EUROGRAPHICS '99 / P. Brunet and R. Scopigno
(Guest Editors)

Volume 18 (*1999*), *Number 3*

Multiresolution Implicit Representation of 3D Objects

Laurent Grisoni

Benoit Crespin

Christophe Schlick

LaBRI †
[grisoni|crespin|schlick]@labri.u-bordeaux.fr

Abstract

This paper presents a generic technique for converting any 3D object into an implicit representation, based on multiresolution implicit grids. A specific hash table structure is proposed, which permits compact storage as well as easy hierarchical evaluation of the object. Finally, a tessellator for these multiresolution implicit objects is presented, that takes advantage of the hash table structure to generate a quick hierarchical mesh representation.

Keywords: geometric modelling, implicit surfaces, multiresolution, level-of-detail algorithms, mesh generation.

1. Introduction

The most commonly used 3D geometric models in Computer Graphics are parametric representations, such as spline surfaces and meshes. These models present very nice properties such as interactive edition, and hardware visualisation. Another common geometric model is the implicit representation, where a surface is generated as an isosurface of a 3D potential field. This model offers the very powerful "blending property" (i.e. geometric continuous combination of objects[3, 32]). This nice property is counter balanced by the fact that implicit objects can not be visualized or manipulated as easily as parametric ones. As a result, a conversion tool between the two models may be interesting for many applications. This paper proposes such a conversion tool acting on both ways. First, it provides the conversion of any geometric object into an implicit representation, more precisely a multiresolution representation based on a data structure that allows efficient storage of the resulting objects. Note that the initial object has to be either a solid or a closed surface to generate a valid implicit representation. Second, a tessellator adapted to such multiresolution implicit objects provides a mesh representation for hardware visualisation and interactive edition.

The first step of our algorithm generates an *in/out grid* (IOG, for short) from the initial object. An IOG is simply a 3D rectangular grid where all the values are boolean and store whether the corresponding grid point is either inside or outside the original object. According to the initial representation of the object (mesh, spline surface, CSG tree, etc...), this step will be more or less efficient. There are mainly two alternatives for this computation. First one may sample the grid and test each point to see if it is inside or outside the object. Second, one may sample the surface and, for each point, mark the grid voxel it belongs to. Afterwards, a scan line algorithm can easily fill the grid between the marked voxels. Sometimes, it may also happen that the initial object is originally provided as a discrete 3D set; in recent years, several devices that achieve a 3D scanning of a physical object, based on ultrasonic or laser technology, have been developed. These devices generate a 3D discrete signal where each sample point contains the boolean information we need.

The second step uses the IOG to generate a *field grid* (FG, for short) which is a rectangular grid containing integer values. A FG is in fact a discrete and quantized approximation of a potential field that would generate, under a given tolerance, the original object as an isosurface for a given isovalue.

As the field functions used by implicit objects are usually

† Laboratoire Bordelais de Recherche en Informatique *(Université Bordeaux I* and *Centre National de la Recherche Scientifique)*. The present work is also granted by the *Conseil Régional d'Aquitaine*.

very smooth, data stored in the FG are good candidates for high level compression schemes. Thus, the third step, which generates the final implicit object, compresses the FG by using biorthogonal B-spline wavelets combined with a specific hash table data structure. the resulting representation, that we call *multiresolution implicit field* (MIF for short), is a true implicit model which can be blended with any usual implicit model (blobs, metabals, distance surfaces, convolution surfaces, etc...). Another nice feature of MIF is that they are well adapted to direct high quality rendering by incremental raytracing [13] but also to fast previewing by discrete shaded point z-buffering[26].

The fourth step generates a *multiresolution mesh* (MM for short) from the MIF. Note that this final step is not mandatory; it is only needed if one wishes to export the MIF to conventional modeling, rendering or animation environnements, for further manipulations.

2. From in/out grids to fields grids

As said before, one important step of our algorithm consists in generating a FG starting from an IOG. Velho and Gomes have been faced to a similar problem[30], for which they proposed to use a wavelet decomposition based technique, adapted from an edge detection technique presented by Mallat and Zhong [19]. Unfortunately this algorithm involves quite heavy mathematics, and can be hardly adapted to different kinds of objects. What we tried to do is to develop a solution that would be easier to use and adapt to any situation. The technique we propose has been detailled in [9], and is briefly recalled in this section.

This technique is a three pass process where the two first ones compute a signed distance for each grid sample, and the last one converts each distance value into a corresponding field value:

1. For each grid sample p inside the object, distance $D(p)$ to the frontier is calculated using some standard discrete distance transformation algorithm [23, 10]. Let Δ be the max of the computed distances.
2. For each grid sample p outside the object, a negative distance value to the frontier is calculated similarly. Computed distances are clamped to $-\Delta$.
3. Finally, distance values $D(p)$ are converted into field values $F(p)$ by using a field transformation

$$F(p) = f\left(\frac{\Delta - D(p)}{2\Delta}\right) \qquad (1)$$

where f is a continuous, positive, monotonously decreasing bijection on $[0,1]$.

2.1. Distance Transformation

The two main discrete distance transformation techniques have been presented years ago by Rosenfeld and Pfaltz[23], and later improved by Danielsson [10]. We will briefly recall them here (in 2D, for the sake of simplicity), and compare their use for the problem we deal with.

Rosenfeld and Pfaltz's technique consists in a two pass algorithm which provides for each sample point an approximation of the euclidean distance to the nearest kernel point. Initial values of the distance $D(p)$ separate the samples in two sets: those that belong to the kernel are set to 0 and those for which the distance to the kernel has to be calculated are set to ∞. This algorithm uses a square mask M depending on the type of distance one wishes to obtain (see below). The first pass works from top to bottom and left to right, and calculates $D(p)$ as follows:

$$D(p) = \min_{p_i}(D(p), D(p_i) + w_i)$$

where p_i are the neighboring samples covered by the mask that have already been treated in the same pass, and w_i are the corresponding weights given by the mask. The second pass of the algorithm processes symmetrically. According to the size of the mask, one may get a better or worse approximation of the euclidean distance. The best approximation with a 3×3 mask is given by the $M_{3,4}$ mask ($M_{3,4,5}$ in the 3D case):

4	3	4
3	0	3
4	3	4

Danielsson's method, which is also a two-pass algorithm, provides exact euclidean distances. It uses three distance values instead of one: $D_x(p)$ and $D_y(p)$ store the horizontal and vertical distance to the closest reference pixel, whereas $D(p)$ stores the actual euclidean distance. Initial values are set again to 0 for the reference samples and ∞ for the others. Instead of using approximated distance values, the comparison is made on the squared sum of the vertical and horizontal distances. Once $D_x(p)$ and $D_y(p)$ have been calculated for the whole image by the two passes, a last additional pass is required to calculate $D(p)$. Note that, as $D(p)^2$ can only take a finite number of integer values, this last step can done by using some look-up table, instead of computing a square root.

2.2. Field transformation

Function f used in our algorithm (see equation 1) to transform the distance value $D(p)$ into a suitable potential field value $F(p)$ plays a critical role in the conversion. Indeed, it determines the blending properties of the final object: soft blending if the function slowly decreases to 0 outside the object, sharp blending if the function quickly decreases outside the object, or even no blending at all, if we use the step function:

$$\forall\ t \in [0,1], \quad f(t) = (t < \frac{1}{2})\ ?\ 1\ :\ 0$$

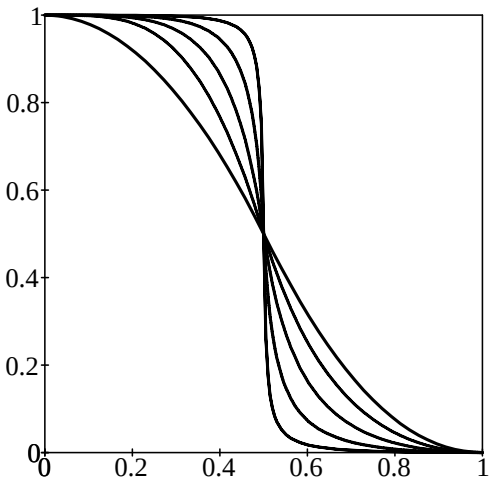

Figure 1: *Field transformation function.*

Some valuable properties for this field function have been exhibited in the literature [16, 2]: $f(0) = 1$, $f(1) = 0$, $f'(t) \leq 0$, $f'(1) = 0$, $f(\frac{1}{2}) = \frac{1}{2}$, etc... An inexpensive function that meets all these conditions is the cardinal step:

$$\forall\ t \in [0,1], \quad f(t) = 1 - t^2(3-2t)$$

Nevertheless it would be better to get a degree of freedom s to control the steepness of the function at $t = \frac{1}{2}$, this parameter may then be used as soft-to-hard slider. We propose the following function:

$$\forall\ t \in [0,1], \quad f_s(t) = 1 - \gamma_s(t)$$

where $\gamma_s(t)$ is Perlin's bias function [22] or its rational polynomial alternative [24]. Several plots of function f_s with different values of parameter s are shown in figure 1. Here again, note that, as the distance value D can only take a finite number of values, the whole field transformation can be implemented as a single look-up table. This means that there is no need to store field values as floating point numbers, since a short index pointing on a floating point "field map" provides straightforward lossless compression.

Starting from the IOG illustrated on figure 2, figure 3 and figure 4 present the FG obtained by this algorithm, using respectively Rosenfeld and Pfaltz's technique (with $M_{3,4}$) and Danielsson's technique. The field has been remapped on a specific greyscale map for better interpretation: field values in the range $[0.5, 1]$ (which belong to the *hard zone* of the implicit object) are linearly mapped to $[255, 0]$ whereas values in the range $[0, 0.5]$ (which belong to the *soft zone*) are linearly mapped to $[0, 255]$. Danielsson's technique provides obviously better results but requires an amount of memory which is three times more important (four times in the 3D case); this may be restrictive for very large grids.

Figure 2: *Test shape.*

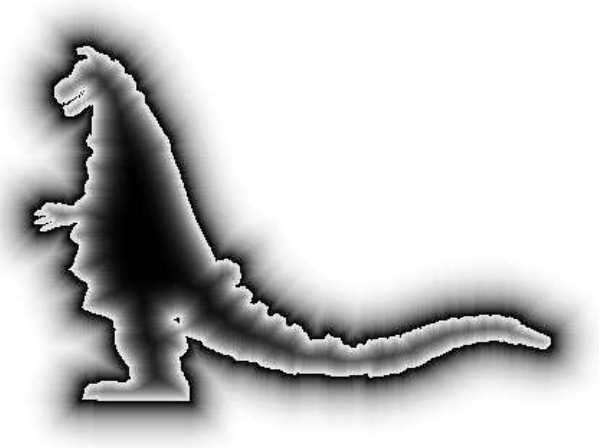

Figure 3: *Field transformation based on Rosenfeld and Pfaltz's algorithm.*

Figure 4: *Field transformation based on Danielsson's algorithm.*

3. From field grids to multiresolution implicit field

At this stage of the conversion process there are two problems that have not been solved yet. First, we have to convert the FG into a continuous representation. A simple idea would be to add an interpolation scheme (trilinear or tricubic, for instance). Such a solution works fine but does not solve our second problem which is the memory cost of the technique described so far. Indeed, potential fields usually involve very few high frequencies (this can easily be checked on figure 4 where the image is blurred almost everywhere, except in a small area around the frontier) which means that there is no need to store a high definition FG everywhere. It seems mandatory to find a compression scheme that lessens the data amount. Wavelet based decomposition schemes naturally provide a useful framework for that, and they straightforwardly generate a continuous representation, which also solves our first problem. Using wavelet technique to compress 3D data grids is not a new idea. Muraki [20] used such an approach for the simplification of IMR data. Gross and Lipppert presented techniques for volume rendering based on wavelet decompositions [13, 17]. In the field of implicit objects, Velho and Gomes [30] have proposed to use wavelets for multiresolution representation of implicit objects. In comparison to previous work, our contribution is based on two innovations: first, a switch in the way to consider wavelet transformation for data compression, and second, a simple and customizable data structure adapted to multiresolution implicit objects, that permits compact storage and fast evaluation. The remainder of this section describes these two ideas.

3.1. Multiresolution representation

We do not plan to give here a complete overview of what wavelets are. The reader might refer to the monography written by Stollnitz, DeRose and Salesin [25] for an introduction, and a list of possible applications in computer graphics. More theoretical considerations may be found in applied mathematics papers [11, 27, 8].

The key notion in wavelet theory stays in the definition of a sequence of nested function spaces V^i called *approximation spaces*:

$$V^0 \subset V^1 \subset \cdots \subset V^n$$

where each space V^i stores an approximation of a given function (V^n stores the original function and V^0 the coarsest approximation). A set of basis functions φ_k^i is usually defined to span the approximation spaces V^i. In the case of compactly supported basis functions, the influence of the basis functions depends on the level of decomposition at which they are involved: typically, the support of φ_k^i is twice as large as the support of φ_k^{i+1}. This property will be used to develop a multiresolution tessellator (see section 4). As the basis functions become more and more localized in the manner they influence the resulting object, one can get an approximation that is as accurate as needed, simply by going far enough in the V^i hierarchy.

In addition to the approximation spaces V^i, the wavelet decomposition also involves a set of *detail spaces* W^i, which are defined as the (usually orthogonal) complement of V^i in V^{i+1}. Practically, each of the W^i stores the error made when going down from the approximation in V^{i+1} to the one in V^i. Some *wavelet* basis functions ψ_k^i that span the detail spaces W^i are defined, similarly to the φ_k^i functions.

In the case of biorthogonal wavelets[8], the theoretical frame that we will use, wavelet multiresolution schemes involve four (usually finite) filters, noted h,g,$\tilde{h}$ and $\tilde{g}$, to achieve the *decomposition* and the *reconstruction* steps. If c^n (respectively d^n) is the set of coordinates of a one dimensional function in space V^n (respectively W^n), the *decomposition* is provided by:

$$c_i^{n-1} = \sum_k \tilde{h}_{k-2i} c_k^n \qquad d_i^{n-1} = \sum_k \tilde{g}_{k-2i} c_k^n$$

and the *reconstruction* by:

$$c_i^n = \sum_k h_{k-2i} c_k^{n-1} + g_{k-2i} d_k^{n-1}$$

The extension of this decomposition and reconstruction process to higher dimension spaces is classically done by tensor product. For our 3D field grids for instance, three consecutive 1D decompositions are performed. The first one is applied on the rows of the grid, each of which being replaced by the c^{n-1} coordinates in the first half and the d^{n-1} coordinates in the other half. The same technique is then applied on the columns of the resulting volume, and finally on the lines of the remainding dimension. In that case, each decomposition step divides the amount of data by eight.

The choice of a wavelet family is of major importance, because many classical wavelet basis functions demand heavy computation[27]. Due to their smooth decrease and local influence, B-splines are good candidates as implicit primitives [29]. Consequently, B-spline based wavelets should be an interesting representation. Starting from this postulate, we have shown that wavelets biorthogonal B-spline wavelets with 4 vanishing and 4 dual vanishing moments appear to be a reasonnable choice[12].

Note also that a major difficulty in usual wavelet decompositions is how to handle boundaries. In our particular case, we have the theoretical guaranty that the potential field smoothly decreases to zero as the distance to the object increases. Consequently, if the FG is large enough, all the boundary voxels will be null, which means that classical boundary handling techniques[27] will provide good results.

The result of our conversion process is illustrated on figure 5. The original object, shown on the upper left corner, is a mesh, provided as a test file for CosmoPlayer on SGI environnements, representing Beethoven's bust (5030 triangles). This mesh has been converted sequentially to an IOG, a FG

and wavelet field representation; each grid has 256^3 samples. The final object, shown on the upper right corner, is an implicit equivalent of the initial mesh, which supports usual implicit transformations. For instance, the middle image illustrates an example of blending with an implicit sphere. The two bottom pictures presents the effect of varying the isovalue: the dilated version is generated using an isovalue of 0.4 and the contracted version, an isovalue of 0.6. As already mentionned above, note that the object gets blurred quite rapidly when leaving the initial isovalue. Another example is given in figure 6. It shows the result of the conversion for high density grid ($512 \times 512 \times 512$). As one can see, the result is better than the conversion at lower resolutions, but the disadvantage is that the amount of data becomes difficult to handle in a simple grid, which is why compression schemes are necessary. The next section discusses this issue.

3.2. Wavelet based compression

Wavelet decompositions provide a very powerful tool for compression, in the sense that a detail space W^i stores what is lost when going from an approximation in space V^{i+1} to a coarser one in V^i. As a result, a small coefficient in this detail space W^i means that, at this location, the difference of the two approximations in V^{i+1} and V^i is negligible and, as a result, the coefficient can be assumed to be null. Hence, a cancellation of detail coefficients according to some threshold provides an easy and efficient compression technique. This principle is the basis of any wavelet-based compression technique proposed so far. Nevertheless when applied on field grids, this process does not provide interesting results. The main reason is that usual wavelets (in our case, biorthogonal cubic B-spline wavelets) are functions with many oscillations. Thus, when representing a smooth function, wavelet coefficients are generated so that neighboring wavelets compensate their oscillations each other; but if the compression process cancels one wavelet, its neighbors do not compensate any longer and some ringing phenomenon appears in the compressed picture (see figure 7). To our knowledge, this drawback has rarely been mentionned and never been explained; maybe because it is much less noticeable in high frequencies images (such as real life photographs, the usual application domain for wavelet compression) than in low frequency images (such as our field grids).

To solve this problem, we propose a new way to consider wavelet transformation for data compression. What we do is simply to express each projection on a detail space W^i generated by the decomposition process into its immediately superior approximation space V^{i+1}. Hence, at the end we get an object that is totally represented as a finite hierarchical sum of B-spline basis functions, without any wavelet function. More precisely, the approximation f_i at level i of the field function is now calculated as follows:

$$f_i(p) = f_0(p) + \sum_{j=0}^{i-1} \delta f_j(p) \qquad (2)$$

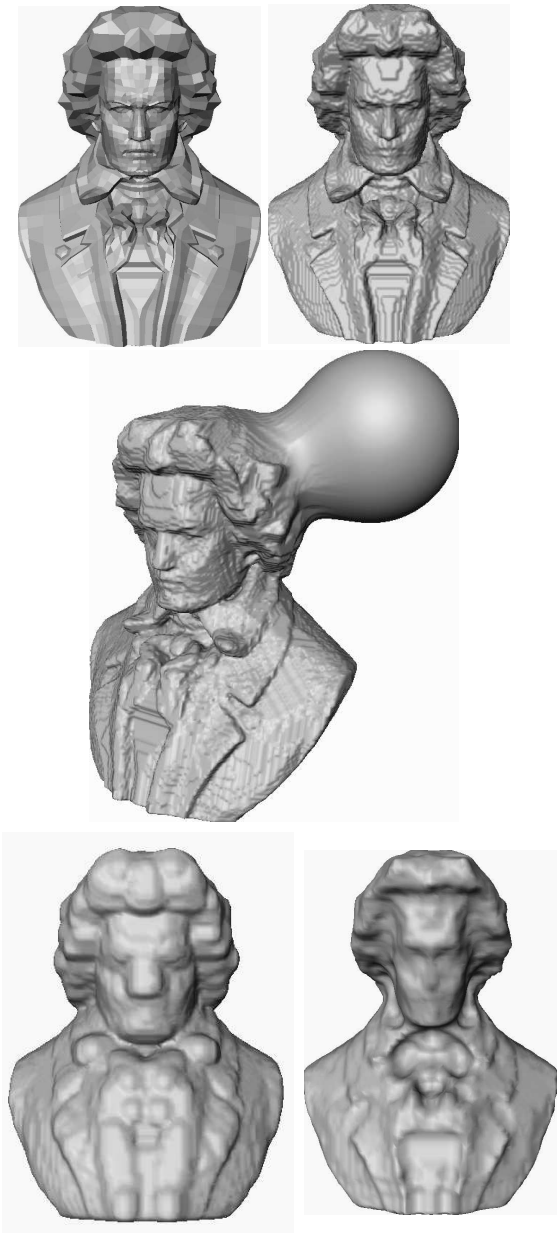

Figure 5: *From up to bottom and left to right: the original mesh, its equivalent implicit representation, and 3 implicit manipulations (blending, dilatation and contraction). All implicit representations are generated on a* $256 \times 256 \times 256$ *grid.*

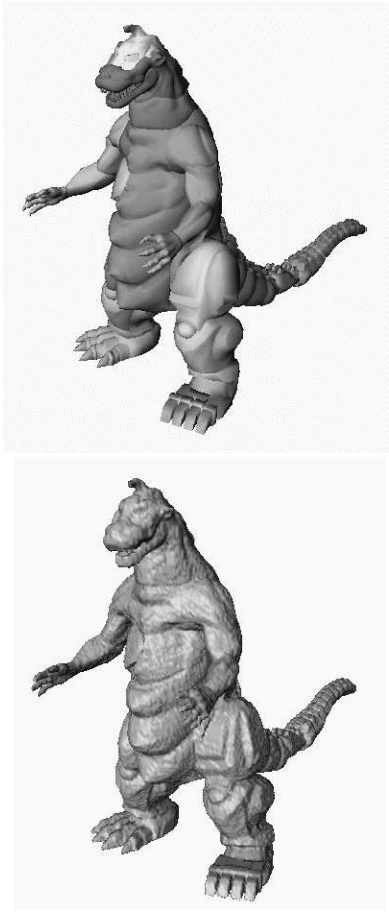

Figure 6: *From top to bottom: the initial mesh (49478 triangles) and its implicit representation using a* $512 \times 512 \times 512$ *grid.*

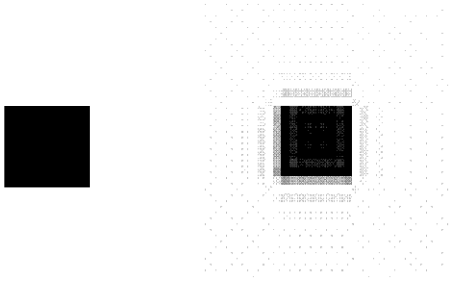

Figure 7: *An initial* 256×256 *image, and its projection on a* 32×32 *grid. Oscillations do not compensate each other anymore.*

or

$$f_i(p) = f_{i-1}(p) + \delta f_{i-1}(p) \tag{3}$$

where $\delta f_j(p)$ stands for the additional B-spline functions resulting from the projection of the original function f onto W^j. Equation 3 suggests some incremental evaluation of the field: this property will be used in our multiresolution tesselator. The main advantage of this transformation comes from the fact that B-spline basis functions are smoothly decreasing functions without oscillations. Consequently, ringing phenomena are totally prevented applying the "cancellation by threshold" compression technique on the B-splines coefficients. As a counterpart, the decomposition can not be done at constant memory space anymore, as with usual wavelet decomposition, because of these additional B-spline coefficients. Fortunately, an implementation trick can be used to generate this representation at very little memory overhead:

- The coefficients of the approximation space V^0 are first stored in the final data structure, described in section 3.3.
- Every coefficient that belongs to space V^0 is then cancelled, and the projection on V^1 is calculated, using the new (null) coefficients in V^0 and the wavelet coefficients in W^0. As a result, the projection on W^0 generated by the decomposition process is now expressed in space V^1.
- This process is recursively repeated until the finest approximation space V^n is reached.

In other words, this process iteratively destructs the grid, while the final data structure is constructed.

3.3. Data structure

This section details the last step of our conversion process, which translates the uncompressed wavelet representation of field grids into our final compressed multiresolution implicit fields (MIF). From now on, we call *source* each elementary B-splines function at a given multiresolution level, and *intensity* its corresponding coefficient generated by the decomposition process.

For each source, the information one has to store is the level at which it is involved, its position in the grid, and its intensity. The data structure we propose is based on two remarks. First, the intensities of the sources may obviously be quantized. Indeed, the precision offered by pure floating point storage of the coefficients generated by wavelet decomposition largely exceeds the needs of practical computer graphics applications. As sources belonging to coarse resolution levels obviously appear as more important for the resulting object than sources in fine resolution levels, adaptative quantization would be desirable. Second, the structure of wavelet decomposition implies that the range of the source coordinates is halved when going from level $i+1$ to level i. From the quantization point of view, this means that only i bits are needed to store a coordinate at the level i. As we manipulate 3D functions, the total number of bits to store the

coordinates for each source at the level i is $3i$. The remaining bits can then be used to increase the quantization precision when the level decreases. Thus, we get a more precise representation for deeper sources, which exactly meets the desire of adaptive quantization we mentionned above.

Our current implementation permits 8 levels of decomposition from space V^{10} (a 1024^3 grid) to space V^3 (a 8^3 grid). But it may easily be adapted to higher resolutions or deeper decompositions. Here are the implementation details:

- A hash table of 4096 entries is constructed. An entry contains a list of 3-byte blocks, where each block corresponds to one source. Hence, we use $log_2(4096) + 8 * 3 = 36$ bits per source. Among these, we use 3 bits to store level i, $i+3$ bits for each source coordinate, and hence $36 - 3*(i+3) - 3 = 24 - 3i$ bits remain for the quantized intensity s. Note that the range of stored levels i varies from 0 to 7 but the range of the actual multiresolution level varies from 3 ($8 \times 8 \times 8$) to 10 ($1024 \times 1024 \times 1024$).
- Each entry key of the hash table is a 12-bit word composed of the four less-significant bits of each coordinates. Concatenating less-significant bits offers a good shuffle for the resulting keys. Note that only 3 bits per coordinate are involved at the coarsest level ($8 \times 8 \times 8$). In that case, coordinates are left padded with 0 to get the 12-bit entry.
- Finally, each 3-byte block is constructed by concatenating the 24 remaining bits used to define the source.

Figure 8 shows a MIF using an initial IOG resolution of $256 \times 256 \times 256$. Several threshold values have been used, providing several compression ratios. One can see that despite the extreme compression ratio, the shape of the original object is nicely preserved. The only artefact that appears is a progressive smoothing of the surface compared to the initial, non-compressed version.

In addition to high compression ability, MIF also offer two other main valuable properties. First, the evaluation of field values at any point is immediate; it only involves bit mask operations to generate the coefficients from the hash table, and some look-up table access to obtain the final value from the coefficients. This should be compared with expensive point-to-primitive distance computations required by usual implicit models. Second, as mentionned in the introduction, MIF may be directly visualized, either by incremental ray tracing for high quality rendering or by discrete shaded point z-buffering (possibly done by hardware) for fast preview.

4. Multiresolution Tessellation

Even though high or low quality direct visualisation of MIF is possible, having a tessellation tool offers valuable benefits, such as hardware rendering and export to standard file formats. Several tessellation techniques exist for generic implicit objects. Bloomenthal presented a classification [21, 6] including most of them: uniform and adaptive marching cubes [32, 4, 5], shrinkwrap [28, 7] critical point based technique [14, 15],

Figure 8: *From up to bottom and left to right: the original object (1706140 sources), and several compressed versions (using respectively 58569, 32071, and 15145 sources) using an initial* $256 \times 256 \times 256$ *grid. The last MIF only takes about 45 kb in memory.*

particle relaxation [31]. Such generic tessellators are not necessarily optimal for our purpose, because we may benefit from some specific properties of MIF to speed up the tessellation:

- First, all sources that belong to a given level can easily be extracted from the hash-table structure. One can thus efficiently generate a partial evaluation of the field up to a given level.
- Second, if one stores partial field values $f_i(p)$ at each considered voxel vertex p, equation (3) offers an efficient incremental process to compute global field values $f(p)$.
- Third, the use of B-splines offers some nice properties about the topological consistency. Their variance diminution property guarantees that there will be no unwanted oscillations if one takes voxels that are equal to the 64th of the smallest involved B-spline support at a given current level (in the case of cubic B-splines). This means that the topology of the resulting mesh will match that of the original object.

Our multiresolution tessellator is based on a marching-cubes algorithm which acts sequentially on each level of the MIF. At a given level i, the voxels of the grid are *treated*, which means:

- evaluate the field at the voxel vertices, up to the sources of

level i, using equation (3) or (2) whether the voxel already exists or not at level $i-1$.
- store the voxel in the octree and flag it as "done" for this level.

Once the treatment of a given level is completed, we *flush* the octree in order to keep only the voxels that are at the boundary of the implicit volume (i.e. at least one vertice inside and one outside).

More precisely, here are the different steps of the algorithm:

- Every source of the coarsest approximation is extracted from the hash-table structure. For each source, the 64 voxels corresponding to its support are *treated* and the resulting octree is *flushed*.
- For each level of resolution, every corresponding source is extracted from the hash-table structure. For each source, the list of octree leaves intersecting the support of the source is constructed. If this list is empty, then all 64 voxels of the source are *treated*. Otherwise, each voxel of the list is recursively divided, unless the lower bound is reached. During the subdivision the field values are evaluated incrementally with equation (3). Once the octree is up to date for this level, it is *flushed*.
- After each octree flush, one can get a mesh if needed. As we have an octree where each leaf is on the surface, we can apply classical tessellation methods, using Kuhn simplices for example [5]. Figure 9 shows an example of several meshes generated at different levels.

5. Conclusion and Future Work

This paper presented a framework for converting any 3D object into a multiresolution implicit representation. This process uses a distance based field grid generation, combined with an adequate data structure for storing the resulting object. These *multiresolution implicit fields* (MIF) may be directly visualized either by ray-tracing for high-quality rendering, or by discrete shaded point z-buffering for fast preview. They may also be tessellated to be included in classical computer graphics environnements. We showed how one can take advantage of the multiresolution structure to generate a fast and incremental tessellation preserving the topology.

We are currently investigating two directions for the MIF model. The first one is to improve the tessellator which suffers from the usual drawbacks (irregularly shaped triangles) of marching cubes methods. The solution may be to insert an iterative process that would let the mesh vertices freely evoluate on the implicit surface, using some mesh relaxation [15] to get more regularity. The second challenging issue for MIF is to define the same kind of tools that are available for multiresolution B-spline curves and surfaces, which provide interactive edition at different multiresolution levels [25]. One possibility may be to use some discrete hierarchical warping algorithms.

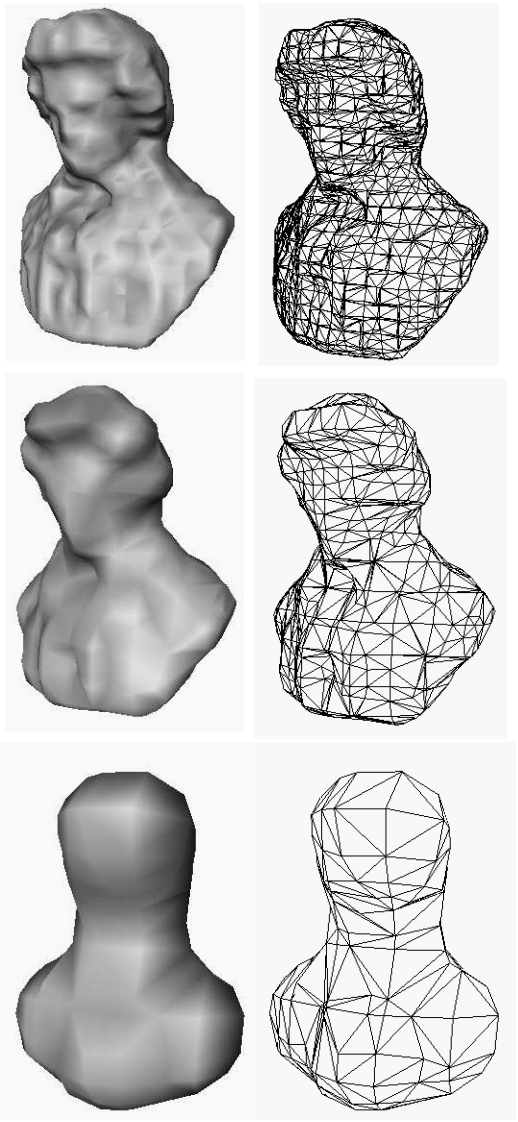

Figure 9: *From top to bottom: three different meshes (shaded, and wireframe versions) generated in beethoven's implicit representation tessellation levels, from fine to coarse, using the representation provided by a* $256 \times 256 \times 256$ *grid.*

References

1. E. Bittar, N. Tsingos, M.P. Gascuel, *Automatic Reconstruction of Unstructured 3D Data : Combining a Medial Axis and Implicit Surfaces*, Proceedings of Eurographics'95, pp. 457–468, 1995.
2. C. Blanc, C. Schlick, *Extended Field Functions for Soft Objects*, Proceedings of Implicit Surfaces'95, pp. 21–32, 1995.
3. J.F. Blinn, *A Generalization of Algebraic Surface Drawing*, ACM Transaction on Graphics, vol. 1, no. 3, pp. 235–256, 1982.
4. J. Bloomenthal, *Polygonization of Implicit Surfaces*, Computer Aided Geometric Design, vol. 5, no. 4, pp. 341–355, 1988.
5. J. Bloomenthal, *An Implicit Surface Polygonizer*, in: Paul Heckbert (ed.), *Graphics Gems IV*, Academic Press, New York, 1994.
6. J. Bloomenthal & Al., *Introduction to Implicit Surfaces*, ISBN 1-55860-233-X, Morgan Kaufmann Publishers Inc., 1997.
7. A. Bottino, W. Nui, K. Van Overveld, *How to shrinkwrap through a critical point: an algorithm for the adaptive triangulation of iso-surfaces with arbitrary topology*, In Proc. Implicit Surface'96, pp. 53–72, 1996.
8. A. Cohen, I. Daubechies, J.-C. Feauveau, *Biorthogonal Bases of Compactly Supported Wavelets* , Comm. on Pure and Applied Mathematics, vol. 15, pp. 485–560, 1992.
9. B. Crespin, C. Schlick, *Generating implicit field function from in/out images*, Implicit Surfaces '98 proceedings, pp. 91–98, 1998.
10. P.E. Danielsson, *Euclidean Distance Mapping*, Computer Graphics and Image Processing, vol. 14, pp. 227–248, 1980.
11. I. Daubechies, *Ten Lectures on Wavelets*, CBMSF-NSF Regional Conf. Series in Appl. Math.w, vol. 61, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.
12. L. Grisoni, C. Schlick, *Multiresolution representation of implicit objects*, Implicit Surfaces '98 proceedings, pp. 1–10, 1998.
13. M.H. Gross, L. Lippert, R. Dittrich, S. Haring, *Two Methods for Wavelet-Based Volume Rendering*. Computers & Graphics , vol. 21, no. 2, pp. 237-252, 1997.
14. B. T. Stander, J. C. Hart, *Guaranteeing the topology of an implicit surface polygonization for interactive modeling*, Computer Graphics (Annual Conference Series), Aug. 1997, pp. 279–286, 1997.
15. J. C. Hart, A. Durr, D. Harsh, *Critical Points of Polynomial Metaballs*, Implicit Surface'98 Proc., pp. 69–76, 1998.
16. Z. Kacic-Alesic, B. Wyvill, *Controlled Blending of Procedural Implicit Surfaces*, Proceedings of Graphics Interface'91, pp. 236–245, 1991.
17. L. Lippert, M. H. Gross, C. Kurmann, *Compression Domain Volume Rendering for Distributed Environments*, Eurographics '97 Proc., COMPUTER GRAPHICS forum, vol. 16, no. 3, pp. 95-107, 1997.
18. T. Lyche, K. Morken, *Spline Wavelets of Minimal Support*, Numerical Methods of Approximation Theory, vol. 9, pp. 177–194, 1992.
19. S. Mallat, S. Zhong, *Characterization of Signals from Multiscale Edges*, IEEE Transactions on P.A.M.I., vol. 14, pp. 710–732, 1992.
20. S. Muraki, *Volume Data and Wavelet Transform*, IEEE Computer Graphics & Applications, vol. 13, no. 4, p. 50-56, 1993.
21. P. Ning, J. Bloomenthal, *An Evaluation of Implicit Surface Tilers*, IEEE Computer Graphics & Applications, pp. 33–41, november 1993.
22. K. Perlin, *An Image Synthesizer*, Siggraph'85 proceedings, pp. 287–296, 1985.
23. A. Rosenfeld, J.L. Pfaltz, *Sequential Operations in Digital Picture Processing*, Journal of ACM, vol. 13, no. 4, pp. 471–494, 1966.
24. C. Schlick, *Fast Alternatives to Perlin's Bias and Gain Functions*, Graphics Gems IV, pp. 379–382, April 1994.
25. E. Stollnitz, T. DeRose, D. Salesin, *Wavelets for Computer Graphics*, theory and applications, ISBN 1-55860-375-1, Morgan Kaufmann Publishers Inc., 1996.
26. N. Stolte, A. Kaufmann, *Visualization of Implicit Surfaces in Spherical Coordinates*, Implicit Surface'98 proceedings, pp. 11–18, 1998.
27. G. Strang, *Wavelets and Dilation equations: a brief introduction*, SIAM Review, vol. 31, no. 4, pp. 614–627, December 1989.
28. K. Van Overveld, B. Wyvill Shrinkwrap: an adaptive algorithm for polygonizing an implicit surface. Tech. Rep. 93/514/19, University of Calgary, Dept of Computer Science, March 1993.
29. L. Vehlo, J. Gomes, D. Terzopoulos, *Multiscale implicit objects*, in proceedings of V SIBGRAPI, 1994.
30. L. Velho, J. Gomes, *Approximate Conversion of Parametric to Implicit Surfaces*, in proceedings of Implicit Surfaces '95, pp. 77–96, 1995.

31. A. Witkin, P. S. Heckbert, *Using Particles to Sample and Control Implicit Surfaces*, Computer Graphics Annual Conference Series, pp. 269–277, 1994.

32. G. Wyvill, C. McPheeters, B. Wyvill, *Data Structure for Soft Objects*, vol. 2, no. 4, pp. 227–234, 1986.