HAL
open science

Triangulation of uniform particle systems: its application to the implicit surface texturing

Florian Levet, Xavier Granier, Christophe Schlick

To cite this version:

Florian Levet, Xavier Granier, Christophe Schlick. Triangulation of uniform particle systems: its application to the implicit surface texturing. 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Jan 2007, Plzen, Czech Republic. inria-00187181v2

HAL Id: inria-00187181

https://inria.hal.science/inria-00187181v2

Submitted on 27 Oct 2013

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.

Triangulation of uniform particle systems: its application to the implicit surface texturing

Florian Levet Xavier Granier Christophe Schlick

IPARLA Project - INRIA/LaBRI - Université Bordeaux 1

{levet,granier,schlick}@labri.fr

ABSTRACT

Particle systems, as originally presented by Witkin and Heckbert [32], offer an elegant solution to sample implicit surfaces of arbitrary genus, while providing an extremely regular distribution of samples over the surface. In this paper, we present an efficient technique that uses particle systems to rapidly generate a triangular mesh over an implicit surface, where each triangle is almost equilateral. The major advantage of such a triangulation is that it minimizes the deformations between the mesh and the underlying implicit surface. We exploit this property by using few triangular texture samples mapped in a non-periodic fashion as presented by Neyret and Cani [16]. The result is a pattern-based texturing method that maps homogeneous non-periodic textures to arbitrary implicit surfaces, with almost no deformation.

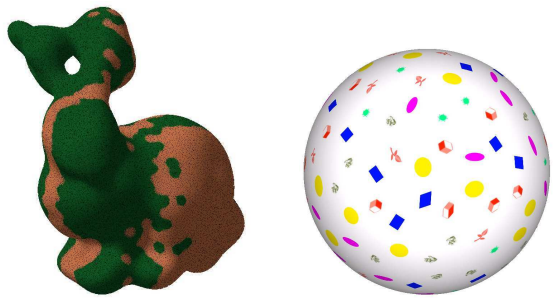
Keywords

Geometric Modeling, Implicit Surfaces, Particle Systems, Texturation

1 Motivation

Starting from the seminal work by Blinn [2] at the beginning of the eighties, implicit surfaces have been recognized as an elegant and powerful representation for 3D modeling. Although the several hundreds of paper that have been published on the subject during the last 25 years, a standard 3D modeling software environment that uses implicit surface as a general geometric paradigm is still not common practice in industry [18, 24]. So even though much work has shown that implicit surfaces combine different mathematical properties that make them very appealing, they are not widely used for real-scale applications in the computer graphics industry. There are mainly two reasons that may explain such a rather confidential use. First, implicit surfaces are not well adapted to the rendering pipeline used in current graphics hardware. Indeed, a rather expensive tessellation step is required to convert the surface into a polygonal mesh that can be efficiently processed by the hardware. Second, conventional texture mapping based on 2D texture patterns, which is by far the most ubiquitous solution to visually enhance geometric models, is really awkward to apply to implicit surfaces as there is no underlying parametrization.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright UNION Agency–Science Press, Plzen, Czech Republic.



(a) (b)

Figure 1: Pattern-based texturing of two different models.

Of course, 3D texturing techniques, such as solid textures [19, 34, 21] or octree textures [1, 12] can always be applied to implicit surfaces as they do not require any 2D parametrization. 3D textures represent a powerful solution to generate visual appearance of objects carved out from heterogeneous material such as marble or wood, for instance. Unfortunately, the visual complexity of most real life objects is much better simulated by 2D pattern-based texture mapping rather than 3D texturing. The main reason is that the normal

vector of 3D textures does not align with the normal of the underlying surface.

Several authors have previously addressed the problem of tuning conventional 2D pattern-based texture mapping techniques for implicit surfaces. Most of these approaches will be briefly recalled in Section 2. The technique presented in this paper belongs to this stream of work. More precisely, the basic idea of this paper is to use the intrinsic features of uniform particle systems to drive a homogeneous non-periodic texture mapping on implicit surfaces with almost no deformation. The process is to rapidly generate a set of particles, apply a limited number of relaxation steps to achieve uniform sampling, and triangulate this set by taking into account the specific characteristics of the sampling. The resulting mesh, where each triangle is almost equilateral, is then used to support the parametrization of a pattern-based texture mapping scheme, similar to the one presented by Neyret and Cani [16].

The remainder of the paper is organized as follows. Section 2 recalls some related previous work on sampling and texturing of implicit surfaces. Section 3 introduces our algorithm to rapidly generate an almost-uniform set of particles over an arbitrary implicit surface. Section 4 focuses on the triangulation of this particle set, while Section 5 shows how to use this triangulation to get a local parametrization of a texture mesh that is then applied to the surface. Section 6 presents several experimental results obtained with our texturing process, and finally, Section 7 concludes and presents some directions that we are currently investigating.

2 Previous work

2.1 Particle systems

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 [26], but in their work there was no underlying implicit surface. Figueiredo et al. introduced [5] a system to sample implicit surfaces using particles. The authors used a similar relaxation process that [26] 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 [30] used a reaction-diffusion method in order to re-tile polygonal surfaces according to the curvature, to get more points in regions of high curvature.

Witkin and Heckbert [32] developed a powerful modeling application by defining an adaptive repulsion and a split/death condition so that particles could either split or be removed from the surface. The technique generates a nice isotropic sampling of the implicit surface. Unfortunately, the convergence of the algorithm is rather slow, even for moderately complex surface, despite the improvement later proposed by Hart et al. [8] for automatic and numerical differentiation of the underlying surface.

Most improvements of particle systems addressed the problem of adaptive sampling of implicit surfaces according to curvature. Crossno and Angel [3] extended the work of Witkin and Heckbert by estimating a different repulsion radius for each particle according to its curvature. Similarly, Rosch et al. [23] used curvature information to sample unbounded surfaces and singularities. Meyer et al. [15] introduced a new class of energy functions for distributing either uniform or non-uniform particles on implicit surfaces. Levet et al. [14] presented an anisotropic sampling of implicit surfaces that locally takes into account the direction and value of principal curvatures.

Other works have presented ideas for sampling implicit surfaces for animation [6]. Some particle systems have also been used to polygonize implicit surfaces [5] or as a comparative quality technique for polygonizing implicit surfaces [11]. Su and Hart [25] presented an object-oriented particle framework designed to rapidly create new particle systems and applications.

Recently, some researches have focused on generating first the particles and, then, performing the relaxation process in order to get a uniform distribution of the particles. So, Galin et al [7] have developed a technique which takes into account the characteristics of the BlobTree [33] in order to realize a very fast uniform sampling of implicit surfaces. Even if a little slower, the method presented by Levet et al. [13] allows to generate a fast sampling of an implicit surface in order to perform a limited number of relaxations. This method works for any type of implicit surface and manage to generate uniform and non-uniform particles.

2.2 Texture mapping on implicit surfaces

Several authors have previously addressed the problem of tuning conventional 2D pattern-based texture mapping techniques for implicit surfaces. Note that most of the approaches are based on particle systems.

Turk [29] proposed to use particle systems to generate natural pattern directly on a generic surface by using a reaction-diffusion process. This approach becomes very expensive for high frequency details that require a huge number of particles.

Pedersen [20] described a texturing method for implicit surfaces by defining a set of patches over the surface. Each patch acts as a local parametrization and can be textured and interactively manipulated. The drawback of this approach is to require laborious and

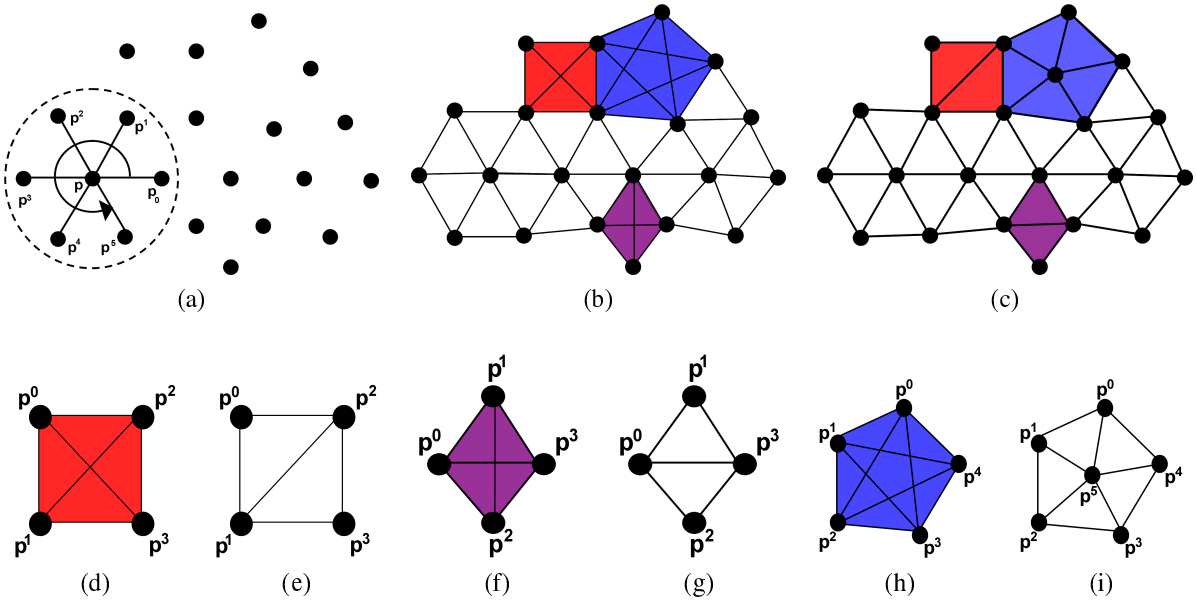


Figure 2: Triangulation of a uniform particle set.(a) Neighboring particles of P are ordered around it. (b) After the neighbors gathering some triangles intersect each others. (c) The simplification step discards all these triangles in order to create a good polygonization. The three problematic configurations are given in (d-f-h) while their solutions are given in (e-g-i).

sometimes tricky hand-tuning by the user to avoid deformation of 2D texture patterns.

Zonenschein et al. [35] proposed a simple texture mapping technique for implicit surfaces that uses a simple parametric support surface (e.g., sphere, cylinder). The mapping from the implicit surface to the support surface is done by a particle system. More precisely, particles are initialized on the implicit surface and then moved towards the support surface through an interpolation of the gradient fields of the two surfaces. While the original technique was rather limited, it has received several further extensions: to offer better control of local mappings [36], to account for composite implicit shapes [37] and for composite skeleton-based support surfaces [27, 28]. Despite this generalization, the technique is still not very robust in the case of implicit surfaces with complex topology or high-order genus.

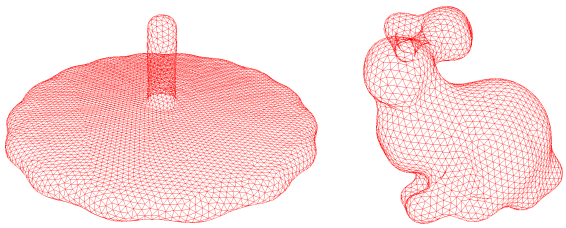
3 Triangulation of the particle set

Crossno and Angel [4] have developed an algorithm that triangulates non-uniform particle set. Because it is not possible to predict the neighbors number of a particle, they have to deal with a lot of special configurations that makes their technique very complex to implement. Because we want to triangulate uniform particle systems, our algorithm is simpler.

The starting point of our technique is a set of uniform particles (which can be obtained by using the particle systems described in [32, 15, 13]). These uniform particle systems have some interesting characteristics. In the ideal scheme, each particle has six neighbors and the distance between each pair of them is $2r$ (r being the global repulsion radius for all the particles). The problem is that this ideal scheme can not be achieved easily. Indeed, as shown in [14], the detection of the convergence of particle systems is a difficult task. Thus, when the sampling of the implicit surface by the particles is stopped, each particle pair is not exactly at a distance of $2r$. Instead, the system has minimized the particle set energy in order that the average particle's number of neighbors is six and that the distance of each particle pair is around $2r$.

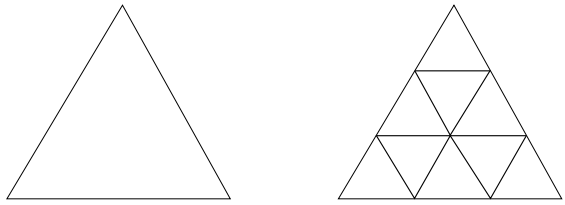
A naive solution would be to gather, for each particle, all the particles located at a distance of $2r$ and triangulate them. The problem is that, as said above, because of the last relaxation, the distance between neighboring particles can be a little more than $2r$. With this naive solution, we can thus miss some neighboring particles leading to uneven triangles. Besides, these configurations are difficult to identify. Our solution is to gather more particles than really needed because, in this case, the identification of the problematic configurations is easier. Thus, in our implementation, the following two-pass process is used: we first triangulate the particles with a simple neighboring search, and second, we simplify the mesh by discarding the intersecting triangles.

In the first step, we gather more neighboring particles than needed by using a $3r$ instead of $2r$ neighborhood, as it avoids some pathological configuration during the



(a) (b)

Figure 3: Triangulation of two models.



(a) Original triangle. (b) Resulting tessellation.

Figure 4: Level 3 tessellation of a triangle.

simplification step. The gathered particles have then to be ordered around P (see Figure 2(a)). All the neighboring particles of P are projected on its tangent plane which allows us to do all the remaining computation in 2D. We select a neighbor of P as the reference particle P^r (in the Figure 2(a), the particle P^0 is defined as the reference particle P^r). For all the other ones, we compute the dot product between the reference vector $\overrightarrow{V^r} = \mathbf{p}^r - \mathbf{p}$ and the vectors $\overrightarrow{V^i} = \mathbf{p}^i - \mathbf{p}$, $\sum_{i=1}^{n}$. Finally, the ordering is done on the 2D cross product between $\overrightarrow{V^r}$ and $\overrightarrow{V^i}$. A simple triangle fan can now be created.

This results in a mesh with some intersecting triangles (see Figure 2(b)). Three different criteria are used to identify these triangles. First, for the configuration shown in Figure 2(d): $\mathbf{p}^3$ is not a neighbor of $\mathbf{p}^0$ but there is a triangle that links them. This configuration arises because we gather a $3r$ neighborhood instead of a $2r$ one. We can have now triangles between particles that are not neighbors. In order to identify such triangles, we have just to gather all triangles that have $\mathbf{p}^0$ as a vertex, and keep those that have $\mathbf{p}^3$ as well. Thanks to the characteristics of the particle systems, we are sure that the three other triangles exist ($(\mathbf{p}^0, \mathbf{p}^3, \mathbf{p}^2)$, $(\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2)$, $(\mathbf{p}^3, \mathbf{p}^2, \mathbf{p}^1)$), and that we recover them. Once we have the four triangles, we just have to discard the $(\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^3)$ and $(\mathbf{p}^0, \mathbf{p}^3, \mathbf{p}^2)$ as P^0 and P^3 are not neighbors (see Figure 2(e)).

After having applied this first criteria, some intersecting triangles still remains. To identify them, we have to look at the neighborhood of a particle. This configuration arises when there is a triangle between three successive neighboring particles(see Figure 2(f)). Contrary to the configuration shown in Figure 2(d), P^0 and P^3 are neighbors. We still gather the four triangles, but this time we discard $(\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2)$ and $(\mathbf{p}^1, \mathbf{p}^2, \mathbf{p}^3)$. Moreover, we have to cancel the neighborhood relation between P^1 and P^2 since they are not neighbors anymore (see Figure 2(g)).

The last configuration is slightly different: this time, the problem occurs because the particle set misses one particle. For instance, in the configuration shown in Figure 2(h), the ideal scheme is achieved by adding a particle at the star's gravity center. Thus, we discard all triangles that belong to the star, add a new particle P^5 located at its gravity center and generate all the corresponding triangles (see Figure 2(i)).

After the application of these three criteria, all the intersecting triangles (see Figure 2(b)) have been discarded leading to a polygonization with near-equilateral triangles. Figure 3 shows the results of our polygonization algorithm on two models. Table 1 shows the efficiency of our algorithm with various number of particles.

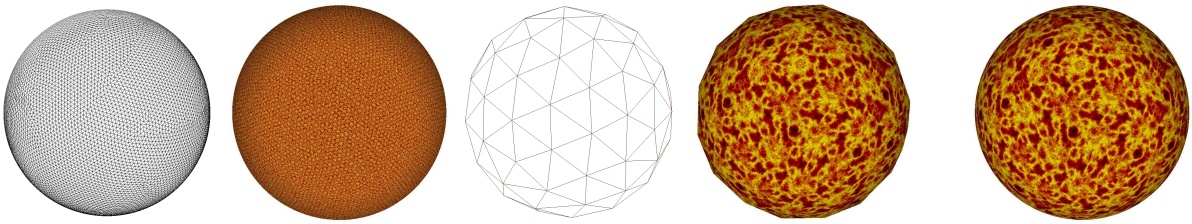
4 Tessellation of the triangulation

With the algorithm detailed above, we can rapidly generate a triangulation from a uniform particle set. So, the initial requirement is to efficiently get such a set of uniform particles. Unfortunately, because of their expensive relaxation process, particle systems proposed in the literature can not easily manage to sample a surface with thousands of particles in order to get the kind of triangulation shown in Figure 3.

As an alternative, we propose a two-step process, in which the implicit surface is first sampled with a small set of particles (this number is later increased by subdivision). The only need is that the general topology of the surface is not missed even if its geometric approximation is not sufficient because of the low number of particles. Thanks to the energy minimization of the particle system, the particles will be placed in order to catch as much details of the surface as possible. One nice characteristic of using the triangulation of the particle set given by the algorithm of Section 3 is that we still have the underlying implicit surface. Thus, we can further tessellate each triangle of the polygoniza-

# particles	Triangulation time	# triangles
682	0.010	1,360
2,813	0.045	5,622
17,178	0.263	34,292
36,464	0.685	72,915
73,379	1.175	146,671

Table 1: Times taken by the triangulation of uniform particle sets. All times are in seconds.

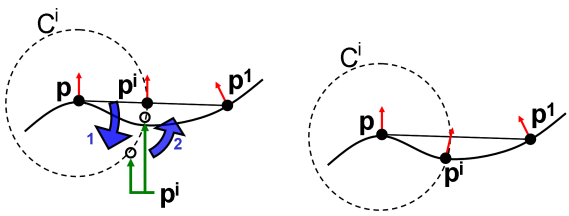


(a) High-detailed texture mesh (b) Direct visualization of the texture mesh (c) Low-detailed texture mesh (d) Direct visualization of the texture mesh (e) Visualization of the parametrized implicit surface

Figure 5: The scaling of the texture patterns is directly related to the texture mesh resolution. (a) Dense texture mesh (b) Texture patterns applied on dense texture mesh (high frequencies of the patterns are lost) (c) Coarse texture mesh (d) Texture patterns applied on coarse texture mesh (e) Coarse texture mesh mapping on the implicit surface (high frequencies of the patterns are preserved).

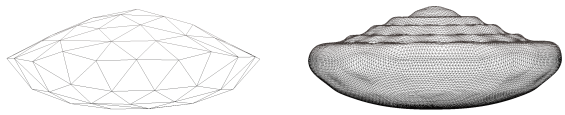
tion of the particles in order to get a good geometric approximation of the model.

Based on [9, 10], we subdivide each triangle as shown in Figure 4. But, contrary to them, we do not reproject the newly inserted vertices on the implicit surface by applying one step of the Newton-Raphson iteration. Indeed, we decided to reproject them by using an improvement of the geodesic reprojection of [13]. The idea is to reproject a newly inserted vertex $\mathbf{p^i}$ on the geodesic of the circle C^i centered on $\mathbf{p}$ (an original vertex of the triangle) which has a radius of $|\mathbf{p} - \mathbf{p^i}|$. But, contrary to [13], we use a dichotomic process. Indeed, because the evaluation of the implicit surface enables us to know if the vertex is inside or outside it, we can apply the iterative process shown in Figure 6(a). At the beginning, the vertex $\mathbf{p^i}$ is outside the surface. During the first reprojection step, a $\pi/4$ rotation is applied on $\mathbf{p}^i$ and the implicit equation is evaluated at this new position. If the vertex is still outside we do not change the direction of rotation. But, if the vertex is now inside, this direction of rotation is inverted (see Figure 6(a)) and its angle is divided by two. This process is repeated until the new vertex lies on the surface (see Figure 6(b)) according to a user-provided threshold on the implicit equation.



(a) (b)

Figure 6: Geodesic reprojection of a vertex: a dichotomic process is used. (a) Successive iterations of the reprojection are applied. (b) Final position of the vertex.



(a) (b)

Figure 7: Example of the tessellation of a model. (a) The implicit surface was sampled with a very low number of particles (74) leading to a polygonization with big triangles. (b) After applying a level 15 tessellation (the final model has 30,600 triangles), all the details of the surface are visible.

5 An application: the texturation of implicit surfaces

Neyret and Cani [16] have designed a pattern-based texturing of triangular meshes. Its principle is to create, over an existing arbitrary mesh, a *triangular texture mesh* composed of a small number of almost-equilateral triangles. The original mesh is used as a local parametrization of each triangle of the texture mesh. Patterns are then mapped on the texture mesh and the parametrization in a non-periodic way with no visible repetition of the patterns.

Thanks to the characteristics of our polygonization and tessellation, this technique can be directly adapted to the texturation of implicit surfaces. Indeed, the polygonization of the particle set (see Section 3) can be used as the texture mesh since each triangle is quasi-equilateral. Nevertheless, care must be taken on the size of the triangles (and thus the number of particles which sample the implicit surface). Indeed, patterns are mapped on the texture mesh. So, a large number of particles generated a good geometric approximation of the surface, but dramatically reduces the size of texture patterns which becomes hardly recognizable (see Figure 5(a)-(b)). On the other hand, a small number of particles, generates recognizable patterns, but does not provide a good geometric approximation of the surface (see Figure 5(c)-(d)).

Triangulation, tessellation and texturation of different models							
Model	# particles	Triangulation Time in sec.	# triangles	Tessellation		Total # of triangles	Total Time
				Level	Time		
Sphere	76	0.002	148	15	0.344	33,300	0.346
Igea	117	0.002	230	10	1.545	23,000	1.547
Rabbit	257	0.004	509	5	0.708	12,725	0.712
Bunny	1,216	0.017	2,425	5	3.358	60,625	3.375

Table 2: Times for the texturing of different implicit surfaces with changing configurations (all given times are in seconds). We give the generation number of uniform particles which is the starting point of our algorithm. The number of triangles of the resulting texture mesh is shown as the time needed to triangulate and simplify this mesh. Finally, we give the time and the level of tessellation leading to the parametrization of these implicit surfaces.

The solution is to use the tessellation of the Section 4 as the parametrization of the texture mesh [16]. The patterns are then mapped from the texture mesh to the triangles resulting of the tessellation. Thus, the geometric density of the texture mesh can be improved as much as desired. The result of using the tessellation this way can be seen in Figure 5(e) which shows similar sized patterns as Figure 5(d), with the same geometric accuracy as Figure 5(b). This indirection between the texture pattern resolution and the geometric density of the texture mesh, offers a very flexible scheme to easily adjust the apparent size of the pattern on the implicit surface.

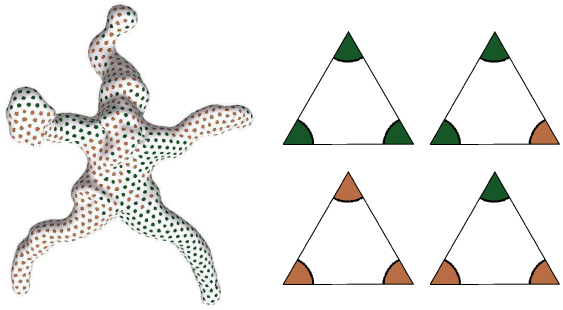


Figure 9: Texturing of the Santa model: the four patterns used (right) and the mapping result (left) are shown.

6 Results and discussion

All example models shown in this paper were built by using a PC with an Intel Pentium IV 3GHz processor and 1GB of main memory.

The sphere model shown in this paper was defined analytically, while the bunny, the Santa and the Igea models were reconstructed using Radial Basis Functions (RBF) [31]. These three models were reconstructed with a limited set of 200 to 500 initial points, because we used the original implementation of RBF with global support functions. But the technique can be easily adapted to more elaborated implicit reconstruction techniques, such as [17, 22] that reconstruct implicit surfaces from several millions of points. Note that our approach scales well since it only depends on the efficiency of the evaluation of the function that defines the implicit surface. This evaluation can be tuned using some efficient spatial subdivision structures. Note also that we removed some points that belong to the bunny ears, in order to increase the genus of the resulting surface, and thus better illustrate the genericity of our texturing technique (see Figure 8).

(a) Texture mesh.

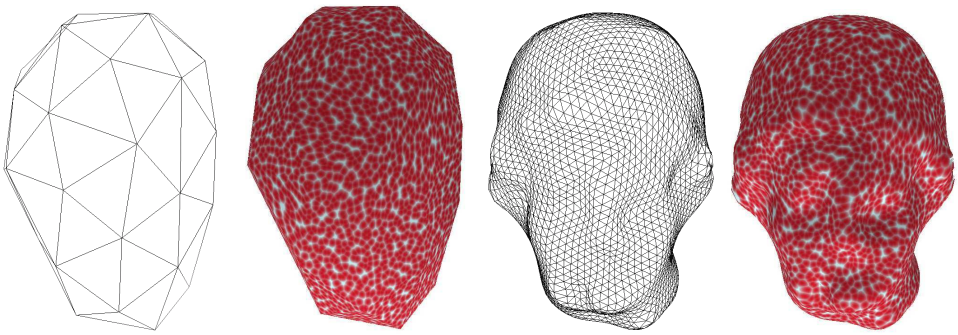
(b) Textured parametrization of the implicit surface.

Figure 8: A closer view of the bunny ears.

Once the texture pattern has been selected, our technique allows the user to finely tune two important parameters. First, changing the radius of the particles provides a full control of the number of desired particles and, thus, of the apparent size of the resulting texture patterns over the implicit surface. Second, changing the tessellation level provides a full control of the geometric accuracy used by the rendering step of the textured surface.

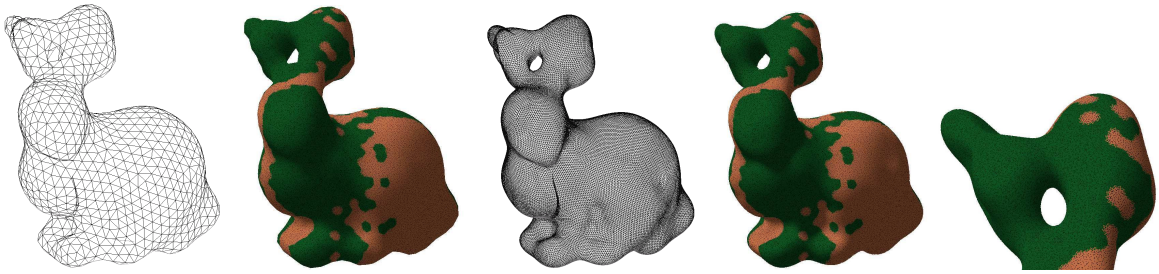
In Table 2, we show some timings taken by our technique to texture implicit surfaces with various level of sampling and tessellation. With our method, simple models can be textured in less than a second without loosing detail of the underlying surface since they can be textured well with only a poor detailed texture mesh (Figure 5(e) shows the texturing of the model presented in Table 2). With more complex models as the bunny, the texture mesh must be a little more detailed in order to not miss topology of the model. So more particles are needed and the overall time of the texturing is larger.

Figure 10 is a good example of our technique. Fig-



(a) Texture mesh (b) Direct visualization of this texture mesh (c) Parametrization of the implicit surface (d) Texturing of this parametrization

Figure 10: Texturation of the Igea model. The texture mesh is very low-detailed. Note how the parametrization adds details on the model geometry with the same texture support.



(a) Texture mesh (b) Direct visualization of this texture mesh (c) Parametrization of the implicit surface (d) Texturing of this parametrization (e) Closer view of the bunny ears

Figure 11: Texturation of the bunny: the texture mesh and the parametrized implicit surface are shown. Note how our method manage to texture the bunny ears even if the genus of the model has been increased.

ure 10(a) shows the support texture mesh for the rendering of the implicit surface. It is composed of 74 triangles. At this time we can hardly recognize the Igea model. We give the parametrization of the implicit surface on which the texturing will be applied in Figure 10(c) (it's a level ten tessellation). In Figure 10(b) and Figure 10(d) the same texture mesh is used. But, while it is directly rendered in Figure 10(b), it is the parametrization of the implicit surface that is visualized in Figure 10(d) which enable us to recognize the Igea model.

7 Conclusion and future works

In this paper, we have presented a new very efficient algorithm to triangulate a uniform particle set. Based on the intrinsic characteristics of these uniform particle systems, our algorithm can triangulate more than 30,000 particles in less than a second. A second contribution of this work is an on-the-fly mesh refinement technique, driven by the underlying implicit equation, that allows to further tessellate each triangle of the polygonization. Finally, this triangulation combined with the tessellation can be directly used in order to map homogeneous non-periodic textures to arbitrary implicit surfaces thanks to a 2D pattern-based texturing method.

Based on this initial technique, we are currently investigating several extensions: for instance, the application of the method to generate out-of-core rendering of huge textured implicit surfaces reconstructed from scanned objects, as well as a local edition of the particle system in the case of animated skeleton-based implicit surfaces.

Acknowledgments

We wish to thank Fabrice Neyret for providing his pattern texture code for meshes.

References

[1] D. Benson and J. Davis. Octree textures. In *Proc. ACM SIGGRAPH '02*, pages 785–790, 2002.

[2] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[3] P. Crossno and E. Angel. Isosurface extraction using particle systems. In R. Yagel and H. Hagen, editors, *IEEE Visualization '97*, pages 495–498, 1997.

[4] P. Crossno and E. Angel. Spiraling edge: Fast surface reconstruction from partially organized sample points. In *VISUALIZATION '99: Proceedings of the*

10th IEEE Visualization 1999 Conference (VIS '99). IEEE Computer Society, 1999.

[5] L. H. de Figueiredo, J. Gomes, D. Terzopoulos, and L. Velho. Physically-based methods for polygonization of implicit surfaces. In *Proc. Graphics Interface '92*, pages 250–257, 1992.

[6] M. Desbrun, N. Tsingos, and M.-P. Cani. Adaptive sampling of implicit surfaces for interactive modeling and animation. In *International Workshop on Implicit Surfaces '95*, avril 1995. Published under the name Marie-Paule Gascuel.

[7] E. Galin, R. Allgre, and S. Akkouche. A fast particle system framework for interactive implicit modeling. In *SMI (Shape Modeling Internatinal)*, 2006.

[8] J. C. Hart, E. Bachta, W. Jarosz, and T. Fleury. Using particles to sample and control more complex implicit surfaces. In *Shape Modeling International 2002*, pages 129–136, 2002.

[9] T. Igarashi and J. F. Hughes. Smooth meshes for sketch-based freeform modeling. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 139–142. ACM Press, 2003.

[10] X. Jin, H. Sun, and Q. Peng. Subdivision interpolating implicit surfaces. *Computers & Graphics*, 27(5):763–772, 2003.

[11] T. Karkanis and A. J. Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics & Applications*, 21(2):60–69, 2001.

[12] S. Lefebvre, S. Hornus, and F. Neyret. *GPU Gems 2*, chapter Octree Textures on the GPU, pages 595–613. 2005.

[13] F. Levet, X. Granier, and C. Schlick. Fast sampling of implicit surfaces by particle systems. In *Shape Modeling International 2006 (Short Papers)*, 2006.

[14] F. Levet, J. Hadim, P. Reuter, and C. Schlick. Anisotropic sampling for differential point rendering of implicit surfaces. In *WSCG'2005*, pages 109–116, 2005.

[15] M. D. Meyer, P. Georgel, and R. T. Whitaker. Robust particle systems for curvature dependant sampling of implicit surfaces. In *Shape Modeling Internatinal 2005*, 2005.

[16] F. Neyret and M.-P. Cani. Pattern-based texturing revisited. In *Proc. ACM SIGGRAPH '99*, pages 235–242, 1999.

[17] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.

[18] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 2(8):429–446, 1995.

[19] D. R. Peachey. Solid texturing of complex surfaces. In *Proc. ACM SIGGRAPH '85*, pages 279–286, 1985.

[20] H. K. Pedersen. Decorating implicit surfaces. In *Proc. ACM SIGGRAPH '95*, pages 291–300, 1995.

[21] P. Reuter, B. Schmitt, A. Pasko, and C. Schlick. Interactive solid texturing using point-based multiresolution representations. In *WSCG'2004*, pages 363–370, 2004.

[22] P. Reuter, I. Tobor, C. Schlick, and S. Dedieu. Point-based modelling and rendering using radial basis functions. In *ACM Graphite 2003*, pages 111–118, 2003.

[23] A. Rosch, M. Ruhl, and D. Saupe. Interactive visualization of implicit surfaces with singularities. *Eurographics Computer Graphics Forum*, 16(5):295–306, 1997.

[24] R. Schmidt, B. Wyvill, M. Sousa, and J. Jorge. Shapeshop: Sketch-based solid modeling with blobtrees. In *2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 53–62, 2005.

[25] W. Y. Su and J. C. Hart. A programmable particle system framework for shape modeling. In *Shape Modeling Internatinal 2005*, 2005.

[26] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *Proc. ACM SIGGRAPH '92*, pages 185–194, 1992.

[27] M. Tigges and B. Wyvill. Texture mapping the blobtree. In *Proc. International Workshop on Implicit Surfaces*, pages 123–130, June 1998.

[28] M. Tigges and B. Wyvill. A field interpolated texture mapping algorithm for skeletal implicit surfaces. In *Computer Graphics International*, pages 25–, 1999.

[29] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91*, pages 289–298, 1991.

[30] G. Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2):55–64, 1992.

[31] G. Turk and J. O'Brien. Variational implicit surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology, 1998.

[32] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. ACM SIGGRAPH '94*, pages 269–277, 1994.

[33] B. Wyvill, E. Galin, and A. Guy. The blobtree. warping, blending and boolean operations in an implicit surface modelling system. *Eurographics Computer Graphics Forum*, 18(2):149–158, 1999.

[34] G. Wyvill, B. Wyvill, and C. McPheeters. Solid texturing of soft objects. *IEEE Computer Graphics & Applications*, 7(12):20–26, 1987.

[35] R. Zonenschein, J. Gomes, L. Velho, and L. H. de Figueiredo. Texturing implicit surfaces with particle systems. In *ACM SIGGRAPH 97 Visual Proceedings*, 1997.

[36] R. Zonenschein, J. Gomes, L. Velho, and L. H. de Figueiredo. Controlling texture mapping onto implicit surfaces with particle systems. In *International Workshop on Implicit Surfaces '98*, pages 131–138, 1998.

[37] R. Zonenschein, J. Gomes, L. Velho, L. H. de Figueiredo, M. Tigges, and B. Wyvill. Texturing composite deformable implicit objects. In *SIBGRAPI'98*, pages 346–353, 1998.