

A Representation of Implicit Objects Based on Multiscale Euclidean Distance Fields

Antônio L. Apolinário Jr. and Claudio Esperança
Laboratório de Computação Gráfica
Rio de Janeiro – Brasil
{alopes, esperanc}@lcg.ufrj.br

Luiz Velho
Instituto de Matemática Pura e Aplicada
Rio de Janeiro – Brasil
lvelho@visgrafimpa.br

Abstract

Objects can be represented at different levels of detail within a required precision. A given level of detail can be defined based on the concept of scale. Not rarely applications need to deal with a single model represented at different scales. Two main approaches are used in Computer Graphics to generate different versions of an object: intrinsic and extrinsic. Both techniques are based on removing details from an object. The main difference between them is that intrinsic methods work directly on a surface whereas extrinsic methods use ambient space. This paper proposes a new way to represent objects in different levels of detail by combining intrinsic and extrinsic methods.

Keywords

Multiscale, Space-scale, Multiresolution, Implicit Objects, Simplification, Fairing, Polygonal Meshes

1. Introduction

Scale is a natural concept that allows us to deal with complex objects in a hierarchical way. An object representation can reveal more or less details as the scale changes from fine to coarse. This idea is vastly explored in science. For example, astronomy, biology and physics, use the concept of scale to describe objects under different perspectives. Finer scale is used to describe micro structures, which are part of the object. On the other hand, coarse scales can represent the relations between objects and their environment.

In Computer Graphics, scale is a powerful tool to deal more efficiently with many tasks, such as visualization, collision detection and animation. Two main approaches are used when objects in different scales have to be generated: intrinsic and extrinsic.

Traditionally, extrinsic methods are applied in Computer Vision and Image Processing, whereas intrinsic techniques are used in Geometric Modeling. Both have the same basic idea: starting with a fine scale representation, apply operators that gradually remove details from the object. This process is called *analysis*. The main issues in this process are how to select information to be removed and how to measure the difference between the original object and the simple model.

On the other hand, we call *synthesis* the process of reconstructing a more complex model, based on a coarse one.

The difference between the intrinsic and extrinsic approaches is the nature of their operators. While extrinsic operators remove details from a function defined on the ambient space where the object is embedded, intrinsic operators act directly on the object surface.

The main contribution of this work is to define a new representation scheme, which blends concepts of intrinsic and extrinsic methods. Starting from an object described as a polygonal mesh, we build a hierarchical implicit representation of it. The implicit representation is based on the Euclidean Distance Function. At each level a smooth version of the polygonal mesh will be used.

In Section 2 we describe the geometric concepts that this work is based on. In Section 3 we discuss the two main methods associated with intrinsic and extrinsic representations. Section 4 presents the main algorithm that builds our hierarchical representation. In Section 5 we discuss some applications that can be improved by this new representation scheme and present some preliminary results. Finally, in Section 6 we point out a few ideas that can be used to enhance our representation.

2. Object Representation

In this Section we briefly describe some basic concepts, techniques and algorithms related to our work.

Three-dimensional objects can be represented in a variety of forms, each one focusing on a class of application requirements. The most traditional forms are extrinsic representations such as *Polygonal Meshes* and *Parametric Objects*, and intrinsic representations such as *Implicit Objects*.

2.1. Polygonal Meshes

A polygonal mesh represents a piecewise linear approximation of an object's surface. It is composed of a set of planar polygonal faces, usually triangles. Simple and flexible, this scheme of representation has been extensively used in Computer Graphics, because graphics hardware can be optimized to process triangles. Complex objects usually require a lot of polygons to produce a good approximation, therefore the mesh size can be a problem.

2.2. Parametric Objects

This scheme represents the surface of an object using a parametric surface such as :

$$F : V \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3 \quad (1)$$

where V is the parametric domain.

The evaluation of F at a point $(u, v) \in V$ generates a point on the surface of the object. Thus, visualization of parametric object is a simple task, since a convenient set of polygons can be generated by sampling the parametric space in a structured manner.

Shape control is another important property of parametric objects. Control points are associated with each object definition, acting like local attractors of the surface. This can be a very useful tool for designers.

Just one parametric surface may not be enough to model complex objects. In such cases, designers could use control patches, each one defining a parametric surface. Continuity should be a major concern in this case. Since it is usually desired that the surface be continuous and smooth [Farin, 1996], these requirements tend to increase the computational complexity.

2.3. Implicit Objects

A surface S of an object can be represented implicitly by a set of points which satisfy

$$S = \{p \in \mathbb{R}^3 : f(p) = c.\} \quad (2)$$

The roots of the equation $f(p) - c = 0$ determine a set of points and represents the surface of the implicit object.

The implicit function f can be interpreted as a *generalized distance function* from a lower dimension geometric element called *skeleton*. Mathematically, a skeleton is the set of points where the gradient of f vanishes.

This approach was first proposed by Jim Blinn[Blinn, 1982]. He used a Gaussian function as a distance function and points in space as skeletons. The global distance function is defined based on the sum of each distance function of each skeleton. Alternative distance function, metrics and skeleton elements were proposed by [Bloomenthal & Shoemake, 1991] and [Blanc & Schlick, 1995], among other researchers, in order to improve shape and control over the object's surface.

Visualization of implicit objects is not a direct procedure, since it requires a root finder algorithm to solve $f(p) - c = 0$. Direct visualization, like Ray Tracing [Hart, 1993], is a straightforward way to render implicit objects, although not suitable to real time visualization. Polygonization algorithms can be used in this case to produce a polygonal approximation. The two major approaches are based on particle systems (e.g. [de Figueiredo *et al.*, 1992], [Witkin & Heckbert, 1994]), and space decomposition algorithms [Velho, 1990].

3. Multiscale and Multiresolution Methods

Multiscale methods are largely used in Computer Vision and Image Processing [Witkin, 1986]. In this context, an object can be characterized as a signal, which should be embedded into a family of derived signals. The analysis of the derived signal allows the identification of fine-scale structures that can be removed.

On the other hand, multiresolution methods work with a discrete approximation. Lower resolution means less geometric detail.

Clearly multiscale and multiresolution methods are closely related with, respectively, extrinsic and intrinsic representations.

In this Section we discuss some methods associated with each kind of representation.

3.1. Simplification Methods

The simplification process is the usual approach for reducing polygonal surface complexity. A global solution is characterized by algorithms based on vertex clustering. Such methods are based on grouping vertices into sets, which will be replaced later by a single vertex [Rossignac & Borrel, 1993]. Clusters are usually defined by the cells of a regular grid. Although simple, this kind of approach could lead to significant changes in topology. However, if the original model is over-sampled [Lindstrom, 2000] this kind of algorithm tends to work very well.

Another way to deal with the simplification problem uses a local approach called *iterative contraction*. For each pair of vertices, a cost function is evaluated and the pair with lowest cost is removed and replaced by a new vertex. Clearly, the two main issues here are the design of the cost function and the choice of the new vertex. Two classical implementations of this kind of method are [Garland & Heckbert, 1997] and [Hoppe, 1996].

An important remark is that simplification algorithms may produce new objects that continue in the same scale of the original.

Two good surveys of simplification methods and multiresolution schemes are [E. Puppo, 1997] and [Garland, 1999].

3.2. Scale-space and Wavelets

Two main mathematical frameworks are used as theoretical support for multiscale representation : scale-space and wavelets.

Scale space representation is a special kind of multiscale representation that comprises a continuous scale parameter and preserves the same spatial sampling at all scales. The theory involved guarantees, among other results, that properties such as isotropy (rotation invariance), homogeneity (invariance under translation) and causality (no new structure can be created in the transformation from fine to coarse scales) [Lindeberg, 1994].

There is a close relationship between scale-space and PDEs, particularly the Heat Equation [Cunha *et al.*, 2001]. This relationship can be used to fair irregular meshes in an intrinsic way [Desbrun *et al.*, 1999].

Wavelets representations decompose the object hierarchically from fine to coarse scales with decreasing spatial sampling [Stollnitz *et al.*, 1995a], [Stollnitz *et al.*, 1995b]. That leads to a rapidly decreasing representation size, reducing computational effort for processing and storage.

3.3. Our Approach

Simplification algorithms can reduce the complexity of objects, removing geometric elements like edges and faces. The algorithm itself can be simple and relatively inexpensive. But in order to obtain good results, more complex criteria must be built to decide where and when information can be removed.

On the other hand, extrinsic methods have a simple and consistent approach to remove detail information, using diffusion propagation. But the mathematical result is defined over functions such as $f : \mathbb{R}^n \rightarrow \mathbb{R}$, not over three dimensional surfaces.

This work presents the main ideas of a new representation scheme that mixes intrinsic and extrinsic methods in order to produce a hierarchical multiscale model.

Starting from an polygonal mesh, we will build an implicit representation based on the Euclidean distance function, generated in a certain resolution. Iteratively we apply a fairing process over the polygonal mesh and construct a new implicit representation, based on a coarse scale. This procedure constructs a hierarchical structure that allows us to reconstruct the object using a simple trilinear interpolation.

4. Hierarchical Multiscale Distance Function Representation

A general procedure to construct the Hierarchical Multiscale Distance Function Representation (HMDF) is presented in *Algorithm 1*.

Algorithm 1 : HMDF Construction

```

bs = EstimateBaseScale(Object,MAXLEVEL)
for s varying from 0 to bs do
    EqualizeObject(s);
    FairObject(s);
    GenerateDistanceFunction(s);
    PolygonizeDistanceFunction();

```

Let us discuss each aspect of this algorithm.

4.1. Building the Implicit Representation

To construct an implicit representation, based on the distance function from a polygonal mesh object we can use two kinds of algorithms : numerical or geometric.

A classical numerical approach solves the *Eikonal equation* :

$$|\nabla u(X)| = F(X) \quad (3)$$

$$u = g(X) \quad (4)$$

where $u(X)$ is a function defined in a domain Ω and $g(X)$ defines the boundary condition along a given curve or surface Γ in Ω .

When $F(X) = 1$ and $g(X) = 0$, the solution of the Eikonal equation gives us the signed distance defined by Γ in Ω . One of the numerical methods to solve (3) is the *Fast Marching Method* [Sethian, 1999], based on Level Set Theory.

Another approach proposed in [Yngve & Turk, 1999] uses the *Variational Implicit Surface*, which uses the vertices of the polygon mesh as constraints of a linear system.

The geometric approach uses a proximity structure to define, for each point in space the closest point on the surface. The *Voronoi Diagram* [F. P. Preparata, 1991] is a classical structure to represent regions of space where a certain point is always the closest one. Based on this idea, Mauch [Mauch, 2000] proposed an algorithm that uses an generalized Voronoi-like structure to define such proximity regions for points, edges and faces.

All the methods described previously represent the distance function as discrete volumetric samples of a scalar field – the Euclidean Distance Field. Adaptive representations, usually based on octrees may also be used [Alyn Rockwood & Jones, 2000].

In our implementation we decided to use the method described in [Mauch, 2000], as a simple solution.

4.2. Estimating the Base Scale

Since we are dealing with a discrete representation of the distance function, the definition of an initial resolution

used to sample the distance function is critical. The complexity of the distance function generation algorithm depends on this initial resolution. Also, this initial sample rate needs to be sufficient to capture as much detail as possible, since the next steps will remove details.

To estimate this initial resolution we use an heuristic based on the maximal tubular neighborhood concept. This can be intuitively defined as the maximum value ε , associated with normal vector length. If this value is applied as a scale factor to every normal vector of every point on the surface, there is no overlap.

Based on this concept, we can define a simple heuristic to estimate the maximal tubular neighborhood. We will search for the smallest cell which contains just one vertex or face. The point is we want to separate elements associated with surface details, like a concavity or a convexity. In these regions the maximal tubular neighborhood tends to be smaller.

To find this minimum cell we build an adaptive subdivision of the space. Although simple and fast, this criterium is sensitive to the object resolution.

4.3. Constructing the Multiscale Object

Although extrinsic methods can not be applied directly on 3D surfaces, the idea of removing details using a smooth Gaussian kernel can be adapted. Applying a discrete Gaussian based filter over a polygonal mesh, Taubin [Taubin, 1995] developed a method that smoothes a surface as a Gaussian filter is applied to scale space.

This kind of method works fine if the polygonal mesh has two characteristics : the faces have a good aspect ratio and the model has a sufficient resolution (i.e. if it is at a proper scale). In order to guarantee these conditions we apply an equalization procedure, that refines the surface until its resolution is compatible with a certain scale level. This step will be presented in the next Section.

Other methods such Discrete Fairing [Kobbelt, 2000] and Implicit Fairing [Desbrun *et al.*, 1999] have been proposed recently in order to fair polygonal meshes, using more sophisticated approaches, such as the Laplace-Beltrami operator and implicit methods. In this work we are using Taubin's method for the sake of simplicity.

4.4. Equalization

Equalization procedures guarantee that the resolution of the polygonal mesh and discrete distance function are compatible. In order to obtain such compatibility we refine the original polygonal mesh, changing its resolution, based on the reference scale used.

We use a simple criterion to decide when a face must be subdivided. The face is projected on the XY, XZ and YZ planes. Then we take the largest projection and compare it with the corresponding grid cell projection. The ratio between the sizes of the cell and face projections indicates if the face must be subdivided.

The algorithm processes the subdivision in two steps. First

the edges are refined by midpoint subdivision. This refinement just inserts a vertex on the triangle's edge, it does not split the face yet. Once this step ends, we may have triangular faces where one, two or all of its 3 edges may have been split. On the second step we promote local face tessellation, to generate again a triangular mesh.

This two step procedure guarantees that non-conformable triangles will be avoided, and that the result triangles will keep a good aspect ratio. This last condition is important because it can interfere in fairing algorithm.

4.5. Reconstruction

At the end of the algorithm we have a hierarchical volumetric representation of the Euclidean distance function. At this point, we are able to reconstruct any intermediate scale level by applying an interpolation procedure. Given a target scale, we determine the two scale levels at the hierarchical representation, that the target falls in-between. We estimate the interpolation factor and apply a trilinear interpolation procedure. More accurate interpolation methods could be used, such as quadratic interpolation.

Once we determine the scale level, we can visualize the model in two different ways, as we can see in figure 1.

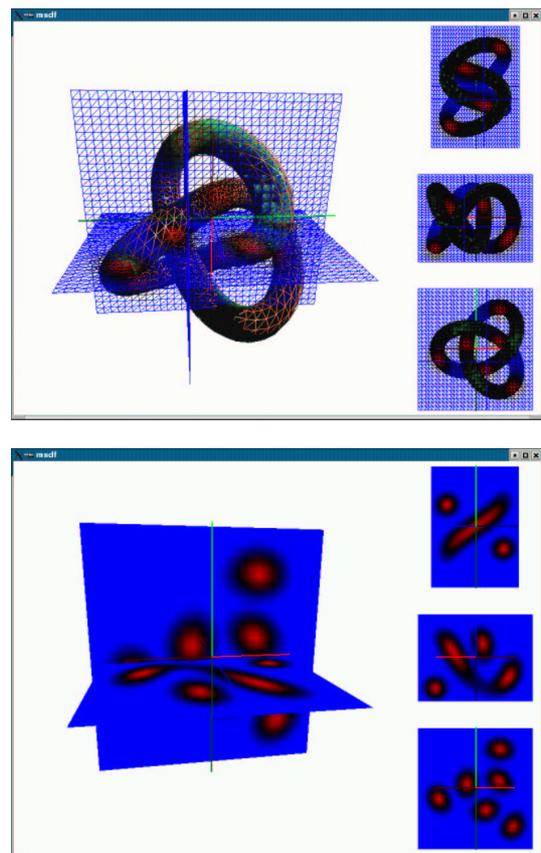


Figure 1. Two visualizations of the same HMDF representation. On the left the original object and on the right the cross sections of the distance function.

To visualize the surface of the object we used the Marching Cubes algorithm [Lorensen & Cline, 1987]. We can select which iso-surface will be generated (default is $f(p) = 0$). The normal vector at each vertex on the polygonization is obtained as an interpolation of the gradient on the grid vertices.

The other visualization mode is by means of a cross section of the volumetric data [Bloomenthal, 1997]. As can be seen in Figure 1, there are three main cross sections projected and a 3D view of them. In the current implementation, the user can only see planes that are orthogonal to the three main axes.

4.6. Implementation Details

As a preliminary result of this research, we implemented a prototype system, coded in C++.

The main class in the system is the Hierarchical Distance Function class (HDF). It manages the execution of the algorithm described in Section 4. It is composed of a set of Regular Distance Function classes (RDF), each one representing a resolution level. The HDF class also controls processes such as interpolation between RDFs, error analysis, etc.

The main task performed by RDF class is to control the interface between system and the algorithm used to generate distance functions. So, we can easily change the distance function generation algorithm in a transparent way.

A class called Distance Function (DF) is also created, in order to store and manage the discrete distance function, which is stored as a 3D matrix. Tasks such as the evaluation of the distance function, gradient and closest surface point are performed by this class. The polygonization algorithm is also part of DF class.

To represent polygonal objects we used a half-edge data structure, because some operations like fairing and equalization need to traverse the faces using neighborhood information. Instead of re-implementing this data structure, we used the Computacional Geometry Algorithms Library – CGAL [CGAL, 2001], which is a robust and stable library that deals with geometric data structures, such as convex hulls, triangulations, topological maps and search structures, among others. This library is very powerful as it can be extended using the inheritance concept of object oriented languages.

The basic data structures, such lists, vectors, etc, needed during the implementation are provided by the Standard Template Library [Stroustrup, 1997].

5. Results

In this section we show some preliminary results and discuss some applications of this representation.

5.1 Experiments

Some preliminary results can be seen in Figures 2, 3 and 4. Each figure shows a model at two different resolutions. We chose to present all three models in wireframe to emphasise the polygonization resolution as it changes with scale.

Figure 2 shows a sphere. As its curvature does not vary along the surface, the smoothing process has no visible effects. In other words, as a sphere model has no surface details, the fairing process produces no changes. But the resolution produced by the equalization process, makes the number of faces increase according to the scale.

The pyramid model, shown in Figure 3, is clearly affected by fairing. It acts mostly at sharp edges, smoothing them as the scale increases.

Finally, the bunny model (Figure 4) presents an object with no sharp edges but with a surface with non constant curvature. The fairing process removes gradually the surface details as the scale increases.

Another interesting experiment is to control the scale based on the distance from the observer. Low resolution models can be used if the object is far away. This idea is presented in Figure 5. It shows a high resolution model when the object is near the camera, and a coarse one when it is far away. In Figure 6 the different resolutions are seen from the same point of view. We can clearly see that in lower resolution some model details were removed.

Figure 6 also shows the polygon mesh (in orange) generated by the equalization and fairing processes. Based on this mesh the implicit model is calculated. The green shaded mesh is built by the polygonization algorithm, applied over the volumetric distance function. As we can see the original mesh from each scale varies as a function of the scale. The difference between the two meshes is the limited precision associated with the interpolation process, and by the inherent finite precision of z-buffer.

5.2 Applications

The first application used to test the hierarchical representation was visualization. As we know, this is a basic problem in Computer Graphics. The main goal is to balance the complexity (resolution) of a model with the visible area. Thus, objects could be represented at lower resolutions when far away from the observer, and at high resolutions when closer.

Our representation can be used in this kind of application. Moreover, it can generate a continuous range of models from coarse to fine resolution, just using an interpolation procedure between consecutive distance functions.

Another straightforward application is Collision Detection. This kind of application can be optimized if we know how far one object is from another. This could be obtained easily, since we have a representation that stores the distance function in a volumetric data structure. The procedure can be also improved by the fact we have different resolutions of the distance function, which can gradually give a more precise information as the object becomes closer.

6. Conclusions and Future Work

This paper introduces a new way to represent objects, based on a Hierarchical Multiscale scheme. This representation is built from an initial object described by a polygonal surface. This object is converted to an implicit rep-

resentation generated by the discrete Euclidean Distance Function.

Some improvements will be carried out as the next steps of our research :

- Once we have a multiscale representation of the object, we plan to evaluate how we can introduce simplification algorithms, ([Lindstrom, 2000], [Garland & Heckbert, 1998]) to reduce the size of the meshes after the equalization/fairing step.
- New polygonization algorithms, such as [Leif Kobbelt & Seidel, 2001] can improve the approximation quality of traditional marching cubes, using additional information derived from distance functions.
- Other algorithms to generate the distance function could be used, like [Sethian, 1999].
- A more efficient data structure could be defined based on the fact that the resolutions can be embedded.

Acknowledgements

The authors are partially supported by research grants from the Brazilian Council for Scientific and Technological Development (CNPq).

References

- [Alyn Rockwood & Jones, 2000] ALYN ROCKWOOD, SARAH FRISKEN, RONALD PERRY, & JONES, THOUIS. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. July.
- [Blanc & Schlick, 1995] BLANC, CAROLE, & SCHLICK, CHRISTOPHE. 1995 (April). Extended fields functions for soft objects. *Pages 21–32 of: Implicit surfaces '95*. is95.
- [Blinn, 1982] BLINN, JAMES F. 1982. A generalization of algebraic surface drawing. *Acm transactions on graphics*, **1**(3), 235–256.
- [Bloomenthal, 1997] BLOOMENTHAL, JULES (ed). 1997. *Introduction to implicit surfaces*. San Francisco, California: Morgan Kaufmann Publishers, INC.
- [Bloomenthal & Shoemake, 1991] BLOOMENTHAL, JULES, & SHOEMAKE, KEN. 1991. Convolution surfaces. *Computer graphics*, **25**(4), 251–256. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).
- [CGAL, 2001] CGAL. 2001 (v.2.3). *Computational geometry algorithms library*. <http://www.cgal.org/Manual>.
- [Cunha et al. , 2001] CUNHA, ANDERSON, TEIXEIRA, RALPH, & VELHO, LUIZ. 2001. Discrete scale spaces via heat equation. *In: Proceedings of sibgrapi 2001 - xiv brazilian symposium on computer graphics and image processing*. Florianópolis, Brazil: IEEE Press, for SBC - Sociedade Brasileira de Computacao.
- [de Figueiredo et al. , 1992] DE FIGUEIREDO, LUIZ HENRIQUE, GOMES, JONAS, TERZOPOULOS, DEMETRI, & VELHO, LUIZ. 1992. Physically-based methods for polygonization of implicit surfaces. *Pages 250–257 of: Proceedings of graphics interface '92*. CIPS.
- [Desbrun et al. , 1999] DESBRUN, MATHIEU, MEYER, MARK, SCHRÖDER, PETER, & BARR, ALAN H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. *Pages 317–324 of: ROCKWOOD, ALYN (ed), Proceedings of the conference on computer graphics (siggraph99)*. N.Y.: ACM Press.
- [E. Puppo, 1997] E. PUPPO, R. SCOPIGNO. 1997. *Simplification, lod and multiresolution - principles and applications*. Eurographics Association.
- [F. P. Preparata, 1991] F. P. PREPARATA, M. I. SHAMOS. 1991. *Computational geometry: An introduction*. New York: Springer Verlag.
- [Farin, 1996] FARIN, G. E. 1996. *Curves and surfaces for computer aided geometric design: A practical guide*. Fourth edn. NY: AP.
- [Garland, 1999] GARLAND, MICHAEL. 1999. Multiresolution modeling: Survey future opportunities. *In: State of the art report(star)*. Eurographics.
- [Garland & Heckbert, 1997] GARLAND, MICHAEL, & HECKBERT, PAUL S. 1997. Surface simplification using quadric error metrics. *Proceedings of siggraph 97*, 209–216. ISBN 0-89791-896-7. Held in Los Angeles, California.
- [Garland & Heckbert, 1998] GARLAND, MICHAEL, & HECKBERT, PAUL S. 1998. Simplifying surfaces with color and texture using quadric error metrics. *Ieee visualization '98*, 263–270. ISBN 0-8186-9176-X.
- [Hart, 1993] HART, JOHN. 1993. Ray tracing implicit surfaces. *Pages 13.1–13.15 of: signotes93*. sigcno93.
- [Hoppe, 1996] HOPPE, HUGUES. 1996 (Aug.). Progressive meshes. *Pages 99–108 of: Siggraph '96 proc*.
- [Kobbelt, 2000] KOBBELT, LEIF P. 2000. Discrete fairing and variational subdivision for freeform surface design. *Pages 142–150 of: The visual computer*, vol. 16(3/4). Springer.
- [Leif Kobbelt & Seidel, 2001] LEIF KOBBELT, MARIO BOTSCH, ULRICH SCHWANECKE, & SEIDEL, HANS-PETER. 2001. Feature sensitive surface extraction from volume data. *Pages 57–66 of: Siggraph 2001 proceedings*. ACM Press, New York.

- [Lindeberg, 1994] LINDEBERG, TONY. 1994. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of applied statistics*, **21**(2), 224–270.
- [Lindstrom, 2000] LINDSTROM, PETER. 2000. Out-of-Core simplification of large polygonal models. *Pages 259–262 of: HOFFMEYER, SHEILA (ed), Proceedings of the computer graphics conference 2000 (SIGGRAPH-00)*. New York: ACM Press.
- [Lorensen & Cline, 1987] LORENSEN, WILLIAM, & CLINE, HARVEY. 1987. Marching cubes: a high resolution 3d surface construction algorithm. *Computer graphics*, **21**(4), 163–169. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).
- [Mauch, 2000] MAUCH, SEAN. 2000 (September). *A fast algorithm for computing the closest point and distance function*. Tech. rept. CalTech. unpublished.
- [Rossignac & Borrel, 1993] ROSSIGNAC, J., & BORREL, P. 1993 (June). Multi-resolution 3D approximation for rendering complex scenes. *Pages 453–465 of: Second conference on geometric modelling in computer graphics*. Genova, Italy.
- [Sethian, 1999] SETHIAN, J.A. 1999. *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision and materials science*. 2nd. edition edn. Cambridge University Press.
- [Stollnitz *et al.* , 1995a] STOLLNITZ, ERIC J., DEROSE, TONY D., & SALESIN, DAVID H. 1995a. Wavelets for computer graphics: a primer, part 1. *Ieee computer graphics and applications*, **15**(3), 76–84.
- [Stollnitz *et al.* , 1995b] STOLLNITZ, ERIC J., DEROSE, TONY D., & SALESIN, DAVID H. 1995b. Wavelets for computer graphics: a primer, part 2. *Ieee computer graphics and applications*, **15**(4), 75–85.
- [Stroustrup, 1997] STROUSTRUP, BJARNE. 1997. *The c++ programming language*. 3rd edition edn. Addison-Wesley Pub Co.
- [Taubin, 1995] TAUBIN, GABRIEL. 1995. A signal processing approach to fair surface design. *Pages 351–358 of: COOK, ROBERT (ed), Siggraph 95 conference proceedings*. Annual Conference Series Addison Wesley, for ACM SIGGRAPH. held in Los Angeles, California, 06-11 August 1995.
- [Velho, 1990] VELHO, LUIZ. 1990. Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraint. *Pages 125–136 of: Proceedings of eurographics '90*. Elsevier Science Publisher.
- [Witkin, 1986] WITKIN, A. P. 1986. Scale space filtering. *Pages 5–19 of: PENTLAND, A. P. (ed), From pixels to predicates: Recent advances in computational and robot vision*. Norwood, NJ: Ablex.
- [Witkin & Heckbert, 1994] WITKIN, ANDREW P., & HECKBERT, PAUL S. 1994. Using particles to sample and control implicit surfaces. *Proceedings of siggraph 94*, July, 269–278. ISBN 0-89791-667-0. Held in Orlando, Florida.
- [Yngve & Turk, 1999] YNGVE, GARY, & TURK, GREG. 1999. *Creating smooth implicit surfaces from polygonal meshes*. Tech. rept. GIT-GVU-99-42. Graphics, Visualization, and Usability Center. Georgia Institute of Technology. unpublished.

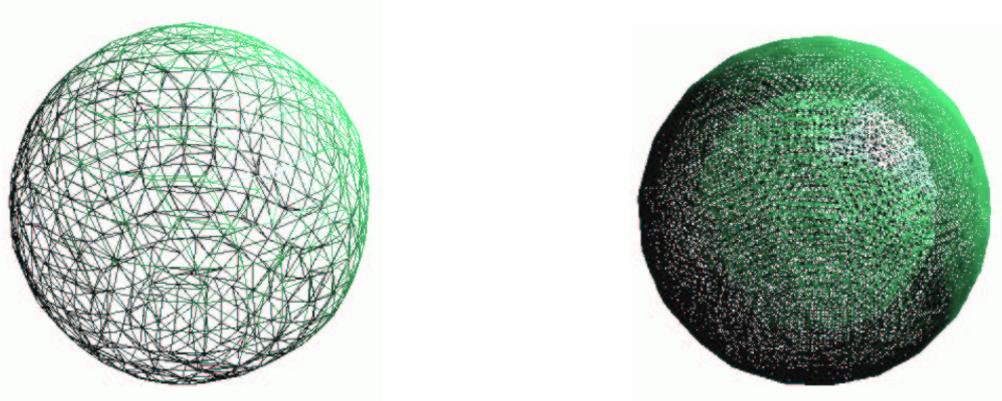


Figure 2. A sphere presented in two different scales: coarse (left) and fine (right). The shape of the two models are practically identical, although the resolution varies.

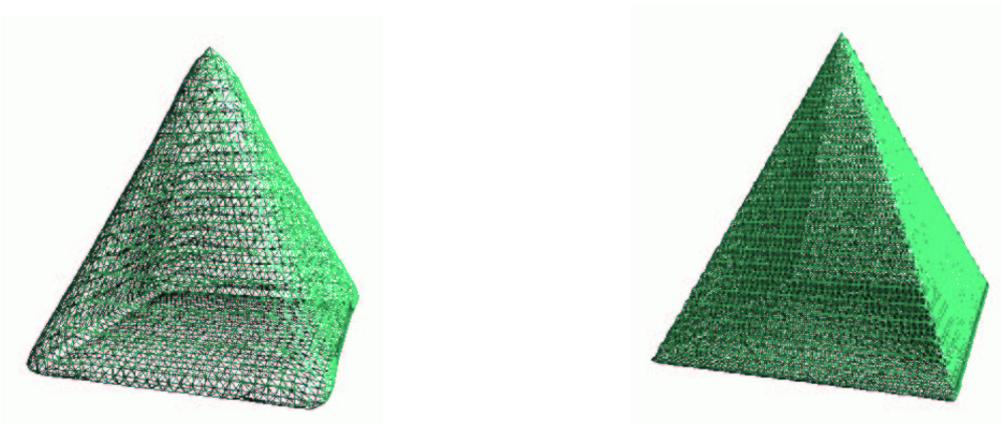


Figure 3. A pyramid shown in two different scales: coarse (left) and fine (right). The edges get smooth as the scale varies. The resolution of the models are proportional to its scale.



Figure 4. The bunny model shown from two different scales: coarse (left) and fine (right). The details of the surface are removed as the scale increases. The resolution of the models are proportional to its scale.

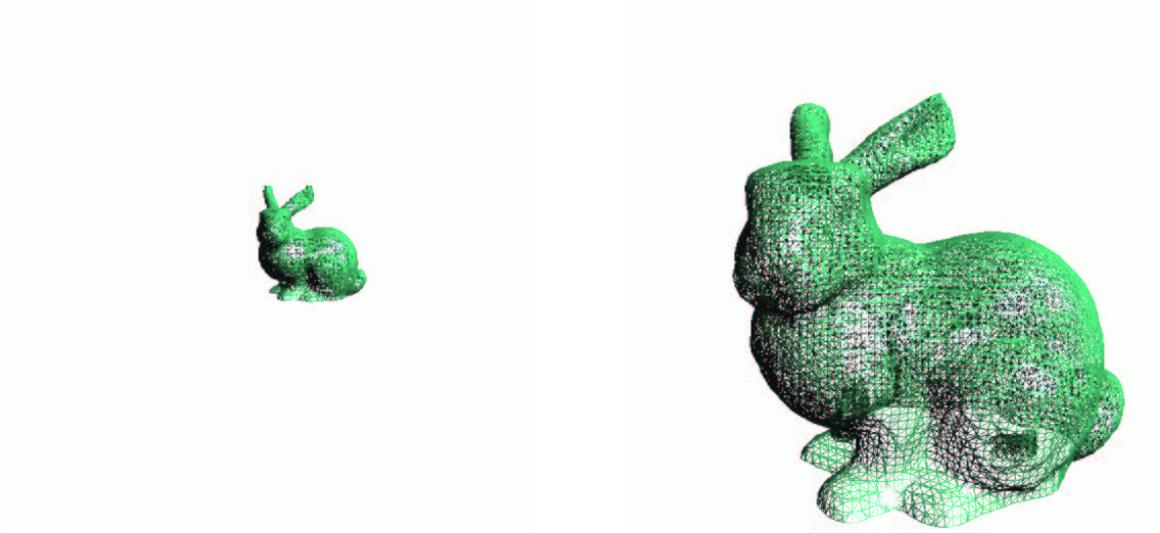


Figure 5. The bunny model is presented in two different points of view: far (left) and near (right). The model scale is proportional to the distance of the camera.

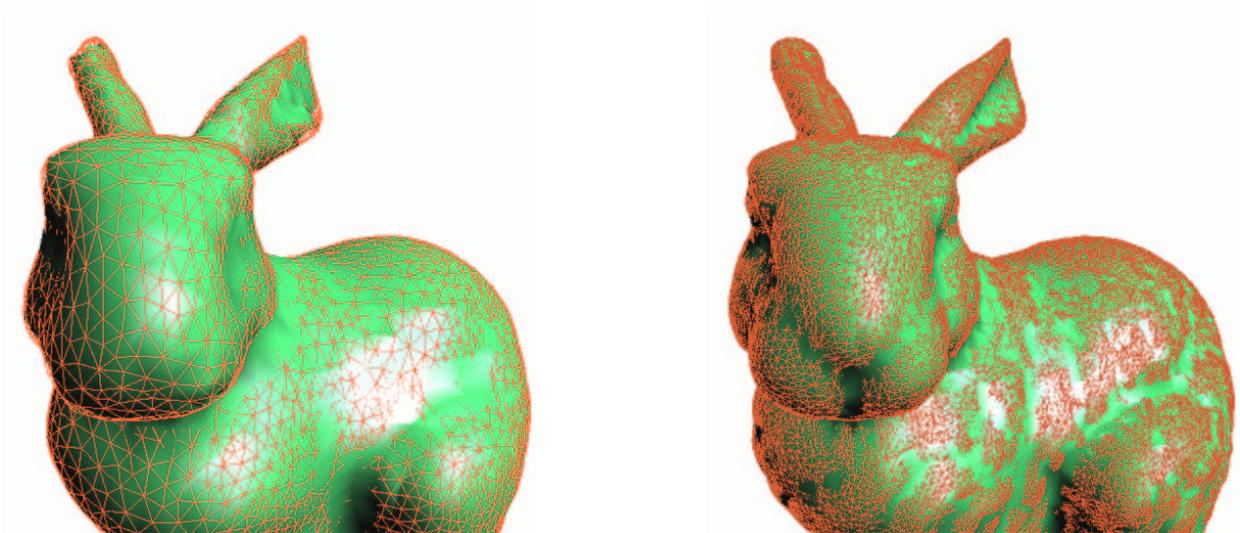


Figure 6. A close view of the bunny models presented in the Figure 5. We can see how the details are removed as the distance increase. The mesh shown in orange represents the original polygon mesh used to generate the implicit representation. The green shaded surface represents the polygonization result. The difference between them is caused by the limit precision of the interpolation process and the inherent finite precision of z-buffer.