Rough Surface Modeling Using Surface Growth

Yootai Kim*
Department of Computer and Information Science,
The Ohio State University

Raghu Machiraju†
Department of Computer and Information Science,
The Ohio State University

David Thompson‡
Center for Computational Systems,
Mississippi State University

Abstract

In this paper, we present a novel modeling method for synthesizing rough surfaces using discrete surface growth models. We employ a two-pass method. Initial point cluster data is generated using discrete simulation models derived from physical surface growth and evolution. Then, a final surface is reconstructed from the point clusters by a level set method. Since many rough surfaces in nature are formed by deposition and diffusion processes, discrete models simulating these processes can reasonably reproduce natural rough surfaces. These discrete models are easy to implement and efficient enough for interactive control. After generating initial data set from the application of a discrete model, a level set method is used to obtain implicit surface representations. This approach allows us to easily handle complex surface topologies and compute intrinsic geometric properties of the surface if needed. We demonstrate that our approach allows a flexible and general way to create various rough surfaces.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Surfaces and object representation ; K.2 [Physical Sciences and Engineering]: Biology and materials science

Keywords: Natural phenomena, Fractals, Particle system, Surface representation, Implicit surface, Level set method

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 for 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 phenomena. 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 having discrete variables with 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 2D simulation result using the diffusion limited aggregation (DLA) cluster growth model 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-size structures are produced. Therefore, a sophisticated methodology is needed to capture and represent complex surfaces. Explicit methods of representation using triangulation will be unable to capture the complexity of the surfaces in its entireity. 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. Level set methods are being increasingly used for reconstruction of smooth surfaces. We consider their use for reconstruction of rough surfaces.

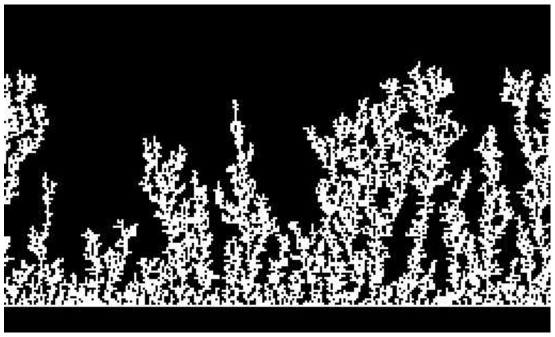


Figure 1: A fractal surface growth simulation: DLA

In this paper, we propose methods for generating rough surfaces using discrete surface growth models. Our goal is to develop easily controllable methods for generating rough surfaces for use in computer graphics applications. We employ a two-pass method. A point set is generated using discrete models based on surface growth and evolution. Then, a final surface is captured 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

*e-mail: yootai@cis.ohio-state.edu
†e-mail: raghu@cis.ohio-state.edu
‡e-mail: dst@erc.msstate.edu

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. Higher resolution surfaces can be obtained by using denser computational grids upon which the level set calculations are performed. In addition, it is easy to deform the shape for animation and combine several objects for an elaborated model.

Our paper is organized as follows. In Section 2, we review some related efforts of 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 surface growth models and 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. Fractal models have been the major methods to model natural surfaces to date. They are classified into one of the following five approaches: Poisson faulting [14, 26], Fourier filtering [14, 15, 26], midpoint displacement [7, 11, 16, 21], successive random additions [26], 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 [18] compares these methods in a comprehensive way. However, all of them produce height field surfaces 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. Global structures are hard to obtain through the deployment of these techniques.

Procedural textures [3] can also be used to simulate a rough surface. Lewis [12] suggested a solid noise synthesis algorithm for surface texturing and stochastic modeling. Worley [28] 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 appropriate 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 [18]. 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 [4].

Implicit surface methods have been used to represent rough surfaces. Interesting and 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 in discrete volume.

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 nonlocal 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 is 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).

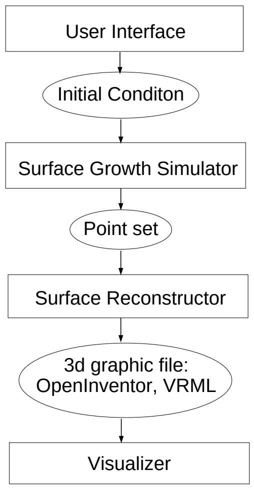


Figure 2: Rough surface generator pipeline

Thus, our two-pass approach divides the problem into two well-separated sub-problems. Each sub-problem has been studied in many areas of science and engineering and, therefore, one can exploit many of the well-developed methods for these sub-problems. We can use not only surface growth models from materials science, but can employ any rule-based method if needed.

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 [24]. Our surface reconstruction method is based on a level set method proposed by Zhao [31]. In [31], an initial surface is continuously deformed toward a final surface in a distance potential flow direction. The final surface can be extracted as a polygonal model using the marching cube method [13], then can be rendered with standard graphics software.

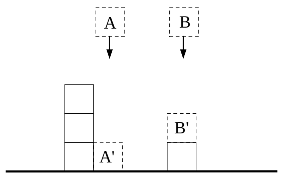
4 Surface Growth Simulation

We now describe two fractal surface growth models: random deposition with surface relaxation (RDSR) and diffusion limited aggregation (DLA). While RDSR is a local growth model, DLA is a nonlocal growth model. DLA generates more diverse surfaces than RDSR. More detailed descriptions of these methods and others can be found in [1].

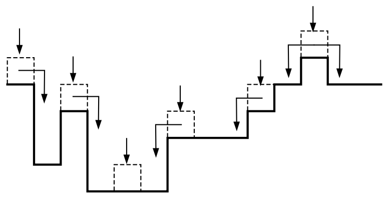
4.1 Random Deposition with Surface Relaxation (RDSR)

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 Figure 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 Figure 3(b)). Due to the relaxation process, the final surface will be smoother than the one from RD [1].

The most important difference between RD and RDSR is that RDSR surface is correlated through the relaxation process. The interface width is another measure for surface roughness, which 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 more natural and smoother.



(a) Random deposition



(b) Random deposition with Surface Relaxation

Figure 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 perimetric branches. (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 [22].

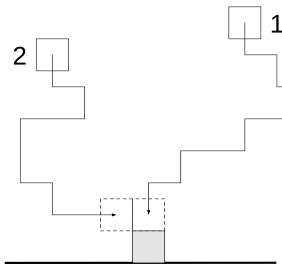


Figure 4: Growth model for DLA

5 Surface Reconstruction

We now explain our surface reconstruction algorithm using level set methods. We mostly follow the level set formulation in [31]. Since the method of [31] is targetted towards the reconstruction of smooth surfaces, we employ a different potential flow for the reconstruction of rough surfaces in the level set formulation.

5.1 Level Set Formulation

In general, the topology of the surface of point sets generated from surface growth simulator is not simple. 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 [20] to capture evolving surfaces by curvature flow and has been successfully used to track interfaces for wide variety of problems. See [19, 24] 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:

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

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

Embed the motion The time evolution PDE for the level set function is obtained by differentiating $\phi(\Gamma(\mathbf{x}, t), t)$.

$$\phi_t + \frac{d\Gamma(\mathbf{x},t)}{dt} \cdot \nabla\phi = 0 \Leftrightarrow \phi_t + v_n|\nabla\phi| = 0 \qquad (3)$$

Here, v_n is the normal velocity of $\Gamma(\mathbf{x},t)$ which may depend on external physics or global, local, or 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

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

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

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

While Zhao used $\mathbf{v}(\mathbf{x}) = -\nabla d(\mathbf{x})$ in [31], 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

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

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).

5.2.1 Computing the Distance Function

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

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

We use the algorithm in [31] 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),

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

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

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

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) satifies

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

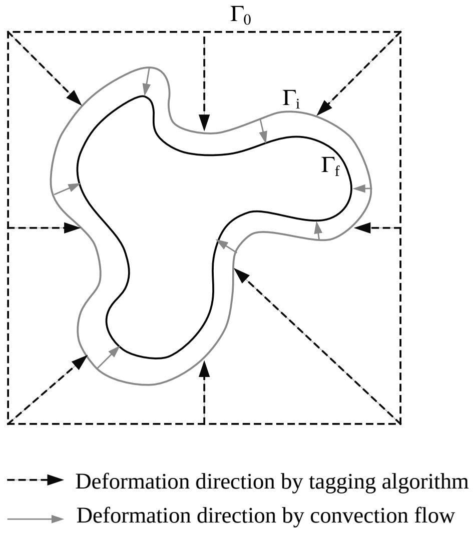

Γ_0
Γ_i
Γ_f
Deformation direction by tagging algorithm
Deformation direction by convection flow

Figure 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

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

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

where $\delta = |x_{min} - y_{min}|$.

Then, the distance function is obtained by solving Eq. (8) on every grid cell in the following four sweeping orderings:

$$\begin{array}{ll} (1) i = 1:n, j = 1:n & (2) i = 1:n, j = n:1 \\ (3) i = n:1, j = n:1 & (4) i = n:1, j = 1:n. \end{array}$$

Usually, the solution converges within five or six sweeps in two dimension, and nine sweeps in three dimension. See [30] for details and proofs.

5.2.2 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 final shape, it will take a long time to evolve the PDE. A good guess for the initial surface helps to reduce the computational cost to get the final surface. A good candidate of initial surfaces is an approximate offset surface such that $\{\mathbf{x} : d(\mathbf{x}) = \epsilon\}$ where ϵ is an offset distance specified by the user. To find such an approximate offset surface, we use a simple tagging algorithm using a region growing method in discrete space.

We always start from an 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 simple tagging algorithm in Alg. 1.

We maintain a priority queue for $\partial\Omega$ so that the largest point can be identified quickly. After tagging, we can obtain the signed

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

```
Require: $S \in \Omega$
  $\epsilon$: offset distance
  while maximum distance of the tagged boundary $\geq \epsilon$ do
    Pick the largest point $\mathbf{x}_{ij} \in \partial\Omega$
    if All interior neighbors of $\mathbf{x}_{ij}$ are closer to $S$ then
      Add $\mathbf{x}_{ij}$ into $\overline{\Omega}$ and Put its interior neighbors into $\partial\Omega$.
    end if
  end while
  The final $\partial\Omega$ is the offset surface.
```

distance function by negating the distance function at all interior cells.

5.2.3 Solving the Level Set PDE

We can continuously deform the initial signed distance function, $\phi(\mathbf{x})$, by solving the level set PDE (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 $O(N^{\frac{2}{3}})$ using the fast local level set method [23]. Instead of computing on every grid cells, it 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 [20, 29].

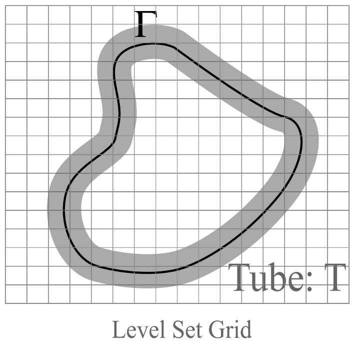


Figure 6: Computation is only performed on the gray region(Tube: T) around the zero level set Γ

We outline the main algorithm.

1. Update tubes, T and N, where

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

   Time advancing step is performed in tube T, while the redistancing step is performed in tube N.

2. Advancing: Update ϕ in tube T for one time step to obtain $\tilde{\phi}$ by an ODE time stepping method. Instead of 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:

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

3. Redistancing: 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 $d_s(\mathbf{x})$:

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

   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:

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

6 Results

We use the *Vispack* library [27] to implement parts of the surface reconstructor. The zero level set or the required implicit surface is extracted by using suitable *Vispack* routines that invoke the marching cube method [13]. The result is available as VRML output. Since VRML is a common 3D 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 MIPS R10000 processor and 1 GByte memory. For 100^3 computational grid, it takes 3-4 minutes to compute distance function and an initial surface respectively. It further takes about 20 seconds for a first-order solver, and a minute for a second-order solver to execute a time step of the level set march. We use a second-order solver to obtain the presented results. Since we use 2-pass algorithm, we employ two computational grids: one is for surface growth simulation, and the other is for surface reconstruction. From now on, *g-grid* stands for surface growth grid and *r-grid* does for surface reconstruction grid.

Fig. 7 shows the deformation of the bunny model by the convection flow given by Eq. (6). The initial surface 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 1. 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 [31]. In the weighed minimal surface model, an additional curvature term regularizes the surface, which is not desired for rough surfaces.

Fig. 8 shows a rough terrain. It is generated from a DLA simulation using a plane as a seed surface. The initial surface with an offset of $\epsilon = 3$ is used for all rough surface examples. Fig. 9 is another example from the DLA simulation with different initial parameters. These examples demonstrate that our method can produce visually appealing natural scenery.

In Fig. 10, we show results generated by both a RDSR simulation(Fig. 10(a)) and a DLA simulation(Fig. 10(b)) on a sphere. Both surfaces are naturally rough and similar; it is noteworthy that the DLA surface looks rougher. This specific result indicates that a RDSR surface can be emulated by a DLA surface by choosing an appropriate reconstruction grid. If the reconstruction grid resolution used is greater than the scale of the underlying surface structure, a smoother surface than the original surface will be obtained. Therefore, the DLA simulation is preferred to RDSR. Also, aggregated structures are generated which can never generated by a RDSR surface.

Fig. 11 illustrates the generative power of our method. In this example, various grids of different scales are used for a DLA simulation on the same sphere as the seed surface. While a global dendritic surface is developed for a coarse grid, fine microstructure on surface is generated for a fine grid. Further, this examples also illustrate how well the level set method captures complex geometry and topology including arches.

Finally, we show some results from a DLA simulation on various objects as initial seed surfaces in Fig. 12. These examples indicate that our technique can be used as a real surface texture generator. It can be used as a form of displacement mapping without the typical weaknesses of those methods.

7 Conclusion and Future Work

We present a new modeling technique for generating rough surfaces. The proposed modeling technique uses a fractal surface growth model and a level set method to extract the surface. Our method is flexible because of the modular design. Fractal surface growth models generate surfaces that are naturally rough. They allows users more control on the final surface characteristics. The implicit representation with a level method allows the handling of complex topology naturally. Thus, we obtain promising results by combining these two methods.

A pressing problem is that some tiny holes are found in the reconstructed surface. We believe that they are caused by the limitation of the Marching Cubes algorithm, which does not guarantee continuous contours. The problem can be fixed by using more robust and efficient isosurface algorithm [17]. Another immediate problem is to control the roughness of the surface perhaps through various parameters including interface width and the fractal dimension. Also, by simulating DLA on complex underlying surfaces, we can produce more interesting results than the typical displacement mapping method. We may use an image as the distribution of initial seeds for DLA. It would be interesting to also consider environmental factors such as gravitational force as well to control surface growth.

A major disadvantage of the level set method is the computation cost and the stability of the PDE solver. Fast and robust methods with suitable tradeoffs for accuracy may be preferable for computer graphics applications. It is likely that we will consider adaptive and semi-lagrangian methods [25]. Finally, it would be useful to add roughness on the level set function grid itself and applying a viable physics-based velocity function such as a dendritic growth of a Stefan problem [2].

Another limitation of our method is the rendering problem of the reconstructed surface. Currently, we extract a polygonal representation from the implicit volume, then render it using a traditional renderer. However, as the grid size becomes bigger, the polygon model may get too large to handle. It would be more efficient to render the surface in the implicit form.

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] Ed D. S. Ebert. *Texturing and Modeling*. Academic Press, New York, 1994.

[4] Julie Dorsey and Pat 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.

[5] Paul 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] Kurt W. Fleischer, David H. Laidlaw, Bena L. Currin, and Alan 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] Geoffrey Y. Gardner. Functional modeling of natural scenes, functional based modeling. *SIGGRAPH Course Notes*, 28:41–49, 1988.

[9] Ned 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] John 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] William E. Lorensen and Harvey 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] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W.H.Freeman and Co., New York, 1982.

[15] Gary A. Mastin, Peter A. Watterberg, and John F. Mareda. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications*, 7(3):16–23, March 1987.

[16] Gavin S. P. Miller. The definition and rendering of terrain maps. *Computer Graphics(Proceedings of ACM SIGGRAPH 86)*, 20(4):39–48, August 1986.

[17] Doug Moore and Joe Warren. *Compact Isocontours from Sampled Data*, pages 23–28. Academic Press, INC, San Diego, 1992.

[18] F. Kenton Musgrave, Craig E. Kolb, and Robert S. Mace. The synthesis and rendering of eroded fractal terrains. *Computer Graphics(Proceedings of ACM SIGGRAPH 89)*, 23(3):41–50, July 1989.

[19] S. Osher and R. Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational Physics*, 169:463–502, 2001.

[20] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based in hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[21] H. O. Peitgen and Saupe Dietmar, editors. *The Science of Fractal Images*. Springer-Verlag, New York, 1988.

[22] H. O. Peitgen, H. Jürgens, and S. Dietmar, editors. *Chaos and Fractals*. Springer-Verlag, New York, 1992.

[23] 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.

[24] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[25] J. Strain. A fast modular semi-lagrangian method for moving interfaces. *Journal of Computational Physics*, 161:512–528, 2000.

[26] Richard F. Voss. Random fractal forgeries. In R. A. Earnshaw, editor, *Fundamental Algorithm for Computer Graphics*. Springer-Verlag, Berlin, 1988.

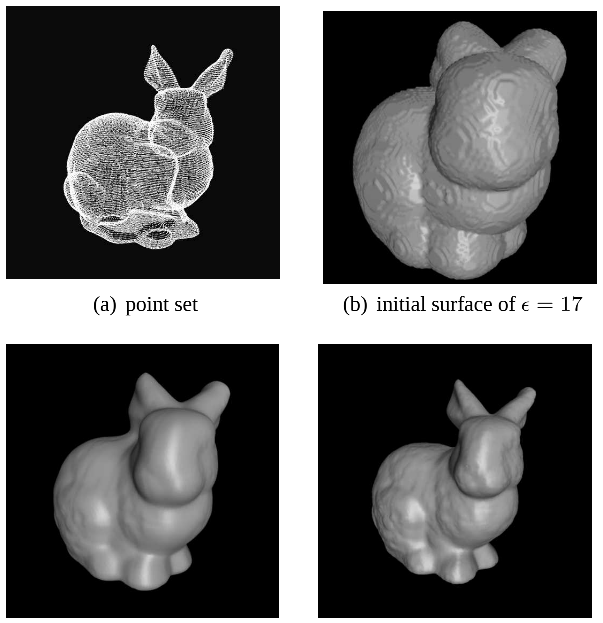
[27] Ross T. Whitaker. Vispack:a c++ object oriented library for processing volumes, images, and level-set surface models. 2002. available from http://www.cs.utah.edu/ whitaker/vispack/index.html.

[28] Steven Worley. A cellular texture basis function. In Holly Rushmeier, editor, *Proceedings of ACM SIGGRAPH 1996*, Computer Graphics Proceedings, Annual Conference Series, pages 291–294, August 1996.

[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] Hong-Kai Zhao. Fast sweeping method for eikonal equations. *preprint*, 2002. available from http://www.math.uci.edu/ zhao/publication/publication.html.

[31] Hong-Kai Zhao, Stanley Osher, and Ronald 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)*, jul 2001.


(a) point set (b) initial surface of $\epsilon = 17$

(c) 100 iterations (d) 200 iterations

Figure 7: Deformation by the convection flow(r-grid: $135 \times 134 \times 112$

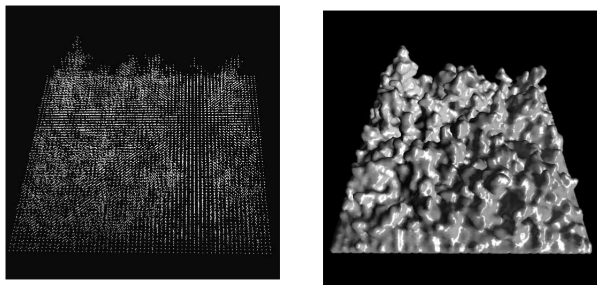

(a) a point set from DLA simulation (b) 30 iterations

Figure 8: DLA simulation 1 on a plane(g-grid: $64 \times 64 \times 64$, r-grid: $115 \times 115 \times 49$)

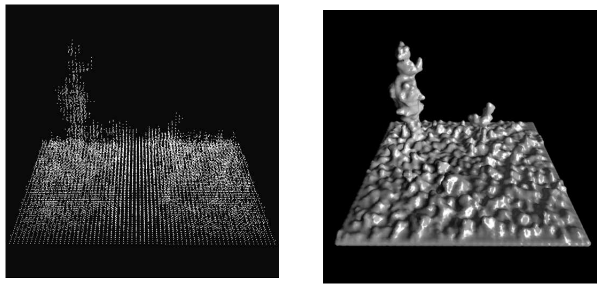
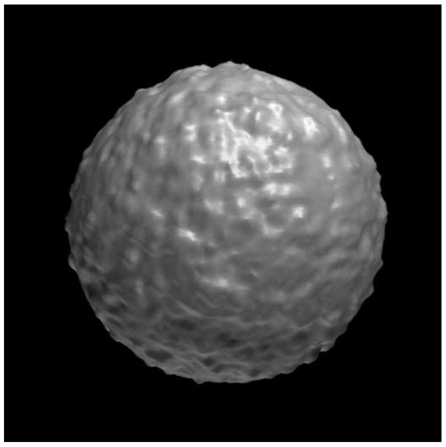

(a) a point set from DLA simulation (b) 30 iterations

Figure 9: DLA simulation 2 on a plane(g-grid: $64 \times 64 \times 64$, r-grid: $115 \times 115 \times 75$)


(a) RDSR on a sphere(r-grid: $112 \times 115 \times 114$)

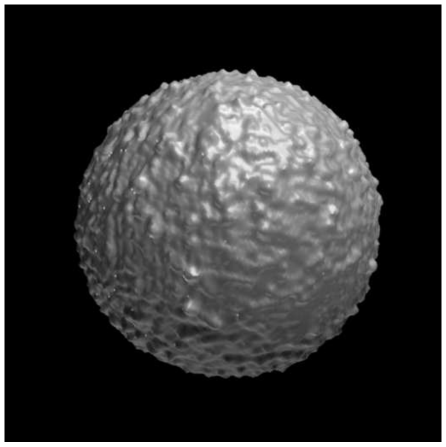
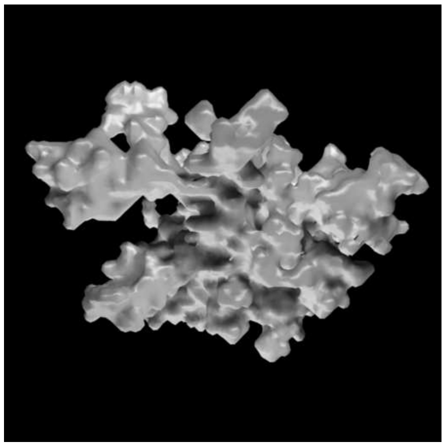
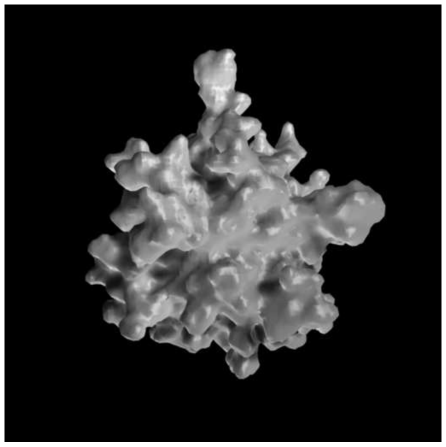
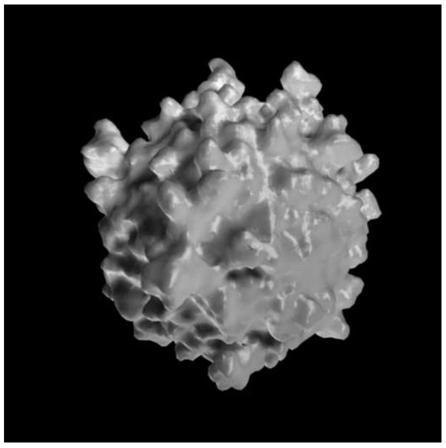

(b) DLA on a sphere(r-grid: $139 \times 139 \times 139$)

Figure 10: Comparison between RDSR and DLA simulation on a sphere


(a) g-grid: 10^3


(b) g-grid: 20^3


(c) g-grid: 40^3

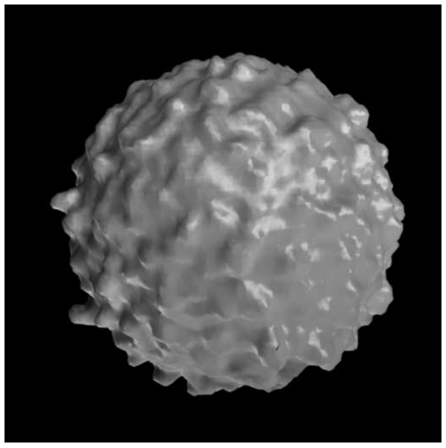
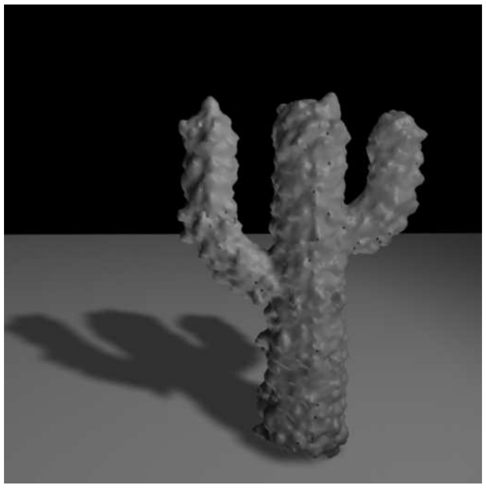
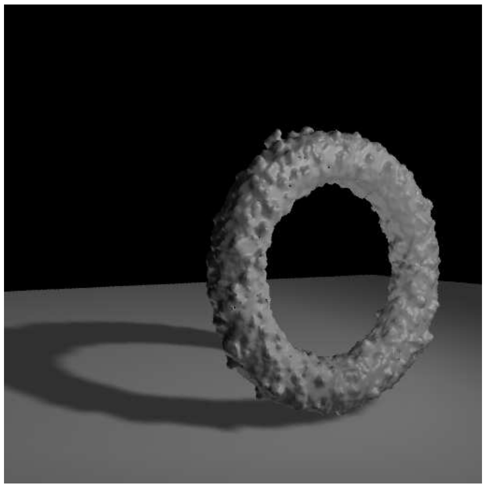
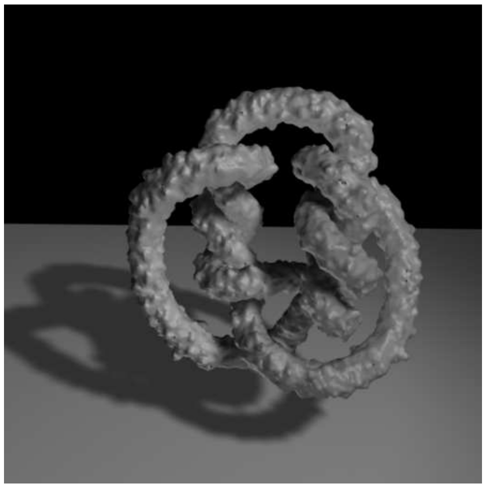

(d) g-grid: 80^3

Figure 11: DLA surfaces generated by using different growth simulation grids on a sphere(r-grid: $89 \times 91 \times 89$)


(a) cactus: $43 \times 95 \times 139$


(b) torus: $139 \times 40 \times 137$


(c) knot: $138 \times 139 \times 78$

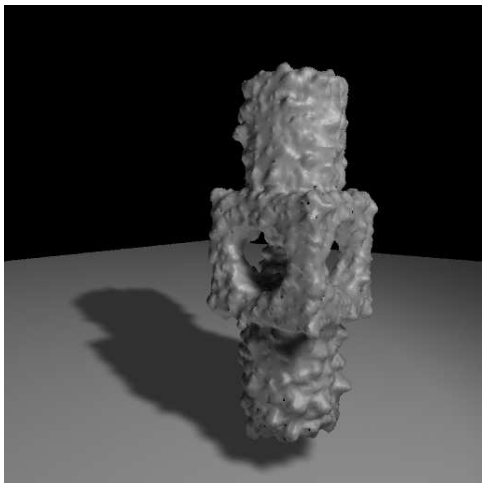

(d) mechanical part: $61 \times 59 \times 139$

Figure 12: DLA on various objects(All grid sizes are r-grid.)