Eurographics Symposium on Point-Based Graphics (2007)
M. Botsch, R. Pajarola (Editors)

Sampling Point-Set Implicits

J. Proença[1], J. A. Jorge[1], M. C. Sousa[2]

[1]Dept. Information Systems and Computer Engineering, TU Lisbon, Portugal
[2]Dept. Computer Science, University of Calgary, Canada

Abstract
We present a novel approach for point-set implicit surface sampling that is able to rapidly distribute particles over the surface of 3D objects. Our methods benefit from the inner structure of a MPU implicit to obtain a near-optimal initial distribution, with higher densities in areas of higher complexity. The adequate number of particles to effectively sample the surface is determined almost automatically and in accordance to the surface characteristics. We also use the MPU information to obtain local surface complexity heuristics that are used in the simulation stage of the particle system. Shape modeling operations are directly supported and the redistribution of particles is very fast for local edits. We present performance results that show that our system is faster than other state-of-the art approaches for the same number of points.

Categories and Subject Descriptors (according to ACM CCS): Sampling, approximation, and interpolation.

1. Introduction

Implicit surfaces provide simple and flexible mathematical representations for 3D objects that allow for precise shape definition and modeling. One particular subgroup, point-set implicits, is able to reconstruct complex surfaces from point clouds that can be obtained from 3D scans of real objects.

However, sampling this type of representation is a difficult problem, because of its complexity and continuous nature. Creating a particle system that scatters points over the surface has been a commonly used solution. In 1994, Witkin and Heckbert [WH94] presented an approach to distribute particles over implicit surfaces using attractor and repulsion forces. This set the standard for other systems that used similar methods to find areas of interest or render the surface through points or splats [Elb98, FJW*05, JWS06, LGS06].

One of the main difficulties when simulating such a particle system is the number of iterations (or relaxation steps) that are necessary to converge into an optimal distribution and the associated evaluations of the potential function that generate a large computational overhead. In more recent years, new methods have emerged that produce near-optimal initial distributions to shorten this simulation stage [LGS06].

In this paper, we present a novel approach that is able to quickly distribute particles over a point-set implicit surface with higher densities in areas of higher detail. We use Multi-level Partition of Unity (MPU) implicits [OBA*03] that are able to represent complex surfaces retrieved from dense point clouds. The MPU is also characterized by its internal octree structure that establishes a 3D cell subdivision of the surrounding space of the surface, where cells are smaller and more numerous in areas of higher complexity. The main contributions of our work are:

- We use the descriptive power of the MPU octree to rapidly generate a near-optimal initial distribution of particles that is adaptive according to the local surface complexity. Therefore, we only need a limited number of relaxation steps in the simulation stage to obtain a good distribution of particles.
- Our method is able to determine the appropriate number of particles according to the surface topology and complexity, without user intervention.

In addition, we use surface complexity heuristics obtained from the octree cells that allow us to avoid computing the principal curvature values during the simulation of the particle system for the same purpose, thus accelerating the process. These heuristics can also be used with other adaptive octree structures, that may be built for various types of point-set implicits. The particle system is able to regenerate itself after modeling operations, by applying a new simulation step that is confined to the area of the shape edit. This regeneration occurs quickly for localized modifications. The initial generation of particles is fast and we are able to obtain excellent performance levels even for high complexity models. These results compare very favourably to other state-of-the-art approaches.

2. Related Work

Implicit surface sampling using particles was first introduced by Figueiredo et al. [dFdMGTV92], who used attraction and repulsion forces to create point distributions for polygonization. Witkin and Heckbert [WH94] developed this concept further in a modeling application, by using an adaptive repulsion and split-and-death criteria for a particle scattering that minimizes an energy criterion. While either papers does not support non-uniform distributions, the elements introduced in terms of physical-based particle systems paved the way for a series of approaches.

Hart et al. [HBJF02] extended the Witkin-Heckbert approach to uniformly sample more complex surfaces, being able to numerically differentiate implicits defined by large numbers of control parameters. Rosch et al. [RRS02] used curvature information to sample unbounded surfaces and singularities. Crossno et al. [CA97] also conceived an extension to Witkin-Heckberth for sampling iso-surfaces extracted from volume data. Meyer et al. [MGW05] presented a new class of energy functions and numerical techniques for obtaining uniform or non-uniform distributions of particles over implicits, but while their methods use less relaxation steps than [WH94], each step consumes more computation time.

Foster et al. [FJW*05] presented pen-and-ink rendering styles over implicits that use a Witkin-Heckbert type particle system for surface sampling, similarly to [Elb98]. They initialize particles through the intersection of randomly generated rays with the surface. The system is also able to distribute particles through attraction and repulsive forces. They provide uniform or non-uniform distributions and in the latter the densities may vary according to the surface curvature or the proximity to the viewing position. More recently, [JWS06] presented an extension of this work, in which particles originate smarticles, i.e. flocks of points that inspect the surface searching for areas of interest or to create groups of lines that represent features spanning specific regions.

One of the major problems common to particle systems is the computational cost of the relaxation process, which becomes expensive beyond a few thousand points. Levet et al. [LGS06] presented an approach applied to analytic and VIS [TO99] implicit representations that addresses the problem by rapidly generating a set of initial particles with near-optimal positions. They perform a pure geometric initialization, in which particles are iteratively added to the point-set in a process akin to surface-tracking polygonization. This optimized initial placement requires less relaxation steps to achieve either uniform or non-uniform distributions.

2.1. The MPU Surface

The MPU [OBA*03] efficiently constructs implicit representations from dense sets of control points sampled on the surface of complex objects, using three elements: an octree of cubic spatial cells that cover the object; quadratic functions that approximate the local shape in each cell and weight functions that blend the local functions, thus providing a precision-controllable approximation to a complex implicit surface efficiently. This process is guided by the subdivision of the octree structure, where cells become smaller and more numerous in areas where the point positions and normals suggest higher variations of curvature.

3. Overview

Our approach uses a Witkin-Heckbert type particle system to create non-uniform distributions, where particles are more concentrated in areas of higher detail. We use the inner octree structure of the MPU implicit to guide the definition of the initial positions of particles and to obtain local surface complexity heuristics to support the adaptive point densities. For this reason, we cannot say that we treat the potential function as a *black box*, such as in [FJW*05, JWS06], since our methods are connected to the specificities of the octree of the MPU. However, in terms of mathematical information, we only query the potential function to extract its value and gradient. Because of this, we can say that our techniques can be used with other point-set implicit representations as long as an adaptive octree structure is constructed for the associated control-point datasets (using various processes for octree sampling such as in [JLSW02, SW04]).

We also store references to every particle in the corresponding octree cell, in order to be able to use this structure for rapid determination of particles within a fixed radius. This is usually done using a regular voxel grid, but we acknowledged the benefits of using the octree, since we expect the particle densities to follow the cell subdivision level along the surface (therefore, we have an adaptive voxel grid).

As soon as an object is loaded from a point cloud, the MPU implicit surface is generated and the associated potential function is defined for the whole surrounding space. The particle system is then initialized and an initial distribution of points is created (Section 4). Afterwards, a physical simulation stage is performed, where particles suffer repulsion and attractor forces (Section 5). It comprises a fixed number of relaxation steps that improve the quality of the overall distribution. If the user applies some sort of modeling operation to the shape of the object, there is a local regeneration of the MPU and the particle system in the affected surface area (Section 6). All of these elements of our approach are performed with very good efficiency and we will provide performance results in Section 7.

4. Creating the Initial Distribution

To create the initial set of particles we use the spatial complexity descriptive power that the MPU's octree gives us and we put a constant number of particles on the surface for each cell. Since cells are smaller and more abundant in areas of higher complexity, we get a higher point concentration where there are more surface details, right from the start.

The first step in this process is to determine which octree cells contain portions of the surface. Since the MPU potential function $f(\mathbf{x})$ covers the entire 3D space, many cells do not cover any surface region, i.e. cells where $f(\mathbf{x}) \neq 0$. We store every control point from the original point cloud (that originated the MPU representation) in the corresponding octree cell that covers its surrounding 3D region. All the cells that contain these points are marked as covering the surface, which gives us a very straightforward process to guarantee that all the surface areas are covered by our distribution (since the MPU is defined by the control points). There is always the possibility that certain octree cells will not contain any control points, but this can only happen when the surface is only lightly captured by a particular cell, because the local surface definition was created from nearby control points. We do not identify these cells explicitly because they are not relevant to the initial distribution of points (the simulation stage however might push particles into these volumes).

In each surface-containing cell, we select a k number of control points and use their positions and surface normals to generate particles. If there is an insufficient number of points in a particular cell (because k is larger than the number of control points that are contained in it), new points are generated with random positions inside the cell. These points undergo a Newton step afterwards to ensure surface adherence (a Newton step is a well known root-finding method for potential functions that iteratively approximates a point to the root using the local gradient).

While this approach is general for any number k of particles initially created per octree cell, we came to the conclusion that we can always use $k = 1$ to effectively cover the surface with an appropriate quantity of particles (all of the models presented in this paper use that value). This means that we create the initial set of particles as a subset of the points in the point cloud that originated the MPU surface, since we already know that each identified cell contains at least one control point. This is a benefit in terms of performance because it allows us to use points that already lay on the surface, so we do not need to use the computationally expensive Newton Step to bring the particles to the surface. Furthermore, selecting the number of particles adequate for an appropriate coverage becomes automatic, because the set of particles naturally reflects the number of smaller or larger octree cells and therefore the overall surface topology and complexity.

This approach also provides a near-optimal initial placing of particles, which drastically decreases the number of subsequent simulation iterations needed to obtain an optimal distribution.

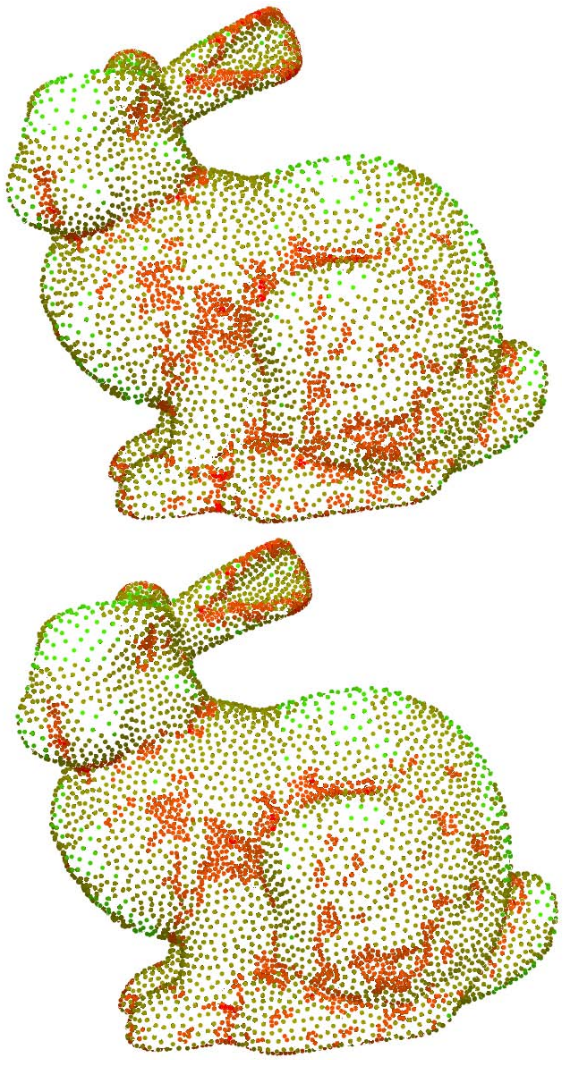
Figure 1: *Particle initialization and simulation in the Bunny model. Top: the set of initial particles obtained from our octree-driven approach. Bottom: after 20 simulation iterations, we obtain an optimal non-uniform distribution.*

5. The Simulation Stage

The particle system simulation process is composed by a set of relaxation steps where at each step every particle suffers attraction forces to the surface and repulsion forces from nearby points and moves accordingly. The calculation of these forces, as well as the movement of each particle, is performed in the same way as in [FJW*05], except for the repulsion distribution factor δ_i, which directly influences the repulsion force and controls the local density of particles. Foster et al. calculate it using Hessian values extracted from the potential function to determine local mean curvature. This poses a significant computational burden; we avoid this altogether by using the octree cell depth information on the MPU instead. The depth indicates the cell subdivision degree and therefore is a good heuristic for determining the local complexity of the surface, which yields a simpler formula for δ_i:

$$\delta_i = weight\left(\frac{d_i - d_{min}}{d_{max} - d_{min}}\right) \tag{1}$$

where d_i is the cell depth of particle P_i and d_{max} and d_{min} are the maximum and minimum cell depths where particles exist in the model (these are determined during the initial particle generation step). The *weight* function is an adapted quadratic B-Spline [OBA*03] that sets a higher value for δ_i if the particle is in one of the lower depth cells (less curved regions of the surface) and vice versa.

$$weight(t) = \begin{cases} 1 - 3(\frac{t}{1.05})^2, & \text{if } t < \frac{1}{3} \\ 1.5(1 - \frac{t}{1.05})^2, & \text{if } t \geq \frac{1}{3} \\ 0, & \text{if } t > 1 \end{cases} \tag{2}$$

The net effect of using this technique is that it is easy and fast to obtain good results where the particles become well distributed over the surface to reflect local complexity with comparatively less steps and a lower cost per particle. However, using the cell level, instead of the Hessian curvature, has the effect of discretizing the values of δ_i while accelerating its computation, which in turn originates discrete particle density levels along the surface, instead of a more continuous density variation. Nevertheless, this is only a particularity of our strategy that does not significantly degrade the final distribution.

6. Regenerating Particles

One of the main advantages of using the MPU implicit surface for representing 3D objects is that its octree structure allows to locally recompute the surface after a shape edit. In such an operation, the octree cells of the affected surface area are marked and new control points are generated to define the new shape characteristics of the object. In these cells, the MPU suffers a regeneration process that resembles the one that is performed upon its creation with the difference that it is confined only to the cells where the shape is really modified. This allows for a faster local redefinition that avoids a total re-computation of the implicit representation.

Since the MPU provides a straightforward way of knowing which parts of the surface are altered in a shape edit, through the marking of affected octree cells, we only have to regenerate particles in those cells. More specifically, after a modeling operation, the relevant octree cells are marked and altered in the MPU regeneration process (the subdivision depth of the cells may even change). During this stage, we delete all of the particles that were contained in those cells. After the MPU regeneration, we insert new points in the particle system for the new octree cells, in the same way as it is described in Section 4. Then a new particle system simulation process is started with the difference that it only affects the particles that belong to the marked cells. This means that we only compute the attractor and repulsion forces for points inside the affected area and do not allow particle movements that result in placing points outside the marked cells (this is done to avoid particle concentration along the border of the affected surface area). Figure 2 presents an example of a shape edit applied to the *Cow* model, where we used 2D stroke over-sketching to apply 3D shape modifications to the MPU surface. In the zoomed in figure of the modeled area, we can see how only in the affected region (the top of the *Cow*'s back) there are modifications in the particle distribution, while in other areas it remains unaltered (the belly).

Since this process only affects a subset of the particles (depending on the size of the modeling operation), its total computation time is shorter than for simulating of the entire particle system. In fact, for the usual editing operations that affect a small portion of the surface, the local redistribution of points only takes a few seconds even for the more complex models (after the local re-computation of the MPU). For shape edits that affect larger areas of the overall surface, rebuilding the MPU takes more time and the particle system regeneration requires almost the same time as the initial distribution process.

7. Results and Discussion

The particle system solution that we have conceived is able to rapidly scatter particles over complex surfaces defined by MPU implicits, using a non-uniform type of distribution. Figures show some examples of the final results we get for some well known models with different complexity levels. In Table 1 we can see the performance results for each of the presented test models. These values were gathered using a 3.6 GHz Pentium IV with 2 gigabytes of RAM and a NVIDIA Quadro FX 3400 graphics card, running Windows XP SP2.

As we can see, the number of particles that are selected for each model is not necessarily proportional to the number of points in the dataset that originates the MPU, because it is also influenced by the surface topology and complexity. Therefore, our particle selection strategy proves to be very effective in terms of finding an adequate amount of particles to sample the surface, without any need for parameter adjustments by the user.

Little time is required to initialize the particle system, taking less than one second for simpler models such as the *Bunny*, *Cow* and *Horse* and about four seconds for more complex examples like the *Phlegmatic Dragon* and *David's Head* (with 60000 particles created from 800000 dataset points).

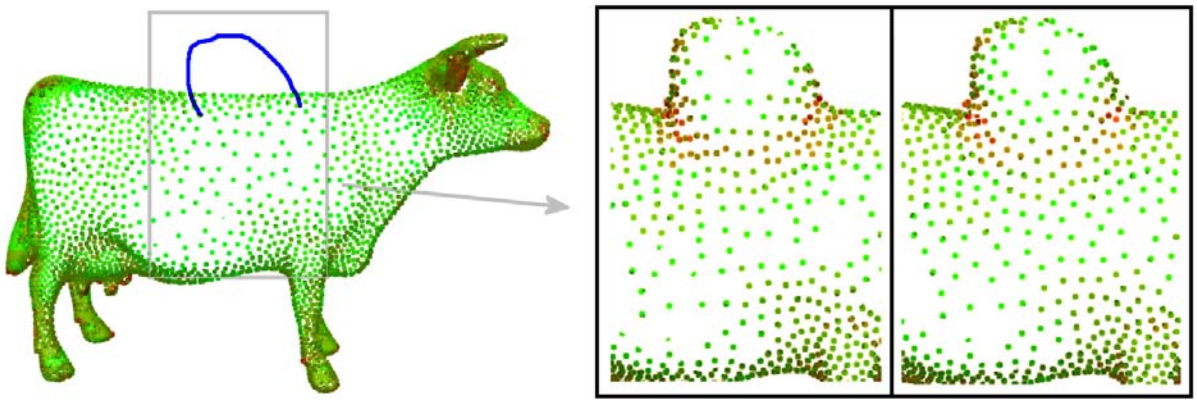

Figure 2: *Particle regeneration in the Cow model after a local shape edit. Left: we apply a 3D shape modification in the back of the Cow through a 2D stroke. Right: the zoomed in figures show the initial re-initialization of particles on the left and the result that is obtained after the local redistribution of particles on the right, which lasts only for a few seconds.*

Model	Dataset Points	Particle System	Init. Time.	Simul. Time
Bunny	69451	10000	0.3 sec	29 sec
Cow	92864	8334	0.4 sec	31 sec
Horse	96966	6668	0.4 sec	17 sec
Igea	268686	13665	1.1 sec	42 sec
Armadillo	345944	28360	1.5 sec	176 sec
Dragon	480076	26984	2.3 sec	112 sec
David	827181	59994	3.7 sec	789 sec

Table 1: *Particle system results. Each row indicates the model, number of points in the dataset, number of points in the particle system, time spent in the particle initialization using one particle per octree cell and time spent by the simulation stage with 20 iterations performed.*

Our Method				
Model	Implicit Repr.	Particle System	Init. Time.	Simul. Time
Bunny	MPU	10000	0.3 sec	15 sec
David	MPU	59994	3.7 sec	433 sec
Levet et al. [LGS06]				
Bunny	VIS	12388	313 sec Total	
Ellipsoid	Analytic	55748	25 sec	589 sec

Table 2: *Particle system comparison to Levet et al. [LGS06]. The top table presents our results and the bottom one the ones from Levet et al. Each row indicates the model, type of implicit representation, number of points in the particle system, time spent in the particle initialization and time spent by the simulation stage with 10 iterations performed.*

After obtaining the initial near-optimal positions for particles, we apply 20 iterations in the physical simulation stage. This number was chosen for all the tests, since it allows us to achieve adequate particle distributions. The results at this stage suggest that the simulation time is affected by the complexity of the surface and number of particles. This is why *David's Head* takes a much larger simulation time in comparison to the other models, since the MPU definition must cover a lot of sharp features.

We can compare our results to those of Levet et al. [LGS06], because they also present a strategy for near-optimal initial placement of points to generate non-uniform distributions of particles. They present the initialization and simulation times for their system using VIS implicit surfaces [TO99] or analytical representations, with a hardware configuration not much worse than ours. In Table 2 the results from both systems using a similar number of particles and 10 simulation iterations are presented. In the *Bunny* model, defined by a VIS by Levet et al., they perform particle initialization and the simulation process in 313 seconds, while we perform the same task in about 15 seconds. Our times are about 20 times faster for a similar number of points, but it can be argued that the VIS implicit evaluations have a greater computational overhead than the MPU.

However, when comparing the ellipsoid results from Levet et al., defined by an analytic implicit (which obviously is a much simpler representation than the MPU), to our *David's Head* results, the advantages of our method become clearer. Indeed, while they need 25 seconds to set the initial positions for 55748 particles, we only require 3.7 seconds to compute the initial positions of 59994 particles in the *David's Head* model. It takes 589 seconds for the simu-

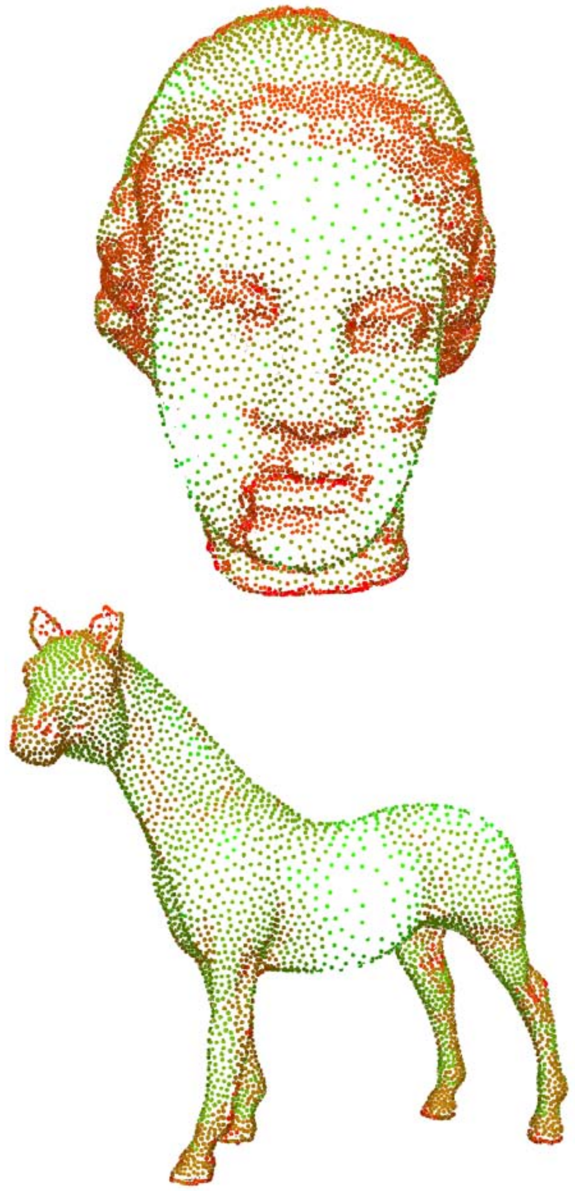
Figure 3: *Final particle distribution results for the Igea (Top) and Horse (Bottom) models.*

lation stage in the ellipsoid, while we distribute particles in 433 seconds in our model. In this case, our results are seven times faster for the initial placing of particles and the simulation stage takes 74% of the time used in [LGS06], while we are using an implicit representation that is much more computationally expensive to evaluate, a more complex surface shape and a greater number of particles.

We can conclude that our performance results compare very favourably to the ones from Levet et al., which can be justified by the fact that, in the simulation stage, they need to perform function evaluations and geometrical computations that we avoid with our octree-driven approach, and in the simulation stage they extract curvature values from the implicit representation, while we use the much faster octree cell depth heuristic. Their approach is more general than ours, as theoretically it can be applied to any type of implicit, but in terms of performance it does not deal well with medium or high complexity surfaces. Our approach is more directly connected to the octree structure and uses the MPU implicit, which allows for fast sampling of surfaces obtained from hundreds of thousands of points and featuring lots of intricate shape details.

However, it is important to refer certain limitations from our approach. Since our particle simulation strategy is somewhat similar to [FJW*05], there are some situations where we require the user to adjust a repulsion force parameter to the surface scale and topology, in order to avoid particles escaping from the surface near discontinuities. While the particle system can handle occasional holes in the MPU, due to defective dataset point clouds, it does not support incomplete open surfaces also.

8. Conclusions and Future Work

We have presented methods for fast sampling of point-based MPU implicit surfaces, where we harness the descriptive power of the surface representation to provide rapid near-optimal initialization of particles. The number of particles is automatically adjusted to the surface complexity and topology, regardless of its size and scale. This initial scattering of particles shortens the subsequent simulation stage that is necessary for achieving an optimal distribution. We also use octree heuristics to adjust the particle density to the local surface complexity, thus avoiding computationally expensive curvature evaluations. Our methods work well for different levels of surface complexity and also support shape edits, being able to redistribute particles in modeled areas. This regeneration occurs rapidly for local edits.

Our approach shows good performance results that improve on the state-of-the-art even for highly detailed models. We compare the particle initialization and simulation times to the strategy from [LGS06], for similar numbers of particles. Our approach is faster even when compared to simpler, easier to evaluate analytic representations. However, some areas of further improvement remain. While we already provide automatic initialization and selection of the number of particles, methods for automatically adapting the repulsive forces would eliminate any need for user intervention at the simulation stage. Another area of improvement is to provide support for distributing particles over open surfaces, since many point-cloud datasets only include limited portions of real surfaces.

Acknowledgments: This work was supported in part by

European Commission grant No. IST-034525 (SATIN) and by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada.

The models are courtesy of the Digital Michelangelo Project 3D Model Repository (*David's Head*), the Stanford 3D Scanning Repository (*Bunny*, *Cow* and *Armadillo*), Cyberware (*Igea*) and UTIA, Academy of Sciences of the Czech Republic, and CGG, Czech Technical University in Prague (*Phlegmatic Dragon*).

References

[CA97] CROSSNO P., ANGEL E.: Isosurface extraction using particle systems. In *IEEE Visualization '97* (1997), pp. 495–498.

[dFdMGTV92] DE FIGUEIREDO L. H., DE MIRANDA GOMEZ J., TERZOPOULOS D., VELHO L.: Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface'92* (Vancouver, Canada, 1992), pp. 250–257.

[Elb98] ELBER G.: Line art illustrations of parametric and implicit forms. *IEEE Trans. Vis. Comp. Graph. 4*, 1 (Jan./Mar. 1998), 71–81.

[FJW*05] FOSTER K., JEPP P., WYVILL B., SOUSA M. C., GALBRAITH C., JORGE J. A.: Pen-and-ink for BlobTree implicit models. *Computer Graphics Forum 24*, 3 (2005), 267–276.

[HBJF02] HART J. C., BACHTA E., JAROSZ W., FLEURY T.: Using particles to sample and control more complex implicit surfaces. In *Proc. of Shape Modeling International* (2002), p. 269.

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. In *Proc. of ACM SIGGRAPH02* (2002), pp. 339–346.

[JWS06] JEPP P., WYVILL B., SOUSA M. C.: Smarticles for sampling and rendering implicit models. *Theory and Practice of Computer Graphics 2006* (2006), 39–46.

[LGS06] LEVET F., GRANIER X., SCHLICK C.: Fast sampling of implicit surfaces by particle systems. In *Proc. of Shape Modeling International* (July 2006), p. 39.

[MGW05] MEYER M., GEORGEL P., WHITAKER R.: Robust particle systems for curvature dependent sampling of implicit surfaces. In *Proc. of the International Conference on Shape Modeling and Applications (SMI'05)* (2005), pp. 124–133.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3 (2003), 463–470.

[RRS02] ROSCH A., RUHL M., SAUPE D.: Interactive visualization of implicit surfaces with singularities. 295–306.

[SW04] SCHAEFER S., WARREN J.: Dual marching cubes: Primal contouring of dual grids. In *Proc. of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)* (2004), pp. 70–76.

[TO99] TURK G., O'BRIEN J. F.: Shape transformation using variational implicit functions. In *Proc. of ACM SIGGRAPH 1999* (Aug. 1999), pp. 335–342.

[WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. In *Proc. of SIGGRAPH '94* (1994), pp. 269–277.

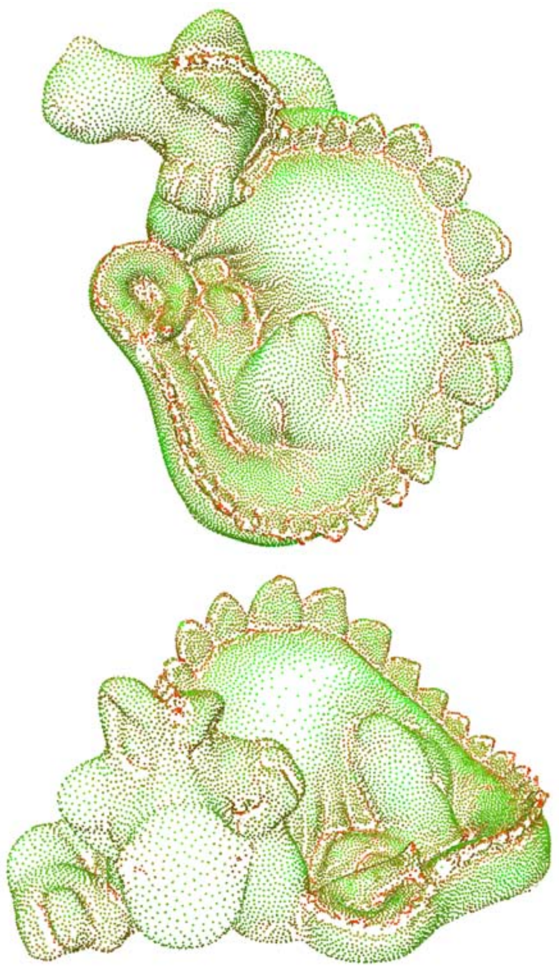
Figure 4: *Final particle distribution results for the Phlegmatic Dragon model.*

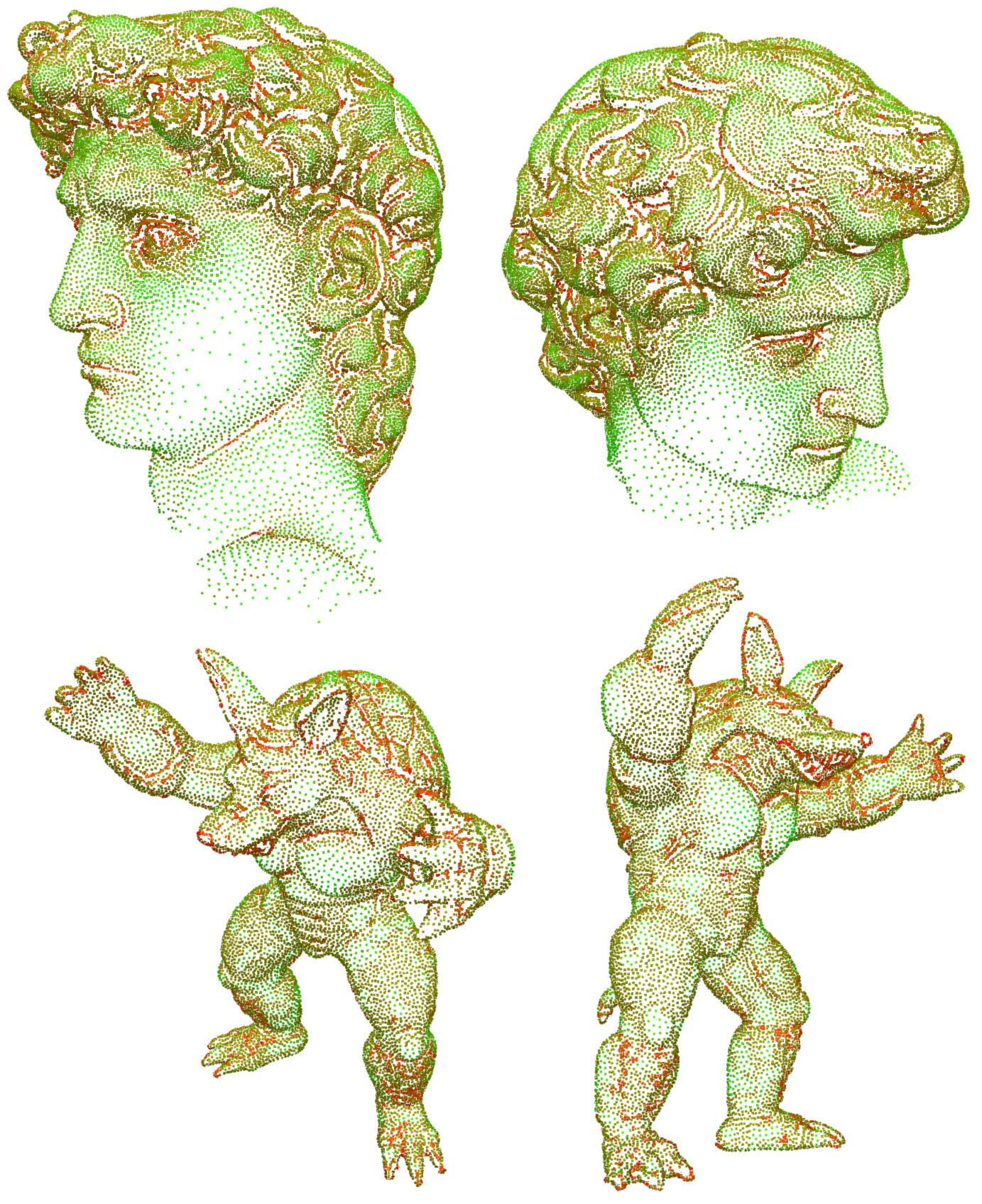

Figure 5: *Final particle distribution results for the David's Head (Top) and Armadillo (Bottom) models. Red points correspond to octree cells of higher depth (particles are more concentrated), green points to lower depth.*