MarchingParticles: Fast Generation of Particles for the Sampling of Implicit Surfaces

Florian Levet, Xavier Granier, Christophe Schlick

▶ To cite this version:

Florian Levet, Xavier Granier, Christophe Schlick. MarchingParticles: Fast Generation of Particles for the Sampling of Implicit Surfaces. Computer graphics & geometry, 2007, 9 (1), pp.18-49. inria-00187183

HAL Id: inria-00187183
https://inria.hal.science/inria-00187183

Submitted on 6 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MarchingParticles: Fast Generation of Particles for the Sampling of Implicit Surfaces

F. Levet, X. Granier, C. Schlick
IPARLA project (INRIA futurs & LaBRI UMR 5800, Université Bordeaux 1)
351, cours de la Libération - 33405 Talence, France

{levet,granier,schlick}@labri.fr

Abstract:

Particle systems, as originally proposed by Witkin and Heckbert [32], are a powerful way to sample implicit surfaces since they generate almost evenly distributed samples over the surface, thanks to a global minimization of an energy criterion. Nonetheless, due to the computational cost of the relaxation process, the sampling process becomes rather expensive when the number of samples exceeds a few thousands.

In this paper, we propose a technique that only relies on a pure geometry processing which enables us to rapidly generate the set of final particles (e.g. half a second to generate 5,000 particles for an analytic implicit surface) with near-optimal positions. Because of its characteristics, the technique does not need the usual splitand-death criterion anymore and only about ten relaxation steps are necessary to get a high quality sampling. Either uniform or non-uniform sampling can be performed with our technique.

1. Introduction

Implicit surfaces are an elegant representation for 3D modeling without explicitly having to account for topology issues. Moreover, it is possible to develop a complete modeling-animating-rendering pipeline with almost no topological constraints by using ray-tracing to render the corresponding surfaces. Unfortunately, for an interactive rendering of implicit surfaces, there is usually no other choice than a conversion into polygonal meshes, which inherently reintroduces heavy topological constraints.

In 1994, Witkin and Heckbert [32] gathered some existing ideas [25,26,8,27] in a powerful tool designed to both sample and control implicit surfaces. Starting from a seed particle, the complete surface is sampled thanks to a repulsion scheme and a split-and-death criterion. Each particle repels nearby particles to minimize a Gaussian energy function. When a particle is on an under-sampled region it will split while it will die on an oversampled region. Finally, each particle maintains an adaptive repulsion radius that can grow or shrink based on its local neighborhood. The final result obtained with their method is a set of uniform particles almost evenly distributed on the surface.

Some following works have improved this basic scheme by achieving either a non-uniform isotropic sampling [7,21], an anisotropic sampling [16] or by changing the Gaussian energy function with forces [20] or with new classes of energy functions [19].

Nonetheless, none of these works have focused on the two main drawbacks of particle systems: the computational cost of the algorithm and the convergence of the system. When determining the next position of a particle, one has to compute its intersections with all its neighboring particles. As this process has to be done for each particle, sampling a surface with more than a few thousand particles is too expensive for practical applications. Besides, even when a surface is well sampled, particles keep moving. Thus, finding a good convergence criterion that works for all configurations is not an easy task.

The goal of this paper is to enable the use of particle systems with more than a few thousands. The basic idea is to rapidly generate the set of final particles with near optimal positions by using pure geometry processing. Therefore,

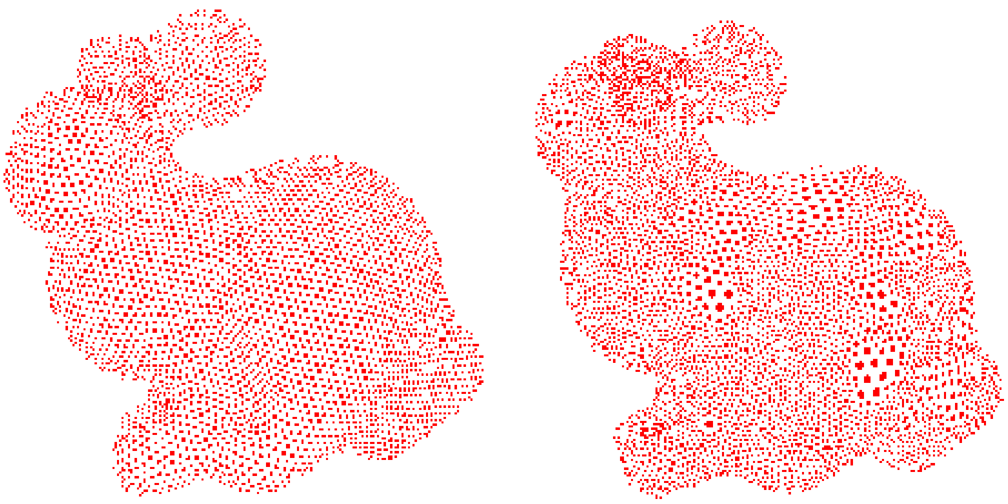


Figure 1 Two samples distributions of the bunny: (left) with uniform particles and (right) with non-uniform particles.

only ten relaxation steps are needed to get a high quality sampling and to avoid the convergence problem. Either uniform or non-uniform sampling can be performed with our technique (as seen in Figure 1).

The remainder of the paper is organized as follows: Section 2 presents some related previous work dealing with sampling techniques for implicit surfaces. Section 3 presents the motivation of our work. Section 4 details our generation algorithm of uniform particles while Section 5 focuses on the generation of non-uniform particles. Section 6 presents several experimental results obtained with our sampling method. Section 7 concludes and presents some directions we are currently investigating.

2. Previous work

Basically, existing sampling techniques of implicit surfaces can be divided into two main families: tessellating techniques and particle system techniques.

Tessellating implicit surfaces

The tessellating techniques of implicit surfaces can themselves be divided into three categories. First, *spatial sampling techniques* subdivide the 3D space into cells, commonly either cubes or tetrahedrons, and search for the cells that intersect the implicit surface. One of the most commonly known spatial sampling techniques is the marching cubes algorithm independently developed by Wyvill et al. [34] and Lorensen et al. [18]. Its principle is to divide the 3D space into cubic cells, and triangles according to the sign of the implicit function at the corners of the cells. Marching tetrahedra algorithms, by Shirley and Tuchman [22] or Hall and Warren [12], further divide the cubic cells into tetrahedra which avoids the ambiguous configurations that may arise with cubic cells. Both the marching cubes and the marching tetrahedra algorithms employ cells of constant size; so both techniques may miss small geometric features, do not adapt to the local geometry of the implicit surface, and require some knowledge about the topology of the implicit surface. To overcome this drawback, adaptive subdivision techniques that converge to the surface recursively have been developed [3], but with such techniques cracks may occur between triangles of adjacent cells of different size. Azernikov and Fisher [2] proposed a technique to deform the volumetric grid toward the object's shape. A high quality anisotropic sampling of implicit surfaces is obtained at the expense of a high computational cost.

The second category are *surface fitting techniques* that create a seed mesh that roughly approximates the implicit surface and progressively adapt and deform it to better fit the implicit surface. For example, Velho [30,31] starts with a coarse polygonal approximation of the surface [11] and subdivides each polygon recursively according to the local curvature. But still there must be some a priori knowledge about the topology of the surface since the coarse polygonal approximation has to capture the correct topology of the implicit surface. Other surface fitting techniques either assume special classes of implicit surfaces created from skeletal elements [10,6], or rely on a search for critical points [29,5,23,33] suffering from inefficiency for complex implicit surfaces.

The third category are *surface tracking techniques* (or *continuation techniques*) that start from a seed element on the surface and iteratively grow a polygonal mesh that approximates the implicit surface. Cellular surface tracking techniques [1,34,4] start from a cell that intersects the implicit surface and iteratively find all intersecting cells among its neighbors. Since the cells are of constant size, cellular surface tracking techniques suffer from the same drawbacks as non-adaptive spatial sampling techniques, and furthermore, in the general case, it can be difficult to determine a seed cell.

Particle systems

The second way to sample implicit surfaces is to use so-called *particle systems* that evenly distribute samples over the implicit surface. The first time that particles were used to sample a surface was in a complete modeling tool based on oriented particle systems developed by Szeliski and Tonnesen [25], but in their work there was no underlying implicit surface. Turk [26] used a similar process in order to generate textures using a reaction-diffusion method. Figueiredo et al. introduced [8] a system to sample implicit surfaces using particles.

Even if the repulsion forces applied to the particles were not the same as the one proposed by Turk, the authors used a similar relaxation process in order to achieve a uniform distribution of the sample points which are then used to compute a polygonal approximation of the implicit surface. Turk's reaction-diffusion method has also be used to re-tile polygonal surfaces [27] according to the curvature, to get more points in regions of high curvature.

Witkin and Heckbert [32] developed a powerful modeling application by putting together some of these existing ideas. Indeed, they demonstrated that particle systems are useful in order to both display and control implicit surfaces. The authors defined an adaptive repulsion and a split/death condition so that particles could either split or be removed from the surface. Hart et al. [13] improved upon this particle system for automatic and numerical differentiation of the implicit surfaces in order to sample more complex implicit surfaces. Moreover, *shape adapters* were introduced that simplify surface deformations. One main limitation of both works is that no information about the curvature of the implicit surface is taken into account, thus only uniform distribution can be generated.

Crossno and Angel [7] derived another extension based on the work of Witkin and Heckbert that they used to sample an isosurface extracted from a 3D density image. The same repulsion forces and relaxation process is used, but the repulsion radius of each particle is estimated according to the curvature at the sample. In other words, particles in regions of higher curvature have a smaller repulsion radius.

Similarly Rosch et al. [21] extended the Witkin and Heckbert scheme to take into account curvature informations in order to sample unbounded surfaces and singularities. Levet et al. [16] presented an anisotropic sampling of implicit surfaces for differential point rendering [14]. They locally take into account the direction and value of principal curvatures, to generate ellipsoidal particles.

Meyer et al. [19] introduced an interesting new class of energy functions for distributing either uniform or nonuniform particles on implicit surfaces. Unfortunately, even if their technique requires less relaxation steps than the original Witkin and Heckbert method, each step takes longer. Besides each example presented in the paper was initialized with the desired number of particles and no clue is given on how this pre-processing is achieved.

Pauly et al. [20] proposed to use particle systems in order to simplify point-sampled surfaces. In their method the distribution of points depends on the curvature of a moving least squares (MLS) surface that approximates the set of points [17]. Using a death condition combined with some repulsion forces enables them to reduce the number of points of the surface.

Other works have presented ideas for sampling implicit surfaces for animation [9] and texture mapping [35]. Some particle systems have also been used to polygonize implicit surfaces [8] or as a comparative quality technique for polygonizing implicit surfaces [15].

Finally Su and Hart [24] presented an object-oriented particle framework designed to rapidly create new particle systems and applications. Some applications of their object-oriented system are provided such as the detection of singularity particles, a tool for dynamic meshing or a way to cluster particles.

3. Motivation

All existing particle systems have more or less based their process on the general scheme proposed by Witkin and Heckbert which can be summarized in following algorithm.

```
Require: An implicit surface
    Create a set of particles lying on the surface
    while Convergence is not reached do
        Compute the repulsion radius (or radii) of the set of particles
        Compute the energy of the set of particles
        Compute the velocity of the set of particles
        Update the position of the particles
        Split or kill particles when necessary
    end while
```

This scheme has two main bottlenecks: the computational cost of the relaxation process and the detection of the convergence of the system.

As soon as there are more than a few thousands particles, each step of the relaxation process becomes very time consuming. Since the computation is done for each particle, the overall complexity of the algorithm is $O(n^2)$. It can be reduced to $O(n \ln n)$ by using a space partitioning but it is usually not worth the effort because systems with several thousand particles require several hundreds of relaxation steps.

The second bottleneck is the detection of the convergence of particle systems. This involves two criterions: first the number of particles has to be stabilized and second, the residual motion has to be null. However, as shown in [16], particle systems never reach a true equilibrium. Even if the residual motion is very low, particles keep moving and sometimes, new particles may be created.

Our aim is to avoid the need of the convergence detection by limiting the number of relaxation steps thanks to a fast generation of a set of particles with interesting characteristics.

4. Uniform sampling

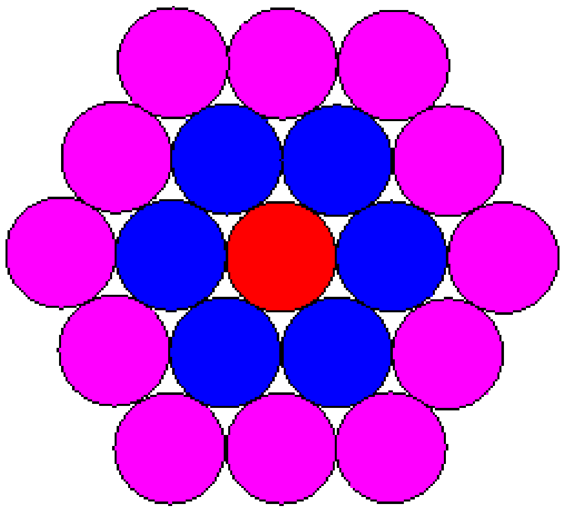


Figure 2 Ideal configuration for a uniform sampling.

In this section we present our method for the generation of uniform particles and the relaxation process used in order to obtain a set of evenly distributed particles.

General approach

For an uniform sampling, the ideal local distribution of particles is given in Figure 2. Each particle has six neighbors which is the ideal configuration in order to minimize the energy. Because we work with 3D models, this ideal 2D configuration is hard to achieve. Thus the key idea of our method is to get an efficient generation of particles by a local 2D process, working on tangent planes defined locally for each particle.

Generation of a particle's children

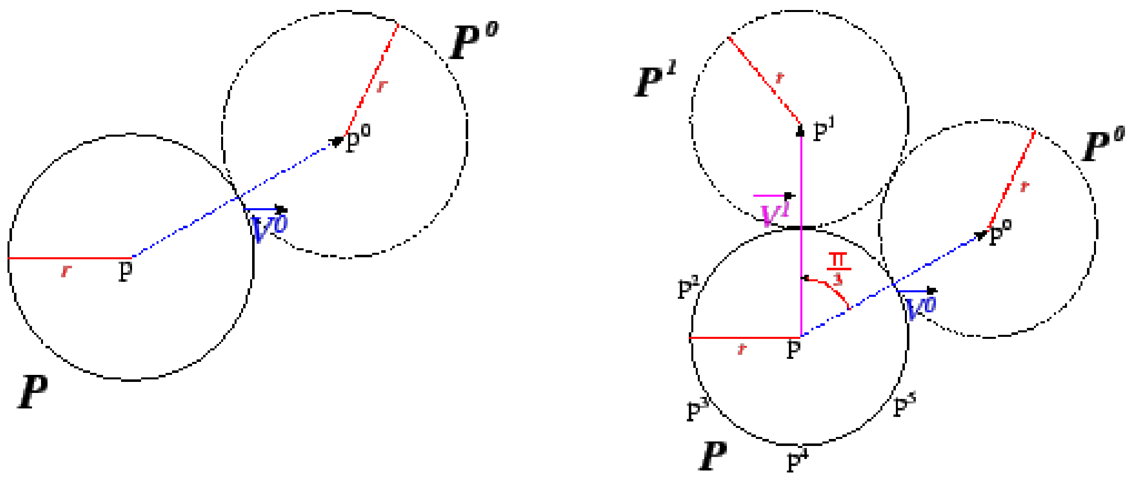


Figure 3 Generation of particles in the 2D tangent plane. (left) The first particle is generated using an arbitrary direction. (right) The second child is placed at an angle of $\pi/3$.

Let S be the implicit surface we want to sample. Each particle P is defined by a 3D position p, a normal $\vec{N}$ at the surface which defines a tangent plane and a repulsion radius r.

Let us start from a seed particle, our goal is to uniformly place its six children particles on the tangent plane of particle P with respect to its repulsion radius.

For each child, we define the tangent vector $\vec{V^i} = p^i - p$ so as the six particles are regularly placed at an angle of $\pi/3$ with its previous and successive particle as can be seen in Figure 3(right). Their 2D positions are given by:

$$p_x^i = \cos\left(\frac{i\pi}{3}\right) V_x^0 - \sin\left(\frac{i\pi}{3}\right) V_y^0$$
$$p_y^i = \sin\left(\frac{i\pi}{3}\right) V_x^0 + \cos\left(\frac{i\pi}{3}\right) V_y^0$$

Note that the first child of the seed article is arbitrary initialized (see Figure 3(left)).

Projecting the children on the surface

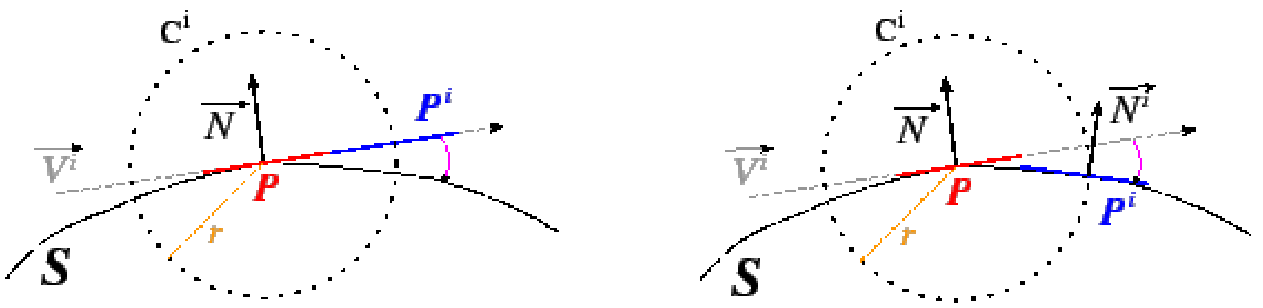


Figure 4 Re-projecting a particle on the surface. (left) Initial position of the child P^i*. (right)* P^i *is reprojected according to circle* C^i.

As we have created the children particles on the tangent plane of the father particle, they have a more or less deviation from S depending on the local curvature (see Figure 4(left)). The most common technique used to reproject a particle on the surface is the iterative Newton-Raphson method. But it does not guarantee that the distance between a particle and its children will be preserved.

Instead, we propose to move each child P^i iteratively on the $C^i(p, 2r)$ circle (see Figure 4(right)) which is the circle centered on p with radius $2r$, defined in the plane $(p, \vec{N}, \vec{V^i})$. After n iteration steps, the positions of particle P^i are:

$$p^i = \cos(\theta^i)\vec{V} + \sin(\theta^i)\vec{N}$$
$$\theta^i = \sum_{j=0}^{n} \alpha_j^i \pi \quad ,$$

where α_j^i are the corrective terms computed for the n n steps. At each step, we evaluate the implicit function at the current position of P^i. This gives us a value that is scaled between -1 and 1 in order to compute α_j. The more distant the particle is from the surface, the larger α_j is. We stop this process when the child position has converged on the surface.

As shown in Figure 4(right) we finally get a reprojected particle on S with a correct normal at the surface.

The complete process

After having detailed the process to place the six children of a particle on the surface, we now explain how to integrate it on the complete sampling process.

Gathering neighboring particles

With the exception of the seed of the system, we never generate the whole set of six children. For each particle, some of its children have already been placed by previous particles and exist in the particle set. Thus, before adding the new particles to the set, we have to gather its neighboring particles. We use a space partitioning scheme in order to accelerate this gathering (as in [32,16]). Any other hierarchical partitioning should work fine.

The algorithm

Starting from an arbitrary chosen seed particle, we generate its six children as explained above. These children are added to the stack of untreated particles. The initial state of the algorithm is now initialized.

For all the particles of the stack, we have to generate its six children. We already know a particle at a desired position: its father. Thus, for each particle, we define its first child as being its father. We have to place only the five remaining children (that we re-project on the surface). In order not to place children too close to existing particles, we check if they belong to a particle of the neighborhood. If an existing particle corresponds, the child is replaced by it and this particle is removed from the neighborhood. If there is no particle, the child is added to the particle set and to the stack of untreated particles. This process is summarized in following algorithm.

```
Require: An implicit surface
Require: A particle seed marked as done
Require: Its six children marked as not done
  while It exists one particle marked as not done do
    Place its six children on the surface
    Gather its neighboring particles
    for all of its six children do
      if an existing particle correspond to the current child then
        Update the child with this particle
        Remove this particle from the neighboring particles
      else
        Add the child (marked as not done) to the particle set
      end if
    end for
    Mark the particle as done
  end while
```

The relaxation process

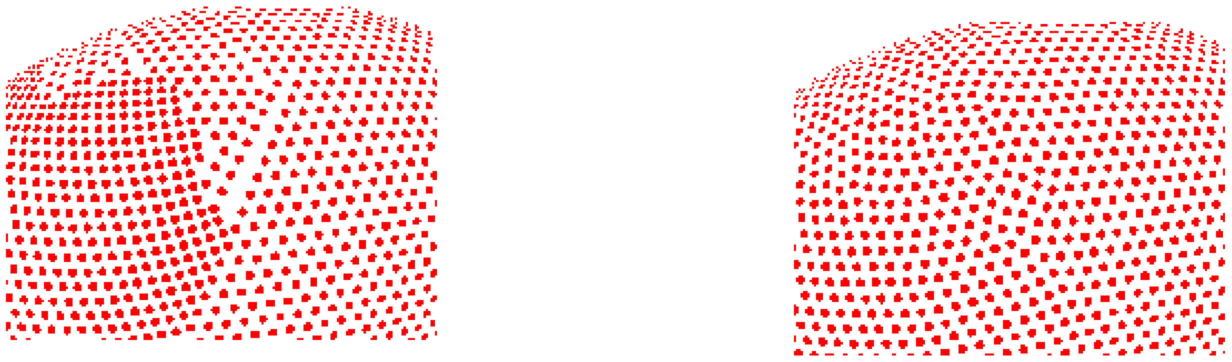


Figure 5 Illustration of the relaxation process. (left) After the generation holes can appear. (right) After the relaxation all holes have disappeared.

The previous step results in a set of particles with positions near the desired ones where each particle has six neighbors. At this moment, we do not have the highest quality sampling of the surface. We have a near-optimal sampling in region of low curvature but holes may appear in regions of high curvature (see Figure 5(left)). This comes from the re-projection of the particles from the tangent plane to the surface. Thus, we need to realize a relaxation on all the particles.

We used the scheme presented by Witkin and Heckbert in [32]. Based on a Gaussian computation of the energy, it is easy to implement and works well for uniform sampling. We just skip from our implementation the adaptive repulsion radius (as we have particles everywhere on the surface) and the split-and-death criterion (as we already have the desired number of particles).

The relaxation process removes all holes of the sampling as can be seen in Figure 5(b).

5. Non-uniform sampling

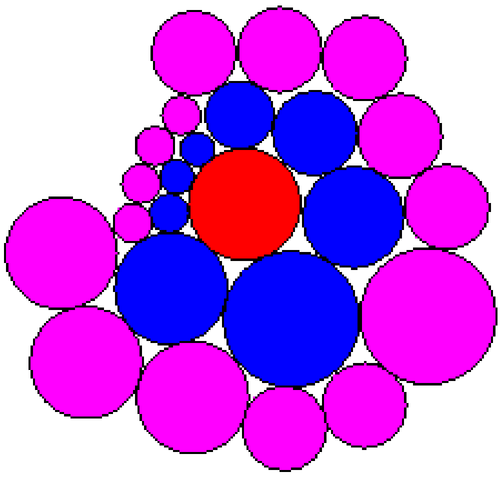
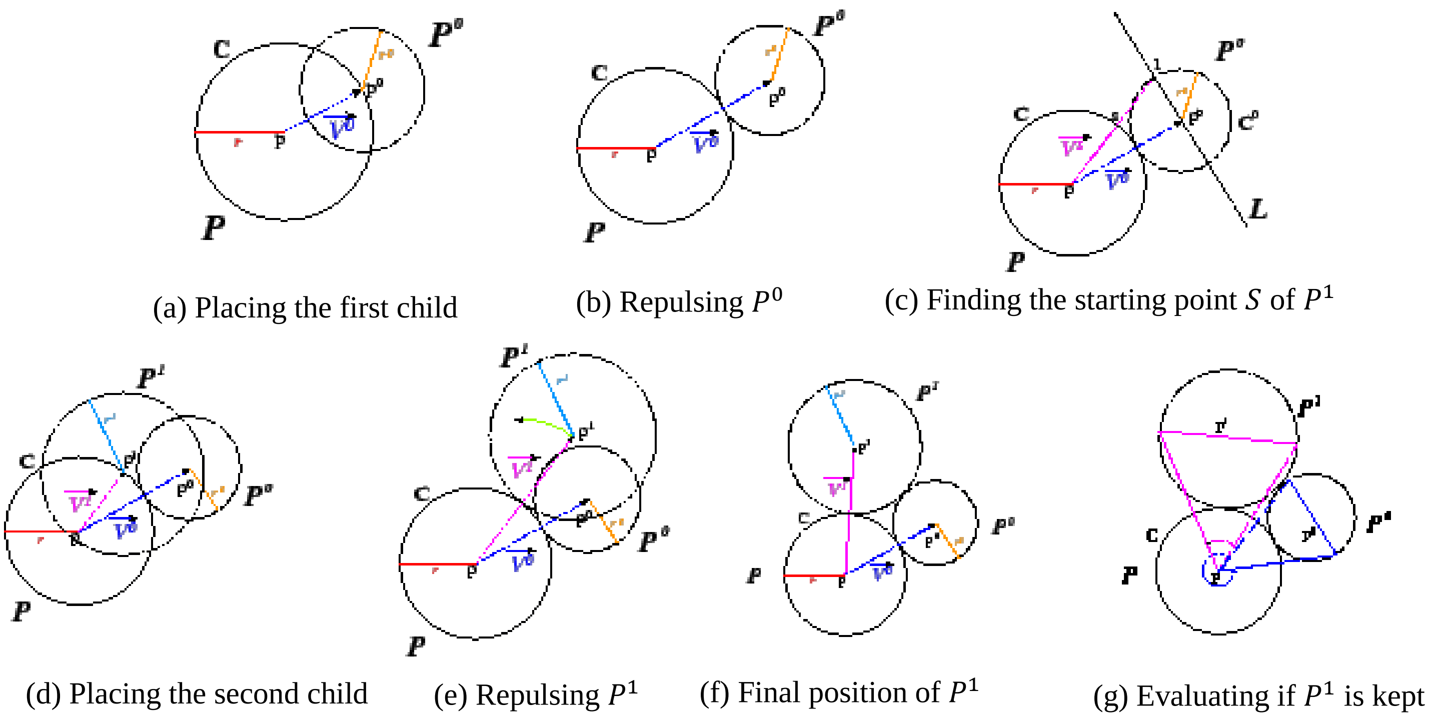


Figure 6 Possible configuration for a nonuniform sampling.

In this section we present a method to efficiently generate non-uniform particles on a surface.

As for the uniform sampling, the principle of the method is to only work locally on 2D planes. In [19], Meyer et al. stated that the ideal configuration is six neighbors even for a non-uniform sampling. This is due to the fact that their non-uniform sampling is dependent of the curvature of the surface and of an energy minimization. Because we are not using a relaxation process based on an energy minimization, we do not know the number of children of


(a) Placing the first child (b) Repulsing P^0 (c) Finding the starting point S of P^1

(d) Placing the second child (e) Repulsing P^1 (f) Final position of P^1 (g) Evaluating if P^1 is kept

Figure 7 Generation of the first two children of P: first, P^0 is placed anywhere on C (a) and then is repulsed in the direction $\overrightarrow{V^0}$ until it reaches its final position (b). Thus we find the starting point S of P^1thanks to the point I (c). The second child P^1 is then created (d) and repulsed in direction $\overrightarrow{V^1}$ (e). This time P^1 intersects P^0so P^1 is repulsed on the circle C (f). Finally, when its final position is reached, we test its projection on C (thanks to φ) in order to determine if $P^1$is added to the set or not.

a particle, which is only dependent of the curvature of the surface. We can have a configuration as shown in Figure 6.

Generation of a particle's children

The main difference with the uniform case is that the repulsion radius of the particles are different. Thus the distance between two particles is no more $2r$ but instead $r^i + r^j$. Because of this change we can not prevail the number of children of a particle anymore. As a consequence we do not know the exact position of the children and we can not place them directly. Our goal is that all the children are still tangent with their father and that there is the smallest overlapping between successive children. Note that we still work on the tangent planes at the surface.

The first step is to determine the starting point of the generation of the child. Two cases exist:

- for the generation of the first child of the seed point, an arbitrary position is chosen (see Figure 7(a)),
- for all other particles at least one child already exist (as the first child of a particle is defined as its father). We determine the straight line perpendicular to the vector $\overrightarrow{V^{i-1}}$ (P^{i-1} is the previous child). I is the intersection between this line and the circle C^{i-1} as shown in Figure 7(c). Finally, the intersection between $\overrightarrow{V^{i-1}} = I - p$ and C gives the starting point s (see Figure 7(c)).

As P^i and its child intersect themselves we have to iteratively move the child in the direction of vector $\overrightarrow{V^i}$. We stop this motion when the child and its father do not overlap anymore as underlined by Figure 7(b-e). At each step we evaluate the curvature of the implicit function at the current 3D position of P^i and change its repulsion radius r^i.

Since the previous repulsion is only along $\overrightarrow{V^i}$, two successive children can intersect themselves as shown in Figure 7(e). In order to reduce the overlapping we iteratively move P^i on the circle C. Two cases are possible:

- if there is no particle in the direction of the repulsion we stop this motion when P^i does not intersect any other child,
- if another child exists in the direction and eventually intersects P^i, we stop the motion when the position minimize the overlapping.

Before adding P^i to the children list and to the particle set we compute the projection of its diameter (given by the angle φ) on C and we determine the actual free circumference of this circle that is given by the angle τ (see Figure 7(g)). If this free circumference is larger than the projection P^i is added.

Finally we re-project the children of P on the surface S by using the method presented in Section 4.3.

The complete process

Some adjustments are necessary in order to integrate this solution in our framework.

First we need to determine the repulsion radius of each particle. Thus, we need to evaluate the curvature of the implicit surface in any point. To achieve this goal we implemented the method given in [16]. Note that this computation gives the principal directions of curvature and their values (that we call κ_1 and κ_2). As we only need a non-uniform isotropic sampling of the surface we define the curvature at a point to be $\kappa = \kappa_1 + \kappa_2$. Plus, this calculation only allows us to determine the implicit field of the implicit surface. This curvature value has to be scaled in order to be used as a distance measurement (e.g., the repulsion radius). Finally a maximum and a minimum radius are defined in order to avoid very small or very large particles.

The algorithm is very similar to the one used for the uniform sampling. Once again we must gather the neighboring particles of the current particle in order to determine if the generated children do not take place at the position of an existing particle.

Determining the fate of the children

The main difference with the uniform case is that we do not place the children and then determine if they are close to existing particles. This time we want to place children between the existing neighboring particles if there is enough available space.

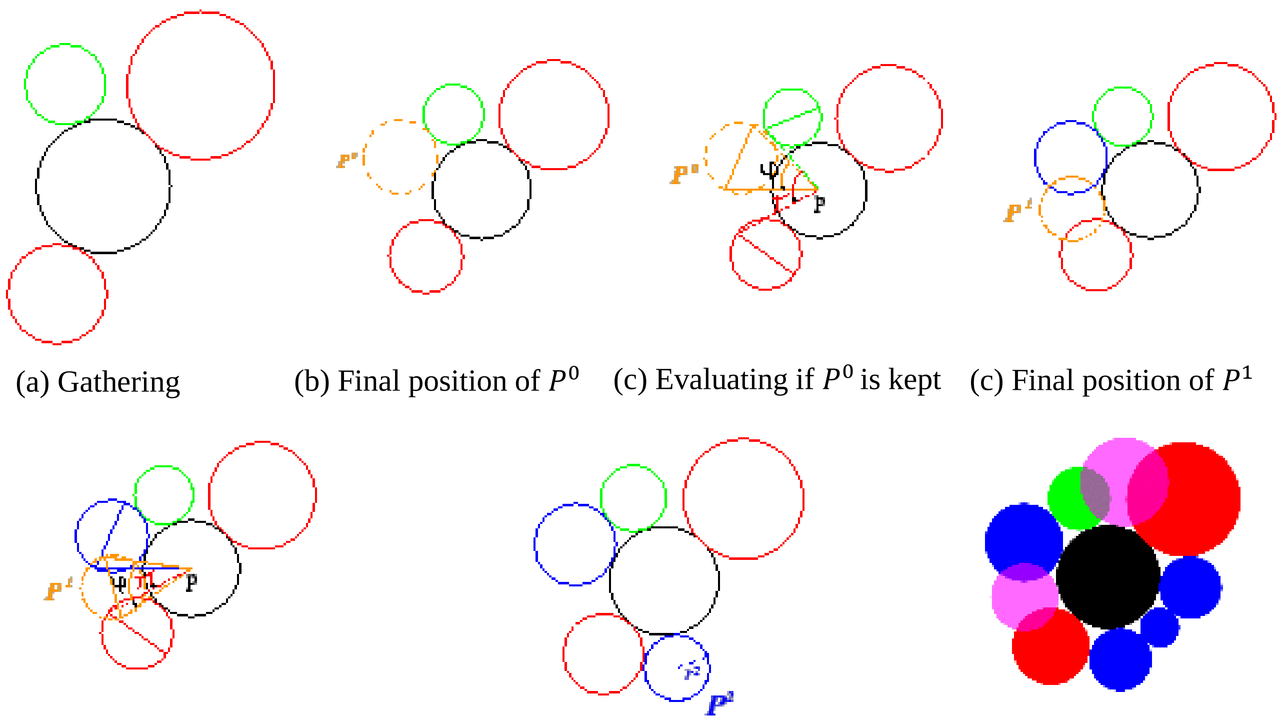
In Section 5.1 we have explained how to generate non-uniform children on a particle P^i. At the beginning of this algorithm each particle (except the seed point) has one child (its father). Thus we can say, because we work on the tangent circle C^i, that its free available space is from its father to its father.

Now, the only difference is that we already have existing particles on C^i. As a consequence if we have n existing particles we have n free available spaces. These spaces are between the existing particles and their successive existing particles. Thus we apply the generation explained in Section 5.1 on each free space. An illustration of this process is given in Figure 8. Figure 8(g) shows the result of this algorithm on a particle P.

The relaxation process

At the end of the process we can have some particles that intersect themselves. This is due to the fact that our generation algorithm only takes into account the direct neighboring of the treated particle. So, in regions of junction between two sampling lines, the problem is that some particles may intersect themselves. This time, the characteristic holes shown in Figure 5(left) do not appear. Since this sampling is not dependent of some directions these typical holes disappear. However, we still have to realize a relaxation process in order to get almost evenly distributed particles.

In [15,21], the authors have stated that by a small modification of the energy of the Witkin and Heckbert scheme (by multiplying by the cosine of the angle between two particles) non-uniform sampling can be achieved. Our experimentations were not satisfactory. Indeed, small particles, because they are more numerous, keep repelling bigger particles. Thus we adopted a relaxation scheme based on a computation of forces [27,20,16] which is easy to implement and was sufficient to validate our algorithm.



(a) Gathering (b) Final position of P^0 (c) Evaluating if P^0 is kept (c) Final position of P^1

(e) Evaluating if P^1 is kept (f) Final position of P^2 (g) Final result

Figure 8 Generation of the children of P (the black particle) with respect to existing particles (red particles). Its father is green while blue particles are generated particles added to the set. Finally, the orange particle is the the current generated particle. First we gather the neighboring particles of P (a). We then find the final position of the current generated particle (b-d-f). We estimate its projection on C (thanks to the angle φ) and compare it with the free circumference of C given by the angle τ (c-e) in order to determine if it is added to the set or not. The final result is that some generated particles have been added to the set (blue particles) while others have not (transparent violet particles) (g).

6. Results and discussion

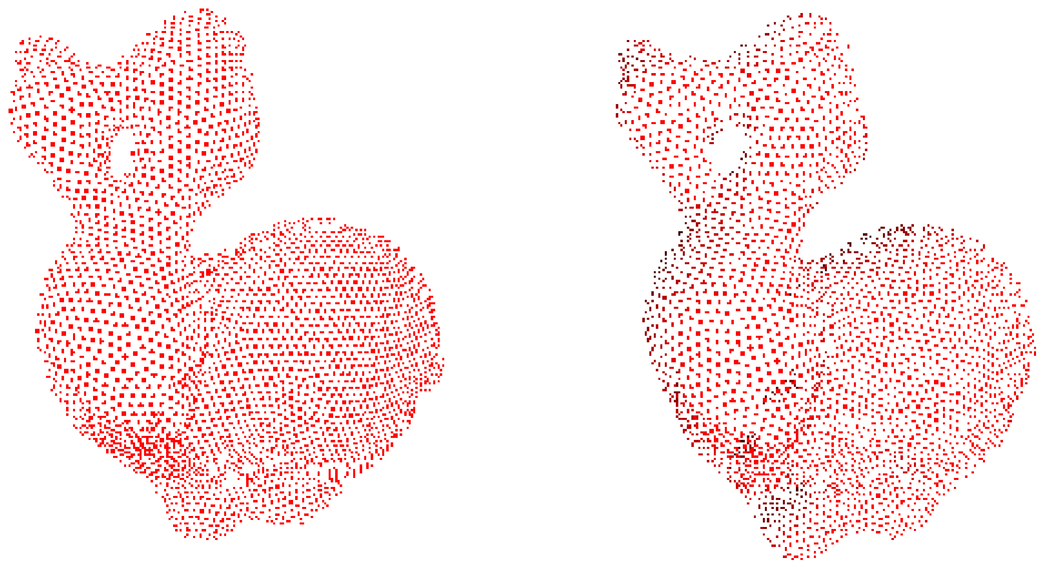
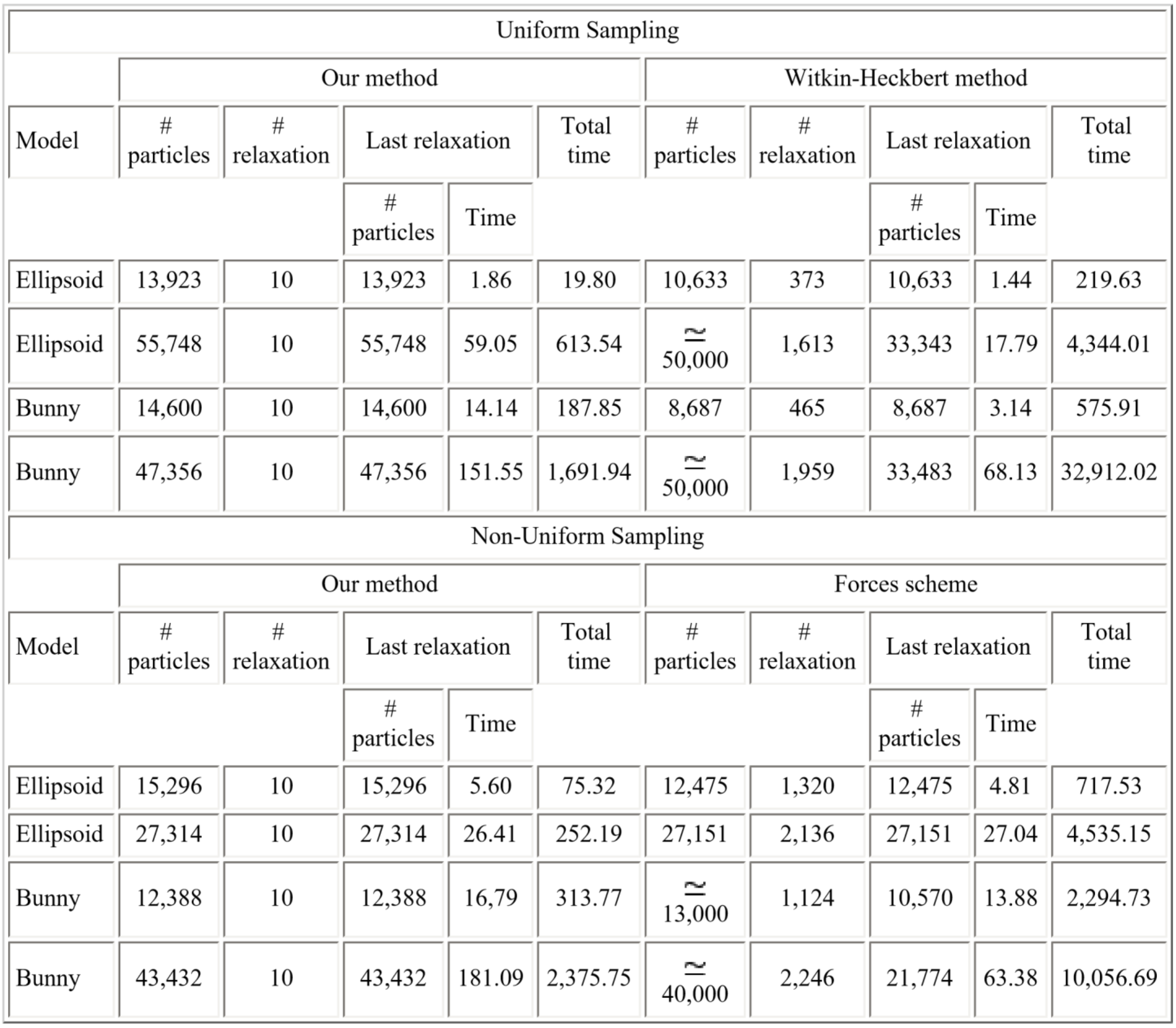


Figure 9 The bunny sampled with around 10,000 particles using (left) our method and (right) the Witkin and Heckbert one.

The prototype system was written in the C++ language. All example models shown in this paper were built with the prototype system by using a PC with 3 GHz processor with 1 GB of main memory (note that we do not use the complete available memory).



Uniform Sampling										
	Our method					Witkin-Heckbert method				
Model	# particles	# relaxation	Last relaxation		Total time	# particles	# relaxation	Last relaxation		Total time
			# particles	Time				# particles	Time	
Ellipsoid	13,923	10	13,923	1.86	19.80	10,633	373	10,633	1.44	219.63
Ellipsoid	55,748	10	55,748	59.05	613.54	≃ 50,000	1,613	33,343	17.79	4,344.01
Bunny	14,600	10	14,600	14.14	187.85	8,687	465	8,687	3.14	575.91
Bunny	47,356	10	47,356	151.55	1,691.94	≃ 50,000	1,959	33,483	68.13	32,912.02
Non-Uniform Sampling										
	Our method					Forces scheme				
Model	# particles	# relaxation	Last relaxation		Total time	# particles	# relaxation	Last relaxation		Total time
			# particles	Time				# particles	Time	
Ellipsoid	15,296	10	15,296	5.60	75.32	12,475	1,320	12,475	4.81	717.53
Ellipsoid	27,314	10	27,314	26.41	252.19	27,151	2,136	27,151	27.04	4,535.15
Bunny	12,388	10	12,388	16,79	313.77	≃ 13,000	1,124	10,570	13.88	2,294.73
Bunny	43,432	10	43,432	181.09	2,375.75	≃ 40,000	2,246	21,774	63.38	10,056.69

Table 1 We compare our uniform sampling with the Witkin-Heckbert method and our non-uniform sampling with a scheme based on forces. All given times are in second. For the two comparative techniques, we start

The sphere and ellipsoidal models shown in this paper were analytic implicit surfaces while the rabbit, the bunny and the Santa models were reconstructed using RBF [28]. These three models were reconstructed with a set of 200 initial points. In the case of the bunny we removed some points of its ears. Thus a part of the implicit surface was reconstructed between these two ears as can be seen in Figure 9 and in Figure 12(bottom-left). As a consequence we increased the genius of the initial bunny model.

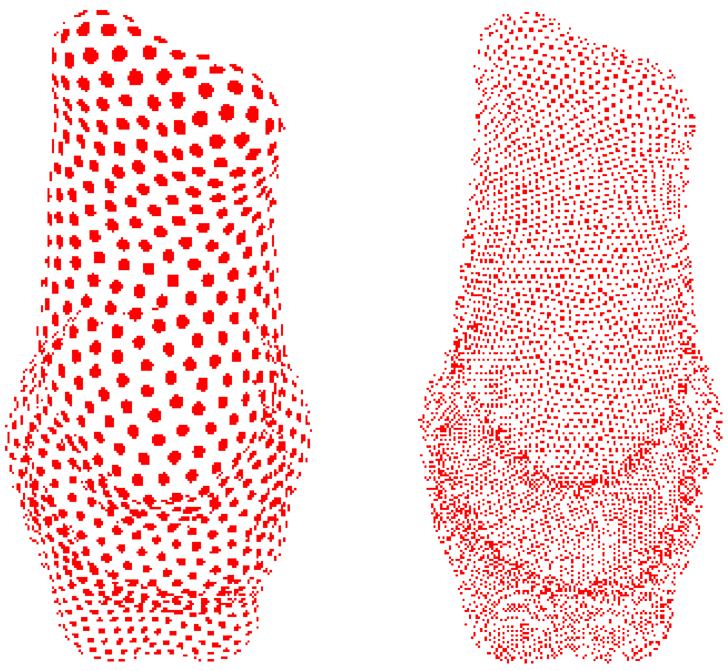


Figure 10 Two samples distributions of the rabbit: (left) with 2,449 particles and (right) with 25,702 particles.

As said before the only parameter that can be changed by users is the scaling factor of the curvature. Changing it enable users to sample a model with more or less particles. Figure 10 shows two uniform sampling of the bunny. In Figure 10(a) the scaling factor used was 2% of the bounding box diagonal of the model while in Figure 10 (b) it was 0.6%.

Since our objective is to fastly generate particles on implicit surfaces, we present some times related to our techniques. Table 1 shows times taken by the sampling of two models: the ellipsoid and the bunny. We compare our uniform sampling with the Witkin-Heckbert technique [32] and our non-uniform sampling with a scheme using only forces [27,20,16]. A different number of particles and last relaxation number of particles means that the targeted number of particles was not achieved. For example, for the second sampling of the bunny with the Witkin-Heckbert method, the targeted number was around 50,000 particles. Because the time needed by the sampling was already too important, we stopped the process before its end when only a part of the surface was sampled (there was only 33,483 particles after 1,959 relaxations).

As said before, our techniques do not need a split criterion (so we keep the number of particles constant during the ten relaxation steps). On the contrary, the two comparative techniques start from one seed particle. Then, the systems are supposed to sample the totality of the surface thanks to their split criterion. The split criterion used for the Witkin-Heckbert method is given in [32]. For the forces scheme, we split a particle when no force is applied on it. Note that we used the same code for the relaxations of our methods and for the comparative techniques (with the exception of the split criterion).

The most interesting observation is that we improve the sampling time of all examples, enabling the use of particle systems with thousand particles. Contrary to other systems that need hundreds of relaxation, the use of only ten relaxations saves a lot of time.

Moreover, Table 1 shows that the type of implicit surface used has a direct impact on the time taken by the sampling. Because the bunny was reconstructed by using RBF, the evaluation of its implicit surface takes much more time than for analytic implicit surfaces. This explains the difference of timing between the two models.

Finally, we can see that the non-uniform sampling is slower than the uniform sampling. Since for the nonuniform sampling two iterative repulsions have been introduced, the time needed to generate the children of a particle is more important than for the uniform sampling. Besides we have to compute the curvature of the surface for each particle.

#particles	Time (in second)		
	Generation	Relaxation	Total
2,225	0.11	0.49	0.60
13,923	1.22	18.58	19.80
55,748	24.80	588.74	613.54
87,043	61.60	1,487.49	1,549.09

Table 2 Times taken by the sampling of the analytic ellipsoid.

In Table 2 we show the times taken by different levels of sampling of the analytic ellipsoid. We can see that the set of particles generation time is far smaller than the time taken by the relaxation process.

Figure 9 shows the bunny sampled with around 10,000 uniform particles with our method (see Figure 9(left)) and the Witkin-Heckbert technique (see Figure 9(right)). In each case no hole appears. One interesting difference can be seen: the alignment of the particles provided by our method. This comes from our hexagonal generation of particles designed to achieve the ideal uniform configuration shown in Figure 2. An illustration of this hexagonal generation can be seen in Figure 11 where each level of particles generation has a different color.

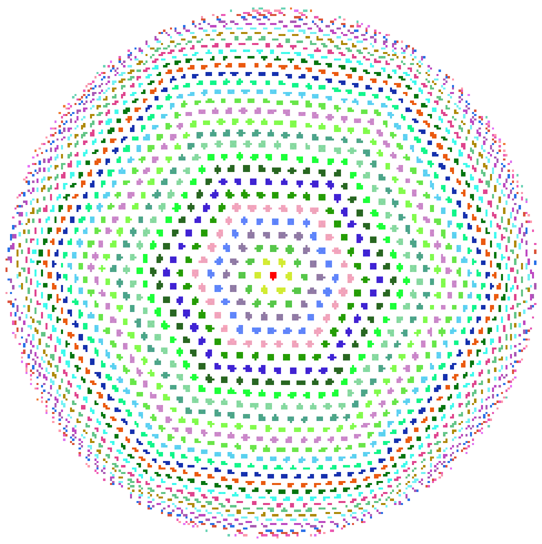


Figure 11 A sphere sampled with uniform particles: each level of generation has a different color.

Even if the times taken by our methods are very interesting we think that it's possible to still improve them. We use a space partitioning scheme [32,16] that works very well for a low number of particles but that is far less efficient with thousands particles. When a lot of particles are generated, the time of traversal of the structure keeps increasing. Thus, with a better spatial as in [7,19], we believe that we could speed-up the generation time.

Finally, we show various examples sampled with our techniques in Figure 12.

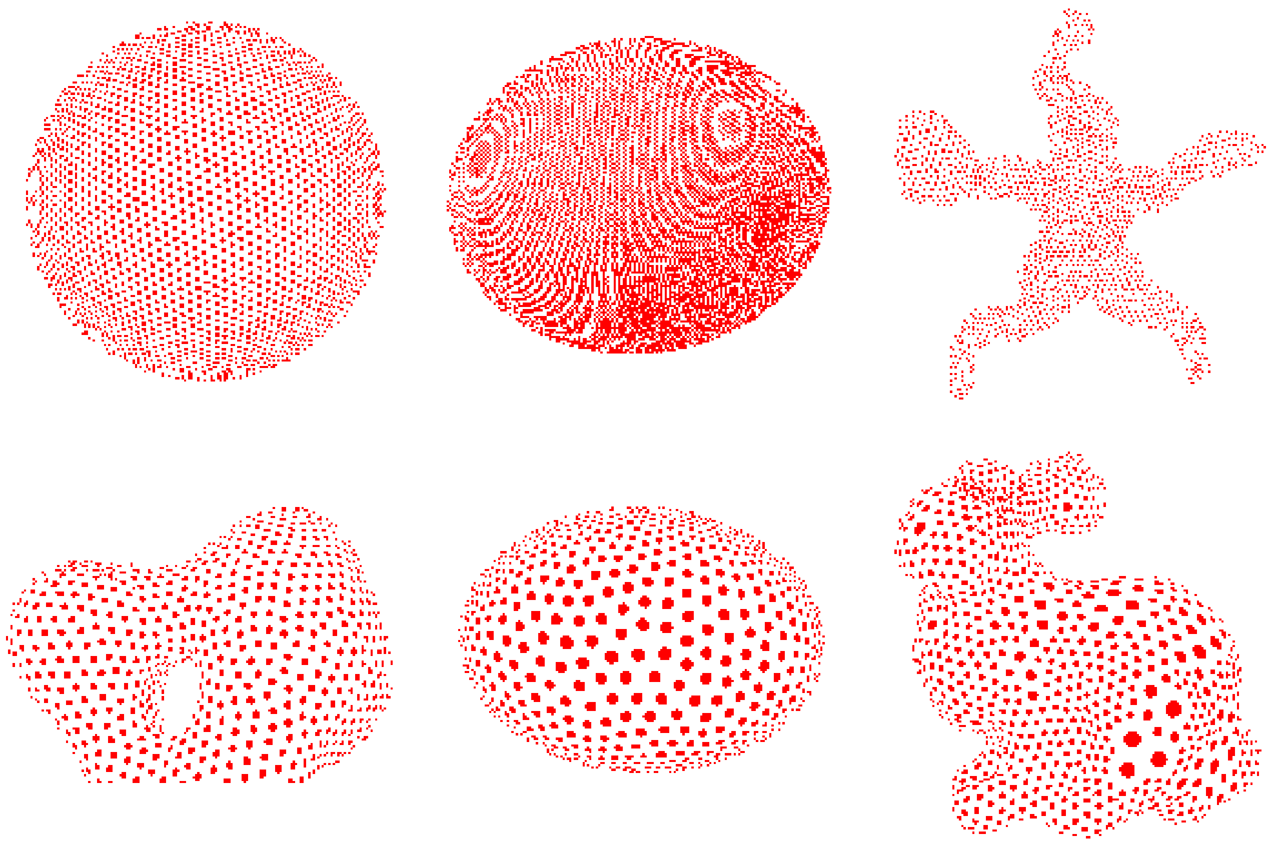


Figure 12 Examples of our two sampling techniques.

7. Conclusion and future works

In this paper, we have presented a new particles generation algorithm enabling the use of particle systems with more than a few thousands. We did not limit our technique to uniform particles since non-uniform distribution of particles can be achieved.

Because our computations only lie on pure geometric processing (by the use of tangent planes) we have developed a very efficient and rapid generation algorithm. Its result is a set with the desired number of particles. At this time, their positions are near optimal.

Thanks to the characteristics of our generation algorithm, we only have to do ten steps of the relaxation process to get a high quality sampling. This is a great improvement compared to existing particle systems that can need hundred steps of relaxation to achieve their sampling.

By combining these two results, we greatly improve the timing of particle systems. Thus, we can sample implicit surfaces with more than a few thousand particles.

Future works

Since we decided to improve particle systems, we have developed our generation algorithm with the idea that a relaxation is performed on the set of generated particles. Thus, because we only work on tangent planes, holes can appear on the surface and, in some areas, particles can overlap themselves. We believe that our sampling quality can be improved by taking into account the local informations of curvature during the generation of children. This will enable us to choose directly a more optimal position for each particle.

At this moment, the generation time of our non-uniform technique is more important than for our uniform technique. This is due to the fact that we introduce two iterative repulsions that need more computation than a direct placement. We are investigating a way to directly place the children even for the non-uniform sampling.

The time of one relaxation remains problematic. Even if we limit this problem by needing only ten relaxation steps, for thousands of particles the time taken by the relaxation process is much larger than the generation time. Because of the characteristics of our sampling, a lot of particles have already an optimal position. Finding a criterion to identify these particles and not move them could greatly improve the speed of the relaxation process.

Finally, we are investigating a way to adapt our uniform technique in order to polygonize implicit surfaces. During the generation step we guarantee a 6-connectivity for each particle. This 6-connectivity may be lost during the relaxation process. Our actual goal is to keep this 6-connectivity during all the steps of our algorithm.

Bibliography

1 Eugene L. Allgower and Stefan Gnutzmann. Simplicial pivoting for mesh generation of implicitly defined surfaces. *Computer Aided Geometric Design*, 8(4):305 - 325, 1991.

2 Sergei Azernikov and Anath Fischer. Anisotropic meshing of implicit surfaces. In *SMI (Shape Modeling Internatinal)*, 2005.

3 Jules Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341-355, 1988.

4 Jules Bloomenthal. An implicit surface polygonizer. *Graphics Gems IV*, pages 324-349, 1994.

5 Andrea Bottino, Wim Nuij, and Kees van Overveld. How to shrinkwrap through a critical point: an algorithm for the adaptive triangulation of iso-surfaces with arbitrary topology. In *Proceedings of Implicit Surfaces '96*, pages 53-73, 1996.

6 Benoit Crespin, Pascal Guitton, and Christophe Schlick. Efficient and accurate tessellation of implicit sweep objects. In *Proceedings of Constructive Solid Geometry '98*, 1998.

7 Patricia Crossno and Edward Angel. Isosurface extraction using particle systems. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization '97*, pages 495-498, 1997.

8 Luiz Henrique de Figueiredo, Jonas Gomes, Demetri Terzopoulos, and Luiz Velho. Physically-based methods for polygonization of implicit surfaces. In *Proceedings of Graphics Interface '92*, pages 250-257. CIPS, 1992.

9 Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Cani. Adaptive sampling of implicit surfaces for interactive modeling and animation. In *Implicit Surfaces*, avril 1995. Published under the name Marie-Paule Gascuel.

10 Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Cani. Adaptive sampling of implicit surfaces for interactive modeling and animation. *Computer Graphics Forum*, 15(5), 1996. Published under the name Marie-Paule Gascuel.

11 Jonas Gomes and Luiz Velho. *Implicit Objects in Computer Graphics*. Série Monografias do IMPA, Rio de Janeiro, 1992.

12 Mark Hall and Joe Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics & Applications*, 10(6):33-42, 1990.

13 John C. Hart, Ed Bachta, Wojciech Jarosz, and Terry Fleury. Using particles to sample and control more complex implicit surfaces. In *Proceedings of Shape Modeling International 2002*, pages 129-136, 2002.

14 Aravind Kalaiah and Amitabh Varshney. Differential point rendering. In K. Myskowski and S. Gortler, editors, *Rendering Techniques 2001 (Proceedings of the Eurographics Workshop on Rendering 2001)*, pages 139-150. Springer Verlag, 2001.

15 Tasso Karkanis and A. James Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics and Applications*, 21(2):60-69, 2001.

16 Florian Levet, Julien Hadim, Patrick Reuter, and Christophe Schlick. Anisotropic sampling for differential point rendering of implicit surfaces. In *WSCG (Winter School of Computer Graphics)*, 2005.

17 David Levin. Mesh-independent surface interpolation. In Guido Brunnett, Bernd Hamann, and H. Müller, editors, *Geometric Modeling for Scientific Visualization*. Springer, Heidelberg, Germany, 2003.

18 William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (ACM SIGGRAPH 87 Proceedings)*, 21(4):163-169, 1987.

19 Miriah D. Meyer, Pierre Georgel, and Ross T. Whitaker. Robust particle systems for curvature dependant sampling of implicit surfaces. In *SMI (Shape Modeling Internatinal)*, 2005.

20 Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization 2002*, pages 163-170, 2002.

21 Angela Rosch, Matthias Ruhl, and Dietmar Saupe. Interactive visualization of implicit surfaces with singularities. *Eurographics Computer Graphics Forum*, 16(5):295-306, 1997.

22 Peter Shirley and Allan Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics (San Diego Workshop on Volume Visualization)*, 24(5):63-70, 1990.

23 Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proceedings of ACM SIGGRAPH 97*, pages 279-286, 1997.

24 Wen Yu Su and John C. Hart. A programmable particle system framework for shape modeling. In *SMI (Shape Modeling Internatinal)*, 2005.

25 Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, 26(2):185-194, 1992.

26 Greg Turk. Generating textures for arbitrary surfaces using reaction-diffusion. *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, 25(4):289-298, 1991.

27 Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2):55-64, 1992.

28 Greg Turk and James O'Brien. Variational implicit surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology, 1998.

29 C.W.A.M. van Overveld and Brian Wyvill. Shrinkwrap: an adaptive algorithm for polygonizing an implicit surface. Technical Report 93/514/19, The University of Calgary, Calgary, Alberta, Canada, 1993.

30 Luiz Velho. Adaptive polygonization made simple. In *Proceedings of SIBGRAPI '95*, pages 111-118, 1995.

31 Luiz Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2):5-25, 1996.

32 Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of ACM SIGGRAPH 94*, pages 269-278, 1994.

33 Shin Ting Wu and Marcelo de Gomensoro Malheiros. On improving the search for critical points of implicit functions. In *Proceedings of Implicit Surfaces '99*, pages 137-144, 1999.

34 Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227-234, 1986.

35 Ruben Zonenschein, Jonas Gomes, Luiz Velho, and Luiz Henrique de Figueiredo. Controlling texture mapping onto implicit surfaces with particle systems. In *Proceedings of Implicit Surfaces '98*, pages 131-138, 1998.

Levet Florian 2007-04-24