Modeling Rough Surfaces

Yootai Kim[1], Raghu Machiraju[1], and David Thompson[2]

[1] Department of Computer and Information Science, The Ohio State University
2015 Neil Avenue, Columbus, OH 43210
{*yootai, raghu* }*@cis.ohio-state.edu*
[2] Computational Simulation and Design Center, Mississippi State University
Box 9627, Mississippi State, MS 39762
dst@erc.msstate.edu

1 Introduction

Many surfaces in nature are rough. A rough surface can be defined as a surface that has a fractal dimension. In fact, one measure of roughness is the fractal dimension [1, 14]. Rough surfaces are observed at all scales independent of their origin; for example, a microscopic view of metal substrate, a cauliflower, ice, and mountains are all rough at some level. Additionally, rough surfaces are frequently generated by various technical processes, such as molecular beam epitaxy (MBE).

Rough surfaces are also common in synthetic environments. Techniques for realistic image synthesis have improved dramatically. However, there is no easy solution to the problem of generating rough surfaces. The recent movie *Ice Age* produced by Blue Sky Studios is a good example. Although the synthetic ice world in the feature film was visually appealing, the computer generated ice models were not realistic enough to match the visual richness of natural ice. Furthermore, the production process still requires much labor and somewhat *ad hoc* methods.

In materials science, numerous models have been developed to study surface growth phenomenon [1]. In general, it is difficult to develop a viable continuum model of surface growth phenomenon and then solve the resulting differential equations. Therefore, discrete models play an important role in the prediction of surface growth phenomena. A discrete model is defined as a system with discrete variables and update rules. Since many rough surfaces in nature are formed by deposition and diffusion processes, discrete models simulating these processes can reasonably reproduce natural rough surfaces. Typical results from these simulations consist of point clusters with nontrivial topologies.

For instance, consider the results of the two-dimensional simulation using the diffusion limited aggregation (DLA) cluster growth model shown in Fig 1. The distribution of the islands and dendrites is unusual and cannot be generated by traditional means. Also, in addition to fractal like microstructures, prominent large-scale structures are produced. Therefore, a sophisti-

cated methodology is needed to capture and represent these complex surfaces. Explicit methods of representation using triangulation are unable to capture the intricacies of these surfaces in their entirety. Implicit methods do possess the capability to represent surfaces with complex topologies. However, it is necessary to consider methods that provide variable amounts of smoothness. Level set methods, through appropriate choice of initial conditions and front velocities, can extract surfaces of differing smoothness while preserving the underlying topology and are routinely used for reconstruction of smooth surfaces. We consider their use for reconstruction of rough surfaces.

Fig. 1. A fractal surface growth simulation: DLA

In this paper, we propose a method for generating rough surfaces using discrete surface growth models. Our goal is to develop easily controllable methods for generating rough surfaces for computer graphics applications. We employ a two-pass method. A point set is generated using a discrete model based on surface growth and evolution. Then, the resulting surface is extracted by a level set method. This two-pass process provides more flexibility to users by separating the surface extraction step from the data generation step. The simple rule-based discrete simulations we employ here have several advantages. First, the results are convincing since they are derived from physical processes. Second, implementation is easy and the computations using the methods are not expensive. Lastly, users can exercise control by simply changing the discrete update rules. After generating an initial data set from the application of a discrete model, a level set method is used to obtain an implicit surface representation. This approach allows us to easily handle complex topologies and compute intrinsic geometric properties of the surface. In addition, it is easy to deform the shape for animation and to combine several objects to generate an elaborate model.

Our paper is organized as follows. In Section 2, we review related efforts for rough surface generation. We then provide an overview of our techniques in Section 3. Later, in Section 4 and 5, we describe the details of several surface growth models and our level-set-based surface extraction method. In Section 6, we provide results that demonstrate the potential of our approach and in the final section we draw conclusions and present a discussion of future work.

2 Related Work

The morphology of rough surfaces can be described by fractal models and concepts. To date, fractal models have been the primary method to model natural surfaces. They can be classified as one of the following five approaches: Poisson faulting [14,25], Fourier filtering [14,15,25], midpoint displacement [7, 11, 16, 20], successive random additions [25], and noise synthesis [8, 16]. The Poisson faulting process is a sum of randomly placed step functions with random heights, which generates a Brownian surface. Musgrave [17] compares these methods in a comprehensive way. However, all produce surfaces defined in terms of height fields and cannot generate arbitrary surfaces. In particular, these techniques cannot generate surfaces of arbitrary genus as required by rough and amorphous materials. These methods generate random fractals which are scale invariant in a statistical sense. Gross and large structures are hard to obtain through the deployment of these techniques.

Procedural textures [4] can also be used to simulate a rough surface. Lewis [12] suggested a solid noise synthesis algorithm for surface texturing and stochastic modeling. Worley [27] obtained good results using a cellular texture basis function for organic skin and tiled stone. Fleischer [6] proposed a cellular development simulation to model organic surface details such as scales, feathers, or thorns. The results are very promising; however, it is hard to devise a cellular automata simulation and conversion functions to obtain desired results.

Computer graphics researchers in the past have used deposition concepts for different purposes. Musgrave performed random deposition followed by surface relaxation to emulate thermal weathering processes [17]. Fearing's accumulation model [5] also employed similar ideas for modeling fallen snow. Finally, Dorsey used fractal growth models such as random and ballistic depositions to model weathering of metallic surfaces [3].

Implicit surface methods have been used to represent rough surfaces. Relevant efforts include Hart's implicit representation of rough surfaces [10] and Greene's voxel space automata [9]. Hart derived implicit formulae for fractal representations. He generated wooden surfaces and blended them to demonstrate the power of implicit representation. Greene simulated the growth of plants using simple relationship rules in discrete volumes.

The efforts we report here are different in several ways. The surfaces we produced are not just height fields. Additionally, our techniques produce both microscopic and macroscopic structural variations. For example, DLA models allow the development of larger gross structures. Further, we employ level set formulations to extract the final surfaces. It should be noted that, while researchers in the physical sciences have used growth models for some time, their results are mostly based on two-dimensional models. Thus, the extraction of a three-dimensional growth surface is certainly novel as reported here.

3 Overview of Rough Surface Generation

Our rough surface generation process consists of two modules: the surface growth simulator and the surface reconstructor. The surface growth simulator generates a point set based on user-specified initial conditions. The initial conditions depend on the surface growth model and include parameters for update rules and the initial configuration of the seed points. In this paper, we use two surface growth models: random deposition with surface relaxation (RDSR) and diffusion limited aggregation (DLA). While RDSR is a simple local growth model, DLA is a non-local growth model. It was previously demonstrated that RDSR and DLA surfaces are fractal in nature [1]. These fractals belong to the class of self-affine fractals which are invariant under anisotropic transformation. The surface reconstructor captures surfaces from the results of the growth simulator and generates a surface representation such as a polygonal model (See figure 2).

Thus, our two-pass approach divides the problem into two well-separated sub-problems. Each sub-problem has been studied extensively and many mature technologies can be brought to bear on its solution. In addition to surface growth models derived from materials science, any rule-based method can be used.

A level set method is used to obtain an implicit surface representation. It is a computational technique for tracking evolving interfaces and is used in a wide range of areas such as physics, materials science, and computer vision [23]. Our surface reconstruction method is based on a level set method proposed by Zhao [30]. In [30], an initial surface is continuously deformed toward a final surface in a potential flow direction. The final surface can be extracted as a polygonal model using the marching cubes method [13] and then rendered with standard graphics software.

4 Surface Growth Simulation

We now describe two fractal surface growth models: random deposition with surface relaxation (RDSR) and diffusion limited aggregation (DLA). While

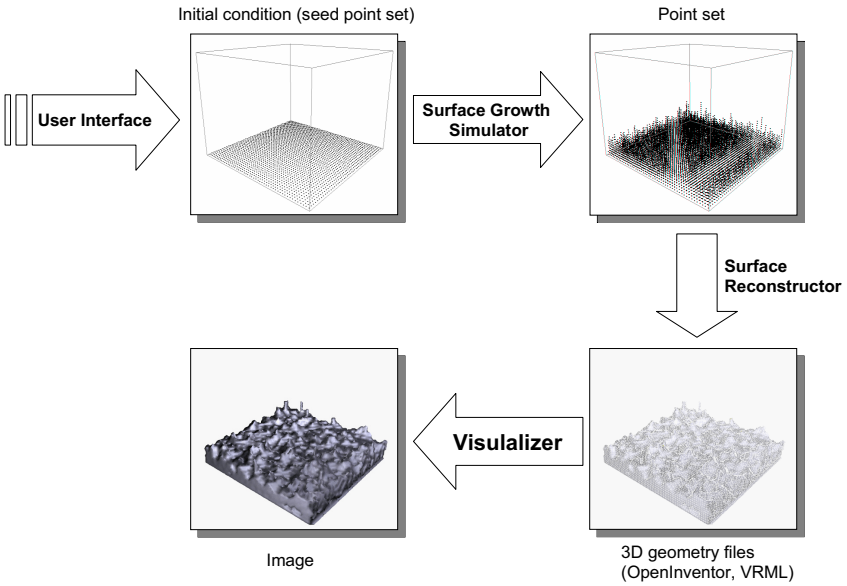


Fig. 2. Rough surface generator pipeline

RDSR is a local growth model, DLA is a nonlocal growth model. DLA generates more diverse surfaces than RDSR. More detailed descriptions of these and other methods can be found in [1].

4.1 Random Deposition with Surface Relaxation (RDSR)

We first explain the random deposition model (RD) because it easily leads to RDSR. RD is the simplest local growth model. From a randomly chosen site over the surface, a particle drops vertically until it reaches the top of the column under it, whereupon it is deposited (see Fig. 3(a)). In RDSR, the deposited particle diffuses along the surface up to a finite distance, stopping when it finds the position with the lowest height (see Fig. 3(b)). Due to the relaxation process, the final surface will be smoother than one generated by RD [1].

The most important difference between RD and RDSR is that an RDSR-generated surface is correlated through the relaxation process. The interface width, another measure of surface roughness, is defined by the root mean square fluctuation in the surface height. The interface width grows indefinitely for RD surfaces, but saturates for RDSR surfaces. RD surfaces look

very rough and protrusive while RDSR surfaces appear smoother and more natural.

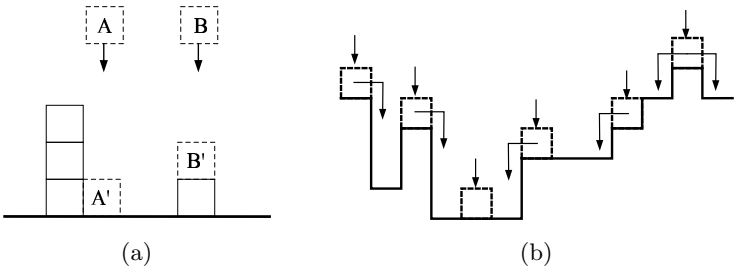


Fig. 3. (a) RD model, (b) RDSR model

4.2 Diffusion Limited Aggregation (DLA)

DLA is the most widely-known nonlocal cluster growth model. The working of the model is illustrated in Figure 4. A seed particle is fixed at a site in the bottom plane. A second particle is then released from a random position distant from the seed. It moves following a Brownian trajectory or a random walk until it reaches one of the four neighbor sites of the stationary seed whereupon it sticks with some probability forming a two-particle cluster. Then, a new particle is released which can stick to any of the five perimeter sites of the two-particle cluster. This process is then repeated. The diffusive effect is achieved through the use of Brownian motion of particles and the release of particles from clusters.

The nonlocality of DLA is due to the shadowing effect generated by the branches of the cluster. There is a much higher probability that a new released particle will be captured by the outlying portions of the cluster than in the interior regions. In other words, the interior region is shadowed by the branches on the perimeter (see Fig. 1). Hence, the growth rate depends not only on the local morphology, but also on the global geometry of the cluster. DLA can generate various surfaces from dendritic structures to a moss-like structure depending on the sticking probability and the nonlocal growth effect [21].

5 Surface Reconstruction

We now explain our level-set-based surface reconstruction algorithm. We follow the level set formulation in [30]. Since the method of [30] is targeted to the reconstruction of smooth surfaces, we employ a different potential flow for the reconstruction of rough surfaces in the level set formulation.

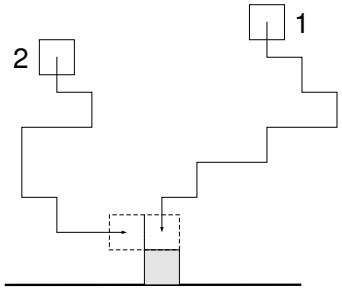


Fig. 4. Growth model for DLA

5.1 Level Set Formulation

In general, the surface growth simulators produce surfaces that do not have simple topologies. This makes explicit surface representation almost impossible to implement. The level set method is a powerful numerical technique for the deformation of implicit surfaces. The level set formulation works in any number of dimensions. The data structure is very simple and topological changes are handled easily.

The level set method was originally introduced by Osher and Sethian in [19] to capture evolving surfaces by curvature flow and has been successfully used to track interfaces for wide variety of problems. See [18, 23] for a comprehensive review. The two key steps of the level set method are described below.

Embed the surface A co-dimension one surface Γ is defined as the zero isosurface of a scalar (level set) function $\phi(\mathbf{x})$, i.e., $\Gamma = \{\mathbf{x} : \phi(\mathbf{x}) = 0\}$. $\phi(\mathbf{x})$ is negative inside Γ and positive outside Γ. In practice, the signed distance function is preferred as a level set function. Geometric properties of the surface Γ, such as the normal and mean curvature can be easily computed from $\phi(\mathbf{x})$ using:

$$\text{outward unit normal:} \qquad \mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{1}$$

$$\text{mean curvature:} \qquad \kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}. \tag{2}$$

Embed the motion The time evolution PDE for the level set function is obtained by differentiating $\phi(\Gamma(t), t)$ to obtain

$$\phi_t + \frac{d\Gamma(t)}{dt} \cdot \nabla\phi = 0 \quad \Longleftrightarrow \quad \phi_t + v_n|\nabla\phi| = 0. \tag{3}$$

Here, v_n is the normal velocity of $\Gamma(t)$ which may depend on external physics or global and local geometric quantities.

To develop the level set PDE, one needs to extend the velocity, v_n in Eq. (3), which is given by the motion of the original surface. Let S denote a point set. Define $d(\mathbf{x}) = distance\_function(\mathbf{x}, S)$ to be the closest distance between the point $\mathbf{x}$ and S. We use the convection model of a surface Γ in a velocity field $\mathbf{v}(\mathbf{x})$ described by the PDE

$$\frac{d\Gamma(t)}{dt} = \mathbf{v}(\Gamma(t)). \tag{4}$$

Then, we can naturally extend the convection to all level sets of $\phi(\mathbf{x}, t)$ to obtain

$$\frac{d\phi}{dt} = -\mathbf{v}(\mathbf{x}) \cdot \nabla\phi. \tag{5}$$

While Zhao used $\mathbf{v}(\mathbf{x}) = -\nabla d(\mathbf{x})$ in [30], we use $\mathbf{v}(\mathbf{x}) = -d(\mathbf{x})$ because the computation of $\nabla d(\mathbf{x})$ on a highly rough surface is very unstable. Thus, the level set formulation of our convection model is

$$\frac{d\phi}{dt} = d(\mathbf{x})|\nabla\phi|. \tag{6}$$

5.2 Numerical Implementation

There are three key numerical elements in our surface reconstruction. First, a fast algorithm is required to compute the distance function to an arbitrary data set on a rectangular grid. Second, we are required to find a good initial surface for our level set PDE to reduce the computational cost of solving the PDE. Third, we need a fast and stable solver for the PDE. As shown in Fig. (5), we obtain an initial surface, Γ_i by deforming the bounding surface Γ_0 following an approximate normal flow of Γ_0. Then, we deform the offset surface Γ_i to get the final surface Γ_f by solving Eq. (6).

Computing the Distance Function The distance function $d(\mathbf{x})$ to an arbitrary data set S is computed by solving the following Eikonal equation:

$$|\nabla d(\mathbf{x})| = 1, \qquad d(\mathbf{x}) = 0, \ \mathbf{x} \in S. \tag{7}$$

We use the algorithm in [30] that combines upwind differencing with Gauss Seidel iterations of alternating sweeping orders to solve the differential equation (7). In two dimension, the following upwind differencing is used to discretize Eq. (7),

$$[(d_{i,j} - x_{min})^+]^2 + [(d_{i,j} - y_{min})^+]^2 = h^2 \tag{8}$$

where h is the grid size, n is the total number of grid points, $i = 1, \ldots, n, j = 1, \ldots, n$,

$$(x)^+ = \begin{cases} x & x > 0 \\ 0 & x \leq 0, \end{cases}$$

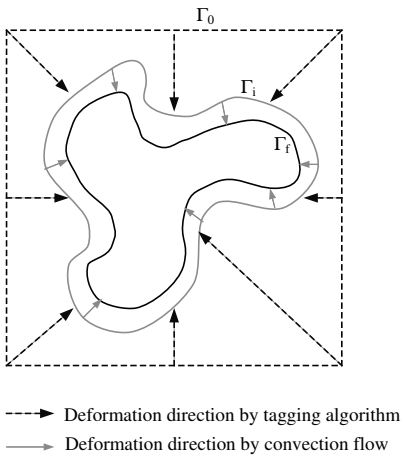


Fig. 5. Deformation methods (Γ_0: an exterior bounding surface, Γ_i: an initial surface for a level set solver, Γ_f: a final surface obtained by a level set solver)

and

$$x_{min} = \min(d_{i-1,j}, d_{i+1,j}) \quad y_{min} = \min(d_{i,j-1}, d_{i,j+1}).$$

The solution for Eq. (8) satisfies

$$\min(x_{min}, y_{min}) < d_{i,j} \leq \min(x_{min}, y_{min}) + h.$$

Hence, the exact solution for the nonlinear Eq. (8) is given by:

$$d_{i,j} = \begin{cases} \min(x_{min}, y_{min}) + h & if\ |\delta| \geq h \\ \dfrac{x_{min} + y_{min} + \sqrt{2h^2 - \delta^2}}{2} & if\ |\delta| < h \end{cases} \tag{9}$$

with $\delta = x_{min} - y_{min}$. Then, the distance function is obtained by solving Eq. (8) on every grid cell in the following four sweeping orderings:

$$(1) i = 1:n, j = 1:n \quad (2) i = 1:n, j = n:1$$
$$(3) i = n:1, j = n:1 \quad (4) i = n:1, j = 1:n.$$

Usually, the solution converges within five or six sweeps in two dimension, and nine sweeps in three dimension. See [28] for details and proofs.

Finding an Initial Surface In our approach, we continuously deform an initial surface to the final surface by following the convection flow direction. If we start with an initial surface that is too far from the real shape, it will take a long time to evolve the PDE. A good guess for the initial surface helps to speed convergence to the final surface. To find an initial surface such that

$\{\mathbf{x} : d(\mathbf{x}) = \epsilon\}$ where ϵ is an offset distance specified by the user, we employ a simple tagging algorithm based on a region growing method in discrete space.

We start from any initial exterior region such as a bounding box. Every grid cell is initially tagged as interior, boundary, or exterior. We denote the interior, boundary, and exterior region as Ω, $\partial\Omega$, and $\overline{\Omega}$ respectively. Let $d_{ij} = d(\mathbf{x}_{ij})$ be the unsigned distance of $\mathbf{x}_{ij}$ to the data set S. We say $\mathbf{x}_{ij} > \mathbf{x}_{kl}$ or $\mathbf{x}_{ij}$ is farther than $\mathbf{x}_{kl}$ or $\mathbf{x}_{ij}$ is larger than $\mathbf{x}_{kl}$ if $d_{ij} > d_{kl}$. We deform the initial tagged boundary $\partial\Omega$ to the final tagged boundary using the tagging algorithm in Algorithm 1.

Require: $S \in \Omega$
1: ϵ: offset distance, $\partial\Omega$: a tagged boundary
2: **while** maximum distance of $\partial\Omega \geq \epsilon$ **do**
3: Pick the most distant point $\mathbf{x}_{<ij} \in \partial\Omega$
4: **if** All interior neighbors of $\mathbf{x}_{ij}$ are closer to S **then**
5: Add $\mathbf{x}_{ij}$ into $\overline{\Omega}$ and Put its interior neighbors into $\partial\Omega$.
6: **end if**
7: **end while**
8: The final $\partial\Omega$ is the offset surface.

Algorithm 1: Tagging algorithm to find an initial offset surface, $d(\mathbf{x}) = \epsilon$

We maintain a priority queue for $\partial\Omega$ so that the most distant point can be identified quickly. After tagging, we recompute the distance function for the tagged boundary. We obtain the signed distance function by negating the distance function at all interior cells.

Solving the Level Set PDE We can continuously deform the initial signed distance function, $\phi(\mathbf{x})$, by solving the level set PDE given in Eq. (3). If we solve the PDE in a brute force way, the computational cost is $O(N^3)$ at each time step for the grid size N. The computational cost reduces to $O(N^{2/3})$ using the fast local level set method [22]. Instead of computing on every grid cell, the computation is restricted to a narrow tube around the zero level set (see Fig. 6). Since the solution of Eq. (3) often becomes very flat or steep at the front $\Gamma(t)$, a redistancing algorithm is needed to keep $\phi(\mathbf{x}, t)$ a signed distance function and smooth in a neighborhood of the front. An upwind scheme is used for space discretization of Eq. (3), and an essentially non-oscillatory Runge-Kutta scheme is used for time approximation. Details for the discretization scheme can be found in [19, 29].

We outline the main algorithm.

1. Update tubes, T and N, where

$$T = \{\mathbf{x} : |\phi(\mathbf{x})| < \gamma\}$$
$$N = \{(x_i, y_i) : \min_{-1 \leq \nu, \mu \leq 1} |\phi_{i+\nu, j+\mu}| \leq \gamma\}.$$

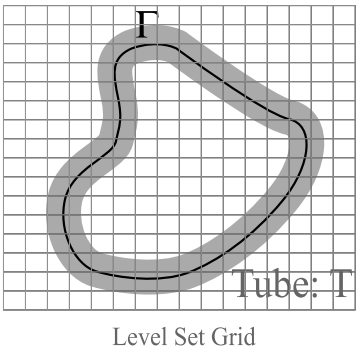


Fig. 6. Computation is only performed on the gray region (Tube: T) around the zero level set Γ

The level set is advanced in time in tube T while the redistancing step is performed in tube N.

2. Advance: Update ϕ in tube T for one time step to obtain $\tilde{\phi}$ by an ODE time stepping method. Instead of using v_n in Eq. (3), $c(\phi)v_n$ is used to prevent numerical oscillations at the tube boundary, where the cut-off function, $c(\phi)$, is defined by:

$$c(\phi) = \begin{cases} 1 & if\ |\phi| \le \beta \\ \dfrac{(|\phi| - \gamma)^2(2|\phi| + \gamma - 3\beta)}{(\gamma - \beta)^3} & if\ \beta < |\phi| \le \gamma \\ 0 & if\ |\phi| > \gamma \end{cases} \tag{10}$$

3. Redistance: Apply the redistancing step to $\tilde{\phi}$ on the tube N. Evolve the following Hamilton-Jacobi equation until $d(\mathbf{x}, \tau)$ reaches a steady state solution $d_s(\mathbf{x})$:

$$\begin{aligned} &d_\tau + S(d)(|\nabla d| - 1) = 0, \\ &d(\mathbf{x}, 0) = d_0(\mathbf{x}) = \tilde{\phi}(\mathbf{x}, t), \\ &S = \frac{d}{\sqrt{d^2 + |\nabla d|^2 h^2}}. \end{aligned} \tag{11}$$

If d_0 is already close to a distance function, the redistancing operation usually takes only one or two iterations within the tube N.

4. Update the new ϕ by:

$$\phi(\mathbf{x}) = \begin{cases} -\gamma & if\ d_s(\mathbf{x}) < -\gamma \\ d_s(\mathbf{x}) & if\ |d_s(\mathbf{x})| \le \gamma \\ \gamma & if\ d_s(\mathbf{x}) > \gamma \end{cases} \tag{12}$$

6 Results

We use the *vispack* library [26] to implement parts of the surface reconstructor. The zero level set or the required implicit surface is extracted using suitable vispack routines that invoke the marching cubes method [13]. The result is available as VRML output. Since VRML is a common three-dimensional format supported by many free and commercial renderers, it provides a flexible choice to users depending on their needs. The computations were conducted on a SGI workstation with a 225MHZ MIPS R10000 processor and 1 GByte of memory. For a 100^3 computational grid, it takes 3–4 minutes to compute a distance function and an initial surface respectively. It takes about 20 seconds for the first order solver, and a minute with the second order solver for one time stepping. We use the second order solver to obtain the presented results.

Fig. 7 shows the deformation of the bunny model by the convection flow given by Eq. (6). The initial surface, Fig. 7(b) is the approximate offset surface from the true surface, i.e. $\{\mathbf{x} : d(\mathbf{x}) = h\epsilon\}$, which is obtained by the tagging algorithm. The initial surface displays aliasing artifacts since the tagging algorithm is a procedural rather than a numerical method. It should be noted that our convection flow is good enough for rough surface characterization, though the result is not as smooth as the one using the weighted minimal surface model in [30]. In the weighed minimal surface model, an additional curvature term regularizes the surface, which is not desired for rough surfaces.

In Fig. 8, we show a rough surface generated by a RDSR simulation on a sphere. An initial surface with an offset of $\epsilon = 3$ is used for all rough surface examples. The final surface in Fig. 8(b) is naturally rough.

Fig. 9 shows a rough terrain. It is generated from a DLA simulation which is performed on a 64^3 grid and the computational grid size is $115 \times 115 \times 49$. This example illustrates how well the level set method captures complex geometries and topologies including arches. The arches are clearly seen in Fig. 10. Fig. 11 is another example of the DLA simulation with different initial conditions. It demonstrates that our method can produce visually appealing natural scenery.

7 Conclusions and Future Work

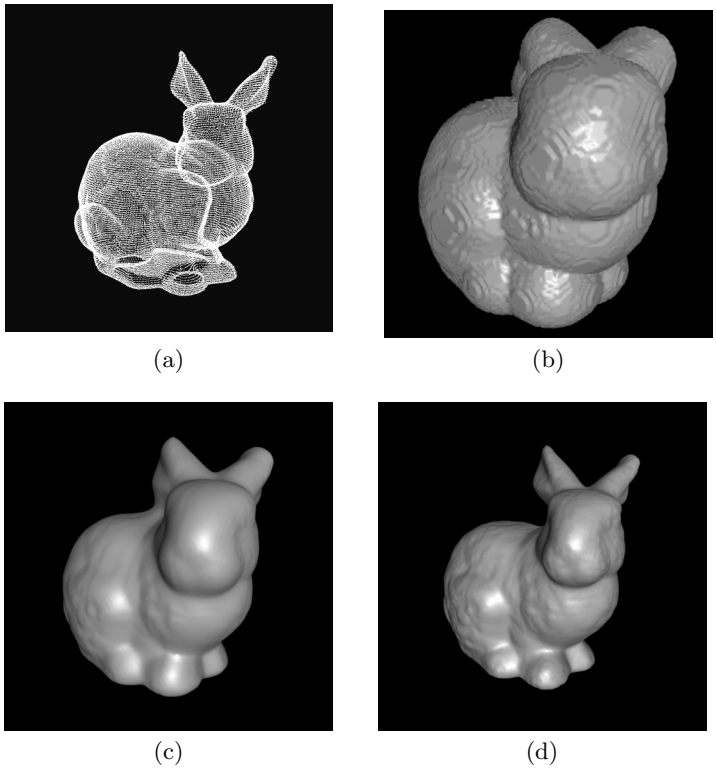
We presented a new rough surface modeling technique using a fractal surface growth model and a level-set-based method for surface extraction. Our method is flexible because of its modular design. A fractal surface growth model guarantees the surface is natural and rough. It also provides the user some control over the shape of the resulting surface. The implicit representation obtained using a level set method handles the resulting complex topologies naturally. It is simple and easy to implement as well. We generated very promising results using these two methods in combination.

An immediate problem is the control of the roughness of the surface, perhaps using the interface width, fractal dimension, or some other measure. One approach is to vary the input conditions. For example, by simulating DLA on a complex object surface, we can generate more interesting results than a typical displacement mapping method. We may use an image as the distribution of the initial seeds for DLA. Another alternative would be to employ environmental fields. Two issues of concern associated with the level set method are the computational cost and the stability of the PDE solver. It may be possible to employ fast and robust methods that are potentially less accurate for computer graphics applications. We are planning to investigate the adaptive, semi-Lagrangian method presented in [24]. Finally, it would be interesting to add roughness directly to the level set function grid by applying a physics-based velocity function such as the dendritic growth in a Stefan problem [2].

References

1. A. L. Barabási and H. E. Stanley. *Fractal Concepts In Surface Growth*. Cambridge University Press, Cambridge, 1995.
2. S. Chen, B. Merriman, S. Osher, and P. Smereka. A simple level set method for solving stefan problems. *J. Comput. Phys.*, 135:8–29, 1997.
3. J. Dorsey and P. Hanrahan. Modeling and rendering of metallic patinas. In Holly Rushmeier, editor, *Proceedings of ACM SIGGRAPH 1996*, Computer Graphics Proceedings, Annual Conference Series, pages 387–396, August 1996.
4. D. Ebert, K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Academic Press, New York, 1994.
5. P. Fearing. Computer modelling of fallen snow. In Kurt Akeley, editor, *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 37–46, New York, July 2000. ACM, ACM Press / ACM SIGGRAPH.
6. K. W. Fleischer, D. H. Laidlaw, B. L. Currin, and A. H. Barr. Cellular texture generation. In Robert Cook, editor, *Proceedings of ACM SIGGRAPH 1995*, Computer Graphics Proceedings, Annual Conference Series, pages 239–248, August 1995.
7. A. Fournier, D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, June 1982.
8. G. Y. Gardner. Functional modeling of natural scenes. *SIGGRAPH Course Notes: 28 Functional Based Modeling*, 28:41–49, 1988.
9. N. Greene. Voxel space automata: Modeling with stochastic growth processes in voxel space. *Computer Graphics(Proceedings of ACM SIGGRAPH 89)*, 23(3):175–184, July 1989.
10. J. Hart. Implicit representation of rough surfaces. *Implicit Surfaces'95*, pages 33–44, April 1995.
11. J. P. Lewis. Generalized stochastic subdivision. *ACM Transactions on Graphics*, 6(3):167–190, July 1987.
12. J. P. Lewis. Algorithms for solid noise synthesis. *Computer Graphics(Proceedings of ACM SIGGRAPH 89)*, 23(3):263–270, July 1989.

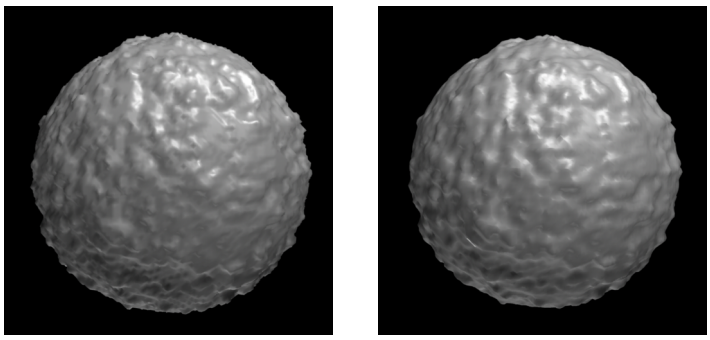
13. W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics(Proceedings of ACM SIGGRAPH 87)*, 21(4):163–169, July 1987.
14. B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Co., San Francisco, 1982.
15. G. A. Mastin, P. A. Watterberg, and J. F. Mareda. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications*, 7(3):16–23, March 1987.
16. G. S. P. Miller. The definition and rendering of terrain maps. *Computer Graphics(Proceedings of ACM SIGGRAPH 86)*, 20(4):39–48, August 1986.
17. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. *Computer Graphics(Proceedings of ACM SIGGRAPH 89)*, 23(3):41–50, July 1989.
18. S. Osher and R. Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational Physics*, 169:463–502, 2001.
19. S. Osher and J. Sethian. Fronts propagating with curvature dependent speed: Algorithms based in hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.
20. H. O. Peitgen and S. Dietmar, editors. *The Science of Fractal Images*. Springer-Verlag, New York, 1988.
21. H. O. Peitgen, H. Jürgens, and S. Dietmar, editors. *Chaos and Fractals*. Springer-Verlag, New York, 1992.
22. D. Peng, B. Merriman, H. Zhao, S. Osher, and M. Kang. A pde based fast local level set method. *Journal of Computational Physics*, 155:410–438, 1999.
23. J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.
24. J. Strain. A fast modular semi-lagrangian method for moving interfaces. *Journal of Computational Physics*, 161:512–528, 2000.
25. R. F. Voss. Random fractal forgeries. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*. Springer-Verlag, Berlin, 1988.
26. R. T. Whitaker. Vispack:a c++ object oriented library for processing volumes, images, and level-set surface models. 2002.
27. S. Worley. A cellular texture basis function. In H. Rushmeier, editor, *Proceedings of ACM SIGGRAPH 1996*, Computer Graphics Proceedings, Annual Conference Series, pages 291–294, August 1996.
28. H. Zhao. Fast sweeping method for eikonal equations. *preprint*, 2002.
29. H. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *Journal of Computational Physics*, 127:179–195, 1996.
30. H. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision (VLSM 2001)*, pages 194 –201, Jul 2001.



(a) (b)

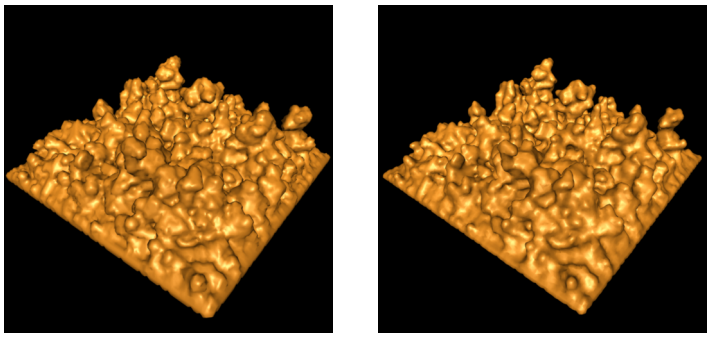
(c) (d)

Fig. 7. Bunny deformed by the convection flow, $135 \times 134 \times 112$ grid: (a) point set, (b) initial surface with $\epsilon = 17$, (c) 100 iterations, (d) 200 iterations



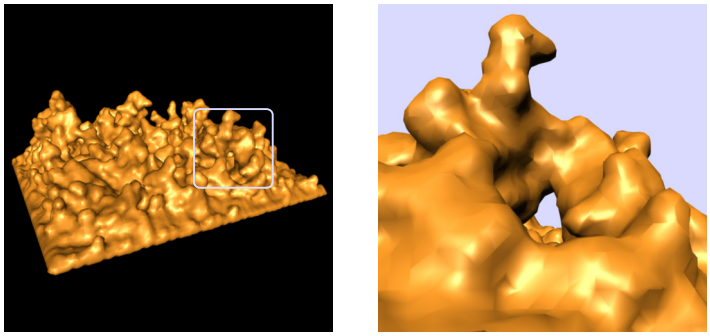
(a) (b)

Fig. 8. RDSR simulation, $112 \times 115 \times 114$ grid: (a) initial surface with $\epsilon = 3$, (b) 30 iterations



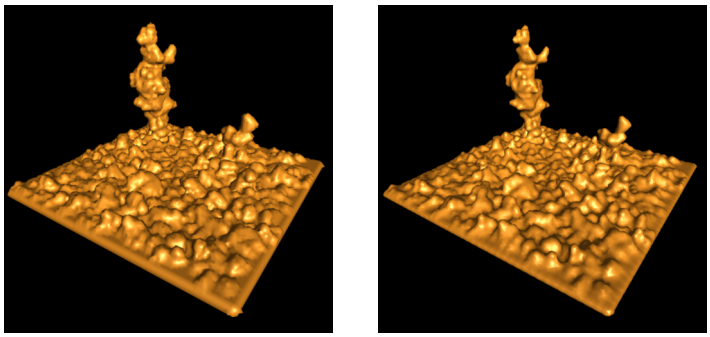
(a) (b)

Fig. 9. DLA simulation 1, $115 \times 115 \times 49$ grid: (a) initial surface with $\epsilon = 3$, (b) 30 iterations



(a) (b)

Fig. 10. Arches from DLA simulations 1: (a) arch structure in the boxed region, (b) zoom-in view of the region



(a) (b)

Fig. 11. DLA simulation 2, $115 \times 115 \times 75$: (a) initial surface with $\epsilon = 3$, (b) 30 iterations