is the principal axis of the particle (particles are usually given as a set of successive sample points.) Since the same set of points produces two opposite directions for the principal axis, we choose the one that corresponds to the drawing orientation of the particle: from the first point to the last point.

7.2. Synthesis

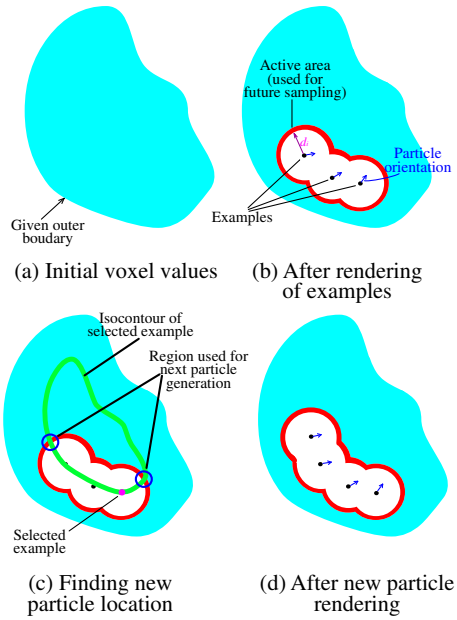
In the synthesis phase, the system first allocates a float-valued voxel grid. This regular voxel grid, called V_s, is aligned with the bounding box of the external surface. We set the resolution of this voxel to be constant: 128 voxels along the longest axis (this number affects the granularity of particle density.) Each voxel in V_s is either initialized as zero (if it is inside of the domain to be filled by particles), or as $-\infty$ otherwise.

Our algorithm generates color particles, whose mutual distances are close to the given examples. This process is analogous to Poisson disc sampling, where the minimum distance between sample points is bounded. In our system, the maximum distance is also controlled. The system starts updating V_s by given example particles (Figure 15a,b). One particle is associated with a spherical region whose radius is d_i and whose center is at the center of the original particle (Figure 15b). The inside of the spherical region holds the value $-\infty$ (which means disabled voxels: colored in white in Figure 15b), while the boundary region is 1, which means active voxels used for future synthesis of particles (colored in red in Figure 15b). These spherical regions, associated with example particles, are added to V_s (Figure 15b). Since adding any number to $-\infty$ makes $-\infty$, the region near particles is filled by $-\infty$, and the boundary region holds a positive value.

After the given example particles are processed, new particles are added one by one so that the distance from existing (already generated) particles to the new object is close to the desired density value. The value stored at each voxel is the probability of the position to be the center of the next synthesized particle. Therefore, the voxels whole value is equal to or less than 0 will not be used as the next particle position.

We describe this process in detail. To synthesize one new particle, we first randomly select one particle from given example particles (we assume the index of the selected particle is j) Then we select voxels in V_s where the distance function (D_f) value at the position is close to f_j (field value of the sample particle - See Figure 15c. Voxels that share the same distance value as f_j are colored in green). This ensures that the synthesized particle is on the same isosurface of D_f. This also means that the isovalue that is not present in the input particles will not be used for synthesis, producing layered appearance of the final particle arrangement.

We randomly choose one voxel in the selected voxels, with the probability proportional to the voxel value in V_s.



(a) Initial voxel values (b) After rendering of examples

(c) Finding new particle location (d) After new particle rendering

Figure 15: *A 2D example of updating V_s by adding one particle. $-\infty$ voxels are colored in light blue, 0 in white, positive values in red (the stronger the red, the larger the value). Small cross indicates the position of particles. The out most square area is the bounding box of the external surface.*

The location of the chosen voxel is the newly generated particle's position. We sample the gradient of D_f at the position and determine the orientation of the new particle, according to the original angle value o_j (rotational degree of freedom along the the gradient is randomly specified.) The shape of the new particle is the same as the original example particle.

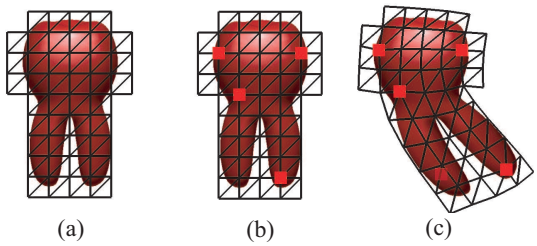
We finally update V_s as exactly in the same way as for example particles (Figure 15d). This process (Figure 15c-d) is repeated until no voxels are selected in searching new location for synthesis.

7.3. Deformation

One of the prominent features of our system is that it supports deformation of the model after the internal texture has been specified. The system stores the source particle information used for each particle multiplication operation and applies it when the outer surface is deformed.

We adopt [IMH05] for the deformation of the surface model. When the user switches to the deformation mode, the screen space is tessellated by a 2D triangular mesh (Figure 16a). Then the user defines some anchor points and interactively moves one of them (Figure 16b,c).

When the user releases the mouse button, the system recomputes the distance function of the new surface model, repositions the original source particles, and finally performs synthesis of the particle once more time.



(a) (b) (c)

Figure 16: *An example of deformation*

8. Results

Figure 1 shows the process of texture detail propagation. The color particles on the cross-sectional plane are multiplied into the 3D space, following the external geometry of the region. In our system, the distance field maxima does not have to be a specific topological type: point, curve or surface. Figure 17 shows more results created by our system. Radial or layered structure is very common and is faithfully designed by our user interface. Our exprience shows our system is more suitable to create non-photorealistic looking objects, because our primary goal was to design a solid texture from scratch, without any predefined textural information. We think that photo-realistic texture generation requires photos taken from the real world. Our system is also suitable to easily create furry models as shown on the right of the Figure 17.

The time to generate particles is proportional to the product of the number of particles and the number of voxels (128 voxels along the longest axis, in our case). Our current implementation generates more than 100 particles per second, with 2.4GHz Core2 duo processor, 4GB memory, on Windows XP.

9. Conclusion and future work

We proposed a sketch-based volume modeling tool using cross-sectional plane as the interface. The modeling process is guided by surface objects, called guiding geometries. These concepts are applied for sketch-based space division and color particle distribution. We also proposed a deformation interface that alters the surface geometry, as well as the internal volumetric structure. Using our system, users can easily and interactively generate solid texture data from scratch.

One of the limitations of our system is the restricted use of field functions. We observed that our distance function is the most significant and meaningful guiding function. However, in some cases, a direct use of the distance function produces counterintuitive results. In such cases, it may be necessary to modulate the distance function.

In the future, we hope to extend this system to support animated objects, aiming at educational purposes. We also hope to support highly structured textures.

References

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. *ACM Trans. Graph. (Proc. Siggraph '07) 26*, 3 (2007), 10.

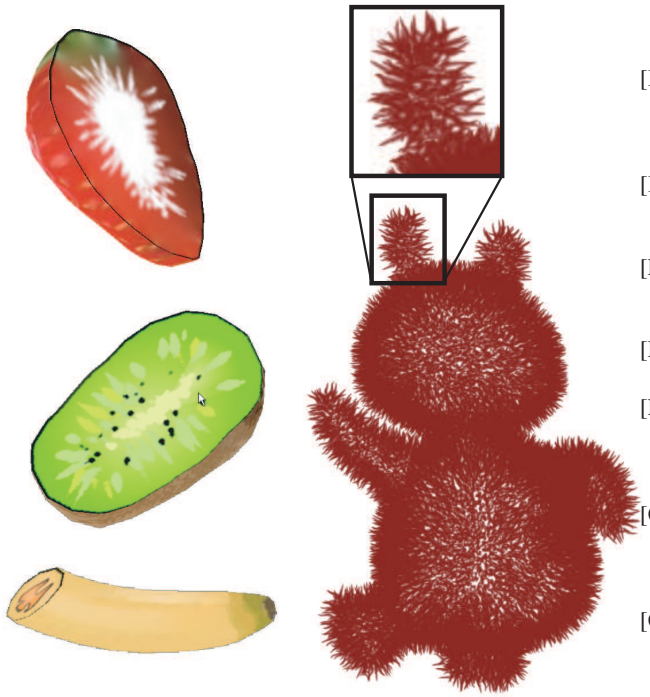


Figure 17: *Results.*

[BBT*06] BARLA P., BRESLAV S., THOLLOT J., SILLION F., MARKOSIAN L.: Stroke pattern analysis and synthesis. In *Computer Graphics Forum (Proc. Eurographics '06)* (2006), vol. 25, pp. 663–671.

[CDM*02] CUTLER B., DORSEY J., MCMILLAN L., MÜLLER M., JAGNOW R.: A procedural approach to authoring solid models. *ACM Trans. Graph. (Proc. Siggraph '02) 21*, 3 (2002), 302–311.

[CSC06] CORREA C., SILVER M.-D., CHEN M.: Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 12*, 5 (2006), 1069–1076.

[FH07] FANG H., HART J. C.: Detail preserving shape deformation in image editing. *ACM Trans. Graph. (Proc. Siggraph '07) 26*, 3 (2007), 12.

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *Proc. Siggraph '95* (1995), pp. 229–238.

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. (Proc. Siggraph '05) 24*, 3 (2005), 1134–1141.

[IMMI08] IJIRI T., MECH R., MILLER G., IGARASHI T.: An example-based procedural system for element arrangement. *Computer Graphics Forum (Proc. Eurographics 2008) 27*, 2 (2008), 429–436.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *Proc. Siggraph '99* (1999), pp. 409–416.

[KFCO*07] KOPF J., FU C.-W., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.-T.: Solid texture synthesis from 2d exemplars. *ACM Trans. Graph. (Proc. Siggraph '07) 26*, 3 (2007), 2.

[KH06] KARPENKO O. A., HUGHES J. F.: SmoothSketch: 3d free-form shapes from complex sketches. *ACM Trans. Graph. (Proc. Siggraph '06) 25*, 3 (2006), 589–598.

[KPHE02] KNISS J., PREMOZE S., HANSEN C., EBERT D.: Interactive translucent volume rendering and procedural modeling. In *Proc. Vis. '02* (2002), pp. 109–116.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. Siggraph '87* (1987), pp. 163–169.

[Mer07] MERRELL P.: Example-based model synthesis. In *Proc. I3D '07* (2007), pp. 105–112.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: FiberMesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. (Proc. Siggraph '07) 26*, 3 (2007), 41.

[ONNI03] OWADA S., NIELSEN F., NAKAZAWA K., IGARASHI T.: A sketching interface for modeling the internal structures of 3d shapes. In *Proc. Smart Graphics* (2003), pp. 49–57.

[ONOI04] OWADA S., NIELSEN F., OKABE M., IGARASHI T.: Volumetric illustration: designing 3d models with internal textures. *ACM Trans. Graph. (Proc. Siggraph '04) 23*, 3 (2004), 322–328.

[Pea85] PEACHEY D. R.: Solid texturing of complex surfaces. In *Proc. Siggraph '85* (1985), pp. 279–286.

[Per85] PERLIN K.: An image synthesizer. In *Proc. Siggraph '85* (1985), pp. 287–296.

[PG06] PHAN L., GRIMM C.: Sketching reaction-diffusion texture. In *Eurographics Sketch Based Interfaces and Modeling workshop (SBIM)* (2006), pp. 107–114.

[POB*07] PIETRONI N., OTADUY M. A., BICKEL B., GANOVELLI F., GROSS M.: Texturing internal surfaces from a few cross sections. *Computer Graphics Forum (Proc. Eurographics '07) 26*, 3 (2007), 295–302.

[SIMC07] SCHWARZ M., ISENBERG T., MASON K., CARPENDALE S.: Modeling with rendering primitives: an interactive non-photorealistic canvas. In *Proc. NPAR '07* (2007), pp. 15–22.

[Ske] SKETCHUP: Google SketchUp (http://sketchup.google.com/) by Google inc.

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH '07) 26*, 3 (2007), 80.

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. (Proc. Siggraph '07) 26*, 3 (2007), 81.

[Wei01] WEI L.-Y.: *Texture Synthesis by Fixed Neighborhood Searching. Ph.D. Thesis*. Stanford University, 2001.

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: Sketch: an interface for sketching 3d scenes. In *Proc. Siggraph '96* (1996), pp. 163–170.

[ZHW*06] ZHOU K., HUANG X., WANG X., TONG Y., DESBRUN M., GUO B., SHUM H.-Y.: Mesh quilting for geometric texture synthesis. *ACM Trans. Graph. (Proc. Siggraph '06) 25*, 3 (2006), 690–697.